
CASAL
(C++ algorithmic stock assessment laboratory)

CASAL User Manual v2.30-2012/03/21

B. Bull
R.I.C.C. Francis
A. Dunn
A. McKenzie
D.J. Gilbert
M.H. Smith
R. Bian
D. Fu

NIWA Technical Report 135
ISSN 1174-2631
2012

CASAL
(C++ algorithmic stock assessment laboratory)

CASAL User Manual v2.30-2012/03/21

B. Bull
R.I.C.C. Francis
A. Dunn
A. McKenzie
D.J. Gilbert
M.H. Smith
R. Bian
D. Fu

NIWA Technical Report 135
ISSN 1174-2631
2012

Published by NIWA
Wellington
2012

Enquiries to:
Science Communication, NIWA,
Private Bag 14901, Wellington, New Zealand

ISSN 1174-2631

Citation:
Bull, B.; Francis, R.I.C.C.; Dunn, A.; McKenzie, A.; Gilbert, D.J.; Smith, M.H.; Bian, R.; Fu, D. (2012). CASAL (C++ algorithmic stock assessment laboratory): CASAL User Manual v2.30-2012/03/21. *NIWA Technical Report 135*. 280 p.

The National Institute of Water and Atmospheric Research
is New Zealand's leading provider
of atmospheric, marine, and
freshwater science

Visit NIWA's website at http://www.niwa.co.nz

CONTENTS

1. INTRODUCTION

CASAL (C++ algorithmic stock assessment laboratory) is a generalised age- or size-structured fish stock assessment model that allows a great deal of flexibility in specifying the population dynamics, parameter estimation, and model outputs.

This manual provides information on how to use CASAL, including how to run CASAL, how to set up the input data files, descriptions of the population dynamics and estimation methods, and how to generate outputs. It also contains a brief overview of the technical specifications of the software, and examples of models using CASAL.

CASAL is designed for flexibility. It can implement either an age- or size-structured model, optionally also structuring the population by sex, maturity, and/or growth-path. It can be used for a single stock for a single fishery, or for multiple stocks, areas, and/or fishing methods. The user can choose the sequence of events in a model year. The data used can be from many different sources of information, for example catch-at-age or catch-at-size data from commercial fishing, survey and other biomass indices, survey catch-at-age or catch-at-size data, and tag-release and tag-recapture data. Estimation can be by either maximum likelihood or Bayes.

As well as generating point estimates of the parameters of interest, CASAL can calculate likelihood or posterior profiles and can generate Bayesian posterior distributions using Monte Carlo Markov Chain methods. CASAL can project stock status into the future using stochastic recruitment and can generate a number of yield measures commonly used in New Zealand stock assessments, including MCY, CAY, F_{max}, $F_{0.1}$, deterministic MSY, and CSP.

2. GETTING STARTED

2.1 CASAL end user licence

Redistribution and use of CASAL (including the documentation, examples, and other ancillary files) in source and binary forms, with or without modification, are permitted provided the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimers.
2. Redistributions in binary form must reproduce the copyright notice, this list of conditions and the following disclaimers in the documentation and/or other materials provided with the distribution.
3. CASAL contains the libraries ADOL-C, newran, and GetPot. ADOL-C is available free of charge from https://projects.coin-or.org/ADOL-C and is available under the terms of the Common Public License (see www.eclipse.org/legal/cpl-v10.html); newran is freely available and can be obtained from www.robertnz.net; and GetPot is distributed under GNU Lesser General Public License (www.gnu.org/licenses/lgpl.html) and is available from http://getpot.sourceforge.net.
4. NIWA is to be acknowledged in publications relating to the use of or from conclusions drawn from the use of CASAL.
5. Neither the name of the NIWA nor the names of CASAL's contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors ‘as is’ and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

2.2 Version

This document details the usage of CASAL version v2.30-2012/03/21. The version number printed by CASAL is suffixed with a date in format `yyyy/mm/dd`. This is the last UTC date on which its source files were officially modified. User manual updates will usually be issued for each minor version or date release of CASAL, and can be obtained, on request, fromCASAL@niwa.co.nz.

2.3 Citing CASAL

A suitable reference for CASAL and this document is:

Bull, B.; Francis, R.I.C.C.; Dunn, A.; McKenzie, A.; Gilbert, D.J.; Smith, M.H.; Bian, R.; Fu, D. (2012). CASAL (C++ algorithmic stock assessment laboratory): CASAL User Manual v2.30-2012/03/21. *NIWA Technical Report 135*. 275 p.

2.4 System requirements

CASAL is available for most IBM compatible machines running Linux, from the command prompt under most Microsoft Windows operating systems.

Several of CASAL's tasks are highly computer intensive and a powerful processor is recommended. We recommend a minimum of 100 megabytes of free RAM for running CASAL (although, depending on the scope of the problem, you may need much more). The program itself requires less than 10 megabytes of hard-disk space but output files can consume large amounts of disk space. Depending on the number and type of user output requests, the output could range from a few hundred kilobytes to several hundred megabytes.

2.5 Necessary files

In Linux, only the executable file `casal` is required to run CASAL. For Microsoft Windows, the executable file `casal.exe` is needed. The Linux version is 64-bit, while the Microsoft Windows is 32-bit.

2.6 Useful add-ons

No software other than the appropriate operating system or emulation package is required to run CASAL. However, as CASAL offers little in the way of post-processing of the output, most users will wish to have a package available that allows tabulation and graphing of model outputs. We recommend the use of software packages such as Microsoft Excel (www.microsoft.com), S-Plus (www.insightful.com), or R (www.r-project.org) (R Development Core Team 2007).

Users may also wish to use the 'extract CASAL output' R functions for post-processing CASAL output (see Section 13). These are distributed as the R package `casal`.

A useful package for post-processing and analysis of Monte-Carlo Markov Chain (MCMC) Bayesian output is the R package "Bayesian Output Analysis Program (BOA)" — see Smith (2001). Information about this package can be found at www.public-health.uiowa.edu/boa.

The utility program `simCASAL` is available for assisting in running simple operating model/estimation model experiments in CASAL. See Section 12.1 for details.

2.7 Getting help

CASAL is distributed as unsupported software. NIWA does not provide help for users of CASAL outside of NIWA, although notification of any problems or errors in CASAL is invited. Updates may or may not correct these problems or errors — see Section 11.3 for how to report errors. Information about CASAL can be found at http://www.niwa.co.nz/our-science/fisheries/tools/casal. The maintainer of this software, documentation, and associated files can be contacted at CASAL@niwa.co.nz.

2.8 Technical specifications

CASAL is compiled on Linux using `gcc`, the C/C++ compiler developed by the GNU Project (http://gcc.gnu.org). The Linux version uses `gcc` version 4.1.2 (SuSE Linux). Note that CASAL is not supported for Linux kernel versions prior to 2.6. The Microsoft Windows version is compiled using Microsoft Visual C++ in Microsoft Visual Studio 2005, and should run on most 32 bit WindowsXP and Windows Vista systems. There are no current plans to port CASAL to Microsoft Windows 64 bit platforms. Note that the output from CASAL may differ slightly on the different platforms due to different precision arithmetic or other platform dependent implementation issues. The source code for CASAL is available on request.

CASAL uses a quasi-Newton optimiser and scalar, vector, and matrix types from the `Betadiff` automatic differentiation software package. `Betadiff` emulates most of the functionality of an early version of AUTODIF (Fournier 1994), and is based on a slightly modified version of the program ADOL-C v1.8.4 “A package for automatic differentiation of algorithms written in C/C++” (https://projects.coin-or.org/ADOL-C) developed by a team including Andreas Griewank (Technical University of Dresden, griewank@math.tu-dresden.de). A suitable reference for ADOL-C is Griewank et al. (1996).

The optimiser used by `Betadiff` is based on the main algorithm of Dennis Jr. & Schnabel (1996).

The random number generator used by CASAL is the `newran` random number generation package (Davies 1998), and uses the Lewis-Goodman-Miller algorithm with Marsaglia mixing.

3. RUNNING CASAL

CASAL is controlled by command line arguments, which initiate various required tasks — for example, running the model, estimating the parameters, or completing a MCMC run. Section 3.1 lists these command line arguments.

CASAL gets most of its information from input data files. The program looks for three files, `population.csl`, `estimation.csl`, and `output.csl`, which contain population, estimation, and output parameters respectively (although the names of these files can be modified, see later). Section 3.4 describes how to construct a CASAL data file — the population, estimation, and output file parameters are listed in Sections 8, 9, and 10 respectively.

Note that the information is read in from the three data files at the start of each CASAL run. As a result, you can change the data files and start another run in the same directory before the first run is finished, provided that the outputs of the two runs are sent to different destinations.

To use CASAL, open a command prompt window (Microsoft Windows) or a terminal window (Linux). Navigate to the directory where your CASAL input configuration files are located. Then type `casal` with the selected arguments. CASAL will print output to the screen, and return to the command prompt when it completes its task. Note that the CASAL executable must be either in the directory where you run it or somewhere in your operating system `PATH`. See your operating system documentation for help on identifying or modifying your `PATH`.

CASAL uses both the standard output and standard error; we suggest redirecting both into files, but not if using CASAL in an interactive mode (for example, with `-Y`).

With the bash shell (on Linux systems), use the command structure,

```
(casal [arguments] > out) >& err &
```

It may also be useful to redirect the standard input, especially if using CASAL inside a batch job or with queuing software, i.e.

```
(casal [arguments] > out < /dev/null) >& err &
```

On Microsoft Windows systems, you can redirect to standard output using,

```
casal [arguments] > out
```

And, on some recent Microsoft Windows systems (e.g., Professional versions of Windows NT, Windows 2000, and Windows XP), you can redirect to both standard output and standard error, using the syntax,

```
casal [arguments] > out 2> err
```

For `casal -r`, `-e`, or `-E`, the standard output dump can be processed using the extract CASAL output R package functions (Section 13).

Microsoft Windows users may find the `start` command helpful, particularly if they wish to run CASAL as low priority (see `start /?`), for example,

```
      start /LOW casal [arguments] > out 2> err
```

Note that CASAL includes a few lines of header information in the output. The header consists of the program name and version, the arguments passed to CASAL from the command line, the date and time that the program was called (derived from the system time), the user name, and the machine name (including the operating system and the process identification number).

3.1 Command line arguments

The call to CASAL takes the following form.:

```
casal [-h] [-l] [-r] [-e] [-E] [-p] [-m] [-M] [-a number]
       [-C filelist -S outfile] [-s number prefix]
       [-v outfile] [-P outfile] [-Y] [-f prefix]
       [-F suffix] [-q] [-Q] [-i infile] [-I infile]
       [-O outfile] [-o outfile] [-g RNG_seed] [-n name]
```

The call should include exactly one of the following "task" arguments.

`-h`	Display *help*.
`-l`	Display the CASAL end user *licence*.
`-r`	*Run* the population section once only and calculate the objective function (see Section 6). Print out the free parameters, the objective function and its components, the fits and residuals if requested (Section 6.8), and the output quantities (Section 7.2).
`-e`	Calculate the point *estimate* of the parameters (Section 6.3). Print outputs as per `-r`.
`-E`	As per `-e` but using finite differences instead of automatic differentiation (Section 6.3).
`-p`	Calculate likelihood or posterior *profiles* (Section 6.4).
`-m`	Use *MCMC* (Section 6.5) to sample the posterior distribution of the parameters. See Section 3.2 for MCMC procedure.
`-M`	Use *MCMC* (Section 6.5) to sample the posterior distribution of the parameters, but using finite differences to find the MPD and the proposal matrix. See Section 3.2 for MCMC procedure.
`-a [`*`number`*`]`	Recover the specified MCMC run from its results files; continue the run and *append* further results to the results files. See Section 3.2 for MCMC procedure (see also `-n`).
`-C [`*`filelist`*`]`	*Concatenate* the MCMC results files for the specified files into a single set of samples from the posterior. Reduce the sample size by random or systematic sub-sampling if requested. Optionally apply prior re-weighting (Section 6.5). The `filelist` argument should be a list of the (full) names of samples files, separated by white space. Use `-S` to specify the file into which to dump the results. See Section 3.2 for the MCMC procedure.
`-s [`*`number prefix`*`]`	Generate *simulated* observations, i.e., use CASAL as a simulator (Section 6.9). You must use `-i` to provide the name of a file containing free parameters, either one set (e.g., a point estimate) or multiple sets (e.g., a posterior sample). For each set of parameters supplied, `number` simulations are carried out. The results are dumped to files whose names are generated by combining the filename `prefix` specified, the number of the parameter set, and the number of the

	individual simulation (e.g., if `prefix=my_simulate`, then the third set of simulated observations for the second set of true parameters will be dumped into a file `my_simulate.par2.sim3`). If `number=1`, then the `.sim[`*`n`*`]` part of the filename is omitted.
`-v [`*`outfile`*`]`	Output the *values* of the output quantities (Section 7.2). The argument `-i` must be used to provide the name of a file containing a posterior sample. Results are dumped into `outfile`. Use this option to analyse the results of a MCMC run.
`-P [`*`outfile`*`]`	Calculate *projected* outputs (Section 7.3). The argument `-i` must be used to provide the name of a file containing free parameters, either one set (i.e., a point estimate) or multiple sets (i.e., a posterior sample). Results are dumped into `outfile`.
`-Y`	Calculate *yield* estimates (Sections 7.4, 7.5), e.g., MCY, CAY, deterministic MSY, CSP. The argument `-i` must be used to provide the name of a file containing free parameters, either one set (i.e., a point estimate) or multiple sets (i.e., a posterior sample).

In addition, any of the following arguments may also be used:

`-f [`*`prefix`*`]`	Use a prefix on the names of the three input parameter files.
`-F [`*`suffix`*`]`	Replace the standard `csl` suffix used on the input parameter filenames with a user defined suffix.
`-q`	Run *quietly*, i.e., suppress printing from within the population section.
`-Q`	Suppress all messages and warnings, i.e., all standard error output. Note that this option is *not* recommended for general use, but can be used to suppress verbose output if CASAL is used as a part of a Monte-Carlo simulation controlled by an external program.
`-i [`*`infile`*`]`	*Input* one or more sets of free parameter values from a text file. With `-r`, run the model with each. With `-e`, do a separate point estimate starting at each. With `-p`, use the first set as the initial minimum. (The actual minimum, not the starting point of the minimiser. See Section 6.4). With `-m` or `-M` use the first set as the starting point for the pre-MCMC point estimate, the second set (if there are two) as the starting point for the chain, and ignore the rest. With `-s`, produce simulated observations for each. With `-v`, calculate output quantities for each. With `-P`, calculate the free parameter values for each. With `-Y`, calculate the yield estimates. (See Section 3.3 for the free parameter file format and Section 3.2 for MCMC procedure.)
`-I [`*`infile`*`]`	with `-m -i`, input the covariance matrix used for MCMC from file. (See Section 3.2 for MCMC procedure.)
`-O [`*`outfile`*`]`	*Output* (no append) a set of free parameter values to a text file. With `-r`, output the free parameter values used for the run (either those specified with `-i`, or the base values in the parameter files if `-i` is not used) With `-e`, `-E`, or `-p`, output the estimated parameter values. The text file is in an appropriate format for use with `-i` (Section 3.3). If the file exists already, it is overwritten.
`-o [`*`outfile`*`]`	*Output* (with append) a set of free parameter values to a text file. Same as `-O` above, except that if the file already exists, it is appended to.

`-S [outfile]`	With `-C`, dump the posterior *sub-sample* into `outfile`.
`-g [RNG_seed]`	With `-m`, `-M`, `-s`, or `-Y`, seed the random number *generator* with this positive (long) integer value. If this is not specified, then the default is defined as a number based on the computer clock time.
`-n [name]`	Used with `-m`, `-M,` or `-a` when chains are being carried out on several computers. The argument is the *name* of the current machine, which is inserted into the names of the MCMC results files. As a result files from all the chains can be copied onto a single computer and, because they have different names, they will not overwrite each other.

3.2 Running a Bayesian analysis in CASAL

A full Bayesian analysis is more time consuming than the other CASAL tasks, and involves editing input files to achieve different tasks. The process of how to get CASAL to run a Bayesian analysis is described here. Section 6.5 describes the algorithms used by CASAL.

The first step in producing Monte Carlo Markov Chain (MCMC) results is to execute a `-m` or `-M` run. This produces a single Markov Chain. An initial point estimate is produced before the chain starts, which is derived from either automatic differentiation (`-m`) or finite differences (`-M`). This is done in order to calculate an approximate covariance matrix of the free parameters, but may also be used as the starting point of the chain. Users can specify the free parameter values used as the starting point of the point estimation (as the first row of the file invoked with `-i`), and, optionally, specify the free parameter values from which to start the chain (as the second row of the file invoked with `-i`). If you have specified that a free parameter is fixed in MCMC, you still need to supply a value for it when using `-i`. Once the MCMC commences (as opposed to the initial point estimate), the parameter will be fixed at the supplied value.

The MCMC run produces two results files. The first is `samples.[run number]`, or if the `-n` option is set, `samples.[name].[run number]`. It uses the free parameter file format described in Section 3.3, i.e., a header row followed by many rows of parameter values. The second file is `objectives.[run number]` or `objectives.[name].[run number]`. It contains the standard output header produced by CASAL, the covariance matrix used (if the covariance matrix is modified at one or more iterations of the chain, only the initial version of the matrix is shown), and a columnar table (with one row per posterior sample, giving the sample number, the posterior, prior, likelihood, and combined penalties (all on the negative log-scale), the current step size, the acceptance rate so far, and the number of times the covariance matrix has been modified so far.

The run number is the first available positive integer, i.e., if the directory already contains a file `samples.[name].1` but not `samples.[name].2` or `objectives.[name].2`, the next run number will be 2. We suggest that the *first* thing users do after an MCMC run is *back up the results files*. It may be distressing to inadvertently lose the only copy of a chain that had been running for some time.

If a chain gets interrupted for some reason, users can use `casal -a [run number]` to continue it rather than starting again from scratch (also include the argument `-n [name]` if this was used). The same parameter files must be used in the rerun as were used in the original run (CASAL does not check). Note that the results are not likely to be the same as you would have if the original chain had continued, because the random number sequence will be different. If the original run was interrupted or crashed, ensure that the last lines of the samples and objectives files are complete and that each has the same number of lines once the headers are removed (the printing process might have stopped partway through a line, in which case CASAL would be confused by the partly finished results).

Users may want to run multiple Markov Chains simultaneously if the hardware resources are available. If using a shared file system, users can run multiple chains in the same working directory. If the chains all use the same data files and you want to combine them later to produce a single posterior sample, use `-n` with each chain with a different ‘machine name’ argument to send the results to a different file. All the chains will then share the same run number. Give each chain a different random number seed using `-g` (or the results may be identical).

Optionally, users can provide a covariance matrix which will be used for the proposal distribution when doing MCMC runs, using `casal -m -I` or `casal -M -I` There are at least two reasons this may be preferred:

1. To save time. Currently CASAL has to complete a point estimate at the start of every MCMC run in order to get an approximation to the covariance matrix. This can be time-consuming. When doing multiple chains of the same model, users can choose to calculate the covariance matrix once, then use it for all chains for that model.
2. When an alternative method of generating a covariance matrix is required, which might lead to better MCMC performance (e.g., by taking the covariance of a sub-sample from an old chain).

When a covariance matrix is provided, CASAL subjects this to the usual transformations (e.g., reducing very strong correlations, increasing very small diagonal elements, zeroing rows and columns corresponding to ‘MCMC-fixed’ parameters, etc.). But this feature may easily be abused. If an inappropriate covariance matrix is submitted, CASAL may either not run or alternatively poor MCMC behaviour may result. Note that CASAL does not run any checks on the user supplied covariance matrix other than to ensure that it has the correct dimensions.

The supplied covariance file should consist of a one-line comment followed by a square array of numbers, without column headings or row labels. An easy way to create such a file is to copy and paste from an `objectives*` file from a previous MCMC run or from the standard output of an `-e` run with the `output.csl` file option `@print.covariance=True`. The starting point of the chain must be supplied with `-i`, which must contain only one line of data.

Use `casal -C` to combine a list of MCMC results files into a single posterior sample, decimate it down to a sub-sample of a specified, manageable size, and apply posterior re-weighting if requested. You need to provide the samples file names, which are supplied after the `-C` (the objective files with the same suffixes should also be present), and the name of the file into which the sub-sample is dumped (in the free parameter file format described in Section 3.3), which is supplied with `-S`. The burn-in period should also be set at this stage.

Use `casal -v` to calculate output quantities for a posterior sample, either the sub-sample generated by `casal -C` or the original single-chain sample generated by `casal -m` or `casal -M`.

An example of running a Bayesian analysis

A typical sequence for a ten-chain MCMC might be as follows. First, generate the chains, specifying random number seeds and machine numbers:

```
(machine 1): casal -m -g 144 -n PC1
(machine 2): casal -m -g 1812 -n PC2
…
(machine 10): casal -m -g 71 -n PC10
```

Files such as `samples.PC4.1` will be generated (assuming this is the first MCMC run in the directory). Back them up. For example, machine 2 suffers a power failure. To resume the run from where it stopped,

```
(machine 2): casal -a 1 -g 1812 -n PC2
```

CASAL finds the previous output files, `samples.PC1.1` and `objectives.PC1.1`, resumes the MCMC where they ended, and appends the results from the rest of the chain to these files.

Following this, copy all the output files onto one of the ten computers and run them through an external MCMC diagnostics package. Next pool the ten chains (and sub-sample to reduce the size of the result):

```
casal -C samples.PC1.1 … samples.PC10.1 -S subsample.dat
```

having first added the following to `estimation.csl`,

```
@MCMC
burn_in 100000
```

A sub-sample file is generated. Summarise the posterior,

```
casal -v quantities.dat -i subsample.dat
```

To explore the effect of using a different prior, change the `estimation.csl` file to specify the new prior and add the following to `estimation.csl`,

```
@MCMC
prior_reweighting 1
```

and then repeat the last commands:

```
casal -C samples.PC1.1 … samples.PC10.1 -S subsample.2.dat
casal -v quantities.2.dat -i subsample.2.dat
```

Finally, note that files in free parameter file format (including the posterior samples output by `casal –m`, `-M` and `-C`), and tables of output quantities (including the output of `casal -v` and `-P`) can be read into R using the functions described in Section 13.

3.3 Free parameter file format used by CASAL

In various situations it is useful to either write sets of free parameters to a file or read sets of free parameters from a file. For example:

- When doing MCMC, a long list of sets of parameter values is generated. They are saved to disk (so that they don't consume memory and so that they can be recovered if the program crashes partway through the chain) and can be re-loaded later.

- When a point estimate has been calculated with `-e`, the user may want to save the parameter values and reload them later. For example, when running the model with `-r` at the 'optimal' point.

- When a point estimate has been calculated by another stock assessment package, the user may want to run CASAL using the parameter values estimated by the other package.

The same *free parameter file format* is used in all cases. There is one header line, consisting of the name of each parameter (in `command[`*`label`*`].subcommand` format), followed by the length if it is a vector parameter, separated by single spaces. The header is followed by 1 or more sets of parameters, each written as a long vector on a single line.

A simple example of this is:

```
initialization.B0 size_at_age.k 1 recruitment.YCS 30
10000 0.1 0.87 0.95 1.12 ... (27 more YCS)
15000 0.2 0.93 0.98 1.14 ...
```

Note that the `1` argument for `size_at_age.k` is compulsory. It is not a scalar but a vector of length 1.

For input, the header must be accurate or the program will reject the file. A header check ensures the right parameters have been provided in the right order. However, there is no check on the number of parameters supplied in the rows of the table.

Note that CASAL generates a line of data as output, suitable for use in a file with `-i`, automatically when doing a run or an estimation (i.e., with `-r`, `-e`, or `-E`). To use this as an input for a subsequent run, copy the appropriate lines (i.e., the lines immediately after "`In a format suitable for -i :` ") into a text file, and rerun CASAL with the `-i [`*`file`*`]` option. CASAL can also generate a free parameter file using the `-o` or `-O` options (the first appends to a file if it already exists, the second replaces it) automatically from a `-r`, `-e` or `-E` run (in this case, if `-e` or `-E` is used with a multi-row `-i` input file, then multiple estimations will be done and multiple rows will be written to the `-o` or `-O` file).

Free parameter files can be read into R using the functions in described Section 13.

3.4 Constructing a CASAL data file

The model is specified to CASAL via the population, estimation, and output parameters. These are specified in the `population.csl`, `estimation.csl`, and `output.csl` input data files (although names can be modified using the `-f` and `-F` command-line options). All the parameters that can be used are listed in Sections 8, 9, and 10 respectively.

The parameter files use the *command-block format*. A parameter file consists of any number of command-blocks in any order. Each command-block consists either of a single command (starting with the symbol `@`) and its arguments, or a command (starting with `@`) and an optional label and one or more subcommands, i.e.:

```
@command arguments
```

or

```
@command [label]
[subcommand arguments]
[subcommand arguments]
[...]
```

Blank lines are ignored, as is extra white space between arguments. Comments beginning with '#' are ignored. To remove a group of commands or subcommands using '#', comment out the whole block, not just the first line. Alternatively, comment out an entire block by placing curly brackets around the text to be commented out. Put in a '{' as the first character on the line to start the comment block, then end it with '}'. All lines (including line breaks) between '{' and '}' inclusive are ignored. (Ideally this should be the first character on a line, but if not, the entire line will be treated as part of the comment block.)

Avoid inserting extra white space before a `@` character (which must also be the first character on a line). Make sure the file ends with a carriage return. Commands and subcommands must consist of letters and/or underscores, and must not contain a full-point ('.').

There is no need to mark the end of a command block. This is automatically recognised by either the end of the file or the start of the next command block, which is marked by the `@` on the first character of a line.

Also note that the commands, sub-commands, and arguments in the parameter files are case sensitive.

Some commands can never have subcommands (such as `@initial`). If a command has no subcommands, then it has to have arguments, which are placed on the same line as the command.

All other commands have no arguments, but have subcommands instead. Also,

- Some commands can be used multiple times in the same parameter file and must have a different label each time (such as `@abundance`).
- Some commands can be used only once and may never have a label (such as `@annual_cycle`).
- Some commands can be used one or more times. If used once they do not need a label, but if used more than once they do need labels (such as `@recruitment`).
- Some commands can be used one or more times and do not need labels: they are internally labelled 1 the first time they are used, then 2, 3 … (such as `@growth`).

The parameter listings say what kind of label or argument each command and subcommand takes. Arguments can be of the following types:

switch	true/false
integer	an integer
constant	a real number
estimable	a real number which can be estimated
constant vector	a vector of real numbers
estimable vector	a vector of real numbers which can be estimated
ogive	an ogive which can be estimated
string	a string

vector of strings a list of strings.

Parameters of type *constant vector*, *estimable vector*, or *vector of strings* contain one or more entries separated by white space (tabs or spaces).

Switches are parameters which are either true or false. Enter 'true' as `true`, `t`, or `1`, and 'false' as `false`, `f`, or `0`. Note that this is one of the few situations where CASAL is not case sensitive.

Ogive parameters (Section 5.6) are the most complex to enter. Users need to specify the type of the ogive, then the ogive parameters. For example, a logistic selectivity ogive with the label 'trawl' with parameter values a_{50}=5, a_{to95}=2 might be entered as:

```
@selectivity trawl
all logistic 5 2
```

where '`all`' specifies that this ogive applies to all fish, i.e., males and females, mature and immature. If you want a size-based ogive in an age-based model, you need to insert the word '`size_based`' between the subcommand and the ogive type. For example,

```
@selectivity trawl
all size_based knife_edge 30
```

See Section 5.6 for an explanation of how the size-based ogive is converted to an age-based ogive.

Not all parameters can be estimated — only those of type *estimable*, *estimable vector* or *ogive* can be estimated. You decide which of these CASAL should estimate, the *free parameters* (Section 6.2). Sometimes an ogive has some non-estimable parameters, for example, an `allvalues_bounded` ogive has two non-estimable parameters, the lower and upper bounds — the remaining parameters give the values between these bounds and can be estimated normally (as a single vector parameter).

When CASAL processes these files, it translates each command and each subcommand into a parameter. Each parameter has a name. For commands, the parameter name is simply the command name. For subcommands, the parameter name format is either,

1. `command[label].subcommand` if the command has a label, or
2. `command[i].subcommand` if the command is occurring for the *i*th time and is auto-numbered, or
3. `command.subcommand` if the command has no label and is not auto-numbered.

The user needs to convert commands to parameter names in this way in several situations. For example, if you have constant natural mortality,

```
@natural_mortality
all 0.3
```

and you want to estimate the mortality rate *M*, you need to tell CASAL that the parameter named `natural_mortality.all` is to be estimated, by inserting the commands,

```
@estimate
parameter natural_mortality.all
```

Similarly, if you have,

```
@selectivity trawl
male logistic 5 2
…
```

and you want to apply some kind of penalty to the logistic selectivity ogive, you will need to tell CASAL that the parameter named `selectivity[trawl].male` is to be penalised

CASAL allows you to supply a single line comment within the input parameter files, which is written to any output files generated by CASAL. Here, use the command `@comment` followed by the comment text. This can be useful in assisting identification of output files from a particular run or series of runs.

3.5 CASAL exit status values

When CASAL completes its tasks (or errors out), it returns an exit status value to the operating system. The exit status value can be useful when, for example, CASAL is being called by another program as a means of identifying the reason CASAL exited.

CASAL can return the following values:

0 CASAL completed its task successfully, and if an estimation run (i.e., `casal -e/-E`), then the minimiser reported successful convergence.
1 CASAL completed an estimation run (i.e., `casal -e/-E`), but the minimiser reported that it was unable to determine if it had converged.
2 CASAL completed an estimation run (i.e., `casal -e/-E`), but the minimiser reported that it failed to obtain convergence.
11 CASAL halted because of a “Fatal error”
12 CASAL halted because of a “Betadiff error”.

Note that CASAL can only return an exit status of 1 or 2 if `casal` was called with –e/-E command. In addition, in some circumstances the minimiser within CASAL can be called multiple times within a single run (for example if command call to CASAL was `casal -e -i pars.dat`, where pars.dat contains more than one line of free parameters). In such cases, the return value from CASAL is greatest number from any of the single estimation steps.

4. OVERVIEW OF THE CASAL MODEL

4.1 Model components

A fisheries model in CASAL consists of three parts.

1. The *population section* is the model of the fish population dynamics. It includes processes such as recruitment, migration, natural and fishing mortality.

2. The *estimation section* carries out the estimation of free parameters. The estimation will be based on an *objective function* (negative log likelihood or negative log posterior). The estimation section is used to find a *point estimate*, which is the set of parameter values that minimises the objective function. It may also be used to characterise the uncertainty in the point estimate, via either profiling or producing a Bayesian posterior.

3. The *output section* produces results for the user. The outputs may include parameter estimates, the objective function, fits and residuals, projections, and yield estimates.

4.2 Parameters

Parameters are quantities that describe how things work. There are three types:

1. *population*: both those related to population structure and biology, e.g., size at age, weight at size, maturation, stock-recruitment relationship, natural mortality, migration parameters, and those concerning the fishery, e.g., catches, tagging events, selectivity ogives, maximum exploitation rates.

2. *estimation*: needed for the estimation procedure, e.g., choice of estimation method, observations and their error structures or weights, which parameters are to be estimated, priors, starting values, and minimiser control values.

3. *output*: indicating which outputs the program should produce, e.g., what should be printed as the model runs, which quantities should be written to file, etc.

Some parameters may function as switches, allowing the user to choose between available options (e.g., between Ricker or Beverton & Holt stock-recruitment relationships, or between normal or lognormal distributions).

Each time a model is run the population and estimation parameters will fall into two classes; those which are assumed *known*, and those that are *free* (i.e., to be estimated). It is up to the user to specify which parameters are free. Not all parameters are *estimable*. Some, such as switches, are never estimated. (Note that while CASAL may allow a parameter to be estimated, this does not mean that the modeller should necessarily allow it to be estimated.)

4.3 Observations

Observations are data which allow inferences to be made about the parameters. Examples include CPUE indices, survey biomass estimates, catch at age, and commercial catch length frequencies The process of estimation in CASAL involves finding values for each of the free parameters so that each observation is as close as possible to a corresponding expected value. Note that catches are treated as population parameters, not observations.

5. THE POPULATION SECTION

5.1 Overview

The basic structure of a CASAL population model is defined in terms of an *annual cycle*, *time steps*, *states,* and *transitions*.

The *annual cycle* defines what processes happen in each model year, and in what sequence. (In line with the New Zealand fisheries management framework, CASAL runs on an annual cycle rather than, for example, a 6-monthly cycle.)

Each year is split into one or more *time steps*, with at least one process occurring in each time step. Each time step can be thought of as representing a particular part of the calendar year, or you can just treat them as an abstract sequence of events.

The *state* is the current status of the population, at any given time. The state can change one or more times in every time step of every year. The state object must contain sufficient information to figure out the future course of the fishery (given a model and a complete set of parameters).

There are a number of possible changes in the state, which are called *transitions*. These include processes such as *recruitment*, *natural mortality*, *fishing mortality*, *disease mortality*, *ageing*, *migration*, *tagging events*, and *semelparous mortality*.

The division of the year into an arbitrary number of time steps allows the user to specify the exact order in which processes and observations occur. The user needs to specify the time step in which each process occurs. If more than one process occurs in the same time step, there is a default order in which they occur (see Section 5.3). If you don’t want things to happen in this default order, then split the processes into different time steps.

The key element of the state is the *partition*. This is a broadly applicable concept that can be used to describe many different kinds of fish model. The partition is simply a breakdown of the total number of fish in the current population into different kinds of fish. (Note that the partition records numbers of fish, not biomass.) The fish are categorised by various *characters*. The permissible characters are: size class or age class, sex, maturity, area, stock, tag, and growth-path. The user chooses:

- whether the partition is subdivided by size class or age class (not both)
- which of the other characters are included in the partition, e.g., the number of areas, stocks, tagging events, or growth paths (if any of these characters are included in the partition).

The resulting partition can be conceptualised as a matrix, where the columns are size or age classes and the rows represent combinations of the other characters. Then the number in each cell of the matrix is the number of fish with the corresponding combination of characters.

For an example of these ideas, consider a model of a single stock with a spawning and non-spawning fishery. The non-spawning fishery happens over most of the year (say 10 months) in the home area. The mature fish then migrate to the spawning area, where the spawning fishery operates. At the end of spawning, these fish, along with the recruits from the previous year, migrate back to the home area. The modeller decides that fish will be divided in the partition by age, sex, maturity, and area (spawning and home grounds). So the partition has 8 rows (2 sexes × (mature or immature) × 2 areas) and one column per age class.

The modeller decides to use the annual cycle in Table 1.

Table 1: The annual cycle of a simple model.

Time	Time step	Area Non-spawning	 Spawning	Activity
Jan–Oct	1	Mature and immature fish	Empty	Fishing in the home area.
End of Oct	2	Immature fish	Mature fish	Mature fish migrate to the spawning area.
Nov–Dec	3	Immature fish	Mature fish	Fishery in the spawning area.
End of Dec	4	Mature and immature fish	Empty	Recruits from the previous year appear in the spawning area. Along with the mature fish, they migrate to the home area.

So they define four time steps, labelled 1 through 4. Step 1 includes the non-spawning fishery. Step 2 includes the migration to the spawning area. Step 3 includes the spawning fishery. Step 4 includes recruitment and the migration back to the home area. (In fact, they could have used only 3 time steps, by using a single step in place of their steps 2 and 3. Because the default order of processes within a time step places migrations before fisheries, the processes would still have occurred in the right order.) There are other details to be sorted out, such as the proportion of natural mortality occurring in each time step, but this gives the basic idea.

This structure can be used to implement complex models, with intermingling of separate stocks, with complex migration patterns over multiple areas, and multiple fisheries using different fishing methods and covering different areas and times. Note that there is little point in using a complex structure to model a stock when there are no observations to support that structure. In other words, use a structure for your model that is compatible with the data available.

The model is run from an *initial* year up to the *current* year. It can also be run past the current year to make *projections* — things that happen in the future — up to the *final* year. Alternatively, for yield calculations, it is run over an abstract *simulation period*.

5.2 The state object and the partition

The key component of the state object is the partition, a matrix of *numbers of fish* by combinations of characters. The columns can either be age or size classes, the rows are combinations of the following characters:

- Sex (male or female).
- Area (any number of areas, named by the user).
- Stock (any number of stocks, named by the user).
- Maturity (immature or mature).
- Growth-path (any number of growth-paths).
- Tag. (any number of tagging events, but note that CASAL will always create a "no_tag" member of the partition in addition to those that you specify).

A stock is defined as a subpopulation of fish which recruits separately. See Section 5.11 for the treatment of maturity when it is not a character in the partition.

Growth-paths are a feature used to implement some persistence of size at age in an age-based model that uses some length/size data. Each growth-path has its own growth curve, and the size-based model features will consequently have different effects on different growth-paths. So, you need to tell CASAL the following:

- Whether the model is age- or size-based.
- The number and nature of size classes in a size-based model.
- The minimum and maximum age classes in an age-based model.
- Whether there is a plus group.
- Whether the partition is divided by sex.
- Whether the partition is divided by maturity.
- Whether the partition has growth-paths, and, if so, how many.
- Whether the partition has multiple stocks, and, if so, how many, and their names.
- Whether the partition has multiple areas, and, if so, how many, and their names.
- Whether the partition includes tagged fish, and, if so, how many, and the names of the tag partitions.

Age classes are always 1 year wide, except that the maximum age group can optionally be a plus group. Users need to choose the minimum and maximum age classes. Size classes are defined by the user, and you need to specify how many size classes there are, the lower bound of each size class, and whether the last size class is a plus group, or if not, what its upper bound is. The relevant parameters are `class_mins` and `plus_group`. The `class_mins` parameter contains the lower bound of each class, and concludes with the upper bound of the last class if it is not a plus group. If, for example, size classes of 30–40, 40–50, 50–60, and 60–70+ cm were desired, you would set `class_mins 30 40 50 60` and `plus_group true`. Whereas if 30–40, 40–50, 50–60, and 60–70 cm were desired, you would set `class_mins 30 40 50 60 70` and `plus_group false`.

The user can specify that some combinations of characters are not possible. For example, immature fish might never occur in the area you have labelled `spawn_ground`. To do this, you use the `exclusions` parameters. In this case, you would set,

```
exclusions_char1 maturity
exclusions_val1 immature
exclusions_char2 area
exclusions_val2 spawn_ground
```

It's a good idea to use the `exclusions` parameter wherever appropriate because it reduces the size of the partition (so, with the above example, there will be no rows in the partition corresponding to immature fish in area `spawn_ground`) and can save memory and calculation time.

The other component of the state object in CASAL is a vector of spawning stock biomasses (SSBs, mid-spawning season biomasses of spawning fish) for each stock. CASAL needs to include this in the state object to calculate future recruitments, if there is a stock-recruitment relationship.

5.3 The time sequence

The time sequence of the population model includes the years over which it is to run and the annual cycle for each year. The model runs from the start of year `initial` and runs to the end of year `current`. Projections extend up to the end of year `final`. The annual cycle can contain the following transition processes:

- Ageing (in an age-based model).
- Recruitment.
- Maturation (if maturity is a character in the partition).
- Migration (if the model includes more than one area).
- Growth (in a size-based model).
- Natural and fishing mortality.
- Disease mortality.
- Tag release events.
- Tag shedding rate.
- Semelparous mortality.

If two or more processes are specified for the same time step then they will happen in the above order. This ordering is imposed only to simplify the specification of the annual cycle. It does not restrict the user because it applies only to processes within the same time step. If, for example, it is desired that maturation occur before recruitment then this can be done by putting these processes in separate time steps.

The basic unit of fishing mortality is a *fishery*, defined as fishing mortality in a single area and time step. You may need to split a single administrative fishery into multiple CASAL fisheries, in which case you will need to partition the catch. (However, this should often be avoidable. If you have an observation partway through a fishery, you can specify that a certain proportion of the mortality occurs before the observation, without needing to split the time step into two.)

If there is more than one stock, recruitment is handled separately for each stock, but all stocks must recruit in the same time step. There can be more than one maturation episode per year, each of which can apply to only one stock, or all stocks equally. Similarly there can be more than one growth episode per year, each of which can apply to only one stock, or all stocks equally. The user can define any number of migrations in a given year.

To specify the time sequence, you need to tell CASAL the following:

- The initial, current, and final years.
- The number of time steps in each year.
- The time step in which recruitment occurs, and the area to which each stock recruits
- How SSB is calculated[1].
- In an age-based model, the time step at which ages are incremented.
- If there are any migrations, the time step at which each migration occurs and the source and destination areas. Note that if there are multiple migrations in a time step and an area is the source of more than one migration, then the migrations will happen in the order that they are defined in the `population.csl` file.

---

[1] The SSB (spawning stock biomass) is a common model output and is also the measure of abundance used in stock-recruitment relationships in CASAL (where applicable). Different models define SSB in quite different ways so we allow several options in CASAL as to how SSB is calculated. By default, SSB is calculated for each stock as the mature biomass (of both sexes), in an area (or areas) of your choice, halfway through the natural and fishing mortality in a time step of your choice. It can alternatively be calculated after some other specified proportion of the mortality (see Section 5.4.6) and/or for one sex only. A ‘proportion spawning’ multiplier can be applied to the mature biomass to get the SSB (in multi-area models this would typically not be done, instead the appropriate proportion of fish would be migrated to the spawning area). If maturity is not in the partition, then the modeller may nevertheless know that all fish in the spawning area should be mature (i.e., because only mature fish are meant to migrate) but the model does not ‘know’ this because maturity is not persistent. In this case the user can specify that the SSB is the total biomass in the area, rather than using the mature biomass.

- If maturity is a partition character, the number of maturation episodes per year, and the time step at which each maturation episode occurs.
- In a size-based model, the number of growth episodes per year, and the time step at which each growth episode occurs.
- In an age-based model, the proportion of the year's growth which has occurred by the start of each time step[1].
- The proportion of the year's natural mortality occurring in each time step.
- The time step and area in which each fishery occurs.
- Whether fishing mortality is instantaneous or uses the Baranov equation[2].
- If there is a disease mortality event, and in which time step this occurs.
- If tagging has been specified, when the tagging event occurs, how many fish by age or size class, in which member of the partition to put the tagged fish, and the tag shedding rates, if defined.

You then need to provide CASAL with details about how each process works. These processes are described individually in Section 5.4.

When defining the annual cycle, there are a number of errors can be made. Some of the less obvious ones are listed here. It is an error if:

- the sum of the proportions of the year's natural mortality over time steps is not 1,
- in an age-based model, any element of `growth_props` is outside [0,1]; or if `growth_props` is not 0 in the time step in which fish age; or if `growth_props` diminishes between consecutive time steps without age incrementation having taken place,
- in a size-based model, more than one growth episode occurs in the same time step, unless they involve different stocks,
- you want to use the Baranov equation and there is a time step that includes two or more fisheries in the same area.

5.4 Transitions between states

This section describes the various transition processes in CASAL. The transition processes, in their default order, are:

1. Ageing i.e., age incrementation (in an age-based model).
2. Recruitment.

---

[1] Fish growth in an age-based model is handled quite differently from a size-based model. The simplest option is to assume that the mean size of a fish is based on its age, rounded down to the next lowest whole number of years. So, for example, 2-year old fish have the same mean size whether they have just passed their 2nd birthday or whether they are about to turn 3. An alternative is to allow some fish growth between birthdays. You can do this using the `growth_props` parameter. This is a vector with one entry per time step. The mean size of fish of age a years (rounded down) in the ith time step is calculated as if their age was (a+`growth_props[i]`). So, if the first entry of `growth_props` is 0.5, then, in time step 1, the mean size of 2-year-old fish is calculated as if they were age 2.5. The default is `growth_props = 0` (i.e., no growth between birthdays).

[2] Natural mortality and fishing mortality occurring in the same area and time step can be sequenced in two different ways. The first option is to apply half the natural mortality, then to apply the mortalities from all the fisheries instantaneously, then to apply the remaining half of the natural mortality. The second options is to use the Baranov catch equation, which implies that natural and fishing mortalities are simultaneous. We prefer the first option — the calculations are more straightforward and the result typically about the same. However you can use Baranov if you want, except that we have not yet implemented the Baranov equation for multiple fisheries in the same area in the same time step. Whichever option you use is applied to all fisheries. More on this in Section 5.4.6.

3. Maturation (if maturity is a character in the partition).
4. Migration (in a multi-area model).
5. Growth (in a size-based model).
6. Mortality (natural and fishing).
7. Disease mortality.
8. Tag release events.
9. Tag shedding rate.
10. Semelparous mortality.

5.4.1 Ageing (in an age-based model)

The ageing process is straightforward. Every fish increases in age by one year, except those already in the plus age group (if it exists), which are unaffected. Note that if there is no plus group in the partition, then all fish older than the maximum age are "dropped off" the end of the partition, i.e., die. Further, note that the CASAL annual cycle ageing subcommand is spelt `annual_cycle.ageing_time`.

5.4.2 Recruitment

In the recruitment process, a number of fish are added to the partition. In an age-based model, all recruiting fish are of the minimum age. In a size-based model, you need to tell CASAL the mean and c.v. of the size distribution of recruiting fish, which is assumed to be a normal distribution (and can depend on sex and stock). Note that fish below the minimum of the range that defines the first size class appear in that size class; and similarly fish above the maximum of the range that defines the last size class appear in that size class.

For each stock, the number of fish added in year y is

$$R_y = R_0 \times YCS_{y-y\_enter} \times SR\left(SSB_{y-y\_enter}\right) \times CR\left(T_{y-y\_enter}\right) \qquad \text{(R1)}$$

where R_0 is the stock's average recruitment (ignoring the stock-recruitment and climate-recruitment functions); YCS are year class strength multipliers (also known as recruitment multipliers); y_{enter} is the number of years after it is spawned that a year class enters the partition; SR is the stock-recruitment function ($SR \equiv 1$ if there is no stock-recruitment relationship); CR is the climate-recruitment function (T is a single exogenous variable such as sea surface temperature, $CR(T) \equiv 1$ if no climate-recruitment relationship). It can happen that y–y_{enter} < `initial`, so that $SSB_{y\text{-}y\_enter}$ is not defined. In this case CASAL sets $SSB_{y\text{-}y\_enter}$ to B_0 (or to $B_{initial}$, if this is defined to be different from B_0 – see Section 5.5).

R_0 is an important parameter because it defines how large the stock would be, on average, if there were no fishing. From R_0, CASAL can calculate B_0, which is defined to be the SSB that would exist if recruitment was equal to R_0 every year and there was no fishing (alternatively, CASAL can calculate R_0 from B_0, if the latter is specified). B_0 has several special roles in CASAL: in the stock-recruitment function (where, by definition, $SR(B_0)$=1); and as a reference biomass in stock projections (see Section 7.3.2) and yield calculations (Sections 7.4.2 and 7.5.1).

YCSs starting from year (`initial`-y_{enter}) and extending up to year (`current`-y_{enter}) must be provided.

In an age-based model the value of y_{enter} depends on the order of recruitment, ageing, and spawning processes within a year:

- If recruitment then ageing then spawning, then y_{enter} should equal `min_age+1`.
- If spawning then ageing then recruitment, then y_{enter} should equal `min_age-1`.
- If any other order is used, then y_{enter} should equal `min_age`.

CASAL will output a warning if the value of y_{enter} supplied does not obey the above rule, but will continue running.

One quantity the user can choose to output is the `true_YCS` (see Section 7.2), which differ from the *YCS* by including contributions from *SR* and *CR*. That is, they are defined as true_$YCS_y = R_{y+y_enter}/R_0$.

Stock-recruitment functions

The stock-recruitment functions available are Beverton-Holt and Ricker (the alternative is no stock-recruitment relationship, $SR \equiv 1$). These are parameterised by the parameter *steepness*, defined as $h=SR(0.2B_0)$. The functional forms for these relationships are:

Beverton-Holt: $$SR(SSB) = \frac{SSB}{B_0} \Big/ \left(1 - \frac{5h-1}{4h}\left(1 - \frac{SSB}{B_0}\right)\right)$$

Ricker: $$SR(SSB) = \frac{SSB}{B_0}\left(\left(\frac{1}{5h}\right)^{\frac{5}{4}\left(\frac{SSB}{B_0}-1\right)}\right)$$

Climate-recruitment functions

The basic climate-recruitment relationships available are *exponential*, *arctan*, and *logistic* (the alternative is no relationship). All three are functions of a single exogenous variable *T*, which should be provided for years (`initial`-y_{enter}) to (`current`-y_{enter}) at least, and can also extend further into the future (for use in projections). Two additional climate-recruitment functions have been added for situations where the 'climate variable' *T* is actually a prediction of year class strength, typically from a climate-recruitment regression analysis (e.g., Bull & Livingston 2001). The *identity* climate-recruitment relationship allows the predictions to be used in an unmodified form. The *linear-combination* climate-recruitment relationship allows the model to decide how much credence to give to the predictions, when you estimate the parameter *p* (which must be between 0 and 1, otherwise you can potentially get negative recruitments).

exponential: $CR(T) = \alpha\exp(\beta T)$

arctan: $CR(T) = \alpha\left(0.5 + \tan^{-1}(\beta T)/\pi\right)$

logistic: $CR(T) = \alpha \big/ \left(1 + \beta\exp(\beta_2 T)\right)$

identity: $CR(T) = T$

linear-combination: $CR(T) = pT + (1-p)$

Note that this formulation allows various levels of relationship between recruitment and climate. At one extreme, when $CR(T) \equiv 1$, there is no relationship. At the other extreme, when the *YCS* are constant and $SR \equiv 1$, recruitment is completely determined by climate (apart from the factor R_0). In between these extremes, climate affects recruitment but does not determine it.

Warning: Since the climate-recruitment relationship was coded into CASAL, it has become apparent that some aspects do not work as intended. The climate-recruitment option has been marked obsolete and will not be usable until it is repaired in a future version.

Modifying recruitment in the initial years

If users choose to initialise the population using a recruitment, $R_{initial}$, that is different from R_0 (see Section 5.5), $R_{initial}$ can also be chosen in calculating the recruitment for the first $n_{rinitial}$ years of the model. This option is used to avoid estimating year class strengths about which there is little information. To calculate the first $n_{rinitial}$ recruitments, CASAL simply defines $YCS_y = R_{initial}/R_0$ (or $R_{initial}$, if $R_{initial}$ is defined as a deviate – see Section 5.5) for years `initial-y_enter` $\leq y \leq$ `initial-y_enter+n_rinitial–1` before using Equation (R1). With this option, the parameter vector `recruitment.YCS` is shorter than the output quantities `YCS` and `true_YCS`, because the latter include the first $n_{rinitial}$ YCS but the former does not.

Ensuring R_0 makes sense

As noted above, R_0 is intended to represent the recruitment that would typically occur if there was no fishing. CASAL provides three ways for the user to ensure that their estimate of R_0 has this meaning. The first, and simplest, is to provide a penalty function (see Section 6.7.6) to force the YCSs to average 1. This ensures that the average recruitment for the years in which YCSs are estimated is close to R_0 (ignoring the stock- and climate-recruitment relationships) Unfortunately, this penalty function may need to be large (i.e., have a large weight), which can lead to poor MCMC performance in the calculation of a Bayesian posterior. Because of that, CASAL supports two alternative parameterisations of YCSs which are intended to ensure that R_0 makes sense.

The first alternative (the Haist parameterisation) was suggested by V. Haist. Here, the model parameter `YCS` is a vector Y, covering years from `initial-y_enter+n_rinitial` to `current-y_enter`. The year class strengths are calculated by $YCS_i = Y_i/\text{mean}(Y_i)$ where the mean is calculated over the user-specified years `first_free` to `last_free`. Then,

$$YCS_y = \begin{cases} Y_y \big/ \text{mean}_{y' \in S}\left(Y_{y'}\right) & \text{for } y \in S \\ Y_y & \text{for } y \notin S \end{cases}$$

where S is the set of years from `first_free` to `last_free`. One effect of this parameterisation is that R_0 is now defined to be the mean estimated recruitment over the years `first_free` to `last_free` (because the mean YCS over these years will always be 1). Typically, the user will define `first_free` and `last_free` to span the years for which they expect to have reasonable estimates of YCSs. Often, the user will wish to force Y_y=1 for some or all years $y \notin S$ (this is equivalent to forcing R_y=R_0 x SR x CR) by setting the lower and upper bounds of these Ys to be 1. An exception to this might occur for the most recent YCSs, which the user may want to estimate, but not include in the definition of R_0 (because the estimates are based on too few data). Note that, optionally, the user may exclude one or more years from within the range from the averaging process of the Haist parameterisation.

The advantage of the Haist parameterisation is that the user need no longer use a large penalty to force the mean of the *YCS* parameter to be 1 (though they should still use a small penalty to stop the mean of *Y* from drifting). This may improve MCMC performance. Simulated and projected *YCS* are not affected by this feature, nor are those *YCS* that are set to $R_{initial}$. A disadvantage with this parameterisation in a Bayesian analysis is that the prior refers to the *Y*'s, *not* the *YCS*.

The second alternative is the Francis parameterisation of *YCS*. This uses two distinct concepts of mean recruitment: R_{mean} is the theoretical mean recruitment over all years (past and future), and, as in the Haist parameterisation, R_0 is the mean over the user-specified years `first_free` to `last_free`. There are two corresponding biomasses: B_{mean} is the biomass that would exist if recruitment was always equal to R_{mean} and there was no fishing, and B_0 is the analogous SSB with constant recruitment R_0. With this parameterisation, R_{mean} is used in place of R_0 in the calculation of R_y, so equation (R1) is replaced by:

$$R_y = R_{mean} \times YCS_{y-y\_enter} \times SR\left(SSB_{y-y\_enter}\right) \times CR\left(T_{y-y\_enter}\right) \qquad \text{(R2)}$$

With this parameterisation, R_0 (and thus B_0) become derived parameters, which are calculated from the user-specified R_{mean} (or B_{mean}) using the equation $R_0 = R_{mean}\bar{Y}$. See Figure 1 for an illustration of this parameterisation. Note that the only use of R_{mean} in CASAL is to calculate R_0 and R_y. Also, the "special roles" of B_0 or the definition of `true_YCS` (see above) are unchanged by the Francis parameterisation.

With this parameterisation, the user may also wish to force the recruitment to be based on R_0 for some years at the beginning and end of the period `initial+n_rinitial` to `current`. This can be achieved by providing the associated YCSs for only a subset of this period (and not, as in the Haist parameterisation, by setting the lower and upper bounds to be 1). CASAL will then define $YCS_y = \bar{Y}$ for the additional years `initial-y_enter+n_rinitial` $\leq y <$ `YCS_years`, and `YCS_years` $< y \leq$ `current-y_enter`, before applying equation (R2). Note that these additional years are included in the output quantities `YCS` and `true_YCS`, but not in the parameter vector `recruitment.YCS`. Figure 1 illustrates the full range of options that can be achieved with this parameterisation.

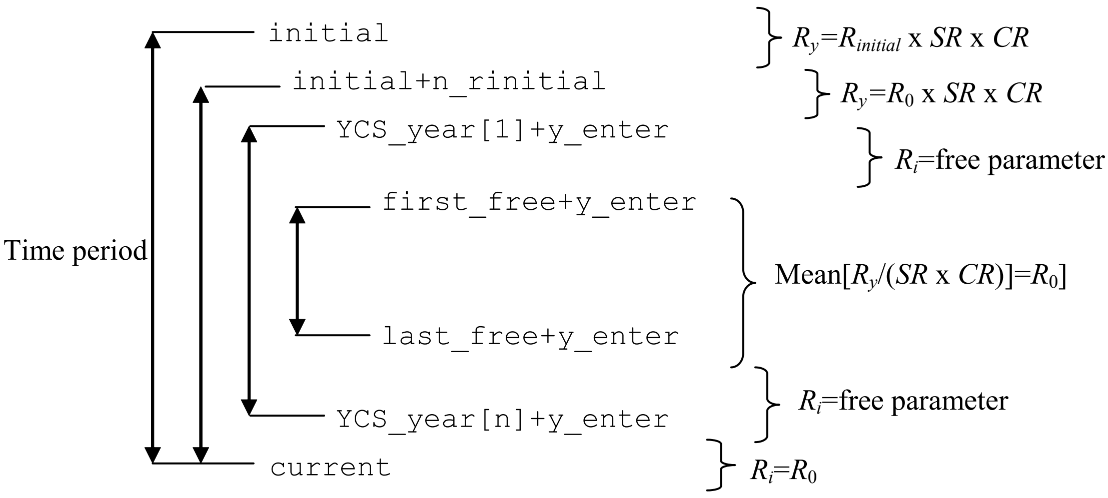

Figure 1: How CASAL determines recruitments (R_y) for the Francis parameterisation.

Two advantages of the Francis parameterisation are that there is no need for a penalty function to constrain the YCSs, and the prior distributions specified for parameter `YCS` do apply to the YCSs (not true for the Haist parameterisation). A disadvantage is the need for the additional parameter R_{mean}. With likelihood estimation, this parameter is not well determined, because if we double R_{mean} and halve all the YCSs we do not affect either the biomass trajectory or the fit to any observations. This will not be a problem with Bayesian estimation unless the priors on R_{mean} and the YCSs are both uniform (not recommended). Because the estimated value of R_{mean} depends on these priors it seems best to treat this parameter, and the associated B_{mean}, as nuisance parameters with little biological meaning.

Specifying recruitment

So, to specify the recruitment for each stock, the following information is required:

1. *YCS*, starting from year (`initial`-y_{enter}+$n_{rinitial}$) and extending up to year (`current`-y_{enter}). With the Francis parameterisation *YCS* may be provided for a consecutive subset of these years (and probably should if you want these set equal to R_0).
2. The value of y_{enter}.
3. The stock-recruitment function (if any) and the `steepness` parameter.
4. The climate-recruitment function (if any) and the values of the climate-recruitment parameters.
5. In a sexed model, the proportion of recruits which are male.
6. In a size-based model, the mean and c.v. of the size distribution of recruiting fish (which can depend on fish sex).
7. In a growth-path model, the proportion of recruiting fish on each growth-path.
8. If $R_{initial}$ is to be used as the recruitment for the first $n_{rinitial}$ years of the model, the value of $n_{rinitial}$.
9. If you want to use the Haist (`@standardise_YCS=True`) or Francis (`@use_mean_YCS=True`) parameterisations of year class strengths are used, they need to be specified, along with the range of free *YCS*.

5.4.3 Maturation

Maturation is the process in which immature fish become mature and are moved accordingly in the partition. See Section 5.11 for how to treat maturity when it is not a character in the partition.

You can specify a single maturation episode in each year, or you can have multiple maturations. Each episode can apply to one stock, or all stocks equally, and can be applied in one area, or all areas equally. Maturation rates are expressed as an ogive (and note that this ogive contains the rates of maturation, not the proportions of mature fish).

If you try to mature fish in an area where fish are constrained to be immature, CASAL will issue a warning, and will not mature those fish.

So, to specify each maturation episode, the following information is required:

- If it applies to only one stock, which is it?
- If it applies to only one area, which is it?
- The maturation rates, as an ogive, optionally by sex.

5.4.4 Migration

Migration is the process of moving fish from one area to another. It only occurs in multi-area models. You can specify any number of migrations occurring in each year. If two or more migrations are specified in the same time step then they take place in the order in which they are given.

A migration can involve only one stock in an area, or all stocks. You can migrate immature fish only, or mature fish only, or both. You can state that a given proportion of these fish migrate (constant across all age or size classes), or provide an ogive of proportions migrating by age or size class.

You cannot migrate fish to an area where their combination of characters is not allowed (CASAL errors out). So, for example, if moving fish to an area where only mature fish are allowed, you need to specify that only mature fish migrate.

CASAL currently supports two-wave migrations. These migrations consist of two waves in different time steps. If p_i is the specified proportion of fish migrating from the ith partition element, proportion (`pwave` p_i) will migrate in wave 1 and proportion (1-`pwave`)×p_i/(1−(`pwave` × p_i)) will migrate in wave 2. Specify these as two separate migrations, give `pwave` for each, and specify that the first is a '1st wave' and that the second is a '2nd wave'. (No checking is currently carried out that there are two matching waves with the same parameters. Remember that to specify `pwave` for each, not `pwave` for the first and `(1-pwave)` for the second. If you want to estimate `pwave`, you need to set the `estimate.same` parameter to make sure that `pwave` takes the same value for both waves.)

CASAL also supports annual variation in migrations and density-dependent migrations. The annual variation allows the migration rate to be modified in a particular year by some factor F. For density dependent migrations, the rate depends on the fish abundance in the source area, the destination area, or both — so, you can encourage fish to move into an under populated area and/or out of an overpopulated area.

Both annual variation and density dependent migration rates are calculated via an odds ratio, and a single factor (F) is applied to all fish in a given migration in a given year, regardless of age, sex, etc. Now let $P^i_{a,b}(y)$ be the proportion of fish in element i of the partition which migrate from area a to area b in year y, prior to the application of an annual variation or density dependence. (These values depend on the migration rate, or ogive of migration rates, etc.) And let the corresponding odds be

$$O^i_{a,b}(y)=\frac{P^i_{a,b}(y)}{1-P^i_{a,b}(y)}.$$

Then the effect of the annual variation or density dependence is to change the odds to

$$\vartheta^i_{a,b}(y)=O^i_{a,b}(y)\times F_{a,b}(y)$$

and hence the proportion of fish migrating to

$$\Pi^i_{a,b}(y) = \frac{\vartheta^i_{a,b}(y)}{1+\vartheta^i_{a,b}(y)}.$$

For annually-varying migrations, the factor F is just $\exp(m_y)$, where m_y is the annual variation value for year y. For density dependent migrations, F is calculated as follows. In each year y, for each density dependent migration from area a to area b

$$F_{a,b}(y) = \exp\left(-S\left(\frac{A_{a,y}-A_{a,0}}{A_{a,0}}\right) - D\left(\frac{A_{b,y}-A_{b,0}}{A_{b,0}}\right)\right)$$

where S is a number expressing the dependence on the abundance in the source area (negative values mean that fish are encouraged to leave an overpopulated area. Set S=0 for no dependence); D is a number expressing the dependence on the abundance in the destination area (positive values mean that fish are encouraged to move to an under populated area — set D=0 for no dependence); $A_{j,y}$ is the total abundance of all fish in area j, year y (with y=0 meaning the unfished equilibrium level) just before the migration occurs.

Neither annual variations nor density dependence are applied during the calculation of the initial state. In the calculation of the initial state for a model with annually-varying migration, the above factor F is set to $\exp(\bar{m})$, where $\bar{m}$ is the mean of the annual variation values, m_y, over a user-specified range of years.

The specification of the annual cycle includes the time step, source area, and destination area of each migration. You also need to tell CASAL the following:

- If there are multiple stocks and only one stock migrates, which is it?
- Do only mature fish migrate, or immature fish, or both?
- If a proportion of these fish migrate (constant across age or size classes), what is it? Or, if fish migrate according to an ogive across age or size classes, what is it?
- Is density dependence applied? If so, what are the values of the density dependence parameters S and D?
- Is an annual variation applied? If so, what are the years (`annual_variation_years`) and values (`annual_variation_values`) of the annual variation, and what range of years should be used in calculating $\bar{m}$?

Two-wave migrations require more details — see earlier.

5.4.5 Growth (in a size-based model)

In a size-based model, growth is the process by which fish move between size classes in the partition. See Section 5.8 for the treatment of fish growth in an age-based model. You can specify a single growth episode in each year, or you can have multiple growths. Each episode can apply to one stock, or all stocks equally, and applies to all areas equally.

There are many possible fish growth increment models, but CASAL implements two. (a) the Francis (Francis 1988) parameterisation of the growth increment von-Bertalanffy curve (see Figure 2), and (b) an alternative form that has an exponential decay. These are referred to as the 'basic' and 'exponential' models respectively. In addition, CASAL allows a (non-estimable) user-specified growth transition matrix (‘matrix’) to be supplied instead of the growth models above.

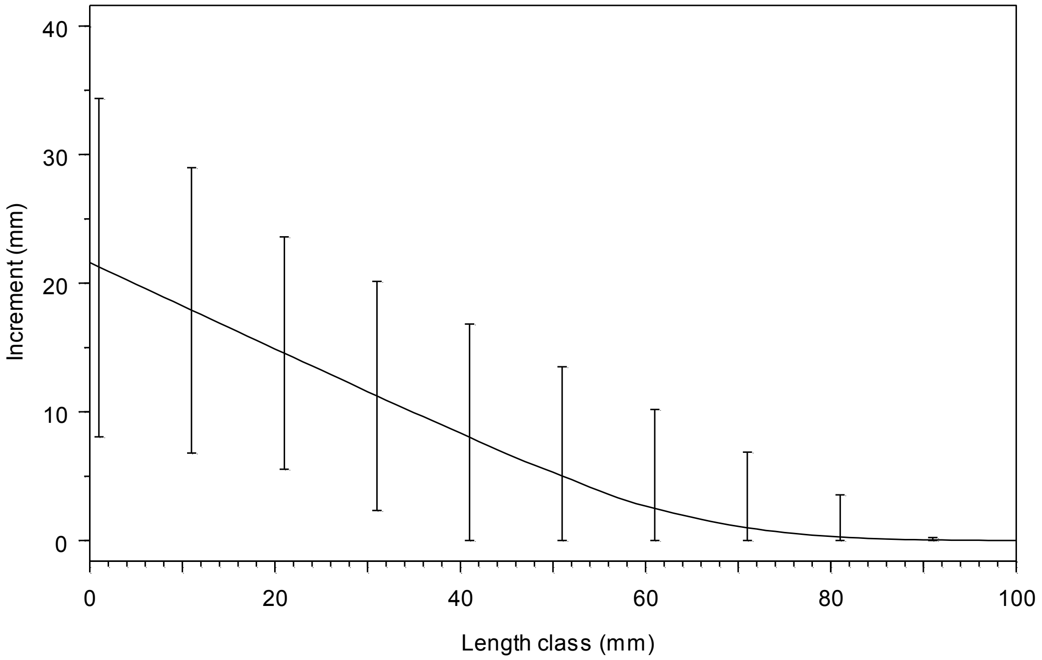

Figure 2: Example of the ‘basic’ growth model (mean and 95% intervals), with parameters *l*=(30, 55), *g*=(11.91, 3.61), *minsigma*=4.45, and *cv*=0.31.

With the ‘basic’ model, it is assumed that the growth of fish in size class *i* is normally distributed with mean

$$\mu = g_\alpha + (g_\beta - g_\alpha)(l_{ci} - l_\alpha)/(l_\beta - l_\alpha),$$

and standard deviation

$$\sigma = \max(c\mu, s_{min}),$$

where l_i is the lower size bound of this size class and $l_{ci}=0.5(l_i+l_{i+1})$.

For the ‘exponential’ model we assume that growth of fish in size class *i* is normally distributed with mean

$$\mu = g_\alpha \left(\frac{g_\beta}{g_\alpha} \right)^{\frac{l_{ci}-l_\alpha}{l_\beta-l_\alpha}},$$

and standard deviation

$$\sigma = \max(c\mu, s_{min}),$$

where l_i is the lower size bound of this size class and $l_{ci}=0.5(l_i+l_{i+1})$.

For both the growth models, where $i<j$, the $[i,j]$th element of the transition matrix (which defines what proportion of the *i*th size class move to the *j*th size class) is simply the integral of this distribution between the bounds (l_j-l_{ci}) and $(l_{j+1}-l_{ci})$. If there is a plus group, the corresponding integrals extend to ∞. The [i,i]th element is the integral between the bounds $-\infty$

and ($l_{i+1}-l_{ci}$). Note that both growth models require the same parameter names but they have different interpretations in each case.

Alternatively, the growth transition matrix can be supplied directly. Supply a square matrix that has one row for each size class in the partition, with each row having length equal to the number of size classes in the partition. The rows of the growth transition matrix must sum to one, and not contain any negative numbers. Here, the matrix element [i,j] defines the proportion of the ith size class that moves to the jth size class. Unlike the growth models above, there are no constraints on the direction of size class movement.

So, you need to tell CASAL the following, for each growth episode:

1. If there are multiple stocks and only one stock grows, which is it?
2. The growth model to be used (either the 'basic' or 'exponential' models above), or supply the growth transition matrix directly.
3. If using a growth model, then the parameters of the growth model with reference sizes l_α and l_β, the corresponding parameters g_α and g_β, a c.v. c and a minimum standard deviation s_{min}. All of these may also depend on sex and maturity.

5.4.6 Mortality (natural and fishing)

Mortality includes natural and fishing mortality — the processes by which fish are removed from the partition. CASAL combines the two processes when they occur in the same time step, hence they are discussed in a single section here.

Each time step can include a proportion of the year's natural mortality and/or one or more fisheries. Natural mortality is applied to all areas and can depend on sex, maturity, stock, and age or size class. A *fishery* is defined as fishing mortality in a specified or area (or group of areas) and time step. You need to supply a catch for each fishery in each year.

CASAL allows for the possibility that the fishery may kill more fish than are reported in the annual catches, but only for non-Baranov models. In this case, the term *removals* refers to all fish killed by the fishery in a year, and *discards* denotes the fish that are killed but not reported in the catch (so removals = catch + discards). These latter fish may be true discards (i.e., fish brought on board the vessel and subsequently returned to the sea), or those killed incidentally by the fishing gear (e.g., fish that escape through the meshes of a trawl net but subsequently die of injuries), or simply unreported landings. The default assumption for each fishery is that there are no discards, so removals = catch.

Natural mortality and fishing mortality occurring in the same area and time step can be sequenced in two different ways. The first option, *instantaneous mortality*, is to apply half the natural mortality, then to apply the mortalities from all the fisheries instantaneously, then to apply the remaining half of the natural mortality. The second option is to use the *Baranov* catch equation, which implies that natural and fishing mortalities are simultaneous. In general, the first option is recommended because it requires much less computation. Note that the use of the Baranov equation for multiple fisheries in the same area in the same time step has not yet been implemented. Whichever option you use is applied to all fisheries.

In the following equations the catch from fishery f is denoted C_f, t is the proportion of the year's natural mortality that occurs in the time step in which the fishery operates and, for the jth element of the partition, S_{fj} is the proportion selected by the fishery, R_{fj} is the proportion of the fish killed by the fishery that are retained (i.e., not discarded), $\overline{w}_j$ is the mean weight, n_j is the pre-mortality number of fish, and M_j is the natural mortality.

With *instantaneous mortality*, the following equations are used.

1. An exploitation rate (actually a proportion) is calculated for each fishery, as the catch over the selected-and-retained biomass,

   $$U_f = \frac{C_f}{\sum_j \overline{w}_j S_{fj} R_{fj} n_j \exp\left(-0.5tM_j\right)}$$

2. The fishing pressure associated with fishery *f* is defined as the maximum proportion of fish taken from any element of the partition in the area affected by fishery *f*,

   $$U_{f,obs} = \max_j \left(\sum_k S_{kj} U_k\right)$$

   where the maximum is over all partition elements affected by fishery *f*, and the summation is over all fisheries *k* which affect the *j*th partition element in the same time step as fishery *f*.

   In most cases the fishing pressure will be equal to the exploitation rate (i.e., $U_{f,obs} = U_f$). This will not be true only if (a) there is another fishery operating in the same time step as fishery *f* and affecting some of the same partition elements, and/or (b) the selectivity S_{fj} does not have a maximum value of 1.

   There is a maximum fishing pressure limit of $U_{f,max}$ for each fishery *f*. So, no more than proportion $U_{f,max}$ can be taken from any element of the partition affected by fishery *f* in that time step. Clearly $0 \leq U_{max} \leq 1$. It is an error if two fisheries which affect the same partition elements in the same time step do not have the same U_{max}.

   For each *f*, if $U_{f,obs} > U_{f,max}$, then U_f is multiplied by $U_{f,max}/U_{f,obs}$ and the fishing pressures are recalculated. In this case the catch actually taken from the population in the model will differ from the specified catch, C_f.

3. The partition is updated using

   $$n'_j = n_j \exp\left(-tM_j\right)\left[1 - \sum_f S_{fj} U_f\right]$$

   where the summation is over all fisheries operating at the current time step.

With *Baranov mortality*, the following equations are used:

1. For each fishery, calculate the associated fishing mortality rate, F_f, by solving the following Baranov equation:

   $$C_f = \sum_j \left[\frac{F_f S_{fj}}{Z_{fj}} \overline{w}_j n_j \left(1 - e^{-Z_{fj}}\right)\right]$$

where $Z_{fj} = tM_j + F_f S_{fj}$. There is no closed form solution for F_f given the other parameters, so this equation must be solved iteratively.

2. The fishing pressure for fishery f is defined as the maximum instantaneous fishing mortality rate for any element of the partition affected by fishery f. Since, with Baranov mortality, there can be no more than one fishery per area per time step, the fishing pressure is

   $$F_{f,obs} = F_f \max_j \left(S_{fj} \right)$$

   There is a maximum fishing pressure limit of $F_{f,max}$ for each fishery. So, $F_{f,max}$ is the maximum instantaneous fishing mortality rate on fish affected by the fishery.

   For each f, if $F_{f,obs} > F_{f,max}$, then F_f is reduced to $F_{f,max}/\max_j(S_{fj})$ and the fishing pressures are recalculated. (Note that $F_{f,max}$ is the maximum permissible value of $F_f \times S_{fj}$ not of F_f. This is confusing, but is allowed in order to maintain compatibility with previous NIWA software. This is another reason why we do not recommend the Baranov option for use in new models.)

3. The partition is updated using

   $$n'_j = n_j \exp\left[-\left(tM_j + F_f S_{fj} \right) \right]$$

   where f is the fishery affecting row i in the time period (if none, then F_f=0).

Your `population.csl` data file should contain a list of selectivities. Each fishery should use one of these selectivities. More than one fishery can share the same selectivity. Also fisheries can share selectivities with observations (for example, a CPUE index could use the same selectivity as the corresponding fishery).

Note that if there are not enough fish to allow the catch to be taken (because one or more fishing pressure limits is exceeded, i.e., $U_{f,obs} > U_{f,max}$ or $F_{f,obs} > F_{f,max}$), CASAL simply reduces the actual catch below the specified catch. If estimating parameters, a parameter set which leads to fishing pressure limits being exceeded is not automatically disallowed. So your point estimate may break fishing pressure limits. To prevent this (as is generally the case), you will need to add catch limit penalties in the estimation section (Section 6.7.6). To see how much the specified catches differ from the actual catches you may request CASAL to output the latter (Section 7.2).

You can specify that observations occur partway through a mortality episode, or that SSBs are calculated partway through mortality. Either way, CASAL needs a method of determining the contents of the partition "after a given proportion p of the mortality". There are two options:

1. *Weighted sum*: after proportion p of the episode, the partition elements are given by $n_{p,j} = (1-p)n_j + pn'_j$. Arguably this is the most natural approach if Baranov is not used, although unless p = 0, 0.5, or 1 it's not logically consistent with the half-M, fishing, half-M sequence used in the instantaneous mortality option.

2. *Weighted product*: after proportion p of the episode, the partition elements are given by $n_{p,j} = n_j^{1-p} n_j'^{p}$. This is the most natural approach if Baranov is used, although it might be desirable to use 'weighted sum' instead for consistency with analyses not using Baranov.

When the Baranov equation is used, CASAL gives the user the option of specifying an F for some years rather than a catch in tonnes. This is intended for modelling the early history of a fishery, if catches were not recorded but the modeller has a vague idea about the level of historical fishing pressure. Be clear that this F is an instantaneous mortality at a selectivity of 1, and that individual partition elements may suffer more or less mortality, depending on the selectivity.

Annual selectivity shifts are also provided for. These allow selectivities to shift to the left or right with changes in an exogenous variable. (In the 2002 hoki assessment, this exogenous variable is either related to the depth being fished or the time of the fishing season, see Francis et al. 2003.) The ogive is shifted by $a_f\left(E_f - \bar{E}_f\right)$, where a_f is a shift factor and E_f is the exogenous variable. This is accomplished by changing the parameters of the ogive, for example, in a logistic ogive, the a_{50} parameter is shifted. Not all ogives support this feature (see Section 5.7 for a complete list). For size-based ogives in an age-based model, the shift is applied before the ogive is converted to age-based. Note that either a_f and/or E_f can be estimated, although it may not be sensible to attempt to estimate both.

So, to specify the mortality processes, you need to tell CASAL the following:

- The value of M, which may depend on sex, maturity, age and/or size.
- The total catch for each fishery in each year.
- Which selectivity is used by each fishery (i.e., S_f).
- Which retention ogive, if any, is used by each fishery (i.e., R_f) (the default is $R_f \equiv 1$).
- The maximum fishing pressure limit for each fishery, as U_{max} for instantaneous mortality, or F_{max} for Baranov mortality.
- Whether the ‘weighted sum’ or ‘weighted product’ approach is to be used to calculate the contents of the partition partway through a mortality episode.
- Optionally if Baranov is used, the instantaneous mortality F to be applied, by year, for a range of years that does not overlap with the range of years for which catches are provided.
- The details of each selectivity, which may include an exogenous shift variable E and a shift parameter a.

5.4.7 Disease mortality

Disease mortality is a special, additional, mortality that is implemented to occur after natural and fishing mortality during a time step. This process removes fish from the partition, is applied to all areas, and can depend on sex/age/size class. It can occur only during one time step in the annual cycle.

The partition is updated using

$$n'_j = n_j \exp\left[-\left(t_{year} M_d S_j\right)\right],$$

where M_d is the disease mortality rate to apply, t is an annual multiplicative scalar (and can be used to index the years in which disease mortality is applied), and S_j is a selectivity to apply to the disease mortality over the sex/age/size classes. As earlier, your `population.csl` data file should contain a list of selectivities. The disease mortality should use one of these selectivities.

5.4.8 Tag release events

Tag release events (also known as mark-recapture events or tag-release events) allow CASAL to incorporate tagging data into the model. These occur as the last transition within a time step during the annual cycle.

To allow tagging to be a part of the model, the number of tagging members of the partition to create must be specified, as must their names. CASAL always creates a "no_tag" member in addition to those specified, and will use this as the source when moving fish into the named tag partition member.

In addition to creating tag members of the partition, you will need to initialise the values by defining a tag-release event (otherwise they will always be zero). This process moves fish from the "no_tag" member of the partition into a named member of the partition. You will need to define how many fish to move, and the year, time step, area, and stock. Also, you may need to define a penalty (see `@fish_tagged_penalty`) to discourage parameter values which do not lead to enough fish being present in the population to allow for the number being tagged (although in cases where only a small proportion of the population is tagged, this is unlikely to be required).

The partition is then updated by moving N fish from the equivalent "no_tag" member of the partition to the named tag member of the partition, where the numbers at age (in an age based model) or numbers at size (in a size based model) are defined by a vector of proportions by age and sex or size and sex respectively. Note that CASAL expects the vector of proportions to sum to 1 over all ages (in an age based model) or sizes (in a size based model) and sex (in a sex-based model).

CASAL allows two methods for determining the proportions at age of the tagged fish in an age-based model, *deterministic* and *free*. In a size-based model, the only option is *free*.

For the *deterministic* method, let l_i be the proportion of fish in size class i in the relevant part of the partition (i.e., area, stock, tag event, etc.), where $\sum l_i = 1$. These are converted into proportions-by-age a_j by generating a conversion matrix M, where each row in M corresponds to an age class and each column to a size class, and M_{ji} is determined from the numbers of fish in the relevant part of the partition, optionally after applying a selectivity, and the current size-at-age distribution of these fish (which may vary between partition rows), i.e.,

$$M_{ji} = N_j \Pr\left(x_j = i\right)$$

where N_j is the number of fish of age j, and $\Pr(x_j=i)$ is the probability that the xth fish of age j has size i. Then a_j is,

$$M_{js,is} = N\left(\text{fish in age class } j, \text{sex } s\right)\Pr\left(\text{each of these fish is in size class } i, \text{sex } s\right)$$

$$a_j = \sum_i \left[\frac{M_{ji} l_i}{\sum_k M_{k,i}} \right]$$

For an age-based model, in the *free* method, the proportions-at-age are simply parameters of the model, which may either be estimated (from observations of proportions at size in an age-based model) or fixed. Additionally, the proportions-at-age may be specified in natural or log space. The later may provide better performance when attempting MCMCs in a Bayesian analysis.

5.4.9 Tag shedding rate

The tag shedding rate transition process applies a tag shedding rate to each tag partition member. Tag shedding transition processes can only be defined if tagging is in the model (see Section 5.4.8).

The partition is updated by removing fish from each tag partition member, where the number removed is defined by `@tag_shedging_rate` and `@tag_shedding_props` commands.

Here, the number of fish in a tagged member of the partition i at time step j is n_{ij}. The partition is updated by applying the tag shedding rate for that tag member of the partition, l_i, by the proportion of tag shedding to apply in that time step t_j, i.e.,

$$n'_{ij} = n_{ij} \exp\left(-t_j l_i\right)$$

5.4.10 Semelparous mortality

Optionally, CASAL can apply semelparous mortality to mature fish, but only if maturity is defined to be a member of the partition (see 5.4.3). Semelparous mortality removes a user-defined proportion (with a default of 1.0) of all mature fish from the partition at the time step defined by the user. The value of semelparous mortality must lie between 0 and 1.

Here, the partition is updated by removing a fixed proportion of mature fish from the partition (defined by `semelparous_mortality`) at the time step defined by `semelparous_time`.

5.5 Setting the initial state

Before setting the initial state of the population you need to supply the equilibrium abundance for each stock. Usually, this is done by specifying either B_0 (equilibrium SSB) or R_0 (equilibrium constant recruitment level, as a number of fish). If you specify B_0, it is used to calculate R_0, and conversely. Alternatively, if the Francis parameterisation of year-class strengths is used you must supply either B_{mean} or R_{mean}, rather than B_0 or R_0. If you specify B_{mean} it is used to calculate R_{mean}, and conversely; in either case, CASAL calculates R_0 from R_{mean} (see Section 5.4.2).

CASAL has an alternative parameterisation of equilibrium abundance for use in two-stock models only. You can specify R_0 or B_0 as the sum over stocks (optionally on the log-scale) and the proportion in each stock. An analogous option based on R_{mean} and B_{mean} is available for when the Francis parameterisation of year-class strengths is used.

CASAL offers the following three methods for setting the initial state of the population.

1. Use the equilibrium state based on constant recruitment R_0.

2. Allow the initial abundance to be different from the equilibrium abundance. You need to supply an initial abundance $B_{initial}$ or $R_{initial}$ for each stock as well as B_0 or R_0 (or, if the Francis parameterisation is used, B_{mean} or R_{mean}). The equilibrium state is calculated, then the numbers of fish of each stock s are multiplied by $B_{initial}(s)/B_0(s)$ if you supplied $B_{initial}$, or by $R_{initial}(s)/R_0(s)$ if you supplied $R_{initial}$. There is also an option

to express $R_{initial}$ as a deviate, i.e., supply $R_{initial}$ relative to R_0, in which case the numbers of fish of each stock s are multiplied by $R_{initial}$.

3. Allow the initial age or size distribution to be different from the equilibrium distribution. (Using this option has approximately the same effect as starting the model some years earlier and estimating the earliest year class strengths.) You need to supply an initial number of fish $C_{initial,\, i}$ for each age or size class i of each stock. The equilibrium state is calculated, then the numbers in each age or size class i are multiplied by a factor such that they sum to the relevant $C_{initial,\, i}$. (Alternatively, you can specify $C_{initial}$ separately for males and females.) Then $B_{initial}$ is calculated for each stock by running the model forwards for one year, with constant recruitment at equilibrium levels and no fishing, and recording the SSB. (The model is put back to the initial state after doing this.) $R_{initial}$ is calculated as $(B_{initial}/B_0)\times R_0$. The SSBs for all years before the initial year are set to $B_{initial}$ (perhaps not ideal, but CASAL needs to fill them in with something in case they are needed for the stock-recruitment relationship or if you ask for them to be printed out).

The algorithm for determining the equilibrium state in a size-based model involves running the model over a number of simulated years with constant recruitment. The number of years to use must be defined; this would usually be the approximate maximum age of the fish.

So, to specify the initial state of the population, you need to supply:

1. R_0 for each stock, or B_0 for each stock (or, if the Francis parameterisation of year-class strengths is used, R_{mean} or B_{mean}). (For a two-stock model you can use the alternative parameterisation above).

2. In a size-based model, the number of years in the constant-recruitment simulations used to determine the equilibrium state.

3. If you want the initial abundance to be able to differ from the equilibrium abundance, then $B_{initial}$ for each stock, or $R_{initial}$ for each stock (optionally, relative to R_0).

4. If you want the initial age or size distribution to be able to differ from the equilibrium age or size distribution, then $C_{initial,\, i}$ (or $C_{initial\_male,\, i}$ and $C_{initial\_female,\, i}$) for each age or size class i of each stock.

5.6 Applying ogives

An ogive is a function with a different value for each age or size class (i.e., for each column of the partition). Ogives are used frequently throughout the CASAL population section: for selectivity curves (Section 5.4.6), rates of migration (Section 5.4.4), and maturation rates (Section 5.4.3).

Ogives have a number of different parametric forms in CASAL and you can use any of these for any ogive parameter. Some common parameterisations are `logistic`, `knife_edge`, `double_normal`, and the most flexible parameterisation `allvalues` where each ogive element is specified separately. See Section 3.4 for instructions on specifying ogives in CASAL. Note also that some ogive forms can be shifted (see Section 5.4.6).

An ogive may be defined to apply to only certain subgroups of fish. For example, `rates_male logistic` would be used to describe a logistic migration ogive for males, and `male_mature logistic` would be used for a logistic selectivity ogive to be applied only to mature males. See Sections 8.7, 8.8, and 8.11 for the permissible subgroup descriptors for maturation, migration, and selectivity ogives, respectively. In the following examples we use *`subgroup`* as a generic subgroup descriptor.

The usage of ogives depends on whether the model is age- or size-based. Ogives can be:

1. *Age-based in an age-based model*

   The ogive is indexed by fish age, with indices from `min_age` to `max_age`.

   For example, you might have an age-based selectivity that was logistic with 50% mark at age 5 and 95% mark at age 7. This would be defined by *`subgroup`* `logistic`, a_{50}=`5`, a_{to95}=(`7-5`)=`2`. Then the value of the ogive at age x=3 is $1\Big/\left[1+19^{(a_{50}-x)/a_{to95}}\right]=1\Big/\left[1+19^{(5-3)/2}\right]$.

2. *Size-based in a size-based model*

   The ogive is indexed by fish size class, with indices from 1 to `n_classes`. The value of the ogive for each size class is a function of the class midpoint. A plus size group has no midpoint, of course, so if you have a plus size group you need to assign it a nominal midpoint using the `plus_group_size` parameter (which is also used to calculate mean weight for the plus group, see Section 5.9).

   For example, you might have size classes of 30–40, 40–50, 50–60, 60–70, and 70+ cm, and want a size-based selectivity that was logistic with 50% mark at 55 cm and 95% mark at 75 cm. This would be defined by *`subgroup`* `logistic`, a_{50}=`55`, a_{to95}=(`75-55`)=`20`. Then the value of the ogive for the second size class is $1\Big/\left[1+19^{(55-45)/20}\right]$.

3. *Size-based in an age-based model*

   This allows the addition of size-based model features to an age-based model, for example a size-based selectivity. The value of the ogive for each element of the partition is the integral of the size-based ogive over the distribution of fish sizes (which depends on age, and potentially on the other partition characters, the year, and the time step, see Section 5.8).

   For example, you might have a size-based selectivity that was logistic with 50% mark at 55 cm and 95% mark at 75 cm. This would be defined by *`subgroup`* `size_based logistic`, a_{50}=`55`, a_{to95}=(`75-55`)=`20`. Suppose the partition is divided by maturity, sex, and age, and that 3-year-old mature male fish in time step 2 have a mean size of 62 cm, and a normal size distribution with a c.v. of 0.2. Then the value of the ogive, for 3-year-old mature male fish in time step 2, is

   $$\int L(x)s(x)\,dx,$$

   where $L(x)$ is the logistic ogive=$1\Big/\left[1+19^{(55-x)/20}\right]$,

   and $s(x)$ is the probability density function of the fish sizes,

$$s(x)= \frac{1}{\sqrt{2\pi}\left(62\cdot 0.2\right)}\exp\left(-0.5\left(\frac{x-62}{62\cdot 0.2}\right)^2\right).$$

CASAL calculates the above integral by a discrete approximation. It takes n_{quant} evenly spaced quantiles of the specified fish size distribution (defined as the quantiles of $((1\ldots n_{quant})-0.5)/n_{quant}$), evaluates the ogive at each, and calculates the average of the ogive values. By default $n_{quant}=5$. This default will generally be adequate, unless the size-based ogives are very steep (e.g., knife-edge). In this case, the resulting age-based ogives may be quite discretised. Fix this problem by increasing the value of n_{quant}. Note that decreasing n_{quant} to 1 effectively bases the ogive on the mean size at age, and ignores the distribution of sizes at age (and reduces the computational cost considerably).

Note that the use of n_{quant} does not affect other uses of variation of size at age in the model, i.e., age-size observations (Section 5.8) or mean weight at size (Section 5.9).

Not all types of ogives can be used as size-based ogives in an age-based model. The permitted types are specified below.

So far, the use of size-based ogives in an age-based model where size-at-age varies from year to year is only implemented for selectivities and proportions maturing (whern maturity is in the partition) — not for proportions mature (i.e., when maturity is not in the partition) or migration rates.

Note that the function values for some choices of parameters for some ogives can result in a computer numeric overflow error (i.e., the number calculated from parameter values is either too large or too small to be represented in computer memory). CASAL implements range checks on some parameters to test for a possible numeric overflow error before attempting to calculate function values. For example, the logistic ogive is implemented such that if ($(a_{50}-x)/a_{to\_95} > 5$) then the value of the ogive at x is zero, i.e., for $a_{50}=5$, $a_{to\_95}=0.1$, then the value of the ogive at $x=1$, without range checking would be $7.1\text{x}10^{-52}$. With range checking, that value is 0 (as $(a_{50}-x)/a_{to\_95}=40 > 5$).

5.7 Ogive descriptions

The available ogives are described below. Table 2 summarises the available ogives and examples of the shapes for most ogives are given in Figure 3.

`constant`

$$f\left(x\right)=C$$

The `constant` ogive has the estimable parameter C. This ogive can be shifted (trivially), and can be used as a size-based ogive in an age-based model.

`knife_edge`

$$\begin{aligned} f(x) &= 0, \quad (x < E) \\ &= 1, \quad (x \ge E) \end{aligned}$$

The `knife_edge` ogive has the non-estimable parameter E, and cannot be shifted. (It might seem straightforward to shift a `knife_edge` ogive, just by changing E, however this cannot work in a gradient-based minimiser, as the test of ($x < E$) is not differentiable). The `knife_edge` ogive can be used as a size-based ogive in an age-based model.

`allvalues`

$$f(x) = V_x$$

The `allvalues` ogive has estimable parameters $V_{low}\ V_{low+1}\ \ldots\ V_{high}$. Here an ogive value for each age or size class is required. The `allvalues` ogive cannot be shifted and cannot be used as a size-based ogive in an age-based model.

`allvalues_bounded`

$$\begin{aligned} f(x) &= 0, && (x < L) \\ &= V_x, && (L \le x \le H) \\ &= V_H, && (x > H)\ (\text{not } f(x) = 1!) \end{aligned}$$

The `allvalues_bounded` ogive has non-estimable parameters L and H. The estimable parameters are $V_L\ V_{L+1}\ \ldots\ V_H$. Here, an ogive value for each age or size class is required. The `allvalues_bounded` ogive cannot be shifted and cannot be used as a size-based ogive in an age-based model.

`logistic`

$$f(x) = 1 \Big/ \left[1 + 19^{(a_{50}-x)/a_{to95}}\right]$$

The `logistic` ogive has estimable parameters a_{50} and a_{to95}. The `logistic` ogive takes values 0.5 at $x=a_{50}$ and 0.95 at $x=a_{50}+a_{to95}$. It can be shifted and can be used as a size-based ogive in an age-based model.

`logistic_capped`

$$f(x) = a_{\max} \Big/ \left[1 + 19^{(a_{50}-x)/a_{to95}}\right]$$

The `logistic_capped` ogive has estimable parameters a_{50}, a_{to95}, and $a_{\max}$,. When $a_{\max}=1$, it is identical to the `logistic` ogive, and otherwise follows a logistic form with values $0.5\times a_{\max}$ at $x=a_{50}$ and $0.95\times a_{\max}$ at $x=a_{50}+a_{to95}$. The `logistic_capped` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`logistic_bounded`

$$
\begin{aligned}
f(x) &= 0, && (x < a_{50} - a_{to95}) \\
&= 1, && (x > a_{50} + a_{to95}) \\
&= 1\Big/\left[1 + 19^{(a_{50}-x)/a_{to95}}\right], && \text{otherwise}
\end{aligned}
$$

The `logistic_bounded` ogive is included to allow CASAL to replicate the ogives in previous NIWA software (`pmod`). It has estimable parameters a_{50} and a_{to95}. The `logistic_bounded` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`double_logistic`

$$
f(x) = \frac{\min\left(a_{\max}\Big/\left[1+19^{(a_{50}-x)/a_{to95}}\right], a_{\max}\Big/\left[1+19^{(x-(a_{50}+b_{50}))/b_{to95}}\right]\right)}{1\Big/\left[1+19^{\left(a_{50}-\left(\frac{a_{50}b_{to95}+a_{to95}(a_{50}+b_{50})}{a_{to95}+b_{to95}}\right)\right)\Big/a_{to95}}\right]}
$$

The `double_logistic` ogive has estimable parameters a_{50}, a_{to95}, b_{50}, b_{to95}, and $a_{\max}$,. The ogive is evaluated as the minimum of a logistic increasing curve (defined by a_{50} and a_{to95}) and a logistic decreasing curve (defined by $a_{50}+b_{50}$ and b_{to95}). The maximum occurs at the intercept of the two logistics, and has value $a_{\max}$. The `double_logistic` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`logistic_product`

$$
f(x) = \frac{a_{\max}\Big/\left(\left[1+19^{(a_{50}-x)/a_{to95}}\right]\left[1+19^{(x-(a_{50}+b_{50}))/b_{to95}}\right]\right)}{\max\left(1\Big/\left(\left[1+19^{(a_{50}-x)/a_{to95}}\right]\left[1+19^{(x-(a_{50}+b_{50}))/b_{to95}}\right]\right)\right)}
$$

The `logistic_product` ogive has estimable parameters a_{50}, a_{to95}, b_{50}, b_{to95}, and $a_{\max}$,. The ogive is the product of two logistic ogives, where the first is increasing (defined by a_{50} and a_{to95}) and the second decreasing (defined by $a_{50}+b_{50}$ and b_{to95}). The `logistic_product` has maximum value of $a_{\max}$, at the function maximum. This value is determined by approximation, and is defined as the maximum value of the 100 step sequence between a_{50}-a_{95} and $a_{50}+b_{50}+b_{to95}$. This approximation should usually be accurate to within 0.1%. The `logistic_product` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`Richards`

$$
f(x) = \left(1\Big/\left[1+19^{(\alpha-x)/\beta}\right]\right)^{\frac{1}{\delta}}
$$

$$
\text{where } \beta = \frac{a_{to95}\log(19)}{\log\left(2^{\delta}-1\right)-\log\left(\left(\tfrac{20}{19}\right)^{\delta}-1\right)}
$$

$$
\text{and } \alpha = a_{50} + \frac{\beta\log\left(2^{\delta}-1\right)}{\log(19)}
$$

The `Richards` ogive has estimable parameters a_{50}, a_{to95}, and an asymmetry parameter δ. When δ=1, it is identical to the `logistic` ogive. The `Richards` ogive takes values 0.5 at $x=a_{50}$ and 0.95 at $x=a_{50}+a_{to95}$. It can be shifted and can be used as a size-based ogive in an age-based model.

`Richards_capped`

$$f(x)=a_{max}\left(1\Big/\left[1+19^{(\alpha-x)/\beta}\right]\right)^{\frac{1}{\delta}}$$

$$\text{where } \beta=\frac{a_{to95}\log(19)}{\log\left(2^{\delta}-1\right)-\log\left(\left(\frac{20}{19}\right)^{\delta}-1\right)}$$

$$\text{and } \alpha=a_{50}+\frac{\beta\log\left(2^{\delta}-1\right)}{\log(19)}$$

The `Richards` ogive has estimable parameters a_{50}, a_{to95}, asymmetry parameter δ, and a_{max} When δ=1, it is identical to the `logistic_capped` ogive, and otherwise follows a `Richards` form with values 0.5×$a_{\max}$ at $x=a_{50}$ and 0.95×$a_{\max}$ at $x=a_{50}+a_{to95}$. It can be shifted and can be used as a size-based ogive in an age-based model.

`double_normal`

$$\begin{aligned} f(x) &= 2^{-\left[(x-a_1)/s_L\right]^2}, && (x\le a_1)\\ &= 2^{-\left[(x-a_1)/s_R\right]^2}, && (x>a_1) \end{aligned}$$

The `double_normal` ogive has estimable parameters a_1, s_L, and s_R. It has values 1 at x=a_1, and 0.5 at $x=a_1-s_L$ or $x=a_1+s_R$. The `double_normal` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`double_normal_capped`

$$\begin{aligned} f(x) &= a_{\max}\times 2^{-\left[(x-a_1)/s_L\right]^2}, && (x\le a_1)\\ &= a_{\max}\times 2^{-\left[(x-a_1)/s_R\right]^2}, && (x>a_1) \end{aligned}$$

The `double_normal_capped` ogive has estimable parameters a_1, s_L, s_R, and $a_{\max}$. When $a_{\max}$=1, it is identical to the `double_normal` ogive, and otherwise follows a double normal form with values $a_{\max}$ at x=a_1, and 0.5×$a_{\max}$ at $x=a_1-s_L$ or $x=a_1+s_R$. The `double_normal_capped` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`double_normal_plateau`

$$
\begin{aligned}
f(x) &= a_{\max} \cdot 2^{-\left[(x-a_1)/s_L\right]^2} &&, \quad (x \le a_1) \\
&= a_{\max} &&, \quad (a_1 < x \le a_1 + a_2) \\
&= a_{\max} \cdot 2^{-\left[(x-(a_1+a_2))/s_R\right]^2} &&, \quad (x > a_1 + a_2)
\end{aligned}
$$

The `double_normal_plateau` ogive has estimable parameters a_1, a_2, s_L, s_R, and $a_{\max}$. When $a_{\max}=1$ and $a_2=0$, it is identical to the `double_normal` ogive, and otherwise follows a double normal form with values $a_{\max}$ at $a_1, < x \le a_1+a_2$, and $0.5 \times a_{\max}$ at $x = a_1 - s_L$ or $x = a_1 + a_2 + s_R$. The `double_normal_plateau` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`double_normal_coleraine`

$$
\begin{aligned}
f(x) &= \exp\left[-(x-a_1)^2 / \sigma_L^2\right], && (x \le a_1) \\
&= \exp\left[-(x-a_1)^2 / \sigma_R^2\right], && (x > a_1)
\end{aligned}
$$

The `double_normal_coleraine` ogive has estimable parameters a_1, σ_L^2, and σ_R^2. This ogive can be shifted and can be used as a size-based ogive in an age-based model. It is designed to replicate the double normal ogive implemented in the Coleraine stock assessment model software (see Hilborn et al. 2001 for detail).

`logistic_producing`

$$
\begin{aligned}
f(x) &= 0, && (x < L) \\
&= \lambda(L), && (x = L) \\
&= \left(\lambda(x) - \lambda(x-1)\right) / \left(1 - \lambda(x-1)\right), && (L < x < H) \\
&= 1, && (x \ge H) \\
\text{where } \lambda(x) &= 1 \Big/ \left[1 + 19^{(a_{50}-x)/a_{to95}}\right]
\end{aligned}
$$

The `logistic_producing` ogive has the non-estimable parameters L and H, and has estimable parameters a_{50} and a_{to95}. The `logistic_producing` ogive cannot be shifted and cannot be size-based. For maturation ogives, $f(x)$ represents the proportion maturing, *not* the proportion mature. This ogive was designed for use in an age-based model with maturity in the partition. In such a model, a `logistic_producing` maturation ogive will (in the absence of other influences) make the proportions mature follow a logistic curve with parameters a_{50}, a_{to9}.

`increasing`

$$
\begin{aligned}
f(x) &= 0, && (x < L) \\
&= f(x-1) + \pi_x\left(1 - f(x-1)\right), && (L \le x \le H) \\
&= f(H), && (x > H) \text{ (note: not 1)}
\end{aligned}
$$

The `increasing` ogive has non-estimable parameters *L* and *H*. The estimable parameters are π_L π_{L+1} … π_H (but if these are estimated, they should always be constrained to be between 0 and 1). The `increasing` ogive cannot be shifted and cannot be used as a size-based ogive in an age-based model. Note that the `increasing` ogive is similar to the `allvalues_bounded` ogive, but is constrained to be non-decreasing.

`increasing_capped`

$$\begin{aligned} f(x) &= 0, && (x < L) \\ &= f(x-1) + \pi_x\left(C - f(x-1)\right), && (L \le x \le H) \\ &= f(C), && (x \ge H) \end{aligned}$$

The `increasing_capped` ogive has non-estimable parameters *L*, *H*, and *C*. The estimable parameters are π_L π_{L+1} … π_{H-1}. Note that the maximum is π_{H-1}, not π_H as for the `increasing` ogive. As for the `increasing` ogive, if these are estimated then they should always be constrained to be between 0 and 1. The `increasing_capped` ogive cannot be shifted and cannot be used as a size-based ogive in an age-based model. Note that the `increasing_capped` ogive is similar to both the `increasing` ogive, but is constrained to be non-decreasing up to a specified cap.

`Hillary`

$$\begin{aligned} f(x) &= 0, && (x \le \alpha) \\ &= \frac{(x-\alpha)^\gamma}{2^{-\gamma}\psi^\gamma}\left(1 + \frac{(x-\alpha)^2}{\psi^2}\right)^{-\gamma}, && (x > \alpha) \end{aligned}$$

The `Hillary` ogive (Payne et al. 2005) has estimable parameters α, ψ, and γ. It has values 0 at $x \le \alpha$, 1.0 at $x = \alpha + \psi$, and a right-hand limb decay rate γ. The `Hillary` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`Hillary_capped`

$$\begin{aligned} f(x) &= 0, && (x \le \alpha) \\ &= a_{\max}\left(\frac{(x-\alpha)^\gamma}{2^{-\gamma}\psi^\gamma}\left(1 + \frac{(x-\alpha)^2}{\psi^2}\right)^{-\gamma}\right), && (x > \alpha) \end{aligned}$$

The `Hillary_capped` ogive has estimable parameters α, ψ, γ, and $a_{\max}$. When $a_{\max} = 1$ it is identical to the `Hillary` ogive, and otherwise follows a `Hillary` form with values 0 at $x \le \alpha$, $a_{\max}$ at $x = \alpha + \psi$, and a right-hand limb decay rate γ. The `Hillary_capped` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`cosh`

$$f(x)=\frac{y}{2}\left(\exp\left(-\alpha\left(x-x_{\min}\right)\right)+\exp\left(\beta\left(x-x_{\min}\right)\right)\right)$$

The `cosh` ogive has estimable parameters y, $x_{\min}$, α, and β. It is 'U-shaped' (unlike most other parameterised ogives which are generally domed) with minimum value y at $x_{\min}$, and increasing left-hand and right-hand limbs described by α and β respectively. Note that the function has been implemented with constraints y, α, and $\beta > 0$. The `cosh` ogive can be shifted and can be used as a size-based ogive in an age-based model.

`double_exponential`

$$\begin{aligned} f(x) &= y_0\left(y_1/y_0\right)^{(x-x_0)/(x_1-x_0)} && (x \le x_0) \\ &= y_0\left(y_2/y_0\right)^{(x-x_0)/(x_2-x_0)} && (x > x_0) \end{aligned}$$

The `double_exponential` ogive has non-estimable parameters x_1 and x_2, and estimable parameters x_0, y_0, y_1, and y_2. It can be 'U-shaped' (like cosh). Bounds for x_0 must be such that $x_1 < x_0 < x_2$. The ogive passes through the points (x_1, y_1), (x_0, y_0), and (x_2, y_2). If both y_1 and y_2 are greater than y_0 the ogive is 'U-shaped' with minimum at (x_0, y_0).

Table 2: Summary of the ogives available in CASAL, their parameters, and if they can be shifted or used as a length-based ogive in an age based model.

Ogive	Parameters		Shift?	Length-based?
	Non-estimable	Estimable		
`constant`		C	Yes	Yes
`knife_edge`	E		No	Yes
`allvalues`		$V_{low}\ V_{low+1}\ \ldots\ V_{high}$	No	No
`allvalues_bounded`	$L\ H$	$V_L\ V_{L+1}\ \ldots\ V_H$	No	No
`logistic`		$a_{50}\ a_{to95}$	Yes	Yes
`logistic_capped`		$a_{50}\ a_{to95}\ a_{\max}$	Yes	Yes
`logistic_bounded`		$a_{50}\ a_{to95}$	Yes	Yes
`double_logistic`		$a_{50}\ a_{\mathrm{to}95}\ b_{50}\ b_{to95}\ a_{\max}$	Yes	Yes
`logistic_product`		$a_{50}\ a_{\mathrm{to}95}\ b_{50}\ b_{to95}\ a_{\max}$	Yes	Yes
`Richards`		$a_{50}\ a_{to95}\ \delta$	Yes	Yes
`Richards_capped`		$a_{50}\ a_{to95}\ \delta\ a_{\max}$	Yes	Yes
`double_normal`		$a_1\ s_L\ s_R$	Yes	Yes
`double_normal_capped`		$a_1\ s_L\ s_R\ a_{\max}$	Yes	Yes
`double_normal_plateau`		$a_1\ a_2\ s_L\ s_R\ a_{\max}$	Yes	Yes
`double_normal_coleraine`		$a_1\ \sigma_L^2\ \sigma_R^2$	Yes	Yes
`logistic_producing`	$L\ H$	$a_{50}\ a_{to95}$	No	No
`increasing`	$L\ H$	$\pi_L\ \pi_{L+1}\ \ldots\ \pi_H$	No	No
`increasing_capped`	$L\ H\ C$	$\pi_L\ \pi_{L+1}\ \ldots\ \pi_{H-1}$	No	No
`Hillary`		$\alpha\ \psi\ \gamma$	Yes	Yes
`Hillary_capped`		$\alpha\ \psi\ \gamma\ a_{\max}$	Yes	Yes
`cosh`		$y\ x_{\min}\ \alpha\ \beta$	Yes	Yes
`double_exponential`	x_1, x_2	x_0, y_0, y_1, y_2	No	Yes

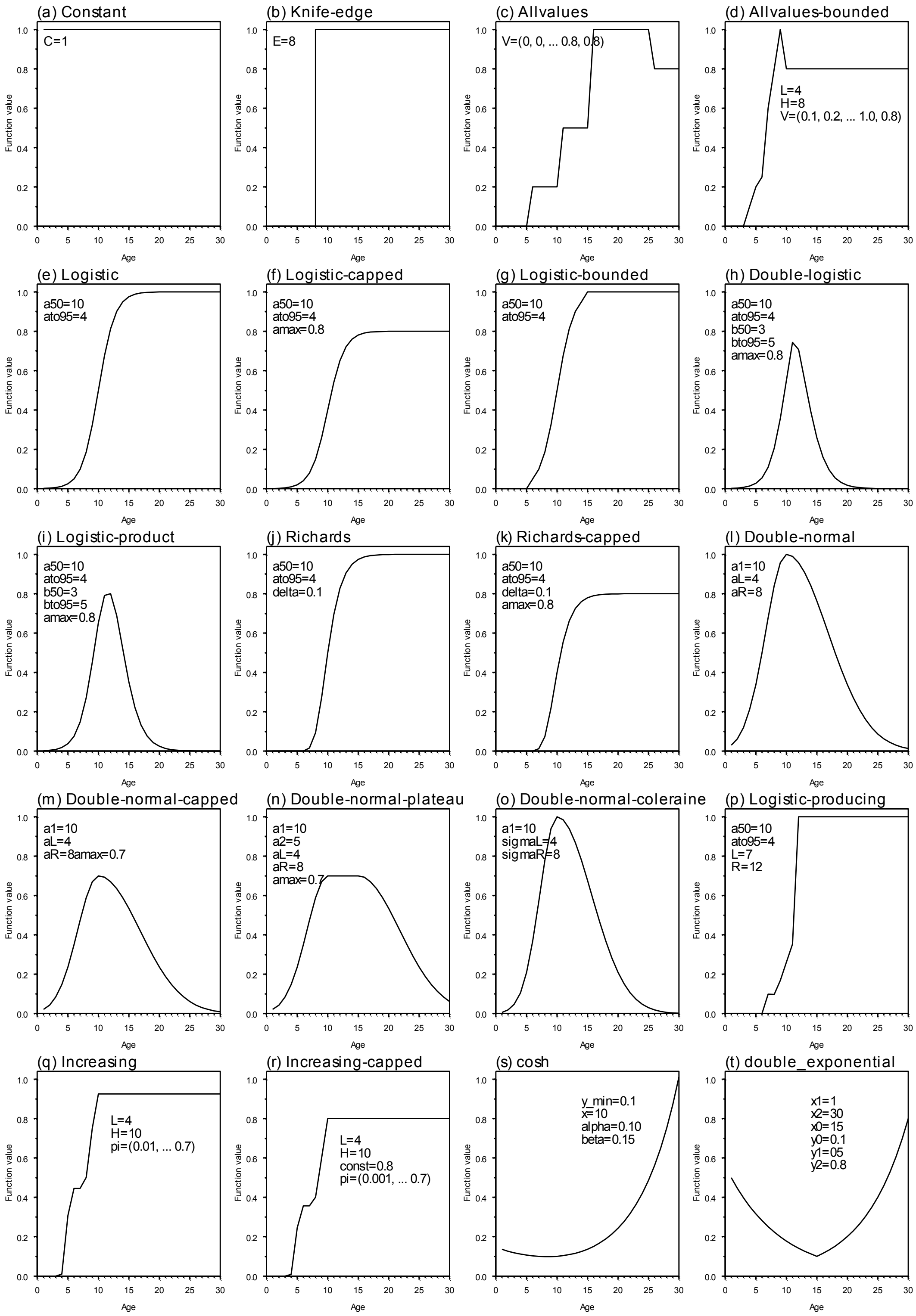

Figure 3: Examples of the ogives available in CASAL.

5.8 Calculation of size-at-age (in an age-based model)

In an age-based model, fish size does not feature in the partition, but size-at-age is still an element of the model. See Section 5.4.5 for a discussion of fish growth in a size-based model.

Size-at-age is based on a growth curve which specifies the mean size at a given age. There are two alternative growth curves in CASAL:

1. von Bertalanffy, where size at age is defined as, $\bar{s}(age) = L_{\text{inf}}\left(1-\exp\left(-k\left(age-t_0\right)\right)\right)$
2. Schnute, where size at age is defined as,

$$\bar{s}(age) = \begin{cases} \left[y_1^b + \left(y_2^b - y_1^b\right)\dfrac{1-\exp\left(-a\left(age-\tau_1\right)\right)}{1-\exp\left(-a\left(\tau_2-\tau_1\right)\right)}\right]^{1/b} & \text{if } a \neq 0, b \neq 0 \\ y_1 \exp\left[\ln\left(y_2/y_1\right)\dfrac{1-\exp\left(-a\left(age-\tau_1\right)\right)}{1-\exp\left(-a\left(\tau_2-\tau_1\right)\right)}\right] & \text{if } a \neq 0, b = 0 \\ \left[y_1^b + \left(y_2^b - y_1^b\right)\dfrac{age-\tau_1}{\tau_2-\tau_1}\right]^{1/b} & \text{if } a = 0, b \neq 0 \\ y_1 \exp\left[\ln\left(y_2/y_1\right)\dfrac{age-\tau_1}{\tau_2-\tau_1}\right] & \text{if } a = 0, b = 0 \end{cases}$$

The von Bertalanffy curve is parameterised by L_{inf}, k, and t_0; the Schnute curve (Schnute 1981) by y_1 and y_2, which are the mean sizes at reference ages τ_1 and τ_2, and a and b (when b=1, this reduces to the von Bertalanffy with k=a). All these parameters can depend on sex, stock, and/or growth-path. (But note, all the parameters should depend on the same thing. You cannot supply L_{inf} by sex and k for both sexes combined.)

The model can incorporate changes in size-at-age during the year — i.e., growth between fish birthdays — by incrementing *age* as specified by the `annual_cycle.growth_props` parameter (see Section 5.3).

Optionally, if tagging is a part of the partition and the growth curve is von Bertalanffy, you can specify a growth loss period for tagged fish in each tag partition member. Here, for each tag partition member i, the period of "no growth" is specified by the parameter g_i, where $g_{\text{no\_tag}} = 0$, i.e., the size at age is defined as,

$$\bar{s}(age) = L_{\text{inf}}\left(1-\exp\left(-k\left(age-t_0-g_i\right)\right)\right)$$

Note that this is not an ideal solution to inclusion of the effect of a tagging event on mean growth within the model, particularly if the model contains data about recaptures at size during the period of the "no growth" period.

Optionally, you can give CASAL mean-size-at-age data instead of a growth curve. For one or more years, provide the mean size at each age. These data should apply to a single time step, which you must specify (using `@size_at_age_step`). For example, you might provide mean size data for 1991, 1992, 1994 and 1997 which are accurate for time step 2. If fish growth occurs between birthdays (i.e., some element of `growth_props` is nonzero), then

CASAL uses interpolation to fill in mean fish sizes in other time steps (described below, but see Section 5.8.1 for details of the method of calculation).

Note that these mean-size-at-age data are not treated like observations, which CASAL attempts to match as closely as possible; rather, CASAL uses the exact figures given.

CASAL can potentially use mean fish size data from any or all years in the range *initial* to *final*. Use `@size_at_age_years` to specify which years you will provide these data for.

If data are not provided for a particular year, then CASAL needs to fill in the missing values. This can happen in several situations:

1. An internal gap. Data are provided for at least one year after that year, and for at least one year before that year (e.g., if data are provided for 1992, 1993, 1995, and 1996, then 1994 is an internal gap)
2. An external gap. Either there are no data for years after that year, or no data for years before that year (in the above example, 1991 and 1997 are external gaps)
3. While calculating a deterministic or stochastic equilibrium, i.e., during MCY/CAY yield calculations, or when calculating the initial state of the partition in year *initial*. In this case the data is never provided and is always filled in with the mean of the values provided.

There are two possible options for filling in the missing values. The first is to use the mean of the values provided. The second is to use interpolation: an internal gap is filled in using interpolation from the two provided years which bracket it, or an external gap is filled in with the values from the closest provided year. The option used is determined by the user through the `@size_at_age_miss` command,

`@size_at_age_miss`	Internal gaps	External gaps
`mean` (the default)	use the mean	use the mean
`interp`	interpolate	use the closest year
`interp.mean`	interpolate	use the mean
`mean.interp`	use the mean	use the closest year

You can provide mean size-at-age data for the projection period (see Section 7.3). And, in some circumstances you may want to provide mean fish size data for year `current`+1, even if doing model runs without projections. The reason is that CASAL needs mean sizes for year `current`+1 in order to calculate (by interpolation) mean sizes for those time steps (if there are any) in year `current` which are after the time step at which your sizes are valid. However, you do not need to provide data for year `current`+1, and it is unlikely to make a substantial difference if you don't.

If you are unsure if CASAL is calculating mean sizes appropriately from the data provided, then the best thing to do is to request the mean-sizes-at-age be printed for each time step in each year. Do this by setting the `@print.every_mean_size` to `True` in the `output.csl` parameter file. Note that you may also need to set `@print.population_section` to `True` as well.

Optionally, you can specify distributions of sizes at age, as well as the mean size at age. These size distributions are used to fit size frequency and age-size observations (Section 6.6), to calculate mean-weights-at-age (Section 5.9), and to convert size-based ogives to age-based (Section 5.6). Two distributional forms are implemented, normal and lognormal, and you can also allow the distribution to vary as a linear function of either age (the default, for backwards compatibility) or mean size (often a more logical choice) . For either distributional form, you need to give the c.v.s or standard deviations of the distribution of the size-at-age, which can depend on sex, stock, and growth-path.

If you use a growth curve, then as an optional feature, CASAL allows you to model annual growth variation. There is an annual growth variable for each year, indicating “how many average years growth” fish achieve in that year. The mean size of a fish of age a in year y is then given by $f(e_{ay})$, where f is the growth curve, e_{ay} the ‘effective’ age, given by

$$e_{ay} = \sum_{y-a+I_{aged}}^{y-1+I_{aged}} r_i$$

where $I_{aged}=1$ if fish have been aged yet in the year or 0 else, and the r_i are annual growth variables (in year y all fish grow as much as they would in r_y years of average growth). If within-year growth is included in the model, and proportion p of annual growth has occurred by a given time step, then the effective age at that time step is $e_{ay}+pr_{y+I\_aged}$. Be careful to avoid off-by-one errors when specifying r_i. They can cover any range of consecutive years. (Note that different annual growth variables for different stocks is not implemented.) If you use this feature, you can only use size-based ogives for selectivity, not for any other model feature such as migration rates.

So, to specify size at age, you need to tell CASAL the following:

1. Which growth curve is to be used — Schnute or von Bertalanffy. The parameters of the growth curve (which can depend on sex, stock, growth-path).
2. Alternatively, mean-size-at-age data for one or more years.
3. Whether size distributions around the mean are to be used. If so, with what distribution, and what c.v.s or s.d.s (which again can depend on sex, stock, growth-path), and if the variation in c.v.s or s.d.s is a linear function of age or mean size.
4. If annual growth variation is used, the growth variable for each year.

Be careful about the scale of the parameters (i.e., L_{inf} for von Bertalanffy growth). This should be in units compatible with the size-weight relationship (Section 5.9). For example, if you provide catch in tonnes and the size-weight relationship on a scale that converts a length in centimetres to a weight in tonnes, then the growth curve should be specified in centimetres.

5.8.1 Interpolation of size at age

This section details the equations used to calculate mean-size-at-age in an age-based model, when mean size data are provided for one or more years. The objective is to determine M_{ays}, the mean size of fish of age a in year y, time step s.

Let P_y be an indicator which takes the value 1 if and only if mean size data are supplied for year y. For years y with $P_y=1$, we have size-at-age values $\{V_{ay}\}$ for $a=a_{min}\ldots a_{max}$ (the range of ages in the partition). We also have `@annual_cycle.growth_props`, indicating the growth between fish birthdays: let g_s be the entry of `growth_props` for time step s. Let t be the time step to which the mean size data applies, and b the fish birthday, i.e., the time step in which age incrementation is carried out.

The simplest case is when $P_y=1$ and $s=t$, in which case $M_{ays}=V_{ay}$.

Now let

$$\begin{aligned} y_1 &= y, & s &\geq t \\ &= y-1, & s &< t \end{aligned},$$

y_2=y_1+1,

$$\begin{aligned} a_1 &= a, & &\text{if } t < s < b, \text{ or } s < b < t, \text{ or } b < t < s, \text{ or } b = t, \text{ or } s = t \\ &= a-1, & &\text{otherwise, and} \end{aligned}$$

a_2=a_1+1

If P_{y1}=P_{y2}=1, then we use interpolation to fill in the mean sizes for this time step,

$$M_{ays} = w_1 V_{a1\,y1} + w_2 V_{a2y2}$$

$$\text{where } w_1 = \begin{cases} g_t - g_s & (a_1 = a-1) \\ 1 + g_t - g_s & (a_1 = a) \end{cases} \text{ and } w_2 = 1 - w_1 .$$

These equations need to be modified for boundary cases, but this is simply done. If $a_1 > a_{max}$ or y_2 > `final`, subtract 1 from a_1, a_2, y_2, w_1; and if, in addition, $y_1 - 1$ > `initial`, subtract 1 from y_1; and add 1 to w_2. If $a_1 < a_{min}$ or y_1 < `initial`, add 1 to a_1, a_2, y_1, y_2, and w_1 and subtract 1 from w_2. Technically, what these modifications do is to change from an interpolation to an extrapolation.

Now if P_{y1}=0 and/or P_{y2}=0, then we need to 'fill in the blanks' for the missing year(s) before carrying out the interpolation. This can happen in ordinary model runs if the user has not provided data for one or more years; also, it always happens if the model is being run to stochastic or deterministic equilibrium (either to find the initial state of the model, or during yield calculations).

There are two possible options at this stage. The first is to replace values for missing years with the average over all years provided,

$$V_{a,\,avg} = \frac{\sum\limits_{y:P_y=1} V_{ay}}{\sum P_y}$$

This option is always used when the model is being run to equilibrium. Also, it is used for internal gaps (years y for which there is at least one u>y for which P_u=1 and at least one l<y for which P_l =1) if the user has set `@size_at_age_miss` to either `mean` or `mean.interp`, and for external gaps (years where the condition above does not hold) if the user has set `@size_at_age_miss` to either `mean` or `interp.mean`.

The second option is to fill in missing years using interpolation. An internal gap is filled in using interpolation from the two provided years bracketing it, if the user has set `@size_at_age_miss` to either `interp` or `interp.mean`. Let u be the earliest year after y for which P_u=1 and l be the latest year before y for which P_l=1, then interpolate with

$$V'_{ay} = \frac{V_{al}(u-y) + V_{au}(y-l)}{u-l}.$$

An external gap is filled in with the values from the nearest year, if the user has set `@size_at_age_miss` to `interp` or either `mean.interp`.

5.9 Calculation of mean weight

In size-based models, the size-weight parameters a and b need to be provided, which can depend on sex and stock. CASAL then calculates the mean weight for each size class as

$$\text{mean weight}=a\times\text{size}^b,$$

where size is approximated by (upper bound plus lower bound of size class)/2. If there is a plus group, you need to specify a nominal mean size for it using the `plus_group_size` parameter.

In age-based models, size-weight parameters a and b also need to be provided, and these can depend on sex and stock. If you don't specify a distribution for sizes-at age (see Section 5.8), then the mean weight for a given partition element is calculated as,

$$\text{mean weight}=a\times(\text{mean size at age})^b$$

where the mean size at age can depend on the other partition characters, the time step, and the year (Section 5.8).

If a size distribution is specified, then the mean weight at age is calculated over that distribution, using the following formula, which is exact for lognormal distributions, and a good approximation for a normal distribution (if the c.v. is not large),

$$\text{mean weight}=a\times(\text{mean size at age})^b \times \left(1+cv^2\right)^{\frac{b(b-1)}{2}}$$

where cv is the c.v. of sizes-at-age for that element of the partition and the term $\left(1+cv^2\right)^{\frac{b(b-1)}{2}}$ corrects for bias in estimating mean weight.

Be careful about the scale of a — this is easily specified incorrectly. If the catch is in tonnes, and the growth curve in centimetres, then a should be on the right scale to convert a length in centimetres to a weight in tonnes. Within the fin fisheries at NIWA, a is more often expressed on a scale to convert length in centimetres to weight in kilograms, and the user needs to divide this figure by 1000. Also note that the command `@size_weight` has the optional subcommand `verify_size_weight` that can be used to help check that the units specified are plausible.

5.10 Weightless model (running CASAL as a numbers only model)

You may wish to use a model which does not involve fish weight at all, but models the number of individuals instead (as can be the case in some shellfish models). For this type of model, abundance and catch data need to refer to numbers of fish, not biomass. If this is the case, set the `@weightless_model` switch to `true` (the default is `false`). The effect of this command is to assume a size-weight relationship of w=1, i.e., each fish is assumed to have a nominal 'weight' of 1 tonne, irrespective of size. Note that CASAL will still label catches and abundances as if they were biomass (e.g., CASAL will report SSBs) but in each case, these values can be read directly as numbers of individuals.

Note that if `@weightless_model true`, then specifying any size-weight parameters in the `population.csl` file will generate an error.

5.11 Maturity, in models without maturity in the partition

When maturity is not a character in the partition, processes may still depend on maturity. You must then make the assumption that the proportion of mature fish in each element of the partition remains constant over time. The proportion of mature fish in each size or age class must be provided, and these can depend on sex (but not on stock). (Also, note that you are providing the proportion of mature fish, not the proportion of maturing fish as in Section 5.4.3).

Once you have done this, you can calculate SSB as a mature biomass (Section 5.3) and calculate fits to observations which relate to maturity (Section 6.6).

It is possible to have migrations which move only immature or only mature fish. If you migrate fish on the basis of maturity when maturity is not a partition character, be aware that the model does not know that the arriving fish are all mature, or all immature. So, if you migrate only mature fish into the spawning area, you need to tell CASAL that the SSB includes all fish in the relevant area — because it does not know that there are no immature fish present. See the subcommand `spawning_use_total_B` (Section 8.2).

You may not want to include maturity in the model in any shape or form, but CASAL still insists that you give it information on proportions mature. In this case, just set the proportion of mature fish to 1 for all age/size classes, as follows,

```
@maturity_props
all constant 1
```

Then the SSB is simply the total biomass in the spawning area at the appropriate time.

6. THE ESTIMATION SECTION

6.1 Role of the estimation section

The tasks carried out by the estimation section are:

1. Get the *point estimate*, i.e., the maximum likelihood estimate (MLE) or maximum posterior density estimate (MPD) (see Section 6.3).

2. *Profile* selected parameters, i.e., find, for each of a series of values of a parameter, allowing all other free parameters to vary, the minimum value of the objective function (Section 6.4). This is called either a likelihood or posterior profile.

3. For Bayesian estimation only, generate an *MCMC* sample from the posterior distribution (Section 6.5).

4. For maximum likelihood or Bayesian estimation, calculate the approximate *covariance* matrix of the parameters as the inverse of the minimiser's approximation to the Hessian, and the corresponding correlation matrix (Section 6.3).

A key decision is between likelihood or Bayesian estimation, which define the *objective function* as the negative log-likelihood or negative log-posterior respectively (Section 9.1).

6.2 Specifying the free parameters

You need to tell CASAL which of the estimable parameters are to be freed by using `@estimate` commands (see Section 9). An `@estimate` command-block looks like this,

```
@estimate
parameter initialization.B0
lower_bound 1000
upper_bound 100000
prior uniform
```

See Section 3.4 for instructions on how to generate the parameter name. At least one free parameter must be specified (or CASAL will report an error). You still provide values for the free parameters as normal. These are used as the starting values for the minimiser (unless you provide alternative starting values using `casal -i`, see Section 3.1).

All parameters are estimated within bounds. For each free parameter (scalar, vector, or ogive), specify the bounds, and, in a Bayesian analysis, the prior (Section 6.7.4). Note that the bounds and prior on an ogive refer to the ogive free parameters, not the actual values of the ogive.

You need to estimate all the estimable parameters of an ogive if you estimate any, but you can fix some of them if you want by setting the lower and upper bounds equal. Similarly, if you want to estimate only some elements of a vector, fix the others by setting the bounds equal.

Catchability coefficients *q* are a special case, because no starting value is provided in the *nuisance* method (see Section 6.7.1) whereas all other free parameters always need starting values. But you still need to provide an `@estimate` block for each *q*, containing the bounds on the *q*, using the following format,

```
@estimate
parameter q[label].q
lower_bound 1e-6
upper_bound 1e-2
```

where `label` is the label of the *q* in the observations blocks. If you're still uncertain how to do this then look at the example in Section 14.

To estimate two (scalar, vector, or ogive) parameters and to constrain them to be the same, you need to use the `same` subcommand. This might arise if, for example, you wanted to use the same migration ogive for two different migrations involving different stocks. Only use one `@estimate` block, for one or other of the two parameters. Put in the `same` command, using the name of the other parameter as the argument. For example,

```
@estimate
parameter growth[1].g
same growth[2].g
…
```

means that the `g` parameters for the first two growth episodes are both estimated, but constrained to be equal. (Don't put in a second `@estimate` block for `growth[2].g` with `same growth[1].g`.)

6.3 Point estimation

Point estimation is invoked with `casal -e`, and also used in several other tasks. Mathematically, it is an attempt to find a minimum of the objective function. CASAL approaches this optimisation problem using a quasi-Newton minimiser built into Betadiff, which is a slightly modified implementation of the main algorithm of Dennis Jr. & Schnabel (1996).

The minimiser has three kinds of (non-error) exit status:

1. *Successful convergence* (suggests you have found a local minimum, at least).
2. *Failure to converge* (you have not reached a local minimum, though you may deem yourself to be 'close enough' at your own risk).
3. *Convergence unclear* (the minimiser has halted and is unable to determine if it is at a minimum. It is possible that a local minimum may have been found, but you should check by restarting the minimiser at the final values of the free parameters).

You can choose the maximum number of quasi-Newton iterations and objective function evaluations allotted to the minimiser. If it exceeds either limit, it exits with a convergence failure. We urge you to use large numbers of evaluations and iterations (at least the defaults of 300 and 1000) unless you successfully reach convergence with less. You can also specify the starting point of the minimiser using `casal -i`.

Is should be stressed that this is a local optimisation algorithm trying to solve a global optimisation problem. This means is that even if a 'successful convergence' message is received, the solution may be only a local minimum and not a global one. To diagnose this problem, try doing multiple runs from different starting points and comparing the results, or (probably better) doing profiles of one or more key parameters and seeing if any of the profiled estimates is actually better than the original point estimate. Otherwise you may have reached a false (local) minimum.

The approximate covariance matrix of the free parameters can be calculated as the inverse of the minimiser's approximation to the Hessian, and the corresponding correlation matrix is also calculated. These results are printed when the `print.covariance` parameter is used (and, optionally, the Eigen values of the Hessian, see Section 10.1). Be aware (i) that the Hessian approximation develops over many minimiser steps, so if the minimiser has only run for a small number of iterations the covariance matrix can be a very poor approximation, (ii) in some cases the inverse Hessian is not a good approximation to the covariance matrix of the free parameters, and in general should not be used, for example, to construct confidence intervals. Also note that if a free parameter has equal lower and upper bounds, it will have entries of '`0`' in the covariance matrix and '`NaN`' or '`-1.#IND`' (depending on the operating system) in the correlation matrix.

Multi-phase estimation is allowed, and is implemented in a manner similar to that by AD-Model Builder (Otter Research Limited 2000). In this case, some free parameters are initially held fixed, and a minimisation is carried out. Next, some or all of the fixed parameters are freed, and another minimisation is carried out, etc. Sensible starting values should be used for the fixed parameters. In some situations, this may be quicker and/or more effective than estimating all the parameters in a single minimisation. (The main idea is that the 'key' parameters should be freed first and the 'nuisance' parameters last, although there is little known about the actual performance improvements that may be expected from this approach.) If this feature is used, then each parameter should be allocated a 'phase'; the default phase is 1. The phase 1 parameters are freed first, then the phase 2 parameters, etc. You can specify that a different maximum number of iterations and/or evaluations is to be used for the 'intermediate' phases, i.e., all but the last. It would probably be advisable to use rather less effort for the intermediate phases than for the final phase.

An option (`casal -E`) is provided in which the point estimate is calculated using finite difference gradients instead of automatic differentiation. This was implemented for three reasons:

1. You can use finite differences to check your results if you suspect the automatic differentiation is misbehaving.

2. If you become aware of problems in the automatic differentiation section of Betadiff and can't fix them, switch to finite differences.

3. You may find that finite differences is faster than automatic differentiation, with comparable accuracy, for problems with small numbers of free parameters. (Although, for large problems, ~100 free parameters, it can be many times slower.)

This option is implemented only for simple point estimation, and not for profiling or for the initial point estimate in MCMC runs.

6.4 Likelihood or posterior profiles

If profiles are requested (`casal -p`), CASAL will first calculate a point estimate. For each scalar parameter or, in the case of vectors or ogives, the element of the parameter to be profiled, CASAL will fix its value at a sequence of n evenly spaced numbers between specified bounds l and u, and calculate a point estimate at each value. By default n=10, and (l, *u*)=(lower bound on parameter plus (range/(2*n*)), upper bound on parameter less (range/(2*n*)). Each minimisation starts at the final parameter values from the previous resulting value of the parameter being profiled. The program reports the objective function for each parameter value, and all the parameter estimates. The initial point estimate is also inserted into the profile (note that this can serve as a check that none of the other points along the profile have a better objective function value than the initial 'minimum').

You specify which parameters are to be profiled, and optionally *n*, *l*, and *u* values for each. In the case of vector or ogive parameters, specify the element of the vector being profiled.

You can also supply the initial point estimate using `casal -i`, so that CASAL does not need to do the first minimisation. Be aware that you are supplying the point estimate, not the minimiser starting point to get to the point estimate (as in other situations where `casal -i` is used).

If you have specified multi-phase estimation (see Section 6.3), it is only used for the initial point estimate. Subsequent minimisations are done single-phase, as they should start reasonably close to the endpoint and so should not need multiple phases.

If you are doing a Bayesian analysis and want likelihood profiles rather than posterior profiles, then either switch to the likelihood objective function (using `@estimator likelihood`) for the duration or make all the priors uninformed.

If you get an implausible profile, it may be a result of not using enough iterations in the minimiser (in this case, increase `max_iters` and/or `max_evals` and retry), or the convergence criteria may not be strong enough (try setting `grad_tol` to a smaller value).

You can direct the profile estimates to an output parameter file with the command line option `-o` (see Section 3.2). Note that if you use the `-o` command line option, then, following the completion of the profile estimates, CASAL will use this file as the input parameter file for a `-r` run. The results from the `-r` run will then be appended to standard output.

6.5 Bayesian estimation

CASAL can:

1. Use a Monte Carlo Markov Chain to generate a sample from the posterior distribution of the free parameters (`casal -m` or `-M`); and output the sampled values to a file, (optionally only every *n*th set of values).

2. If the run is interrupted, recover the results from the file and continue the run from where it left off, appending the results to the file (`casal -a`).

3. Combine the results of one or more chains into a single posterior sample by removing samples from the 'burn-in' periods and concatenating the results; allow the user to reduce the size of the resulting sample by sub-sampling; and optionally, apply prior re-weighting in the sub-sampling process, i.e., apply probability weights to generate a sample from a posterior based on a different prior (`casal -C`). The sub-sampling may be either systematic (every *n*th point) or randomly (with replacement). The former is recommended (to minimise autocorrelation) except with prior re-weighting, when the latter must be used.

4. For a posterior sample, calculate the values of various output quantities at each sample point and export these so that they can be plotted and/or summarised using an external package (use `casal -v`).

Two major steps are best done by an external package, as CASAL has no post-processing capabilities. CASAL cannot:

1. Produce MCMC convergence diagnostics (use a package such as BOA, see www.public-health.uiowa.edu/boa).

2. Plot/summarize the posterior distributions of the output quantities (for example, using a general-purpose statistical or spreadsheet package such as S-Plus, R, or Microsoft Excel).

Bayesian methodology and MCMC are both large and complex topics, and we do not describe either properly here. See Gelman et al. (1995) and Gilks et al. (1998) for details of both Bayesian analysis and MCMC methods. In addition, see Punt & Hilborn (2001) for an introduction to quantitative fish stock assessment using Bayesian methods.

This section only briefly describes the MCMC algorithms used in CASAL. See Section 3.2 for a better description of the sequence of CASAL commands used in a full Bayesian analysis.

CASAL uses a straightforward implementation of the Metropolis algorithm (Gelman et al. 1995, Gilks et al. 1998). The Metropolis algorithm attempts to draw a sample from a Bayesian posterior distribution, and calculates the posterior density π, scaled by an unknown constant. The algorithm generates a 'chain' or sequence of values. Typically the beginning of the chain is discarded and every Nth element of the remainder is taken as the posterior sample. The chain is produced by taking an initial point x_0 and repeatedly applying the following rule, where x_i is the current point:

1. Draw a candidate step s from a proposal distribution J, which should be symmetric i.e., $J(-s)=J(s)$.

2. Calculate $r=\min(\pi(x_i+s)/\pi(x_i),1)$.

3. Let $x_{i+1}=x_i+s$ with probability r, or x_i with probability $1-r$.

An initial point estimate is produced before the chain starts, which is done so as to calculate the approximate covariance matrix of the free parameters (as the inverse Hessian), and may also be used as the starting point of the chain.

The user can specify the starting point of the point estimate minimiser using `casal -i`. Do not start it too close to the actual estimate (either by using `casal -i`, or by changing the parameter values in `population.csl`) as it takes a few iterations to form a reasonable approximation to the Hessian.

There are three options for the starting point of the Markov Chain:

1. Start from the point estimate.

2. Start from a random point near the point estimate (the point is generated from a multivariate normal distribution, centred on the point estimate, with covariance equal

to the inverse Hessian times a user-specified constant). This is done to prevent the chain from getting 'stuck' at the point estimate.)

3. Start from a point specified by the user with `casal -i`.

The chain moves in natural space, i.e., no transformations are applied to the free parameters. The default proposal distribution is a multivariate normal centred on the current point, with covariance matrix equal to a matrix based on the approximate covariance produced by the minimiser, times some `stepsize` factor. The following steps define the initial covariance matrix of the proposal distribution:

1. The covariance matrix is taken as the inverse of the approximate Hessian from the quasi-Newton minimiser.

2. The covariance matrix is modified so as to decrease all correlations greater than `max_cor` down to `max_cor`, and similarly to increase all correlations less than -`max_cor` up to -`max_cor` (the `max_cor` parameter defaults to 0.8). This should help to avoid getting 'stuck' in a lower-dimensional subspace.

3. The covariance matrix is then modified either by,

    i. `adjustment_method=covariance`: that if the variance of the *i*th parameter is nonzero and less than `min_diff` times the difference between the parameters' lower and upper bound, then the variance is changed, without changing the associated correlations, to *k*=`min_diff(upper_bound`$_i$–`lower_bound`$_i$). This is done by setting $\mathrm{Cov}(i,j)' = \mathrm{sqrt}(k)\mathrm{Cov}(i,j)/\mathrm{sd}(i)$ for $i \neq j$, and $\mathrm{var}(i)' = k$.

    ii. `adjustment_method=correlation`: that if the variance of the *i*th parameter is nonzero and less than `min_diff` times the difference between the parameters' lower and upper bound, then its variance is changed to `min_diff(upper_bound`$_i$–`lower_bound`$_i$). This differs from (i) above in that the effect of this option is that it also modifies the resulting correlations between the *i*th parameter and all other parameters.

    This allows a free parameter to move in the MCMC even if its variance is very small according to the inverse Hessian. In both cases, the `min_diff` parameter defaults to 0.0001.

4. The `stepsize` (a scalar factor applied to the covariance matrix to improve the acceptance probability) is chosen by the user. The default is $2.4d^{-0.5}$ where d is the number of free parameters, as recommended by Gelman et al. (1995), though experience has shown that this is often too high, leading to a very low acceptance rate.

The proposal distribution can also change adaptively during the chain, using two different mechanisms. Both are offered as means of improving the convergence properties of the chain. It is important to note that any adaptive behaviour must finish before the end of the burn-in period, i.e., the proposal distribution must be finalised before the kept portion of the chain starts (CASAL enforces this). The adaptive mechanisms are as follows:

1. You can request that the step size change adaptively at one or more sample numbers. At each adaptation, the step size is doubled if the acceptance rate since the last adaptation is more than 0.5, or halved if the acceptance rate is less than 0.2. (See Gelman et al. 1995 for justification.) The new step size is recorded in the `objectives` file.

2. You can request that the entire covariance matrix change adaptively at one or more sample numbers. At each adaptation, it is replaced with a matrix based on the sample covariance of an earlier section of the chain. The theory here is that the covariance of a portion of chain could potentially be a better estimate of the covariance of the posterior distribution than the inverse Hessian.

   The procedure used to choose the sample of points is as follows. First, all points on the chain so far are taken. All points in an initial user-specified period are discarded. The assumption is that the chain will have started moving during this period. If this is incorrect and the chain has still not moved by the end of this period, it is a fatal error and CASAL stops. The remaining set of points must contain at least some user-specified number of transitions. If this is incorrect and the chain has not moved this often, it is again a fatal error. If this test is passed, the set of points is systematically sub-sampled down to 1000 points (it must be at least this long to start with).

   The variance-covariance matrix of this sub-sample of chain is calculated. As above, correlations greater than `max_cor` are reduced to `max_cor`, correlations less than -`max_cor` are increased to -`max_cor`, and very small nonzero variances are increased (`covariance_adjustment` and `min_diff`). The result is the new variance-covariance matrix of the proposal distribution.

   The step size parameter is now on a completely different scale, and must also be reset. It is set to a user-specified value (which may or may not be the same as the initial step size). We recommend that some of the step size adaptations are set to occur after this, so that the step size can be readjusted to an appropriate value which gives good acceptance probabilities with the new matrix.

   All modified versions of the covariance matrix are printed to the standard output, but only the initial covariance matrix (inverse Hessian) is saved to the `objectives` file. (As a consequence, a lapsed chain cannot be continued using `-a` if adaptive covariance is used.) The number of covariance modifications by each iteration is recorded as a column on the `objectives` file.

The probability of acceptance for each jump is 0 if it would move out of the bounds, or 1 if it improves the posterior, or (new posterior/old posterior) otherwise.

You can specify how often the position of the chain is recorded using the `keep` parameter. For example, with `keep 10`, only every 10th sample is written to file.

You have the option to specify that some of the free parameters are fixed during MCMC. If the chain starts at the point estimate or at a random location, these fixed parameters are set to their values at the point estimate. If you specify the start of the chain using `-i`, these fixed parameters are set to the values in the file.

A multivariate *t* distribution is available as an alternative to the multivariate normal proposal distribution. If you request multivariate *t* proposals, you may want to change the degrees of freedom from the default of 4. As the degrees of freedom decrease, the *t* distribution becomes more heavy tailed. This may lead to better convergence properties.

Having produced one or more Markov chains and looked at the diagnostics, reload all the chain output files into CASAL and use them to generate a single posterior sample (using `-C`). At this stage, the first `burn_in` iterations for each chain are discarded (so, with `keep 10`, `burn_in 1000`, the first 1000 recorded samples are discarded for each chain). Unless a very large value of `keep` was originally chosen, it will be necessary to further reduce the size of the posterior sample (possibly down to several hundred) such that it can be analysed in a reasonable amount of time. This is done by sub-sampling. You specify the size of the sub-sample to be produced (or else no sub-sampling is done). You have the option to generate a systematic sub-sample (i.e., every *n*th point is kept) or a random sub-sample (the former is recommended except with prior re-weighting, when the latter must be used).

Given a posterior (sub)sample, CASAL can calculate a list of output quantities for each sample point (see Section 7.2). These quantities can be dumped into a file (using `casal -v`) and read into an external software package where the posterior distributions can be plotted and/or summarised.

The posterior sample can also be used for projections (Section 7.3) and stochastic yield calculations (Section 7.5). The advantage of this is that the parameter uncertainty, as expressed in your posterior distribution, can be included into the risk and yield estimates.

It is possible to investigate the results that would have been obtained if a different prior had been specified This is called prior re-weighting and is done by calculating the ratio of the new prior to the original prior for each point in the posterior sample, then using these ratios as probability weights when generating a random (not systematic) sub-sample with `casal -C`. Prior re-weighting is applicable only if the new prior is zero in every part of the parameter space for which the original prior was zero. Also, it is likely to be numerically unstable unless the new prior is very small in every part of the parameter space for which the original prior was very small.

6.6 Observations

The objective function is based on the goodness-of-fit of the model to your observations. In the current release of CASAL, most observations are different kinds of time series, i.e., data which were recorded for one or more years, in the same format each year. Examples of time series data types include relative abundance indices, commercial catch length frequencies, and survey numbers-at-age.

Generally, time series must relate to a specified time step, and a specified area if the model is spatial, and one or more years in which they were recorded. These are the exceptions; (a) catch-at data (see below) can be based on more than one fishery and hence can cover multiple areas or time steps, and (b) age-at-maturation data (see below) are not associated with a year, time step, or area.

6.6.1 Types of observations

Each time series of observations belongs to one of the following types:

1. *Abundance*: including survey biomass indices and CPUE. Can be absolute abundance or relative abundance. Can be expressed as biomass or numbers of fish.
2. *Catch-at*: including commercial catch proportions-at-age and proportions-at-size. Can be split by sex.

3. *Numbers-at*: including survey numbers-at-age and numbers-at-size. Can be absolute numbers or relative numbers. Can be split by sex.

4. *Proportions-at*: including survey proportions-at-age and proportions-at-size. Can be split by sex.

5. *Proportions mature*: i.e., data on the proportion of fish, by age or size class, which are mature. If using age frequency observations in an age-based model, say which age classes are included and whether the last age class is a plus group. For example, your partition might include ages from 1 to 20+ but you might have observations only for 2, 3, 4, 5, and 6+ aged fish. The same applies to size frequency observations in a size-based model.

6. *Proportions migrating*: i.e., data on the proportion of fish, by age or size class, which went on a particular migration. More specifically, for each age/size class, the number of migrating fish (in the source area of the migration, including all stocks) divided by the total number of fish (in the source area of the migration, including all stocks). Can be provided for either sex, or both combined.

7. *Age size*: i.e., observations of the ages and sizes of individual fish — primarily used to fit size-at-age parameters in age-based models. See below.

8. *Age-at-maturation*: a specialised observation type, used for modelling orange roughy and potentially other fish with similar characteristics. The age at which a mature orange roughy became mature can be estimated by examining the otolith after capture. The age-at-maturation data type allows CASAL to use this information for estimation of maturation parameters. See below.

9. *Selectivity-at*: an unusual observation type, and should be used only when you have some direct observations of the values of a selectivity ogive (to date we have only seen this in a single shellfish model).

10. *Tag-release*: observations on the sizes of tagged fish in tag-release event, and should only be used when tagging is part of the partition in an age-based model and if the tag release type is *free* (see Section 5.4.8)

11. *Tag-recapture*: observations on the numbers of tagged (and scanned) fish recaptured from a tag-release event, and should only be used when tagging is part of the partition.

For each time series, as well as the above, you need to provide CASAL the following information:

1. A label. (This should be unique, neither "`Bpre`" or "`Bpost`", and should not contain a full stop.)

2. The years in which they were observed.

3. For *catch-at* observations, the fishery or fisheries they cover.

4. For all observations except *catch-at* and *age-at-maturation*:
    - The area in which they were observed (in a multi-area model).

- After what proportion of the mortality in the time step they occurred. (This is a useful option as it allows insertion of an observation partway through a time step, which can sometimes avoid splitting the time step into two.).
- The name of the selectivity ogive (except for *tag-release* and some options for *tag-recapture* observations) which should be applied, if any (trawl survey data should use the selectivity of the research vessel; CPUE data could arguably use the selectivity of the commercial fleet).

5. If using age frequency observations in an age-based model, say which age classes are included and whether the last age class is a plus group. For example, you might have observations of 2, 3, 4, 5, and 6+ aged fish. The same applies to size frequency observations in a size-based model.

6. If using a set of size frequency observations in an age-based model, provide the size classes. (In a size-based model, the size bins used in the observations must be consecutive groupings of the bins defined in the population section of the model.)

7. For *catch-at* and *proportions-at* observations, specify whether the observations and fits for each year should sum to 1.

   If you say that proportions must sum to 1, CASAL calculates the expected values using a denominator constructed from the number of fish in the age or size classes included in the range specified in the observations.

   Otherwise, if you do not force the proportions to sum to 1, then the expected values are calculated using a denominator of the total number of fish in all age or size classes in the partition. This approach was intended for backwards compatibility with previous least-squares estimation software and may not always work in likelihood analysis.

   The setting `sum_to_one true` is the default and is recommended. If the observations don't sum to 1 (and this is determined for each year independently), a warning is printed and the observations (for that year) are rescaled to sum to 1. (Note the test for observations summing to 1 is implemented with a tolerance of ±0.01.)

8. For relative observations, you need to provide the label of the catchability coefficients q. Several time series can share the same q (see Section 6.7.1). You can also provide a curvature parameter b for non-linear relationships (see Section 6.7.1).

9. For *at-age* data, whether ageing error should be applied (Section 6.7.7). It may be that you have defined an ageing error but do not want to apply it to one or more of your *at-age* observations, perhaps because the ageing error is meant to be included in the likelihood.

10. For tag-release observations, the numbers released by size class.

11. For tag-recapture observations, the detection probability, and the numbers scanned and recaptured.

12. For likelihood or Bayesian analysis, provide the error distribution and its parameters (the error distributions are listed in Section 6.7.1). For variability parameters (c.v.s, standard deviations, and effective sample sizes N), there can either be one value for all years, or one value per year, or (for at-age or at-size data), one value per age/size class per year. Note that c.v.s are expressed as a proportion not a percentage. For

example, if you put a c.v. of 40 then you probably meant 0.4. See also specifying the process error (Section 6.7.2).

13. And of course, provide the observation values. Abundance values are straightforward — one number per year. At-age or at-size data are a bit more complicated. There is one row of numbers for each year, one column per age/size class. If the observations are sexed, then there are male columns followed by female columns, rather than separate male and female tables. Age-size and age-at- maturation data are input as several rows of data, one per variable: age, size and potentially sex for age-size, and age at capture, age at maturation and potentially sex for age-at-maturation

6.6.2 Age-size observations

Age-size data are observations of the ages and sizes of individual fish. They are primarily used to fit size-at-age parameters in age-based models.

Age-size data cannot currently be used in growth path models, and can never be used in size-based models. An error message will be issued if age-size observations are used without valid size-at-age parameters (i.e., no `@size_at_age_dist` or with `@size_at_age.cv`=0).

The data include a list of ages, a list of sizes, and (in a sexed model) a list of sexes, plus information on when, where, and how the observations were collected. So, the i^{th} elements of the lists contain the age, size, and sex of the i^{th} fish observed.

There are several possible sampling regimes, i.e., assumptions about how the observed fish were sampled from the general population of fish available at that time and place. The options are:

- `random`: fish were a simple random sample from the available population
- `random_at_sex`: fish were a simple random sample within each sex
- `random_at_size`: fish were a simple random sample within each size class
- `random_at_sex_and_size`: fish were a simple random sample within each size class of each sex
- `random_at_age`: fish were a simple random sample within each age class
- `random_at_sex_and_age`: fish were a simple random sample within each age class of each sex

We believe the `at_age` and `at_sex_and_age` options are quite unlikely to be true, yet they are widely applied in fisheries (for example, in the Coleraine (Hilborn et al. 2001) stock modelling software). Probably the `at_size` and `at_sex_and_size` options are most likely to hold for most NIWA finfish programmes.

In age-size data, there should be no observations for which the age is outside the age range defined for the partition (this will generate a fatal error message), nor any non-integer ages. Observations with ages below the minimum age in the partition should be removed. Observations where an age exceeds the maximum age in the partition could either be included in the plus group (i.e., with the observed age changed to that of the plus group) or removed, depending on the circumstances. If there is no plus group in the partition they should be removed. For `random_at_age` or `random_at_sex_and_age` samples they may be either included or removed in the plus group. For all other sample types they could be included in the plus group.

In addition, the user can specify a selectivity ogive which was applied in the sampling process, perhaps due to the sampling gear that was used, or the areal availability of fish at that place or time. The ogive can be age- or size-based: the choice has direct bearing on the likelihood of the observations.

Under some sampling regimes, a size-based selectivity has no effect on the likelihood and hence should not be used (since it adds computational time). This occurs when the character on which the selectivity acts was not randomly chosen in the sample: for instance, if 10 fish of each sex were chosen from each size class, then a size-based selectivity will have had no effect (except perhaps to make it easier/harder to find the 10 fish!). In general, a size-based selectivity has no effect under the `random_at_sex_and_size` sampling method, and it has no effect under the `random_at_size` method unless the selectivity is specified by sex. If you attempt to use a size-based selectivity in this situation, CASAL will issue a warning and will not apply the ogive. In cases where the size-based selectivity does have an effect, CASAL will issue a warning that the selectivity adds to the computational time, and will apply it as requested.

Similarly, under some sampling regimes, an age-based selectivity has no effect on the likelihood, and should not be used. This applies under the `random_at_sex_and_age` sampling method and under the `random_at_age` method unless the selectivity is specified by age. If you attempt to use an age-based selectivity in this situation, CASAL will issue a warning and not apply the ogive.

The user must additionally specify the year, time step, and proportion of mortality when the observations were collected, the area in which they were collected, and whether ageing error is to be applied (by default it is applied, if ageing error parameters are supplied in the input parameter files).

The user does not need to specify an error distribution or its parameters: instead, the choice of size-at-age distribution sampling regime, selectivity ogive, and ageing error determines the likelihood equation. The appropriate likelihood for a single observation, (a,l,s), depends on the nature of the sample.

With a `random` sample covering a single stock,

$$L=\mathrm{P}(a,l,s)=\left[\sum\nolimits_{a'} N_{a's} M_{a'a} f_{a's}(l)\right] \Big/ \left[\sum\nolimits_{a's'} N_{a's'}\right],$$

where $N_{a's}$ is the number of fish of true age a' and sex s (in the specified stock and area, and after the specified selectivity, if any, is applied), $M_{a'a}$ is the probability that a fish of true age a' is observed as age a, and $f_{a's}(l)$ is the probability density function describing the distribution of sizes for a given (true) age a' and sex s. When there is no ageing error the numerator of the above equation simplifies to $N_{as}f_{as}(l)$ and the denominator to $\sum\nolimits_{a's'} N_{a's'}$.

If there is a mixture of stocks in the area sampled, then the N and f terms are stock-dependent and the numerator and denominator are each summed over stocks

For all the other sample types the likelihood is a conditional probability, and is calculated as a fraction whose numerator is the same as for $\mathrm{P}(a,l,s)$ and with the denominator given in Table 3.

If the user specifies a selectivity for an age-size observation, this is easy to deal with if the selectivity is age-based. If $N'_{a's}$ is the number at true age a' and sex s before the selectivity is applied then $N_{a's}=N'_{a's}S_s(a')$, where S is the selectivity function.

It is more complicated with a size-based selectivity because of the need to distinguish between the distribution of size at age before [$f'_{a's}(l)$] and after [$f_{a's}(l)$] the selectivity is applied (note: it is the former which is defined by the model parameters). The appropriate equations are

$$f_{a's}(l) = S_s(l) f'_{a's}(l) \Big/ \int_{l'} S_s(l') f'_{a's}(l') dl',$$

and

$$N_{a's} = N'_{a's} \int_{l'} S_s(l') f'_{a's}(l') dl'$$

The integrals in these equations are calculated by discrete approximation (using 5 points), in the same way as size-based ogives are converted to age-based ogives.

Fits and residuals are not displayed for age-size data, instead CASAL shows the contribution to the objective function of each individual age-size pair.

Table 3: Age-size likelihoods for the different sample types.

Sample	Conditional probability	Denominator	
		With ageing error	Without
random	$L=\mathrm{P}(a,l,s)$	$\sum_{a's'} N_{a's}$	$\sum_{a's'} N_{a's}$
random_at_sex	$L=\mathrm{P}(a,l \mid s)$	$\sum_{a'} N_{a's}$	$\sum_{a'} N_{a's}$
random_at_age	$L=\mathrm{P}(l,s \mid a)$	$\sum_{a's'} N_{a's'} M_{a'a}$	$\sum_{s'} N_{as'}$
random_at_size	$L=\mathrm{P}(a,s \mid l)$	$\sum_{a's'} N_{a's'} f_{a's'}(l)$	$\sum_{a's'} N_{a's'} f_{a's'}(l)$
random_at_sex_and_age	$L=\mathrm{P}(l \mid a,s)$	$\sum_{a'} N_{a's} M_{a'a}$	N_{as}
random_at_sex_and_size	$L=\mathrm{P}(a \mid l,s)$	$\sum_{a'} N_{a's} f_{a's}(l)$	$\sum_{a'} N_{a's} f_{a's}(l)$

6.6.3 Age-at-maturation observations

This type of observation makes sense only for species like orange roughy, in which it is possible to tell (by examining a mature fish) the age it was when it matured. It is available only in an age-based model with maturity in the partition and no more than one maturation episode per year.

The use of this observation requires the important assumption that mortality is independent of maturity status (i.e., the mortality experienced in a given year by fish of a given age is independent of whether the fish is mature or not). This observation differs from most others in that it is not associated with any particular area, year, time step, or ogive.

Suppose we have a sample of n fish and let A_{sj} be the age at which the jth fish was sampled and A_{mj} its age at maturation (the age which it matured). If the fish is not mature we will signal this by setting A_{mj}=0. It is not necessary that this be a fully random sample, but it is necessary that, amongst fish of the same age, the probability of selection does not depend on the age at maturation. That is, the sample must be random, conditional on the age at sampling.

The likelihood that CASAL associates with these observations is the conditional likelihood P(A_{mj}|A_{sj}). (Note that if the sample were fully random it would be sensible to calculate the joint likelihood, P(A_{sj},A_{mj})=P(A_{mj}| A_{sj})P(A_{sj}). Users can, in effect, achieve this by also providing the A_{sj} as a `proportions_at` observation.)

Further assume that mortality is independent of maturity status. This will often not be strictly true. For example, it is false if migration to a fishing ground is dependent on maturity status. The assumption is necessary because without it, the calculation of the likelihood, though still possible in principle, would require fundamental structural changes to CASAL. CASAL does not check this assumption so it is up to the user to decide whether it is warranted.

For orange roughy, the age at maturation cannot be determined until, say, k years after maturation (because the maturation mark in the otolith is not clearly apparent until additional otolith material has been deposited — see Francis & Horn 1997). This means that there would be no observations with A_s–A_m < k. The user may specify a value k.

If there is no ageing error the likelihood is as,

$$\mathrm{P}\left(A_{mj}=a \mid A_{sj}=A\right)=\begin{cases}\dfrac{O_a U_{a-1}}{U_{A-1}+\sum_{a'=a_{\min}}^{A-1} O_{a'}U_{a'-1}} & a_{\min}\le a\le A-k \\ 0 & A-k<a<A \\ \dfrac{U_{A-1}+\sum_{a'=A-k}^{A-1} O_{a'}U_{a'-1}}{U_{A-1}+\sum_{a'=a_{\min}}^{A-1} O_{a'}U_{a'-1}} & a=0\end{cases}$$

where O_a is the proportion maturing at age a, $a_{\min}$ is the minimum age in the partition, $U_a=\prod_{a'=a_{\min}}^{a}\left(1-O_{a'}\right)$, and $U_{a_{\min}-1}=1$. Note that this is independent of the year in which the observation was made (which is not true if mortality depends on maturity status).

Ageing error can have a substantial effect on this type of observation (it makes the maturation ogive appear to be less steep than it really is) so it is important to be able to allow for it. However, it is not straightforward to allow for error in both A_{mj} and A_{sj} (in fact, if the sample is not fully random we are not sure if it is possible). Thus, we allow only for error in the A_{mj}; we do not allow for error either in the A_{sj} or in the detection of maturity. With ageing error, the adjusted likelihood is given by

$$\mathrm{P}'\left(A_{mj}=a \mid A_{sj}=A\right)=\begin{cases}\dfrac{\sum_{b=a_{\min}}^{A-k} M_{ba}P_{bA}}{\sum_{a'=a_{\min}}^{A-k}\sum_{b=a_{\min}}^{A-k} M_{ba'}P_{bA}} & a_{\min}\le a\le A-k \\ 0 & A-k<a<A \\ \dfrac{U_{A-1}+\sum_{a'=A-k}^{A-1} O_{a'}U_{a'-1}}{U_{A-1}+\sum_{a'=a_{\min}}^{A-1} O_{a'}U_{a'-1}} & a=0\end{cases}$$

where P_{bA}=P(A_{mj}=b| A_{sj}=A) and M_{ba} is the probability that a fish with true maturity age b is observed as having matured at age a (this is the ageing-error misclassification matrix — see Section 6.7.7).

If the calculated likelihood for an observation is equal to zero then CASAL replaces this with 10^{-6} to avoid errors from taking the logarithm of zero.

Fits and residuals are not displayed for age-at-maturation data; instead CASAL shows the contribution to the objective function of each individual age-at-maturation observation.

For these observations, the user must specify:

- age at maturation for each fish sampled,
- age at capture for each fish sampled,
- optionally, the sex of each fish sampled,
- k, the number of years after which maturity can be detected,
- whether ageing error is used.

Note that users can moderately improve the performance of CASAL in cases where there are observations on fish with an identical set of maturation ages, sampled ages, and sexes in sexed observations, by placing such observations immediately adjacent to each other in the `age_at_maturation` subcommands within the `estimation.csl` file. In this situation (i.e., when an observation on a fish is identical to the fish immediately preceding) CASAL simply copies the contribution of that observation to the likelihood rather than attempting to recalculate it.

6.6.4 Tag-release observations

When a tagging episode is defined in an age-based model, the user needs to specify the proportions-at-age of tagged fish. If only a size frequency of tagged fish was collected, then the proportions-at-age are not known and must either be estimated or calculated. CASAL provide two methods for determining the proportions-at-age, either *deterministic* or *free* (see Section 5.4.8 for detail). When the *free* method is used in an age-based model, the proportions-at-age can be estimated using a size frequency of tagged fish. The tag-release observations type is used to supply this size frequency

The contribution to the negative-log-likelihood depends on the closeness of the match between the observed size frequency and the fitted size frequency (which is obtained by applying the size-at-age distribution to the estimated age frequency of tagged fish).

Each tag-release observation corresponds to a single tagging episode as defined in the population section (Section 5.4.8). The observed size frequency must be provided (optionally by sex, but only if it is a partition character). The bounds on the size classes are also specified. The user also needs to specify a likelihood function, which can be any of those appropriate for proportions data summing to 1 (e.g., multinomial, Coleraine, Fournier).

Note that if the model includes tagging mortality, the tag-release likelihood is calculated before tagging mortality is applied.

6.6.5 Tag-recapture observations

This is the key observations class for tagging in CASAL. Tag data is primarily used to estimate the population abundance of fish. In some models, this estimation can only be made outside the model and the result is used as an estimate of abundance in the model. But in CASAL the tagging data can, alternatively, be fitted within the model.

Before adding a tag-recapture time series, you will need to define a tag-release event (Section 5.4.8) and possibly a tag-release observation (Section 6.6.4). Tagging events list the labels of the tags which are modelled, and define the events where fish are tagged (i.e., CASAL moves fish into the section of the partition corresponding to a specific tag).

The observations are divided into two parts: (i) the number of fish that were scanned, and (ii) the number of tags that were recaptured. Each can be specified by sex, or for both sexes combined. The precise content of the scanned and recaptured observations depends on the sampling method, and the available options are:

1. *age*: both scanned and recaptured are vectors containing numbers-at-age. Only available in an age-based model. The selectivity ogive is redundant and cannot be supplied.

2. *size*: both scanned and recaptured are vectors containing numbers-at-size. Can be used in either an age- or size-based model. The selectivity ogive is redundant and cannot be supplied.

3. *age_size*: recaptured fish are supplied as numbers-at-age, but scanned fish as numbers-at-size. Only available in an age-based model. A selectivity ogive that applies to the scanned fish would usually be supplied.

4. *growth*: recaptured fish are supplied as numbers-at-size only. This observation type is only available in a size-based mode, and when the numbers of scanned samples are not used. Instead, the expected size distribution of the released fish is compared with the size distribution of recaptured fish as a means of estimating growth.

When defining the tag-recapture time series, you also need to specify:

- the time step,
- the years (unlike a tag-release event, the tag-recapture observations can occur over several years),
- the probability that each scanned tagged fish is detected as tagged (may be less than 1 if the observers are not infallible). The expected number of tags detected is calculated by multiplying this number by the number of tagged fish in the sample,
- the area where the fish are recaptured (in a multi-area model),
- optionally the stock of the recaptured fish (in a multi-stock model),
- whether only mature fish are recaptured,
- optionally a selectivity ogive used in the recapture process (for some sample methods, so far *age* and *size*, this is unnecessary and cannot be supplied),
- the size classes if the observations are size-based in an age-based model,
- whether ageing error is to be used, if it is included in the model (but note that ageing error is redundant and ignored for some sampling options).

Unlike some observations classes, the tag-recapture observations must cover the same age classes as the partition if they are age-based in an age-based model, or the same size classes as the partition if they are size-based in a size-based model.

The tag-recapture likelihoods are specified below. Note that these likelihoods do not have any user-set precision parameters such as *N* or *c.v.* (though there are user-specified robustification and dispersion parameters available). Note that factorials are calculated using the log-gamma function, to allow for non-integer arguments where necessary (and avoid overflow errors).

***age*: recaptures by age and scanned by age**

Designed for situations where the age frequencies of the recaptured tagged fish and of the scanned fish are known (only available in an age based model).

Here we define the likelihood as a binomial, i.e.,

$$-\log(L) = -\sum_{i,n_i>0} \left[\log(n_i!) - \log((n_i - m)!) - \log(m_i!) + m_i \log\left(Z\left(\frac{M_i}{N_i}, r \right) \right) \right.$$
$$\left. + (n_i - m_i) \log\left(Z\left(1 - \frac{M_i}{N_i}, r \right) \right) \right]$$

where n_i = number of fish at age i that were scanned
m_i = number of fish at age i that were recaptured
N_i = number of fish at age i in the available population (tagged and untagged)
M_i = number of fish at age i in the available population that have the tag.

where $Z(x,r)$ is a robustifying function with parameter $r>0$ (to prevent division by zero errors), and is defined as

$$Z(x,r) = \begin{cases} x \text{ where } x \geq r \\ r/(2 - x/r) \text{ otherwise} \end{cases}$$

***size*: recaptures by length and scanned by length**

Designed for situations where the size frequencies of the recaptured tagged fish and of the scanned fish are known. Available in both age or size based models.

Here we define the likelihood as a binomial, but based on sizes, rather than ages,

$$-\log(L) = -\sum_{i,n_i>0} \left[\log(n_i!) - \log((n_i - m)!) - \log(m_i!) + m_i \log\left(Z\left(\frac{M_i}{N_i}, r \right) \right) \right.$$
$$\left. + (n_i - m_i) \log\left(Z\left(1 - \frac{M_i}{N_i}, r \right) \right) \right]$$

where n_i = number of fish at size i that were scanned
m_i = number of fish at size i that were recaptured
N_i = number of fish at size i in the available population (tagged and untagged)
M_i = number of fish at size i in the available population that have the tag.

where $Z(x,r)$ is a robustifying function with parameter $r>0$ (to prevent division by zero errors), defined as

$$Z(x,r) = \begin{cases} x \text{ where } x \geq r \\ r/(2 - x/r) \text{ otherwise} \end{cases}$$

In an age-based model the number of fish at length in the population are determined from the population state and the appropriate growth parameters.

age-size: recaptures by age and scanned by size

Designed for situations where the age frequency of recaptured tagged fish is known, but only the size frequency of scanned fish is known. Only available in an age based model. Uses a selectivity if provided (the default is "none").

The likelihood is quite complex and is calculated as two components. The age frequency of scanned fish is calculated as proportional to the product of the selectivity-at-age and the age distribution of available fish in the population. The first component is the likelihood of the observed age frequency of recaptured tags, given the age frequency of scanned fish calculated above. The second component is the likelihood of the observed size frequency of scanned fish, again given the age frequency calculated above.

1. Likelihood for the age frequency of recaptured tagged fish. This is calculated as for the "age" sample type earlier,

$$-\log(L) = -\sum_{i,n_i>0}\Bigg[\log(n_i!) - \log((n_i - m)!) - \log(m_i!) + m_i \log\left(Z\left(\frac{M_i}{N_i}, r\right)\right) \\ + (n_i - m_i)\log\left(Z\left(1 - \frac{M_i}{N_i}, r\right)\right)\Bigg]$$

where n_i = number of fish at age *i* that were scanned
m_i = number of fish at age *i* that were recaptured
N_i = number of fish at age *i* in the total population (tagged plus untagged)
M_i = number of fish at age *i* in the total population that have tags

where $Z(x,r)$ is a robustifying function with parameter $r>0$ (to prevent division by zero errors), defined as

$$Z(x,r) = \begin{cases} x \text{ where } x \ge r \\ r/(2 - x/r) \text{ otherwise} \end{cases}$$

The values of n_i are not provided by the user, but instead are calculated as the product of the proportions-at-age of available fish (tagged or untagged) by the selectivity, scaled to sum to the total number of fish scanned.

2. Likelihood for the size frequency of scanned fish. This is a multinomial likelihood, i.e.,

$$-\log(L) = -\log(N!) + \sum_i \left[\log((NO_i)!) - NO_i \log(Z(E_i, r))\right]$$

where N = the number of scanned fish in that year
O_i = the observed proportions from scanned_[year]
E_i = expected proportions calculated by converting the n_i above to proportions-at-size using the size-at-age distribution.

Where $Z(x,r)$ is the robustifying function defined above.

growth: recaptures by size, with no information on the number of fish scanned

Designed for situations where you intend to estimate growth using the size frequency of recaptured fish, without using any information on scanned fish. This is only available in a size-based model.

The likelihood is a simple multinomial of the proportions observed at size, given the expected proportions at size in the tagged population, with the sample size determined by the number of individuals recaptured, i.e.,

$$-\log(L) = -\log(N!) + \sum_i \left[\log((NO_i)!) - NO_i \log(Z(E_i, r)) \right]$$

where N = the number of recaptured fish in that year
O_i = proportion of fish at size *i* that were recaptured
E_i = proportion of fish at size *i* in the tagged population

and $Z(x,r)$ is the robustifying function defined above.

Note that if you are also applying fishing or natural mortality to the population between the time of release and recapture, then the number of fish at size may be biased (although this may depend on the various mortality or fishing selectivities applied). A work-around is to define the tagged fish as being from a separate stock in a separate area (i.e., where they are not subject to fishing mortality, and the natural mortality is either zero or applied as a constant rate over all size classes).

6.7 The objective function

In *maximum likelihood* estimation, the objective function is a negative log-likelihood,

$$\text{Objective }(\mathbf{p}) = -\sum_i \log\left[L(\mathbf{p} \mid O_i) \right]$$

where **p** is a vector of the free parameters, L the likelihood function and O_i the *i*th observation.

In *Bayesian* estimation, the objective function is a negative log-posterior,

$$\text{Objective }(\mathbf{p}) = -\sum_i \log\left[L(\mathbf{p} \mid O_i) \right] - \log\left[\pi(\mathbf{p}) \right]$$

where π is the joint prior density of the parameters **p**.

Under either estimation method, penalties can be added to the objective function (see Section 6.7.6). You will usually want to use penalties to ensure that the exploitation rate constraints on the fisheries are not breached (otherwise there is nothing to prevent the model from having abundances so low that the recorded catches could not have been taken). A penalty to force the YCS to average to 1 (i.e., to have mean 1) may also be necessary.

6.7.1 Likelihoods

CASAL has five different kinds of likelihoods:

1. Those used for proportions data, where the proportions should sum to 1 over the columns of the partition. This includes commercial and survey proportions-at-size and proportions-at-age data, but not proportions mature or proportions migrating.

2. Those used for proportions data, where the proportion can be between 0 and 1 in each cell, and need not sum to 1 over the columns of the partition. This includes selectivity at, proportions mature, and proportions migrating.

3. Those used for absolute index data. These likelihoods are used for absolute abundance and can also be used for selectivity at, proportions mature, and proportions migrating.

4. Those used for relative index data, including relative abundance and relative survey numbers-at-age.

5. Specialized likelihoods used for age-size data, age-at-maturation data, or tag-release/recapture events. See Section 6.6 for details about the specialized likelihoods.

See also Section 6.7.2 for detail about process error.

Likelihoods for proportions data (with proportions summing to 1 across columns of the partition)

These likelihoods are used for commercial and survey proportions-at-size and proportions-at-age data, but not proportions mature or proportions migrating. They apply to data which are distributed across columns of the partition, typically summing to 1 across columns.”

Let ***O*** be the observations for a single year in a proportions time series, expressed as a vector of *n* proportions summing to 1; let ***E*** be the corresponding fitted values; let *N* be the “effective sample size” parameter. Then you can use the following likelihoods, which are expressed on the objective-function scale of −log(*L*):

1. Multinomial

$$-\log(L)=-\log(N!)+\sum_i\left[\log\left((NO_i)!\right)-NO_i\log\left(Z(E_i,r)\right)\right]$$

   where *Z*(*x*,*r*) is a robustifying function with parameter *r*>0 (to prevent division by zero errors), defined as

$$Z(x,r)=\begin{cases}x \text{ where } x\geq r\\ r/(2-x/r) \text{ otherwise}\end{cases}$$

2. Fournier

$$-\log(L)=0.5\sum_i\log(E_i')-\sum_i\log\left[\exp\left(\frac{-(O_i-E_i)^2}{2E_i'/N'}\right)+0.01\right]$$

where $E_i' = (1-E_i)E_i + 0.1/n$ and $N' = \min(N, 1000)$.

This is a robustified multivariate normal (it would be the usual multivariate normal with $\sigma^2=(1-E_i)E_i/N$ if the 0.1/n and 0.01 terms are omitted and if $N' = N$). See Fournier et al. (1990).

3. Coleraine. As per Fournier above, but replace E_i' with $O_i' = (1-O_i)O_i + 0.1/n$. A recent reference is Starr et al. (1999).

4. (Robustified) lognormal

$$-\log(L) = \sum_{i=1}^{n}\left(\log(\sigma_i) - \log\left(\exp\left(-0.5\left(\frac{\log(O_i/E_i)}{\sigma_i} + 0.5\sigma_i\right)^2\right) + r\right)\right)$$

where $\sigma_i = \sqrt{\log\left(1+c_i^{\,2}\right)}$, the c_i's are c.v.s by age/size class, and r is a robustifying constant.

The robustification term r is intended to reduce the influence of outliers, in the same way as the robustified Fournier likelihoods above. We recommend r=0.01, though smaller values (0.001, 0.0001, …) could be used for a lesser robustifying effect.

Likelihoods for proportions data (with proportions between 0 and 1 in each cell)

These likelihoods are used for proportions mature and proportions migrating data. They apply to data which can be between 0 and 1 in each cell, not necessarily summing to 1 across columns.

So far there is just one likelihood in this category, the binomial, but this is implemented in two forms; the standard binomial (termed “binomial”), and the normal approximation to the binomial (“binomial-approx”).

Let $\boldsymbol{O_i}$ be the observations for a single year in a time series, expressed as a vector of n proportions between 0 and 1; let $\boldsymbol{E_i}$ be the corresponding fitted values; let $\boldsymbol{N_i}$ be the “effective sample size” parameter for each observed proportion. A single effective sample size may be specified for all observations in a given year, or different sample sizes may be specified for each observation in the year. Then the binomial likelihood, which is expressed on the objective-function scale of −log(L), is:

$$\begin{aligned}-\log(L) = -\sum_i \Big[&\log(N_i!) - \log\big((N_i(1-O_i))!\big) - \log\big((N_iO_i)!\big) + N_iO_i\log\big(Z(E_i,r)\big)\\ &+N_i(1-O_i)\log\big(Z(1-E_i,r)\big)\Big]\end{aligned}$$

where $Z(x,r)$ is a robustifying function with parameter r>0 (to prevent division by zero errors), defined as

$$Z(x,r) = \begin{cases} x \text{ where } x \geq r \\ r/(2-x/r) \text{ otherwise}\end{cases}$$

And the binomial-approx likelihood is:

$$-\log(L)=\log\left(\sqrt{(E_i+r)(1-E_i+r)/N_i}\right)+\frac{1}{2}\sum_i\left(\frac{O_i-E_i}{\sqrt{(E_i+r)(1-E_i+r)/N_i}}\right)^2$$

Here *r* is a non-negative robustifying constant (a nonzero value of *r* is recommended if only to prevent division by zero errors).

This likelihood corresponds to the assumption that, for each observation in each year, a simple random sample of size N_i was taken from the partition and that the observation value was the proportion of fish in the sample who migrated/matured/etc.

Likelihoods for absolute index data

Let ***O*** be the observations for a single year in an time series of absolute indices, expressed as a vector of *n* elements (with *n*=1 for abundance indices, *n*>1 for proportions mature or proportions migrating); let ***E*** be the corresponding fitted values; express the variability of each observation O_i in terms of its c.v. c_i (or in one case, its standard deviation s_i). Then you can use the following likelihoods, which are expressed on the objective-function scale of −log(*L*):

1. Normal

   $$-\log(L)=\sum_{i=1}^{n}\left(\log(c_iE_i)+0.5\left(\frac{O_i-E_i}{c_iE_i}\right)^2\right)$$

   This reflects the distributional assumption that O_i has the normal distribution, with mean E_i and c.v. c_i.

2. Normal parameterised by standard deviation rather than c.v.

   $$-\log(L)=\sum_{i=1}^{n}\left(\log(s_i)+0.5\left(\frac{O_i-E_i}{s_i}\right)^2\right)$$

   This reflects the distributional assumption that O_i has the normal distribution, with mean E_i and standard deviation s_i.

3. Lognormal:

   $$-\log(L)=\sum_{i=1}^{n}\left(\log(\sigma_i)+0.5\left(\frac{\log(O_i/E_i)}{\sigma_i}+0.5\sigma_i\right)^2\right)$$

   where $\sigma_i=\sqrt{\log\left(1+c_i^2\right)}$.

   This reflects the distributional assumption that O_i has the lognormal distribution, that the mean of O_i is E_i and the c.v. of O_i is c_i.

4. Normal-log

$$-\log(L)=\sum_{i=1}^{n}\left(\log(\sigma_i)+0.5\left(\frac{\log(O_i/E_i)}{\sigma_i}\right)^2\right)$$

where $\sigma_i=\sqrt{\log\left(1+c_i^{\,2}\right)}$.

This reflects the distributional assumption that log(O_i) has the normal distribution, that the mean of log(O_i) is log(E_i) and the c.v. of O_i is c_i.

We make the distinction between lognormal and normal-log because they represent subtly different assumptions. With the lognormal, *O* has mean *E* and hence the mean of log(*O*) is less than log(*E*): whereas with the normal-log, log(*O*) has mean log(*E*) and hence the mean of *O* is more than *E*.

Catchability coefficients *q*; likelihoods for relative index data

The log-likelihoods of relative observations depend on the error distribution and the way in which *q*'s are treated in the model. There are two approaches to modelling *q*'s:

1. The *q*'s can be treated as 'nuisance' parameters. For each set of values of the free parameters, the model uses the values of the *q*'s which minimise the objective function. These optimal *q*'s are calculated algebraically (see Section 6.7.3). If one of the *q*'s falls outside the bounds specified by the user, it is set equal to the closest bound. This approach reduces the size of the parameter vector and hence should improve the performance of the estimation method. It is the default in CASAL. However, it is not correct when calculating a sample from the posterior in a Bayesian analysis (except asymptotically, see Walters & Ludwig 1994) and we offer the following alternative;

2. The *q*'s can be treated as ordinary free parameters.

For both options, we it is necessary to evaluate the contribution of ***O*** to the negative log-likelihood for a given value of *q*. Let ***O*** be the observations for a single year in a time series of relative indices, expressed as a vector of *n* elements (with *n*=1 for relative abundance indices, *n*>1 for relative numbers-at-age or at-size). Let ***E*** be the corresponding fitted values and *q* the catchability coefficients for the time series. It is possible for two or more time series collected in similar ways to use the same *q*. Each observation O_i varies about qE_i — express the variability of O_i in terms of its c.v. c_i (or in one case, its standard deviation s_i). Here are the likelihoods, which are expressed on the objective-function scale of −log(*L*):

1. Normal

$$-\log(L)=\sum_{i=1}^{n}\left(\log(c_iqE_i)+0.5\left(\frac{O_i-qE_i}{c_iqE_i}\right)^2\right)$$

This reflects the distributional assumption that O_i has the normal distribution, with mean qE_i and c.v. c_i.

2. Lognormal

$$-\log(L)=\sum_{i=1}^{n}\left(\log(\sigma_i)+0.5\left(\frac{\log(O_i/qE_i)}{\sigma_i}+0.5\sigma_i\right)^2\right)$$

where $\sigma_i=\sqrt{\log\left(1+c_i^{\,2}\right)}$.

This reflects the distributional assumptions that O_i has the lognormal distribution, that the mean of O_i is qE_i and the c.v. of O_i is c_i.

3. Normal-log

$$-\log(L)=\sum_{i=1}^{n}\left(\log(\sigma_i)+0.5\left(\frac{\log(O_i/qE_i)}{\sigma_i}\right)^2\right)$$

where $\sigma_i=\sqrt{\log\left(1+c_i^{\,2}\right)}$.

This reflects the distributional assumption that $\log(O_i)$ has the normal distribution, that the mean of $\log(O_i)$ is $\log(qE_i)$ and the c.v. of O_i is c_i.

4. Robustified lognormal

$$-\log(L)=\sum_{i=1}^{n}\left(\log(\sigma_i)-\log\left(\exp\left(-0.5\left(\frac{\log(O_i/qE_i)}{\sigma_i}+0.5\sigma_i\right)^2\right)+r\right)\right)$$

where $\sigma_i=\sqrt{\log\left(1+c_i^{\,2}\right)}$ and r is a robustifying constant.

This modification to the lognormal is intended to reduce the influence of outliers, and is analogous to the robustified normal distributions of Fournier et al. (1990). We recommend r=0.01, though smaller values (0.001, 0.0001, …) could be used for a lesser robustifying effect. We believe that this likelihood may be most appropriate for relative numbers-at-age or at-size but not for relative abundance indices (its effects are equivalent to dropping observations with large normalised residuals, which may be undesirable for abundance data).

Optionally, a curvature parameter, b, can be used for relative-abundance or relative numbers at observations. This is intended for modelling hyper-depletion or hyper-stability in CPUE (see Harley et al. 2001). In this case the expected value of O_i is assumed to be $qE_i^{1/b}$. [In earlier versions of CASAL this expected value was assumed to be $q(E_i/\max(E))^{1/b}$, which reduces correlation between q and b. However, this had the unforeseen disadvantage of modifying the interpretation of the prior on q in Bayesian estimation. This option is still available (see command `@q_type`) but is not recommended.] Also note that if you wish to *estimate* the curvature parameter, then the q's cannot be treated as 'nuisance' parameters — you will need to set `@q_method=free`.

6.7.2 Process error

In a likelihood-based or Bayesian analysis, you can specify a 'process error' for each set of observations. This has the effect of increasing the error in the data (by increasing c.v.s or standard deviations, or decreasing effective sample sizes), and hence of decreasing the weight given to the data in the fitting process.

For data where the likelihood is parameterised by the c.v., you can specify the process error for a given set of observations as a c.v., in which case all the c.v.s c_i are changed to

$$c_i' = \sqrt{c_i^2 + c_{process\_error}^2} \,.$$

Similarly, if the likelihood is parameterised by the standard deviation,

$$\sigma_i' = \sqrt{\sigma_i^2 + \sigma_{process\_error}^2} \,,$$

and by the effective sample size,

$$N_i' = \frac{1}{1/N_i + 1/N_{process\_error}} \,.$$

In all three cases, the process error has more effect on small errors than on large ones. Be clear that a large $N_{process\_error}$ means a small process error.

In tag-recapture observations, you can specify the process error through the use of a dispersion parameter *ø*. Here, the tag-recapture log-likelihood is modified by multiplying by 1/*ø*. Note that values of *ø*<1 are allowed, but such values would result in an *increase* in the effective sample size and a resulting increase of the weight given to the tag-recapture observations in the fitting process. This is probably not sensible in most applications.

CASAL allows you to estimate process error, although whether this is advisable is another matter. To make several sets of observations share the same process error, use the `same` subcommand in the `estimate` block (see Section 6.2).

6.7.3 Calculating nuisance *q*'s

This section describes the equations used to calculate nuisance catchability coefficients *q*'s (see Section 6.7.1). From the user's point of view, the essence is that you can use nuisance *q*'s in the following situations:

1. With maximum likelihood.

2. With Bayesian estimation, providing that your prior on the *q* is one of the following:
   - Uniform
   - Uniform-log
   - Lognormal with observations distributed lognormal, robustified lognormal, or normal-log.

The equations used for calculating nuisance *q*'s in maximum likelihood or Bayesian analysis are indexed in Table 4.

Table 4: Equations used to calculate nuisance *q*'s. (*=no analytic solution found.)

Distribution of observations	Maximum likelihood	Bayesian with specified prior on q			
		Uniform	Uniform-log	Normal	Lognormal
Normal	(1)	(1)	(4)	*	*
Lognormal	(2)	(2)	(5)	*	(6)
Normal-log	(3)	(3)	(7)	*	(8)

Note that q's are calculated for robustified lognormal likelihoods as if they were ordinary lognormal likelihoods.

The equations and their derivations follow. Let $\sigma_i = \sqrt{\log\left(1+c_i^2\right)}$ throughout, and let n be the number of observations in the time series. The case of multiple time series sharing the same q, and the modifications required for the assumption of curvature, are addressed at the end of this subsection.

First, consider maximum likelihood estimation. When the (O_i) are assumed to be normally distributed,

$$-\log(L) = \sum_i \log\left(c_i q_i E_i\right) + 0.5\sum_i \left(\frac{O_i - qE_i}{c_i q E_i}\right)^2$$

The value of q which minimises the objective function is found by solving $\partial/\partial q\left(-\log(L)\right) = 0$.

$$\frac{\partial}{\partial q}\left(-\log(L)\right) = \frac{n}{q} + \frac{1}{q^2}\sum_i \frac{O_i}{c_i^2 E_i} - \frac{1}{q^3}\sum_i \left(\frac{O_i}{c_i E_i}\right)^2$$

hence

$$\hat{q} = \frac{-S_1 + \sqrt{S_1^2 + 4nS_2}}{2n} \tag{1}$$

where $S_1 = \sum_i \left(O_i / c_i^2 E_i\right)$ and $S_2 = \sum_i \left(O_i / c_i E_i\right)^2$.

When the (O_i) are assumed to be lognormally distributed,

$$-\log(L) = \sum_i \log\left(\sigma_i\right) + 0.5\sum_i \left(\frac{\log\left(O_i\right) - \log\left(qE_i\right) + 0.5\sigma_i^2}{\sigma_i}\right)^2,$$

$$\frac{\partial}{\partial q}\left(-\log(L)\right) = \frac{-1}{q}\sum_i \left(\frac{\log\left(O_i/E_i\right) - \log(q) + 0.5\sigma_i^2}{\sigma_i^2}\right),$$

$$\hat{q} = \exp\left(\frac{0.5n + S_3}{S_4}\right) \tag{2}$$

where $S_3 = \sum_i \left(\log\left(O_i / E_i \right) / \sigma_i^2 \right)$ and $S_4 = \sum_i \left(1/\sigma_i^2 \right)$.

When the (O_i) are assumed to be distributed normal-log, the equations are similar,

$$-\log(L) = \sum_i \log(\sigma_i) + 0.5 \sum_i \left(\frac{\log(O_i) - \log(qE_i)}{\sigma_i} \right)^2,$$

$$\frac{\partial}{\partial q}\left(-\log(L)\right) = \frac{-1}{q} \sum_i \left(\frac{\log(O_i/E_i) - \log(q)}{\sigma_i^2} \right),$$

$$\hat{q} = \exp\left(\frac{S_3}{S_4} \right) \tag{3}$$

Next consider Bayesian estimation, where we must also specify a prior for q.

The effects of the prior on the equations are to replace likelihood L by posterior P throughout, to add $-\log(\pi(q))$ to the equation for $-\log(P)$ and $\partial / \partial q \left(-\log\left(\pi(q) \right) \right)$ to the equation for $\partial / \partial q \left(-\log(P) \right)$.

This last term is 0 for a uniform prior on q, $1/q$ for a log-uniform prior, $\frac{q - \mu_q}{\left(\mu_q c_q \right)^2}$ for a normal prior, and $\frac{1}{q}\left(1.5 + \frac{\log(q) - \log(\mu_q)}{\sigma_q^2} \right)$ for a lognormal prior,

where μ_q and c_q are the mean and c.v. of the prior on q and $\sigma_q = \sqrt{\log\left(1 + c_q^2 \right)}$. Clearly, if the prior is uniform, the equation for $\hat{q}$ is the same as for maximum likelihood estimation.

When the (O_i) are assumed to be normally distributed and the prior is log-uniform, equation (1) becomes

$$\hat{q} = \frac{-S_1 + \sqrt{S_1^2 + 4(n+1)S_2}}{2(n+1)} \tag{4}$$

but we cannot solve for $\hat{q}$ with either a normal or lognormal prior.

When the O_i are assumed to be lognormally distributed and the prior is log-uniform, equation (2) becomes

$$\hat{q} = \exp\left(\frac{0.5n - 1 + S_3}{S_4} \right) \tag{5}$$

and if the prior is lognormal,

$$\hat{q}=\exp\left(\frac{0.5n-1.5+\log\left(\mu_q\right)/\sigma_q^2+S_3}{S_4+1/\sigma_q^2}\right), \tag{6}$$

but it is not possible to solve for $\hat{q}$ with a normal prior.

When the (O_i) are assumed to be distributed normal-log and the prior is log-uniform,

$$\hat{q}=\exp\left(\frac{S_3-1}{S_4}\right) \tag{7}$$

and if the prior is lognormal,

$$\hat{q}=\exp\left(\frac{-1.5+\log\left(\mu_q\right)/\sigma_q^2+S_3}{S_4+1/\sigma_q^2}\right) \tag{8}$$

but again it is not possible to solve for $\hat{q}$ with a normal prior.

The above equations have been written for a single time series (O_i) and no curvature assumption. Suppose now that there are m time series, all with the same q, and all with the same error distribution. This has little effect on the above equations. Simply extend the summations in S_1, S_2, S_3, and S_4 over all observations in the m time series, and let $n=\Sigma_j n_j$. To adjust the equations to allow for curvature we replace E_i by $E_i^{1/b}$ or, if the original curvature assumption is used (see Section 6.7.1), by $(E_i/\max(E))^{1/b}$.

6.7.4 Priors

In a Bayesian analysis, you need to give a prior for every free parameter. There are no defaults.

Note that when some of these priors are parameterised in terms of mean, c.v., and standard deviation, these refer to the parameters of the distribution before bounds are applied. The moments of the prior after the bounds are applied may differ.

For a single scalar parameter p, you can choose between the following priors (expressed in terms of their contribution to the objective function):

1. Uniform,

   $$-\log\left(\pi\left(p\right)\right)=0\,.$$

2. Uniform-log (i.e., log(p) ~ uniform),

   $$-\log\left(\pi\left(p\right)\right)=\log\left(p\right).$$

3. Normal with mean μ and c.v. c,

$$-\log\left(\pi\left(p\right)\right)=0.5\left(\frac{p-\mu}{c\mu}\right)^2.$$

4. Normal with mean μ and standard deviation σ,

$$-\log\left(\pi\left(p\right)\right)=0.5\left(\frac{p-\mu}{\sigma}\right)^2.$$

5. Lognormal with mean μ and c.v. c. $s=\sqrt{\log\left(1+c^2\right)}$, is the standard deviation of log(p).

$$-\log\left(\pi\left(p\right)\right)=\log\left(p\right)+0.5\left(\frac{\log\left(p/\mu\right)}{s}+\frac{s}{2}\right)^2.$$

6. Normal-log with log(p) having mean m and standard deviation s,

$$-\log\left(\pi\left(p\right)\right)=\log\left(p\right)+0.5\left(\frac{\log\left(p\right)-m}{s}\right)^2.$$

7. Beta with mean μ and standard deviation σ, and range parameters A and B.

$$-\log\left(\pi\left(p\right)\right)=\left(1-m\right)\log\left(p-A\right)+\left(1-n\right)\log\left(B-p\right),$$

where $\nu=\frac{\mu-A}{B-A}$, and $\tau=\frac{\left(\mu-A\right)\left(B-\mu\right)}{\sigma^2}-1$

and then $m=\tau\nu$, and $n=\tau\left(1-\nu\right)$. Note that the beta prior is undefined when $\tau\le 0$.

Vectors of parameters can be independently (but not necessarily identically) distributed according to any of the above forms, in which case the joint negative-log-prior for the vector is the sum of the negative-log-priors of the components. Values of each parameter need to be specified for each element of the vector.

In addition, for a vector **p** of n identically distributed parameters (for example, YCS) the following priors are allowed:

1. Multivariate normal from a stationary AR(1) process with parameters

μ=E(p_i), σ=sqrt(Var(p_i)), and ρ=Cor(p_i, p_{i+1}),

$$-\log\left(\pi\left(\mathbf{p}\right)\right)=\frac{\left(p_1-\mu\right)^2}{2\sigma^2}+\frac{\sum_{i=2}^{n}\left(p_i-\rho p_{i-1}-\mu\left(1-\rho\right)\right)^2}{2\sigma^2\left(1-\rho^2\right)}+n\log\left(\sigma\right)+0.5\left(n-1\right)\log\left(1-\rho^2\right)$$

In other words, there are $(n-1)$ i.i.d. normal variates z_i with mean μ and variance σ^2, such that $p_i = \rho p_{i-1} + \left(\sqrt{1-\rho^2}\right) z_i$.

If ρ=0, then the p_i's are i.i.d. normal.

2. Multivariate normal-log, where log(**p**) forms a stationary AR(1) process as per 1. above, with parameters

   m=E(log(p_i)), s=sqrt(Var(log(p_i))), and r=Cor(log(p_i), log(p_{i+1})),

$$-\log\left(\pi(\mathbf{p})\right) = \frac{\left(\log(p_1)-m\right)^2}{2s^2} + \frac{\sum_{i=2}^{n}\left(\log(p_i) - r\log(p_{i-1}) - m(1-r)\right)^2}{2s^2\left(1-r^2\right)}$$
$$+ n\log(s) + 0.5(n-1)\log\left(1-r^2\right) + \sum_i p_i.$$

3. Multivariate normal-log with mean 1, where E(p_i)=1 and log(**p**) forms a stationary AR(1) process as for the multivariate normal above, with parameters

   s=sqrt(Var(log(p_i))) and r=Cor(log(p_i), log(p_{i+1})),

$$-\log\left(\pi(\mathbf{p})\right) = \frac{\left(\log(p_1)+0.5s^2\right)^2}{2s^2} + \frac{\sum_{i=2}^{n}\left(\log(p_i) - r\log(p_{i-1}) + 0.5s^2(1-r)\right)^2}{2s^2\left(1-r^2\right)}$$
$$+ n\log(s) + 0.5(n-1)\log\left(1-r^2\right) + \sum_i p_i.$$

   (i.e., $m=-0.5s^2$)

6.7.5 Priors on ratios of *q*'s

Sometimes it will be useful to be able to specify a prior distribution on the ratio of two catchabilities (q's). For example, suppose relative abundance observations are available from two different survey vessels, and it is believed that the fishing powers of the two vessels are unlikely to differ by more than 30%. In this case we would like to specify a prior distribution for the ratio of the two catchabilities, q_B/q_A, with mean 1 and c.v. about 0.15. CASAL allows this, but (because the method of specifying priors is associated with individual parameters, rather than functions of several parameters) implements it using penalty functions.

In this simple case, involving just two q's, the user would (a) provide whatever prior they like for q_A, (b) specify a uniform prior for q_B (this is required), and (c) specify a `ratio_qs_penalty` for q_B/q_A with mu = 1 and cv = 0.15. Currently, the prior on q_B/q_A must be lognormal. The mathematics underlying this simple case are first described, followed by discussion on how to deal with more complicated cases involving three or more q's.

Case 1: Two *q*'s

Suppose we want to specify two priors – f_A on q_A, and f_R on the ratio $R = q_B/q_A$ – and assume that these priors are independent. The aim is to express the joint prior for q_A and q_B in terms of these two distributions. This can be done as follows.

$$P(q_A = x_1, q_B = x_2) \;\; = P(q_A = x_1) \,.\, P(q_B = x_2 \mid q_A = x_1)$$

$$= \mathrm{P}(q_A = x_1)\ .\ |\mathrm{g}'(x_2)|\ .\ \mathrm{P}(R = \mathrm{g}(x_2) \mid q_A = x_1)$$

where $\mathrm{g}(q_B) = r = q_B/q_A$

so, at $q_A = x_1$, $\mathrm{g}(x_2) = x_2/x_1$ and $\mathrm{g}'(x_2) = 1/x_1$

So $\mathrm{P}(q_A = x_1, q_B = x_2) = \mathrm{P}(q_A = x_1)\ .\ (1/x_1)\ .\ \mathrm{P}(R = x_2/x_1 \mid q_A = x_1)$
$= \mathrm{P}(q_A = x_1)\ .\ (1/x_1)\ .\ \mathrm{P}(R = x_2/x_1)$

In other words, the joint prior for q_A and q_B is $f_A(q_A)\ .\ (1/q_A)\ .\ f_R(q_B/q_A)$.

The contribution to the objective function is the negative log of the joint prior, which is $-\log(f_A(q_A)) + \log(q_A) - \log(f_R(q_B/q_A))$

In the case that f_R is lognormal with mean, μ, and c.v., c, the contribution to the objective function (which is the negative log of the joint prior) becomes (ignoring constants) $-\log\left(f_A\left(q_A\right)\right) + P_{BA}$, where

$$P_{BA} = \log\left(q_B\right) + 0.5\left[\frac{\log\left(\left(q_B/q_A\right)/\mu\right)}{s} + \frac{s}{2}\right]^2$$

is the penalty on q_B/q_A and, as usual, $s = \sqrt{\log\left(1 + c^2\right)}$.

Case 2: Three *q*'s, version 1

With three *q*'s there are two ways to specify priors involving ratios. In the first version, the same *q* appears in the denominator of both ratios. That is, specify the priors f_A, on q_A, f_{R1}, on the ratio $R_1 = q_B/q_A$, and f_{R2}, on the ratio $R_2 = q_C/q_A$. As before, we assume that these three priors are independent.

In this case, an argument analogous to the above shows that the joint prior for q_A, q_B, and q_C is $f_A(q_A)$. $(1/q_A)\ .\ f_{R1}(q_B/q_A)\ .\ (1/q_A)\ .\ f_{R2}(q_C/q_A)$. When the f_{Ri} are lognormal, the contribution to the objective function becomes (ignoring constants) $-\log\left(f_A\left(q_A\right)\right) + P_{BA} + P_{CA}$, where P_{CA} is the penalty on q_C/q_A.

To use this version the user must: (a) provide whatever prior they like for q_A, (b) specify uniform priors for both q_B and q_C, and (c) specify a `ratio_qs_penalty` for q_B/q_A and another for q_C/q_A.

Case 3: Three *q*'s, version 2

In the second version with three *q*'s, f_A and f_{R1} are as previously, but f_{R2} is a prior on the ratio $R_2 = q_C/q_B$. It is straightforward to show that the joint prior for q_A, q_B, and q_C is

$$f_A(q_A)\ .\ (1/q_A)\ .\ f_{R1}(q_B/q_A)\ .\ (1/q_B)\ .\ f_{R2}(q_C/q_B).$$

When the f_{Ri} are lognormal with means, the contribution to the objective function becomes (ignoring constants) $-\log\left(f_A\left(q_A\right)\right) + P_{BA} + P_{CB}$.

To use this version the user must: (a) provide whatever prior they like for q_A, (b) specify uniform priors for both q_B and q_C, and (c) specify a `ratio_qs_penalty` for q_B/q_A, and another for q_C/q_B.

General case: multiple *q*'s

In the general case, with $n+1$ q's, $q_0,q_1,...,q_n$, we want to specify a prior, f_0, for q_0, and also priors $f_1,..., f_n$, for the ratios $R_1,...,R_n$, where $R_i = q_i/q_{Ai}$. There are many different ways to define these ratios. However, to make the mathematics work, we need the condition that, for each i, $q_{Ai} = q_j$ for some $j < i$.

For this general case, the user must (a) provide whatever prior they like for q_0, (b) specify uniform priors for $q_1,...,q_n$, and (c) specify a `ratio_qs_penalty` for each of $R_1,...,R_n$.

6.7.6 Penalties

Penalties can be added to any objective function. You will usually want to use a *catch limit penalty* for each fishery to ensure that the exploitation rate constraints on your fisheries are not breached (otherwise there is nothing to prevent the model from having abundances so low that the recorded catches could not have been taken). A *vector average penalty* to force YCS to average to 1 is also very common.

For most penalties, you need to specify a *multiplier*, and the objective function is increased by this multiplier times the penalty as described below. In some cases you will want to make the multiplier quite large to prohibit some model behaviour. For penalties without multipliers, the objective function is just increased by the value of the penalty. So far, the penalties implemented in CASAL are:

1. *Ogive smoothing penalty*:

    Applied to an `allvalues` or `allvalues_bounded` ogive parameter (Section 5.6). Sum of squares of rth differences, optionally on a log scale. This encourages the ogive to be like a polynomial of degree($r-1$). For compatibility with previous NIWA software, you can choose to exclude indices outside a given set of bounds (these indices are ignored during differencing).

2. *Catch limit penalty*:

    Sum of squares of (actual catch less specified catch), optionally on a log scale, for a single fishery. These are intended to avoid parameter values that cause the specified fishing pressure limits to be exceeded. The penalty is only applied if some fishing pressure limit has been exceeded (since inaccuracy in the iterative solution for F in the Baranov equation leads to actual catches slightly less than specified, and you don't want to penalise that).

3. *Vector average penalty*:

    Applied to a vector parameter. Square of (mean(vector)$-k$), or of (mean(log(vector))$-l$), or of (log(mean(vector)/m)). Encourages the vector to average arithmetically to k or m, or geometrically to exp(l). Typically used for YCS with k=1 or m=1 or l=0, to encourage the YCS to centre on 1. Optionally, you can choose to exclude indices outside a given set of bounds.

4. *Vector smoothing penalty*:

   Applied to a vector parameter. Sum of squares of *r*th differences, optionally on a log scale. This encourages the vector to be like a polynomial of degree ($r-1$). Note a range of the vector to be "smoothed" can be specified (and if not, the smoother is applied to the entire vector), but this must be specified by an index of the vector and must be between 1 and the length of the vector, inclusive.

5. *Element difference penalty*:

   Applied to two vector parameters. Square of ($\text{vector}_1[\text{i}]-\text{vector}_2[\text{i}]$). Encourages the *i*th elements of the two vectors to be equal.

6. *YCS difference penalty:*

   Applied to the YCS of two different stocks. Squared difference between the YCS values for a given year in the two stocks. Used to encourage the two stocks to have the same YCS for that year. If the Haist YCS parameterisation is used, then the penalty applies to the YCS (as one would expect) and not the Y's.

7. *Similar qs penalty*:

   Applied to two catchabilities q (Section 6.7.1). Square of ($\log(q_1)-\log(q_2)$). This is intended to encourage q_1 and q_2 to be similar, perhaps because they belong to observations collected in similar ways.

8. *Ratio qs penalty*:

   Applied to the ratio, q_1/q_2, of two catchabilities, and intended to encourage this ratio to be near μ (Section 6.7.5). The penalty is defined as a lognormal with mean μ and c.v. c:

   $$\text{penalty} = \log\left(q_1\right) + 0.5\left(\frac{\log\left(\left(q_1/q_2\right)/\mu\right)}{s} + \frac{s}{2}\right)^2,$$

   where s is the standard deviation of $\log(q_1/q_2)$, so $s = \sqrt{\log\left(1+c^2\right)}$.

9. *Ogive comparison penalty*:

   Applied to two ogive parameters (Section 5.6). Sum of squares of $\max(\text{ogive}_1-\text{ogive}_2,0)$. Encourages ogive_1 to be at or below ogive_2. Typically ogive_1 is a selectivity for males, ogive_2 is a selectivity for females. This is intended to encourage female selectivities to be greater than those of males at the same age/size. For compatibility with previous NIWA software, you can choose to exclude indices outside a given set of bounds (these indices are dropped off before comparing). Note that this penalty may not be applied to size-based ogives in age-based models.

10. *Ogive difference penalty*:

    Applied to two ogive parameters (Section 5.6). Square of ($\text{ogive}_1-\text{ogive}_2$) for a single size or age class. This is intended to encourage the two ogives to take the same value for that class. Note that this penalty may not be applied to size-based ogives in age-based models.

11. *Fish-tagged penalty:*

    Applied to a specific tagging episode. Square of min(0, number of fish meant to be tagged less number of fish actually tagged). This penalty is intended to discourage parameter values which lead to not enough fish being present at the tagging episode. Considering that the tonnage of fish actually tagged is usually very small, this penalty will generally be zero.

6.7.7 Ageing error

In age-based models, we allow ageing error in at-age observations to be modelled explicitly. After *E* (expected) values are calculated for at-age observations, misclassification rates are applied to them, which has the effect of 'smearing' the age frequencies. The resulting 'smeared' age frequencies are used in calculating the objective function.

Ageing error is optional, and if it is used, it may be omitted for any individual time series. However, CASAL does not yet implement changes in ageing error over time, or different ageing error regimes for different time series.

The ageing error models implemented in CASAL are as follows:

1. *Off by one*:

    Proportion p_1 of fish of each age a are misclassified as age $a-1$ and proportion p_2 are misclassified as age $a+1$. Fish of age $a < k$ are not misclassified. If there is no plus group in the population model, then proportion p_2 fish of the oldest age class will 'fall off the edge' and disappear.

2. *Normal*:

    Fish of age a are classified as ages which are normally distributed with mean a and constant c.v. c. As above, if there is no plus group in the population model, some fish of the older age classes may disappear. If c is high enough, some fish of the younger age classes may 'fall off the other edge' too.

3. *Misclassification matrix*:

    A complete misclassification matrix M is provided, such that M_{ij} is the probability that a fish of age i will be classified as age j. Rows need not sum to 1, but a warning will be issued if they don't.

Note that the expected values (fits) reported by CASAL for an individual time series with ageing error, have had the ageing error applied.

6.8 Residuals

CASAL can generate three kinds of residuals: (1) the usual residuals (i.e., observed less fitted), (2) *Pearson* residuals, and (3) *normalised* residuals.

Let O be an observation and F the corresponding fit (=qE for relative observations), then:

1. *Residuals* are defined as (O−F).
2. *Pearson residuals* attempt to express the residual relative to the variability of the observation, and are defined as (O−F)/std.dev.(O), where std.dev.(O) is calculated as,
    - $F \times cv$ for normal, lognormal, robustified lognormal, and normal-log error distributions.
    - s for normal-by-standard deviation error distributions.
    - $\sqrt{\frac{Z(F,r)(1-Z(F,r))}{N}}$ for multinomial or binomial likelihoods.
    - $\sqrt{\frac{(F+r)(1-F+r)}{N}}$ for the binomial-approx likelihood.
    - $\sqrt{\frac{F'}{N'}}$ for Fournier likelihoods (where $F_i' = (1-F_i)F_i + 0.1/n$ and $N' = \min(N,1000)$, on the basis that they would be equivalent to a multivariate normal with this standard deviation if the final (+0.01) term was omitted.)
    - $\sqrt{\frac{O'}{N'}}$ for Coleraine error likelihoods (similarly).
3. *Normalised residuals* attempt to express the residual on a standard normal scale, and are defined as:
    - Equal to the Pearson residuals for normal error distributions.
    - (log(O/F)+0.5σ^2)/σ for lognormal (including robustified lognormal) error distributions, where $\sigma = \sqrt{\log(1+cv^2)}$.
    - (log(O/F)/σ for normal-log error distributions, again with $\sigma = \sqrt{\log(1+cv^2)}$.
    - And are otherwise undefined.

6.9 Generate simulated observations

CASAL can generate simulated observations from a parameter fit, i.e., generate simulated observations which are randomly distributed (according to the error assumptions defined for the observations) around fits calculated from one or more sets of the 'true' parameter values. This is a form of parametric bootstrap.

One use of this feature is to investigate the uncertainty in CASAL parameter estimates, using a bootstrapping approach:

1. Get one or more sets of free parameters, either using an assumed set of values, a point estimate or a sample from the posterior distribution.

2. Use CASAL to generate many sets of simulated observations, on the assumption that the free parameter estimates are the true values.

3. For each set of simulated observations, generate a simulated estimate of the free parameters (replacing the real observations with the randomised observations).

4. The variability in the simulated estimates is a bootstrap estimate of the uncertainty in the estimation process.

This approach allows the user to assess estimator performance in varying conditions. For example, the simulated estimates could be carried out using a simplified estimation procedure (perhaps fixing some previously free parameters), and the effect of this simplification on estimator performance could then be investigated.

The way in which the above process could be undertaken might be:

1. Estimate the free parameters using `-e`, `-E` or `-m`, `-M`. Generate a file of free parameter values using the usual format (described in Section 3.3)

2. Run CASAL in simulator mode (`-s`), supplying the file of free parameter values with the command line switch `-i`, using the same input parameter files (`population.csl` and `estimation.csl`). The results are files of simulated observations.

3. CASAL creates one file for each set of simulated observations. The total number of files is equal to the number of free parameter sets supplied, multiplied by the number of simulations per parameter set (the latter is the number that you supply as an argument to `-s` on the CASAL command line).

4. Each file contains a set of observations, using the standard CASAL syntax. Note that you can specify what observations you wish to simulate by turning off bootstraps for each set of observations by setting the observation subcommand `do_bootstrap` to `False`.

5. All commands and subcommands will be unchanged from the original estimation parameter file, except for the original observation values which will be replaced with randomised values. (although the subcommands will appear in alphabetical order, and comments and white space will have been removed.)

6. A comment is appended to the top of each file, listing the free parameter values that were used to generate it.

7. Outside of CASAL, attach (append or prepend) a 'stub' estimation parameter file to each file of simulated observations. This stub file should contain all the estimation parameters *other than the observations*, and include a list of parameters to be estimated, etc., using the usual CASAL syntax.

8. Using each (stub plus simulated observations) file along with standard input parameter files (`population.csl` and `output.csl`), carry out an estimate of the free parameters using `-e, -E`, or even `-m` or `-Y`.

The utility program `simCASAL` is available to assist with such a process. See Section 12.1.

The remainder of this section describes the method used to produce the simulated observations for a single set of 'true' free parameter values.

First, the model is run using the true free parameter values, and a set of fits is produced for each set of observations. If a set of observations uses ageing error, then ageing error is applied to the fits as per normal. If there are relative observations, then the catchability coefficient q is applied to the fits as per normal.

Second, each set of observations is randomised, based on

- the fitted values,
- the type of likelihood specified,
- the variability parameters (c.v. N, or σ). Variability is increased by the process error associated with that time series, if any (see Section 6.7.2). If the process error parameter is a free parameter, then the 'true value' of the process error is used.

Age-size and age-at-maturation observations cannot be simulated. You need to remove these observations or set `do_bootstrap` to `False` before using CASAL as a simulator.

The following text describes the process of generating simulated observations for each type of likelihood.

1. Normal likelihood parameterised by c.v.: Let E_{yi} be the fitted value for observation i in year y and c_{yi} be the corresponding c.v. (adjusted by process error if applicable). Each simulated observation value S_{yi} is generated as an independent normal deviate with mean E_{yi} and standard deviation $E_{yi}\ c_{yi}$.

2. Normal likelihood parameterised by standard deviation: Let E_{yi} be the fitted value for observation i in year y and s_{yi} be the corresponding standard deviation (adjusted by process error if applicable). Each simulated observation value S_{yi} is generated as an independent normal deviate with mean E_{yi} and standard deviation s_{yi}.

3. Log-normal likelihood: Let E_{yi} be the fitted value for observation i in year y and c_{yi} be the corresponding c.v. (adjusted by process error if applicable). Each simulated observation value S_{yi} is generated as an independent lognormal deviate with mean and standard deviation (on the natural scale, not the log-scale) of E_{yi} and $E_{yi}\ c_{yi}$ respectively. The robustification parameter r is ignored.

4. Normal-log likelihood: Let E_{yi} be the fitted value for observation i in year y and c_{yi} be the corresponding c.v. (adjusted by process error if applicable). Each simulated observation value S_{yi} is generated as an independent lognormal deviate, such that the mean of $\log(S_{yi})$ is $\log(E_{yi})$ and the c.v. of S_{yi} is c_{yi}.

5. Multinomial likelihood: This is only allowed if the same N value is used for all observations of the same time series in the same year. Let E_{yi} be the fitted value for observation i in year y, for i between 1 and n, and let N_y be the equivalent sample size for that year (rounded up to the next whole number, and adjusted by process error if applicable).Any robustification is ignored. The following process is carried out for each year y:

    a. A sample of N data values from 1 to n is generated using the multinomial distribution, using sample probabilities proportional to the values of E_{yi}.

    b. Each simulated observation value S_{yi} is calculated as the proportion of the N sampled values equalling i.

   c. The simulated observation values S_{yi} are then rescaled so that their sum is equal to the sum of E_{yi}. (The sum of the fitted values for the year may not be equal to 1 if `sum_to_one=False` and the age/size range of the observations does not cover all columns in the partition.)

6. Coleraine or Fournier likelihood: These are not 'proper' likelihoods in the technical sense, and we do not use them as distributions for generating simulated values. Instead, as they are analogous to the multinomial likelihood, we apply the above procedure for the multinomial, using the supplied value of the N parameter.

7. Binomial likelihood: Let E_{yi} be the fitted value for observation i in year y, for i between 1 and n, and N_{yi} the corresponding equivalent sample size (rounded up to the next whole number, and adjusted by process error if applicable). Any robustification is ignored. The following process is carried out for each observation i in each year y:

   a. A sample of N_{yi} independent binary variates is generated, equalling 1 with probability E_{yi}.

   b. The simulated observation value S_{yi} is calculated as the sum of these binary variates divided by N_{yi}.

7. THE OUTPUT SECTION

This section contains three main topics.

1. Section 7.1 describes the printouts from CASAL, which are dumped to standard output (and can be redirected to a file, and imported into R using the functions described in Section 13).

2. Section 7.3 describes projections in CASAL.

3. Sections 7.4 and 7.5 describe yield calculations in CASAL, including deterministic MSY, various yield per recruit statistics, MCY, CAY, and CSP.

Information about MCMC output file formats is given earlier in Section 3.1.

7.1 Printouts from CASAL

CASAL prints out a bunch of different things to standard output. Some of them appear automatically, others must be asked for. It is important to note that the exact content of these outputs can be expected to change without notice. The best way to find out exactly what CASAL prints is to run it and find out.

The main types of printouts are:

- An initial header, giving the command by which CASAL was invoked, the date, the version numbers of the key source files used to build that copy of CASAL, the version number of CASAL itself, user login, and machine name.

- The names of any additional output files that were generated, such as MCMC output dumps.

- The results of the particular task asked for. If you run the model or estimate the parameters (`casal -e`, `-E`, `-r`), CASAL will print out the free parameters, the objective function and its components. If you profile some parameters (`casal -p`), CASAL will print out the objective function value and the free parameter estimates, for each value of each profiled parameter. If you carry out an MCMC, CASAL will print the initial point estimate, the approximate covariance matrix, the lower and upper bounds on the free parameters during MCMC, the start of the chain, and any changes in step size, etc.

- Printouts from the population section of CASAL. You can ask for printouts of the requests sent to the population section by the estimation and output sections and the corresponding results. You can request printouts of the initial state, the final state, the state after every year or every step. These are mostly useful in debugging, i.e., you can inspect them to figure out whether the population dynamics are what was intended. The most important of these printouts is ‘`population_section`’ which gives a text explanation of how CASAL interprets the population section in the `population.csl` file — always look at this printout when you develop a model to ensure that you have correctly specified the model.

- These population printouts only appear if you ask for them in the `output.csl` file. You may want to use the `-q` switch to suppress these printouts, because a major job can generate a huge amount of them. (You can achieve the same result by turning all these requests off in `output.csl`.)

- Printouts from the estimation section of CASAL. You can ask CASAL to print out the parameter lists generated from your `population.csl` and `estimation.csl` files, a good way of checking that your files were read as intended. You can ask for fits, residuals, and standardised residuals. For debugging purposes, you can ask CASAL to print out the objective, parameters, or fits every time the objective function is calculated (so they are printed at each step of a minimization). You can ask for a text explanation of how CASAL thinks your estimation section works — always look at this printout when developing a model. Also check out the list of parameters that were never accessed by CASAL. Presence of a parameter on the list may indicate that the parameter name was spelt incorrectly. (Or it may just mean that the parameter is not used for the task you were doing. For example `max_iters` is not used by `casal -r`). All these estimation printouts only appear if you ask for them in the `output.csl` file.

- Output quantities. These are model outputs calculated from the parameters. They can be produced for any set of free parameters, whether it comes from a model run (`casal -r`), a point estimate (`-e`, `-E`), for values sampled from a Bayesian posterior (`-v`), or for projections (`-P`). The output quantities CASAL can produce are listed in Section 7.2. Output quantities only appear if you ask for them in the `output.csl` file.

7.2 Output quantities

A variety of CASAL outputs are classed as 'output quantities'. They can be produced for any set of free parameters, whether it comes from a model run (`casal -r`), a point estimate (`-e`, `-E`), for values sampled from a Bayesian posterior (`-v`, see Section 3.1), or for projections (`-P`, see Section 7.3).

Output quantities produced by model runs or point estimates are printed in a verbose format. They are marked with asterisks (*) in the output, which clearly identifies them for reading into statistical packages such as R. When output quantities are produced for samples from a Bayesian posterior or for projections, many sets of quantities are generated, so the results are dumped to a file in a columnar format instead.

Output quantities include the following:

1. The values of parameters. You can ask for 'all free parameters', and/or the names of parameters, which need not be free. If you ask for an ogive, CASAL supplies the values of the ogive rather than the ogive arguments. If you ask for a size-based ogive in an age-based model, CASAL supplies the values of the ogive at the sizes given in the output parameter `print_sizebased_ogives_at`. If the size-based ogive is a selectivity, then a better way to extract its values is to use the `selectivity_at` pseudo-observations class (see below).

2. The arguments of ogive parameters (as opposed to the values, which see (1) above).

3. Spawning stock biomasses, for each stock in each model year (SSBs, see Section 5.3).

4. Recruitments, as absolute numbers of fish of each stock, by the year in which they recruit (see Section 5.4.2).
5. YCS, as deviates, by the year in which they spawn (see Section 5.4.2).
6. True YCS', defined as the associated recruitment divided by R_0 (see Section 5.4.2),
7. The climate variable T by year (see Section 5.4.2).
8. Actual catches, optionally by stock (see Section 5.4.6).
9. Removals, optionally by stock (see Section 5.4.6).
10. Discards, optionally by stock (see Section 5.4.6).
11. Fishing pressures, by fishery, for each year (see Section 5.4.6).
12. B_0, R_0, B_{mean}, R_{mean}, $B_{initial}$, and $R_{initial}$ for each stock (see Section 5.5).
13. Nuisance q's (see Section 6.7.1).
14. The 'stock crash' quantity used to calculate stock risk (see Section 7.3).
15. Proportions-at-age tagged if in an age-based model and release type is *deterministic*.
16. Fits, residuals, Pearson residuals, and normalised residuals.
17. Pseudo-fits (see below).

Pseudo-fits are a special case. A *pseudo-fit* is an output defined as the fits to a set of *pseudo-observations*, 'fake' observations which did not occur. This seems like an odd way of doing things, but in fact enables the production of a number of useful outputs. For example, the total biomass for each model year, in a particular area and time step, can be generated as a pseudo-fit to an abundance series. If you want to see the selected biomass, add a selectivity to the pseudo-observations. If you want mature biomass, specify that the pseudo-observations include mature fish only. Similarly you can generate a combined biomass over all areas, biomass of a particular stock, total numbers rather than biomass, etc. You can also inspect the age or size composition of the fish by using numbers-at, proportions-at, or catch-at pseudo-observations. You cannot use pseudo-fits to age-size data, however.

The observation, `selectivity_at`, can be used as a pseudo-observation to extract the values of selectivity ogives, for each age/size class in the partition, in a particular year, time step, area, etc. It is particularly useful for extracting the values of a size-based ogive in an age-based model, because it converts them into values-at-age. This provides a one-step method for finding the actual ogive values being used by the model.

To ask for pseudo-fits, include the pseudo-observations in your `output.csl` file in the same way that you include real observations in your `estimation.csl` file. The only differences are:

1. Do not use relative observation types, i.e., `relative_abundance` and `relative_ numbers_at`. Use the absolute equivalents instead.
2. Do not supply the actual observation values — there are none.

3. Do not supply an error distribution, c.v.s, effective *N*'s, weights, etc.

For example, insert these commands in the `output.csl` file to get outputs of total biomass across all areas, halfway through the mortality in time step 2, for all model years (1970 to 2000).

```
@abundance total_biomass
# output quantity: total biomass in all areas
biomass true
all_areas true
step 2
proportion_mortality 0.5
years 1970 1971 1972 1973 1974 1975 1976 1977 ... 1998 1999 2000
```

For projected output quantities (see Section 7.3.2), the range of years should extend into the future (up to year `final`).

7.3 Projections

Projection is the process of running the model forwards into the future, using randomised recruitments and hypothetical catches. CASAL does this in three situations:

1. Calculation of current annual yield (CAY) (see Section 7.5.1).

2. Calculation of current surplus production (CSP) (see Section 7.5.2).

3. Producing projected fishery performance estimators (FPIs), such as stock risk, or expected biomass at some future time .

All three situations use the same method for generating projections, which is described in Section 7.3.1. The calculation of FPIs is discussed in Section 7.3.2.

For models with annually-varying migrations, CASAL also randomises these migrations when producing FPIs or CSP (but not when calculating CAY).

7.3.1 Carrying out projections

Projections can either be *point-based* (i.e., using a single point estimate of the free parameters), or *sample-based* (using a sample from the posterior distribution, typically generated by MCMC using `casal -m`, `-M`, or `-C`).

For point-based projections CASAL does a large number of simulations, each using the same parameters. The simulations will differ only in terms of the randomised recruitments (and, where appropriate, randomised annually-varying migrations). Year class strengths will be randomised for the cohorts that will recruit in the 'projection period', i.e., the years `current+1` to `final`. You can also choose to randomise YCS for cohorts which have recently recruited (perhaps because there is no information about the abundance of these cohorts). You can also modify the YCS values by an arbitrary multiplier. If there is an explicit climate-recruitment relationship, CASAL uses the climate data *T* up until the last year for which it is provided (which could be as late as the assessment year, or might even be a forecast for the future) and then randomises *T* for years after that.

For sample-based projections CASAL does one simulation for each posterior sample point. Each simulation will use a different set of parameters and a different set of randomised recruitments. YCS and *T*'s will be randomised as above (the only difference is that the user might not need to randomise some recent YCS if their uncertainty was incorporated in the posterior distribution).

In either case, the 'projected expectation' of a quantity refers to an average over all the simulations.

When doing projections to calculate fishery performance indicators, you need to specify future catches or future mortalities (either Baranov or non-Baranov) for each fishery in each year. CASAL does not implement adaptive harvest strategies in projections. To assess a different catch scenario, you need to change the future catches in the data file and rerun the program.

You can choose between four methods of randomising the YCS, :

1. *Lognormal*: The randomised YCS are lognormally distributed, with mean 1, and specified standard deviation and autocorrelation on the log-scale. $YCS_i=\exp(X_i)$, where (X_i) are generated as a Gaussian AR(1) process with standard deviation σ_R and mean$-0.5\sigma_R^2$ (so that the mean of YCS_i is 1), and autocorrelation ρ. Set $\rho=0$, the default, if you don't want autocorrelation. If the randomised YCS are modified by an arbitrary multiplier, then the only change is that X_i will have mean μ, where μ is the recruitment multiplier.

2. *Lognormal-empirical*: The randomised YCS are lognormally distributed as per the *lognormal* method above. The only difference is that the standard deviation parameter is chosen to give variability equal to that of the estimated YCS. CASAL uses σ_R as the standard deviation of the log of the estimated YCS. Optionally, the calculation of σ_R can be based on a sub-range of the estimated YCS (since not all YCS are well estimated and some may even be fixed). As for the *lognormal* method, the randomised YCS can be multiplied by an arbitrary multiplier, with the standard deviation calculated as described.

3. *Empirical*: The randomised YCS are resampled from the estimated YCS. Again, they can optionally be resampled from a sub-range of the estimated YCS. If the randomised YCS are modified by an arbitrary multiplier, then the each of the randomised YCS are multiplied by the recruitment multiplier.

4. *None*: All the randomised YCS are 1, but if modified by an arbitrary multiplier, then the YCS will all equal the recruitment multiplier. Used for deterministic projections only.

If a nonzero autocorrelation parameter ρ is used with lognormal or lognormal-empirical randomisations, then the randomised values must depend on the last fixed value YCS_f. This can get a bit 'messy'.

Let $\mu_R = -0.5\sigma_R^2, X_f = \log\left(\frac{YCS_f - \mu}{\sigma_R^2}\right)$ and (Z_i) be standard normal random deviates, then

$$X_1 = \mu_R + \sigma_R\left(\rho X_f + \sqrt{1-\rho^2}Z_1\right), \text{ and } X_{i+1} = \mu_R + \sigma_R\left(\rho X_i + \sqrt{1-\rho^2}Z_{i+1}\right).$$

Now if the user specifies a very small or zero σ_R, probably in an effort to generate constant YCS_i=1, and a nonzero ρ, and YCS_f is substantially different from 1, then the above formula gives an unexpected result, the YCS_i are not 1, but decay exponentially from YCS_f towards 1. This is because under these assumptions the value of YCS_f is highly implausible. CASAL avoids this situation by erroring out if $|X_f|>5$ with "last non-random year has implausible value". The user can fix the error by setting ρ=0, increasing σ_R, or turning off randomisation (using method `none`).

Only one method of randomising annually-varying migrations is currently available — the empirical method. Annual variation values for future years are resampled from the estimated values for a user-specified range of years.

We provide the same four methods for randomising the *T*s associated with a climate-recruit relationship (see Section 5.4.2). The only difference is that the randomised *T*s need not come from a distribution with mean 1. For lognormal randomisation, or no randomisation, the mean of the *T*s is specified by the user. For lognormal-empirical randomisation, it is the mean of the estimated *T*s.

So, to define the method of doing projections, you need to tell CASAL the following:

1. If projections are point based, the number of projections to be done.
2. Which is the first year for which YCS are randomised? The default is (`current`-y_{enter}+1), which is the first year for which YCS are not provided. But you can specify an earlier year if they want to randomise abundance of some recently recruited cohorts. (Note that this is the year in which the fish spawn, not the year in which they recruit.)
3. Future catches or future mortalities (either Baranov or non-Baranov) for each fishery in each projected year. (This is only necessary for producing FPIs, not for CAY or CSP.)
4. The methods used to randomise YCS and *T*s, and the relevant parameters.
5. The years from which to resample annual variation values (needed only in models with annually-varying migrations).

Of the above, only the first point is specified in the `output.csl` file. All the others are in `population.csl` since they relate to the recruitment variability of the population, and the catches.

7.3.2 Calculating projected fishery performance estimators (FPIs)

There are many fishery performance estimators (FPIs) commonly used in current New Zealand stock assessment. These include:

- Stock risk.
- E($B_{current+k}/B_{current}$).
- E($B_{current+k}/B_{initial}$).
- E($B_{current+k}/B_0$).
- P($B_{current+k}>B_{current}$).

It is impractical to code all conceivable FPIs in CASAL, and it is more useful to dump the results of each individual projection into a text file, where you can use to generate your own FPIs in an external package such as R or Excel. Then if you want to calculate a different set of FPIs, you can do it without needing to redo the projections in CASAL.

Projected abundances and catches are requested from CASAL as output quantities (see Section 7.2). Call `casal -P` to run projections and generate the requested output quantities for the projected years. Use `-i filename` to pass CASAL a parameter estimate or a list of samples from the posterior. Projected actual catches and SSBs can be requested using the `quantities.actual_catches` and `quantities.SSBs` switches in `output.csl`. Various kinds of projected abundances can be requested by asking for abundance 'pseudo-fits' covering a range of years extending into the future. The projected results will then be sent to the output quantities file (the user must specify a filename as the argument of `-P`), which can be imported and processed by another package. There will be one row per projection.

When projections are point based, `casal -P` will also print out the expectation of each output quantity. This is intended as a shortcut so that some FPIs, such as $E(B_{current+k}/B_0)$, can be calculated without using a second software package. (Just divide the expected SSB for year `current+k` by B_0.) On the other hand some FPIs cannot be calculated using this method, such as $P(B_{current+k} > B_{current})$. You will need to use an external package to calculate these FPIs. The stock risk is a commonly used output quantity, defined as the probability that the SSB will fall below 20% B_0 in the projection period (for each stock). To allow a shortcut method for calculating stock risk, we provide a projected output quantity `stock_crash`, which is defined as 1 if the SSB falls below 20% B_0 in the projection period or 0 otherwise (for each stock). Then you can read off the stock risk as the expectation of `stock_crash`.

7.4 Deterministic yield calculations

CASAL implements two kinds of deterministic yields; *per-recruit* analyses (Section 7.4.1) and *deterministic MSY* (Section 7.4.2). They are deterministic in the sense that they are based on simulations which use non-random recruitment with $YCS=CR(T)=1$ (and hence the recruitment in year y is $R_y=R_0\times SR(SSB_{y-y\_enter})$). The calculations are based on a single set of parameters (i.e., a point estimate), supplied with `-i`.

Deterministic yields can be calculated only for single-stock models in CASAL. However, some limited YPR/SPR output is available for these models (see Section 7.4.1).

Deterministic yield calculations are based on simulations at a constant *mortality rate F*. This "mortality rate" can be defined in several ways:

1. If there is only one fishery, then the mortality rate can be defined as an exploitation rate, which is the catch divided by a pre-fishery measure of biomass B_{pre}. Thus, the catch for each year is FB_{pre}. By default, B_{pre} is defined for all areas combined, in the time step of the fishery, before any mortality is applied. But you will need to specify the selectivity and if B_{pre} is for mature fish only. You can change the default definitions, but if you change the proportion mortality, then it must be in a time step before the time step of the mortality episode in which the catch is taken.

2. If there are multiple fisheries, then the mortality rate must be defined as an exploitation rate as above. Again, B_{pre} is defined as the biomass in all areas combined in the time step of the fishery, before any mortality is applied, but you can change this definition. As for the single fishery case above, you will need to specify the

selectivity and if B_{pre} is for mature fish only. You have to provide a *catch split*, i.e., the proportion of the annual catch which must come from each fishery. Once the catch for the year has been calculated, it is split between fisheries according to this 'catch split'.

3. Alternatively, if there is only one fishery and the Baranov catch equation is used, then you can opt to define the mortality rate as the instantaneous mortality rate of the Baranov equation. This is a more conventional method and may be required for comparability with other modelling work.

Note that for options 1 and 2, it may be impossible to take the catch even when $F < 1$, or alternatively it may be possible to take the catch even when $F > 1$, depending on the definition of B_{pre}.

B_{pre} is also used in CAY calculations (Section 7.5.1). The catch split is also used for MCY/CAY calculations and for CSP (Section 7.5.2).

Note that CASAL does not estimate deterministic yields for weightless models.

7.4.1 Yield per recruit analyses

Per-recruit analyses are based on yield per recruit (YPR) and/or SSB per recruit (SPR). You can ask for any or all of the following:

- Data to plot a YPR curve (YPR versus mortality rate) or an SPR curve (SPR versus mortality rate).
- F_{max}, the mortality rate which maximizes YPR.
- $F_{0.1}$, the mortality rate at which the slope of the YPR curve is 0.1 times its slope at the origin (Gulland & Boerema 1973).
- $F_{x\%}$, the mortality rate at which the SPR is x% of its unfished value (Clark 1991).

$F_{0.1}$ should be calculated only if the mortality rate is an instantaneous rate (an $F_{0.1}$ based on exploitation rates could be calculated, but it is not clear that this would be a "safe" rate of fishing, in the way that the $F_{0.1}$ base on instantaneous rates is believed to be).

Each calculation of YPR or SPR works as follows. A single simulation run is done, starting from an unfished equilibrium state, and running until the catch and SSB stabilize. Having reached convergence, the total annual catch, SSB, and annual number of recruits are recorded, and YPR (total annual catch divided by number of recruits) and/or SPR (SSB divided by number of recruits) are calculated.

Traditionally, per-recruit analyses are done without a stock-recruitment relationship. However, it makes no difference either way, as long as the model divides by the actual number of recruits when calculating per-recruit statistics. However, we have found that finding the deterministic equilibrium with a high fishing pressure and a strong stock-recruitment relationship can take many, many simulated years. We recommend turning the stock-recruitment relationship off for per-recruit analyses to speed up calculations.

You need to provide an initial guesstimate of F, which is used to start off the minimiser for the estimates of F_{max}, $F_{0.1}$, and $F_{x\%}$,. Providing an approximate starting value will help the minimiser find a more accurate solution.

For multi-stock models it is not clear how to define F_{max}, $F_{0.1}$, and $F_{x\%}$, so these quantities are not currently calculated by CASAL for these models. However, it is possible to output the data to calculate YPR and SPR curves for each stock.

7.4.2 Deterministic MSY

MSY_{det} is the maximum constant annual catch (using the specified catch split if there is more than one fishery) that can be sustained under deterministic recruitment. The corresponding mortality rate is F_{MSYdet}, and the corresponding SSB is B_{MSYdet}. Both MSY_{det} and B_{MSYdet} are expressed as percentages of B_0.

Simulations for deterministic MSY work in the same way as the per-recruit simulations in Section 7.4.1. For each simulation run with mortality F, the equilibrium total annual catch C_F and spawning stock biomass SSB_F are calculated. CASAL searches over mortality rates to find F_{MSYdet}, the value of F that maximizes C_F. Then MSY_{det} and B_{MSYdet} are C_F and SSB_F respectively (expressed as percentages of B_0).

As well as calculating the MSY, you can request data with which to plot a yield versus SSB curve. You need to tell CASAL the mortality rates F at which SSB and yield are to be calculated.

The results of a deterministic MSY analysis depend heavily on the stock-recruitment relationship used. You have to specify one, even if it is ‘none’.

You need to provide an initial guesstimate of F_{MSYdet}, which is used to start off the minimiser. Providing a value in the right ballpark will help the minimiser find a more accurate solution.

7.5 Stochastic yield estimates

CASAL implements two kinds of stochastic yields, MCY and CAY (Section 7.5.1) and current surplus production (CSP, Section 7.5.2). They are stochastic in the sense that they are based on simulations that use randomised recruitments. They can be either point-based or sample-based. Unlike deterministic yields, they can be calculated for multi-stock models.

7.5.1 MCY/CAY

Calculation of these yields is based on (and extends) the current NIWA procedures described by Francis (1992). Simulations are carried out to maximise yields, under either constant-catch or constant-mortality-rate harvesting, subject to the constraint that SSB should not fall below pB_0 more than proportion q of the time (defaulting to the traditional p=0.2, q=0.1).

By default, the risk constraint in the MCY/CAY analysis specifies that the spawning stock biomass falls below pB_0 less than $q \times 100\%$ of the time. There is also an option to replace B_0 by a different reference biomass, which must be the spawning stock biomass (SSB) of the stock for some year between *initial* and *current*. For example, if the stock was believed to be in good condition in 1985, you could specify that the spawning stock biomass falls below pB_{1985} less than $q \times 100\%$ of the time (where B_{1985} is the spawning stock biomass in 1985).

CAY calculations are based on simulations at a constant *mortality rate F*. As per deterministic yields, this mortality rate is either an exploitation rate — catch/pre-fishery biomass B_{pre} — or if the Baranov equation is used, as the instantaneous mortality rate of the Baranov equation.

For both MCY and CAY calculations, if there are multiple fisheries, you have to provide a *catch split*, i.e., the proportion of the annual catch which must come from each fishery. These issues are discussed in more detail in Section 7.4. Note that B_{pre} is also used in deterministic yield calculations (Section 7.4) and the catch split is also used for deterministic yield calculations and for CSP (Section 7.5.2).

For each of a series of *harvest rates*, H (either a constant catch or a constant mortality rate) many simulation runs are carried out. Each simulation starts from a state which has stabilised under harvest rate H with deterministic recruitment (as per the deterministic simulations in Section 7.4). The run extends over $n_{discard}$+n_{keep} years with stochastic recruitment. You need to choose both $n_{discard}$ and n_{keep}. Hopefully in the long term we will determine good default values. You need to choose a value of $n_{discard}$ that is large enough to allow the population to stabilize under harvest rate H by the end of $n_{discard}$ years. Francis (1992) recommends n_{keep} = the approximate maximum age of the species=$\log_e(100)/M$ (the natural mortality rate). We print $E(SSB_{n\_discard})$ and $E(SSB_{n\_discard+1})$ as diagnostics. If the two are about equal, then $n_{discard}$ may be large enough. Try also using different values of $n_{discard}$ and n_{keep} and seeing if it makes any difference to the results.

With one stock, for each run, CASAL will calculate, over the final period of n_{keep} years, the mean catch taken over all fisheries C_{av}, the mean SSB B_{av}, and the proportion P_{risk} of years in which the SSB falls below pB_0. These quantities will then be averaged over all runs with harvest rate H to calculate $C_{av}(H)$, $B_{av}(H)$, $P_{risk}(H)$. The program then searches for the "optimal" harvest rate H_{opt}, which is the value of H that maximises $C_{av}(H)$, subject to the constraint that $P_{risk}(H) \leq q$. Note that the search may take quite a while, depending on how many simulations you do, and you may want to interrupt it once it reaches a solution which is good enough for your purposes. You may alternatively want to search manually, interactively supplying a sequence of harvest rates. In this case, you will be prompted to input a trial H, CASAL will print $C_{av}(H)$, $B_{av}(H)$, and $P_{risk}(H)$, you will be prompted for a new H, etc. When you are satisfied, enter a negative value, meaning 'stop here'. The last value of H provided will be used to calculate yields.

CASAL then calculates yields. Constant-catch simulations give MCY=H_{opt} (a target catch in tonnes) or if, according to the current assessment, the stock is depressed (i.e., E(current SSB/B_0)<0.2, where the expectation is over the parameter sets provided), then MCY=$H_{opt}\times E(\mathrm{SSB}_{\mathrm{current}}/0.2B_0)$ (see Section 4.5 of Francis 1992). This adjusted value is sometimes labelled the 'current MCY' to distinguish it from the 'long-term MCY' H_{opt}. Whichever MCY is calculated, $B_{MCY}=B_{av}(H_{opt})$. Constant-mortality-rate simulations give $F_{CAY}=H_{opt}$, MAY=$C_{av}(H_{opt})$, and $B_{MAY}=B_{av}(H_{opt})$. The calculation of CAY for next year requires a 1-year projection (Section 7.3). It is the expected catch in the projected year, under a mortality rate of F_{CAY}.

If an exploitation rate constraint is broken during a deterministic simulation, then the catch level is clearly too high. We don't do the following stochastic simulation, and instead take $C_{av}=0$, $B_{av}=0$, $P_{risk}=1$.

For multiple stocks, we had to reinvent the definitions of MCY and CAY. The quantities C_{av}, B_{av}, and P_{risk} are calculated for each stock separately. The "optimal harvest rate" H_{opt} is now the value of H which maximizes $\Sigma_s C_{av,s}$, subject to the constraint that $P_{risk,\ s}(H) \leq q$ for all s (where s indexes the stocks). There is no obvious way to split the MCY between stocks, and nor is it clear how, if at all, the MCY should be modified if one or more of the stocks is depressed. However we can calculate the B_{MCY} for each stock, $B_{MCY,s}=B_{av,s}(H_{opt})$. For CAY simulations, $F_{CAY}=H_{opt}$, $\mathrm{MAY}_s=C_{av,s}(H_{opt})$, $B_{\mathrm{MAY},s}=B_{av,s}(H_{opt})$, and next years CAY_s is the expected catch in the projected year from stock s, under a mortality rate of F_{CAY}.

Where there are multiple stocks with one TAC per stock and no multi-stock fisheries, the natural approach is to calculate MCYs and CAYs separately for each stock using a catch split in which all the catch comes from a single stock. If you do this, you will need to make sure that the abundance measure B_{pre} used in the catch equation

$$\text{catch}=H^{*}B_{pre}$$

is defined appropriately. B_{pre} should refer to the biomass of fish which is targeted by the fishery. In a multi-area multi-stock assessment, when MCY and CAY are being calculated for, say, stock A, then the catch split should sum to 1 for fisheries targeting only stock A and sum to 0 for all other fisheries. B_{pre} should be calculated for the area in which stock A is located at the time step(s) in which the fishery is carried out, otherwise you will get incorrect results (because the catch of stock A is proportional to an abundance measure which includes fish of other stocks, and so high catches may be taken even if few stock A fish remain).

CASAL provides four methods for generating random recruitments for the simulation period — “lognormal”, “lognormal-empirical”, “empirical”, and “none”. These methods are described in Section 7.3.1. The only major difference is that there is no ‘last non-random year’. All the YCS and T’s are random.

As well as recruitment variability, simulations can incorporate uncertainty in several different ways:

1. For sample-based simulations, the Bayesian posterior is meant to express the uncertainty in the free parameters. One simulation run is done for each sample from the posterior (c.f. point-based simulations where many simulations are done using the single set of parameters). Note that if either “empirical” method of randomising recruitment is used for sample-based simulations then the recruitment variability will differ between individual simulations.

2. For point-based MCY simulations, the uncertainty associated with virgin biomass can be incorporated (as in the stock assessment software *pmod*, R.I.C.C. Francis, unpublished). For each year in the ith simulation run with target catch H, the actual catch taken will be $H\varepsilon_i$. This is intended to simulate what would happen if the true virgin abundance was B but was thought to be $B\varepsilon_i$. The ε_i are assumed to be i.i.d., either lognormal or normal with negative values increased to 0, with mean 1 and default c.v. 0.2 (as assumed by Francis 1992, where the normal distribution was used). These errors are not used in sample-based simulations because uncertainty in virgin abundance is meant to be incorporated in the posterior.

3. For both point- and sample-based CAY simulations, the annual stock-assessment uncertainty can be incorporated (as in the stock assessment software *pmod*, R.I.C.C. Francis, unpublished). For each year y in the ith simulation run with target mortality rate H, the catch will be calculated using $F=H\varepsilon_y$. This is intended to simulate what would happen if the true abundance was B but was thought to be $B\varepsilon_y$. The ε_y are assumed to be i.i.d., either lognormal or normal with negative values increased to 0, with mean 1 and default c.v. 0.2 (as assumed by Francis (1992), where the normal distribution was used). Note, that this options has not been implemented for the case where F is an instantaneous mortality rate.

The same random numbers are used for the simulation runs at each harvest rate H. This increases comparability (e.g., between $C_{av}(H_1)$ and $C_{av}(H_2)$) and removes random noise from the $C_{av}(H)$ and $P_{risk}(H)$ curves.

The results of an MCY or CAY analysis depend heavily on the stock-recruitment relationship used. You have to specify one, even if it is ‘none’.

You need to provide initial guesstimates of MCY and F_{CAY}, which are used to start off the minimiser. Providing a value near the ‘true value’ will help the minimiser find a solution faster.

7.5.2 Current surplus production (CSP)

CASAL defines the current surplus production (CSP) as the catch in year `current+1` that would make the projected expectation of post-fishery biomass B_{post} in year `current+1` equal to that in year `current`. The calculation of CSP is hence based on one-year projections (Section 7.3), so you must set `final` to at least `current+1`.

CASAL defines the post-fishery biomass B_{post} as the unselected mature biomass in all areas combined, in the time step of the last fishery, after all mortality has been applied. You can change this definition if you want. If there are multiple fisheries then you have to specify a catch split (the proportion of the catch that must come from each fishery in year `current+1`). The same catch split will be used for deterministic yield calculations (Section 7.4) and for MCY/CAY (Section 7.5.1).

For a multiple-stock model, you have the choice of two approaches. You can either request an overall CSP (as above) or a CSP for each stock, in which case CASAL does the following for each stock *s*:

1. Redefines B_{post} to only include fish of stock *s*.
2. Finds the total catch in year `current+1` which would make the projected expectation of B_{post} in year `current+1` equal to that in year `current`.
3. Returns the expected amount of that catch which comes from stock *s*.

It is possible there will be no CSP, i.e., even if no catch is taken, there is a drop in expected B_{post}.

You need to provide an initial guesstimate of CSP, which is used to start off the minimiser. Providing a value near the ‘true value’ will help the minimiser find a more accurate solution.

8. THE POPULATION.CSL FILE

The population parameters are specified in the `population.csl` file. See Section 5 for information about the population section, and Section 3.4 for instructions on writing a CASAL data file.

8.1 Defining the partition

@size_based	**Should the model be size-based rather than age-based?**
Type	Switch
Default	`False` (i.e., age-based)
Effects	Defines the model as either age-based or size-based.

@n_classes	**Number of size classes**
Conditions	Must be specified in a size-based model. Ignored in an age-based model.
Type	Integer
Effects	Defines the number of size classes

@class_mins	**Size class lower limits (plus the upper limit of the last class if it is not a plus group)**
Conditions	Must be specified in a size-based model. Ignored in an age-based model.
Type	Constant vector
Effects	Defines the lower limits of each of the `n_classes` size classes. If there is no plus group then an additional value defines the upper limit of the last size class.

@min_age, @max_age	**Minimum and maximum age limits**
Conditions	Must be specified in an age-based model. Ignored in a size-based model.
Type	2 x integer
Effects	Defines the minimum and maximum fish age classes.

@plus_group	**Should a plus age or size group be used?**
Type	Switch
Default	`True` (use a plus group)
Effects	Defines the last age or size class as a plus group.

@plus_group_size	**Mean size of plus group**
Conditions	Must be specified in a size-based model with a plus group. Otherwise ignored.
Type	Constant
Effects	Defines the nominal size of the plus group. Used for ogives and for mean weight calculations.

@sex_partition	**Is the partition sex-structured?**
Type	Switch
Default	`False` (the partition is not sex-structured)
Effects	Defines whether sex is a character in the partition

@mature_partition	**Is the partition structured by maturity?**
Type	Switch
Default	`False` (the partition is not structured by maturity)
Effects	Defines whether maturity is a character in the partition

@n_areas	**Number of areas in the partition**
Type	Integer
Default	1 (i.e., a single-area model)
Effects	If `n_areas=1`, then area is not a character in the partition. Otherwise, `n_areas` is the number of areas in the partition.

@area_names	**Area names**
Conditions	Necessary if `n_areas` > 1, otherwise ignored
Type	Vector of strings
Effects	Defines the text label to be associated with each area.

@n_stocks	**Number of stocks in the partition**
Type	Integer
Default	1 (i.e., a single-stock model)
Effects	If `n_stocks=1`, then stock is not a character in the partition. Otherwise, `n_stocks` is the number of stocks in the partition.

@stock_names	**Stock names**
Conditions	Necessary if `n_stocks` > 1, otherwise disallowed
Type	Vector of strings
Effects	Defines the text label to be associated with each stock.

@n_growthpaths	**Number of growth-paths in the partition**
Type	Integer
Default	1 (i.e., not a growth-path model)
Effects	If `n_growthpaths=1`, then growth-path is not a character in the partition. Otherwise, `n_growthpaths` is the number of growth-paths in the partition.

@exclusions_char1	**Partition exclusion term 1**
@exclusions_val1	**Partition exclusion value 1**
@exclusions_char2	**Partition exclusion term 2**
@exclusions_val2	**Partition exclusion value 2**
Conditions	All partition exclusion commands must be used if an exclusion is defined
Type	String vector
Effects	Defines what combinations of characters are excluded from the partition. There is no row in the partition for which the character `exclusions_char1[i]` takes the value named `exclusions_val1[i]` and the character `exclusions_char2[i]` takes the value named `exclusions_val2[i]`
Notes	Exclusions are never necessary but can improve the model's execution speed.
Example	If no females are allowed in area "home", use entries of "sex", "female", "area", and "home" respectively.

@n_tags	**Number of tagging partitions to include in the model**
Type	Integer
Default	0 (i.e., none)
Effects	Number of different tagging events to use in the model. The default specifies that no tagging partition is included
Notes	If defined, you will usually need to define a tag-release event (see `@tag` below) for each of `@tag_names`

@tag_names	**Names of the tagging partition members**
Conditions	Supply only if `@n_tags` > 0
Type	Vector of strings
Effects	Defines the text labels of the tagging partition members.
Notes	The number of entries should equal `@n_tags`. Note that CASAL will always include a partition member with the label `no_tag`. The `no_tag` partition member will be used as both the source and sink for moving fish to or from one of the named tagging partition members. If defined, you will usually need to define a tag-release event (see `@tag` below) for each of `@tag_names`

@tag_shedding_rate	**The tag shedding rate to apply to the tagging partitions in the model**
Conditions	Can be used only if `@tag` is defined
Type	Estimable vector
Default	Vector of zeros of length `@n_tags` (i.e., none)
Effects	Removes fish from the tagging partition members as an instantaneous rate. The default specifies that no tag shedding occurs
Notes	Can only be defined if `@tag` is defined

@tag_loss_props	**Proportion of tag loss that has occurred by each time step**
Conditions	Values must sum to 1. Can be used only if `@tag` is defined
Type	Constant vector
Default	Vector equal to `@annual_cycle.M_props`
Effects	Defines the proportion of the years tag shedding which has occurred by the start of each time step.
Example	If the first entry of `tag_shedding_props` is 0.5, then, in time step 1, the tag shedding applied in time step 1 is calculated as 0.5*`tag_shedding_rate`.

@tag_growth_loss	**The growth loss period to apply to the tag partition member**
Label	The name of a tag partition member
Conditions	Can be used only if `@tag` is defined, and if the growth curve is von Bertalanffy.
Effects	Defines the tag partition member for which to apply a growth loss period
Notes	The `@tag_growth_loss` command should be repeated for each tag partition member where a no growth period is specified.

nogrowth_period	**The period of no growth**
Command	`tag_growth_loss[label]`
Type	Estimable
Effects	Defines the no growth period (i.e., the shift in the t_0 parameter for the von Bertalanffy parameter used for calculating size at age).

8.2 Defining the annual cycle and the time sequence

@initial	**Initial assessment year**
Type	Integer
Effects	Defines the first year of the assessment period.

@current	**Current assessment year**
Type	Integer
Effects	Defines the last year of the assessment period, excluding the projection period (if there is one).

@final	**Final projection year**
Type	Integer
Effects	Defines the last year of the projection period.

@annual_cycle	**Annual cycle block command**
Effects	Defines any following commands as `@annual_cycle` subcommands

time_steps	**Number of time steps**
Command	`annual_cycle`
Type	Integer
Effects	Defines the number of time steps in the annual cycle

recruitment_time	**Time step in which recruitment occurs**
Command	`annual_cycle`
Type	Integer
Effects	Defines the time step in which recruitment occurs

recruitment_areas **Area in which where recruitment occurs, for each stock**

Command	`annual_cycle`
Conditions	Necessary if `n_areas>1`, otherwise ignored
Type	Vector of strings
Effects	Defines the area in which recruitment occurs, for each stock
Notes	Each entry should be an area label as per `area_names`. You need one entry per stock, even if all stocks recruit in the same area.

spawning_time **Time step for recording SSB**

Command	`annual_cycle`
Type	Integer
Effects	Defines the time step in which the value of the spawning stock biomass is recorded

spawning_part_mort **Proportion of mortality in the time step before recording SSB**

Command	`annual_cycle`
Type	Constant
Default	0.5
Effects	Defines the proportion of the time step's mortality episode to apply before recording SSB.
Notes	Should be a real number in [0,1]. Has no effect if there is no mortality in the time step.

spawning_areas **Area for recording SSB, for each stock**

Command	`annual_cycle`
Conditions	Either `spawning_areas` or `spawning_all_areas` or command `@spawning_grounds` is necessary if `n_areas` > 1, otherwise ignored.
Type	String vector of length `n_stocks`
Effects	Defines the area in which to record SSB for each stock.
Notes	Each entry should be an area label as per `area_names`.

spawning_all_areas **Is SSB recorded for all areas combined?**

Command	`annual_cycle`
Conditions	Either `spawning_areas` or `spawning_all_areas` or command `@spawning_grounds` is necessary if `n_areas>1`, otherwise ignored. Only usable in a single-stock model.
Type	Switch
Effects	Defines that SSB is recorded for all areas combined.

spawning_ps **Spawning proportions by age/size class**

Command	`annual_cycle`
Conditions	Specify either `spawning_ps` or `spawning_p`, but not both
Type	Estimable vector
Effects	Defines the factor applied to mature biomass to get the SSB for each stock.
Notes	This must have length equal to the number of stocks. In the special case with only one stock, then use either `spawning_ps` OR `spawning_p`.

spawning_p **Spawning proportion**

Command	`annual_cycle`
Conditions	Specify either `spawning_ps` or `spawning_p`, but not both
Type	Estimable
Effects	Defines the factor applied to mature biomass to get the SSB for each stock.

spawning_use_total_B **Should SSB be defined as total biomass rather than mature biomass?**

Command	`annual_cycle`
Conditions	Can be specified only if maturity is not a partition character
Type	Switch
Default	In single-area models, `false` (SSB = mature biomass). Otherwise, no default.
Effects	Defines the SSB as the total biomass in the spawning area rather than the mature biomass.

Notes	The only reason to use this, we believe, is if you have migrated 'mature' fish to a spawning area when maturity is not a partition character. You know all the fish in the spawning area are mature, but CASAL does not (because the state does not keep track of maturity).

spawning_sex	**The sexes to include in the calculation of spawning stock biomass**
Command	`annual_cycle`
Conditions	Can be used only in a sexed model.
Type	String
Default	`all`
Effects	Defines the sexes to use in the calculation of spawning stock biomass, either `male`, `female` or `all`.
Notes	Only implemented in models with maturity in the partition.

semelparous_time	**Time step to apply semelparous mortality**
Command	`annual_cycle`
Conditions	Can be used only in a model with maturity in the partition
Type	Integer
Effects	Defines the time step in which semelparous mortality is applied.

semelparous_mortality	**Proportion of mature fish assumed to die when applying semelparous mortality**
Command	`annual_cycle`
Conditions	Can be used only in a model with maturity in the partition
Type	Estimable
Default	1.0
Effects	Defines the proportion of all mature fish assumed to die at `annual_cycle.semelparous_time`
Notes	This option will have no effect unless semelparous_time has been defined.

n_growths	**Number of growth episodes per year**
Command	`annual_cycle`
Conditions	Can be used only in a size-based model
Type	Integer
Effects	Defines the number of growth episodes per year

growth_times	**Time step in which each growth episode occurs**
Command	`annual_cycle`
Conditions	Can be used only in a size-based model
Type	Integer
Effects	Defines the time step in which each growth episode occurs
Notes	Number of entries = `annual_cycle.n_growths`

ageing_time	**Time step when age is incremented**
Command	`annual_cycle`
Conditions	Can be used only in an age-based model
Type	Integer
Effects	Defines the time step when ageing occurs

growth_props	**Proportion of growth that has occurred by each time step**
Command	`annual_cycle`
Conditions	Can be used only in an age-based model
Type	Constant vector
Default	Vector of zeros, i.e., no growth between fish birthdays
Effects	Defines the proportion of the year's growth that has occurred by the start of each time step.
Notes	The mean size of fish of age *a* years (rounded down) in the *i*th time step is calculated as if their age was (*a*+`growth_props[i]`). `growth_props[ageing_time]` must be 0, and the entries of `growth_props` have to be non-decreasing between fish birthdays.
Example	If the first entry of `growth_props` is 0.5, then, in time step 1, the mean size of 2-year-old fish is calculated as if they were age 2.5.

M_props	**Proportion of natural mortality that occurs in each time step**
Command	`annual_cycle`
Conditions	Values must sum to 1
Type	Constant vector
Default	No default
Effects	Defines the proportion of the year's natural mortality that occurs in each time step

baranov	**Should fishing mortality be applied simultaneously with natural mortality using the Baranov equation, rather than instantaneously?**
Command	`annual_cycle`
Type	Switch
Default	`False` (i.e., use instantaneous mortality)
Effects	Defines fishing and natural mortality to be applied simultaneously using the Baranov equation, rather than as half a time step's natural mortality, then instantaneous fishing mortality, then the remaining half of the natural mortality.
Notes	You cannot use the Baranov equation if you have more than one fishery in the same area in the same time step.

midmortality_partition	**Method to calculate mortality within the time step**
Command	`annual_cycle`
Type	String
Default	`weighted_sum` if fishing mortality is instantaneous, `weighted_product` if the Baranov equation is used
Effects	Defines how the partition is calculated partway through a mortality episode. Must be either `weighted_sum` or `weighted_product`.
Notes	See Section 5.3. The defaults are usually sensible.

fishery_names	**Names of the fishery**
Command	`annual_cycle`
Type	Vector of strings
Effects	Defines the text labels of the fisheries.

fishery_times	**Time step when each fishery occurs**
Command	`annual_cycle`
Type	Constant vector
Effects	Defines the time step in which each fishery occurs.
Notes	One entry per entry of `fishery_names`.

fishery_areas	**Area where each fishery occurs**
Command	`annual_cycle`
Conditions	Must not be used in the same input file as command `@fishing_grounds`
Type	Vector of strings
Effects	Defines the area in which each fishery occurs.
Notes	If any fishery occurs in more than one area, use command `@fishing_grounds` rather than `fishery_areas`.

n_migrations	**Number of migrations in each year**
Command	`annual_cycle`
Conditions	Can be used only if `n_areas>1`
Type	Integer
Effects	Defines the number of migrations in each year

migration_names	**Names of the fishery**
Command	`annual_cycle`
Conditions	Can be used only if `n_migrations>0`
Type	Vector of strings
Effects	Defines the text labels of the migrations.
Notes	Number of entries should be `n_migrations`.

migration_times	**Time step of each migration**
Command	`annual_cycle`

Conditions	Can be used only if `n_migrations>0`
Type	Constant vector
Effects	Defines the time step in which each migration occurs.
Notes	Number of entries should be `n_migrations`.

migrate_from	**Area from which each migration departs**
Command	`annual_cycle`
Conditions	Can be used only if `n_migrations>0`
Type	Vector of strings
Effects	Defines the source area of each migration
Notes	Each entry should be an area label as per `area_names`. Number of entries should be `n_migrations`.

migrate_to	**Area where each migration arrives**
Command	`annual_cycle`
Conditions	Can be used only if `n_migrations>0`
Type	Vector of strings
Effects	Defines the destination area of each migration
Notes	Each entry should be an area label as per `area_names`. Number of entries should be `n_migrations`.

n_maturations	**Number of maturation episodes in each year**
Command	`annual_cycle`
Conditions	Can be used only if maturity is a character in the partition
Type	Integer
Effects	Defines the number of maturation episodes in each year

maturation_times	**Time step of each maturation episode**
Command	`annual_cycle`
Conditions	Can be used only if `n_maturations>0`
Type	Constant vector
Effects	Defines the time step in which each maturation occurs.
Notes	Number of entries should be `n_maturations`.

disease_mortality_time	**Time step to apply disease mortality**
Command	`annual_cycle`
Conditions	Should only be used if disease mortality is to be applied
Type	Integer
Default	None
Effects	If defined, then the disease mortality is applied in that time step

@fishing_grounds	**Where a fishery takes place**
Label	The name of a fishery
Conditions	If `n_areas` > 1, fishing grounds must be specified using EITHER one `@fishing_grounds` command block for each fishery OR a single subcommand `fishery_areas` of command `@annual_cycle`. The former option must be used if any fishery occurs in more than one area. It is a fatal error to use both `@fishing_grounds` and subcommand `fishery_areas` of command `@annual_cycle` in the same input file.
Effects	Defines the area(s) in which a fishery occurs
Notes	This command is intended for use when there is at least one fishery that occurs in multiple areas. Otherwise, use subcommand `fishery_areas` of command `@annual_cycle`, to specify the area for each fishery.

areas	**The area, or areas where a fishery takes place**
Command	`fishing_grounds[label]`
Type	String vector
Effects	Each entry should be an area label as per `area_names`.

@spawning_grounds	**Where spawning takes place for one or all stocks**
Label	The name of a stock (optional)
Conditions	If `n_areas` > 1, either this command, or one or the other of subcommands `spawning_areas` or `spawning_all_areas` of command `@annual_cycle`, must be used. Ignored if `n_areas` = 1.
Effects	Defines the area(s) in which to record SSB
Notes	If `n_stocks` > 1, use one labelled `spawning_grounds` command for each stock, or a single command for all stocks.

areas	**The areas in which to record SSB**
Command	`spawning_grounds[label]`
Type	String vector
Effects	Each entry should be an area label as per `area_names`.

8.3 Defining recruitment

@y_enter	**Number of years after its birth that a year class enters the partition**
Type	Integer
Effects	Defines the number of years after its birth that a year class enters the partition.
Notes	In an age-based model, there is probably a ‘correct’ value of `y_enter`. You will be warned if you use any other value.
Notes	In an age-based model, the value of `y_enter` depends on the order of the recruitment, ageing, and spawning within the year (see Section 5.4.2). It is a fatal error to use a wrong value.

@standardise_YCS	**Use the Haist parameterisation for YCS?**
Type	Switch
Default	`False` (do not use the Haist parameterisation)
Effects	Defines YCS to use the Haist parameterisation in Section 5.4.2.
Note	Do not use both `standardise_YCS` and `use_mean_YCS`. You will also need to set `@recruitment.first_free` and `@recruitment.last_free` for each stock.

@use_mean_YCS	**Use the Francis parameterisation for YCS?**
Type	Switch
Default	`False` (do not use the Francis parameterisation)
Effects	Defines YCS to use the Francis parameterisation in Section 5.4.2.
Note	Do not use both `standardise_YCS` and `use_mean_YCS`. You will also need to set `@recruitment.first_free` and `@recruitment.last_free` for each stock.

@recruitment	**Recruitment block command**
Label	The name of a stock
Effects	Defines any following commands as `@recruitment` subcommands for the stock
Notes	Omit the stock label if there is only one stock in the model.

YCS	**Year class strengths for the stock**
Command	`recruitment[stock_name]`
Type	Estimable vector
Effects	Defines the year class strengths for the stock
Notes	Entries should correspond to the years given in `YCS_years`. If `@standardise_YCS` is set to `true`, the Haist parameterisation of YCS is used. And If `@use_mean_YCS` is set to `true`, the Francis parameterisation of YCS is used.

YCS_years	**Years for which YCS are provided**
Command	`recruitment[stock_name]`
Type	Constant vector

Effects	Defines the years for which the YCS are provided
Notes	For each entry of YCS there should be a corresponding year in `YCS_years`. The years should cover the range `initial-y_enter+n_rinitial` to `current-y_enter,` or a consecutive subset of these years if the Francis parameterisation is used.

n_rinitial	**Number of years for which $R_{initial}$ is to be used as the YCS**
Command	`recruitment[stock_name]`
Type	Integer
Default	0
Effects	If `n_rinitial=0`, do not use $R_{initial}$ as a YCS. If `n_rinitial>0`, use $R_{initial}$ as the YCS for the first `n_rinitial` years. You then do not need to supply YCS for these year classes.

SR	**Stock-recruitment relationship**
Command	`recruitment[stock_name]`
Type	String
Default	`none`
Effects	Defines the stock-recruitment relationship. Should be `BH` (Beverton-Holt), `Ricker`, or `none`.

steepness	**Steepness parameter of the stock-recruitment relationship**
Command	`recruitment[stock_name]`
Conditions	Only used if `SR=BH` or `Ricker`
Type	Estimable
Effects	Defines the steepness parameter h of the stock-recruitment relationship.

CR	**Climate-recruitment relationship**
Command	`recruitment[stock_name]`
Type	String
Default	`none`
Effects	Defines the stock-recruitment relationship. Should be `exponential`, `arctan`, `logistic`, `none`, `identity`, or `linear-combination`.

Ts	**Climate variable *T***
Command	`recruitment[stock_name]`
Conditions	Only used if there is a climate-recruitment relationship
Type	Constant vector
Effects	Defines the climate variable T for each year
Notes	Entries should correspond to the years given in `Ts_years`.

Ts_years	**Years for which the climate variable *T* is provided**
Command	`recruitment[stock_name]`
Conditions	Only used if there is a climate-recruitment relationship
Type	Constant vector
Effects	Defines the years for which the Ts are provided
Notes	For each entry of Ts there should be a corresponding year in `Ts_years`. The years should at least cover the range `initial-y_enter` to `current-y_enter`, or can extend further forwards at your option.

CR_alpha, CR_beta	**Climate-recruitment parameters alpha and beta**
Command	`recruitment[stock_name]`
Conditions	Only used if there is a `logistic`, `arctan`, or `exponential` climate-recruitment relationship
Type	Estimable
Effects	Defines the values of the α and β parameters in the climate-recruitment relationship

CR_beta2	**Climate-recruitment parameter beta2**
Command	`recruitment[stock_name]`
Conditions	Only used if there is a `logistic` climate-recruitment relationship
Type	Estimable

Effects	Defines the value of the β_2 parameter in the `logistic` climate-recruitment relationship

CR_p **Climate-recruitment parameter p**

Command	`recruitment[stock_name]`
Conditions	Only used if there is a `linear-combination` climate-recruitment relationship
Type	Estimable
Effects	Defines the values of the *p* parameter in the `linear-combination` climate-recruitment relationship

initial_size_mean **Mean size at recruitment (both sexes)**

initial_size_cv **c.v. of size at recruitment (both sexes)**

Command	`recruitment[stock_name]`
Conditions	Only used in a size-based model. Use either `initial_size_mean` and `initial_size_cv`, or the sex-based equivalents.
Type	Estimable
Effects	Define the mean and c.v. of the size of recruiting fish

initial_size_mean_male **Mean size at recruitment (male)**

initial_size_mean_female **Mean size at recruitment (female)**

initial_size_cv_male **c.v. of size at recruitment (male)**

initial_size_cv_female **c.v. of size at recruitment (female)**

Command	`recruitment[stock_name]`
Conditions	Only used in a size-based, sexed model. Alternatively use `initial_size_mean` and `initial_size_cv`.
Type	Estimable
Effects	Define the mean and c.v. of the size of recruiting fish, by sex.

p_male **Proportion of recruits that are male**

Command	`recruitment[stock_name]`
Conditions	Only used in a sex-based model.
Type	Estimable
Default	0.5
Effects	Defines the proportion of recruits that are male

growthpaths **Proportion of recruits on each growth-path**

Command	`recruitment[stock_name]`
Conditions	Only used for growth-path models.
Type	Constant vector
Effects	Defines the proportions of recruits that follow each of the growth-paths
Notes	Should be a vector with one element per growth-path, summing to 1.

first_free, last_free **Range of free YCS defining R_0, with the Haist or Francis YCS parameterisation**

Command	`recruitment[stock_name]`
Conditions	Can be used only if either `standardise_YCS` or `use_mean_YCS` is `true`.
Type	2 x integer
Default	`initial-y_enter+n_rinitial`, `current-y_enter`
Effects	Defines the range of years [`first_free` … `last_free`] over which the average recruitment is defined to be R_0.
Notes	Check that the YCS have come out right by using the `quantities.YCS` output quantity parameter.

exclude_free **Exclude years from the range of free YCS defining R_0 with the Haist YCS parameterisation**

Command	`recruitment[stock_name]`
Conditions	Can be used only if `standardise_YCS` is `true`.
Type	Vector of integers

Default	No default
Effects	Defines years to exclude from within the range of years `[first_free` … `last_free]` over which the average recruitment is defined to be R_0.
Notes	Check that the YCS have come out right by using the `quantities.YCS` output quantity parameter.

8.4 Defining recruitment variability

@randomisation_method	**Randomisation method for recruitment variability in stochastic simulations and projections**
Type	String
Default	No default
Effects	Defines the randomisation method to use in stochastic simulations or projections. Should be `lognormal`, `lognormal-empirical`, `empirical`, or `none`.
Notes	Cannot be `none` for stochastic simulations or projections.

@first_random_year	**For projections, the first year for which YCS are randomised**
Type	Integer
Default	The default is the first year for which YCS are not specified, i.e., `current-y_enter+1`.
Effects	Defines the first year for which YCS are randomised in projections.
Notes	Ignored unless you are doing projections. You may want to set it earlier than the default if the last one or more YCS are poorly estimated (and you are not operating in a Bayesian framework). You cannot set it later than the default. This is the year in which the fish are spawned, not the year in which they recruit.

Recruitment subcommands (specified separately for each stock) follow:

sigma_r	**Standard deviation on the log scale of randomised YCS**
T_sigma_r	**Standard deviation on the log scale of randomised climate data *T***
Command	`recruitment[stock_name]`
Conditions	Only used if `randomisation_method=lognormal`. `T_sigma_r` is only used if there is a climate-recruitment relationship.
Type	Constant
Effects	`sigma_r` defines the standard deviation of log-YCS (σ_r) parameter for use in projections and stochastic simulations. `T_sigma_r` likewise for the climate variable T.
Notes	See also the `rho`, `T_rho`, `T_mean` parameters

rho	**Lag-1 log-scale autocorrelation of randomised YCS**
T_rho	**Lag-1 log-scale autocorrelation of randomised climate data *T***
Command	`recruitment[stock_name]`
Conditions	Only used if `randomisation_method=lognormal` or `lognormal-empirical`. `T_rho` is only used if there is a climate-recruitment relationship.
Type	Constant
Default	0 (i.e., no autocorrelation)
Effects	`rho` defines the autocorrelation of log-YCS (ρ) parameter for use in projections and stochastic simulations. `T_rho` likewise for the climate variable T.

year_range	**Year range from which randomised YCS are resampled**
T_year_range	**Year range from which randomised climate data *T* are resampled**
Command	`recruitment[stock_name]`
Conditions	Only used if `randomisation_method=empirical` or `lognormal-empirical`.
Type	Constant vector
Default	all years in the original YCS, or all years in the original *T*'s
Effects	Defines the year range from which the resampling (nonparametric bootstrap) methods draw their samples, in projections and stochastic simulations.
Notes	Provide just two numbers, the first and last years, for each parameter. The program resamples YCS and *T*'s from these ranges of years (inclusive)

T_mean	**Mean on the linear scale of randomised climate data *T***
Command	`recruitment[stock_name]`
Conditions	Only used if `randomisation_method=lognormal` or `none` and there is a climate-recruitment relationship.
Type	Constant
Effects	`T_mean` defines the mean of randomised climate data *T*, to be used in projections and stochastic simulations.
Notes	You need to provide `T_mean` even if there is no randomisation of climate data. Then all 'randomised' *T*'s are simply set to `T_mean`. See also the `T_sigma_r` parameter.

recruitment_multipler	**Multiply the randomised YCS by an arbitrary multiplier**
Command	`recruitment[stock_name]`
Type	Constant
Default	1
Effects	The recruitment multiplier modifies the randomised YCS by an arbitrary factor. See Section 7.3.1 for detail.

8.5 Defining growth (in a size based model)

@growth	**Growth block command**
Conditions	Only used in a size-based model.
Effects	Defines any following commands as `@growth` subcommands for an episode. The *i*th `@growth` block relates to the *i*th growth episode in a year.

stock	**Stock that the growth episode applies to**
Command	`growth[i]`
Conditions	Only used in a size-based model.
Type	String
Default	All stocks
Effects	If supplied, defines the single stock that the growth episode applies to. If not supplied, the growth episode applies to all stocks equally.

type	**Growth model used by the growth episode**
Command	`growth[i]`
Conditions	Only used in a size-based model.
Default	No default.
Type	String
Effects	Defines the growth model used (either `basic` or `exponential`) or a user supplied growth transition matrix (`matrix`).

g **Reference growths for the growth model**

l **Reference sizes for the growth model**

cv **c.v. for the growth model**

minsigma **Lower bound on sigma for the growth model**

Command	`growth[i]`
Conditions	Only used in a size-based model. Alternatively use the equivalents suffixed by `_male` and `_female`, `_mature` and `_immature`, or `_male_mature`, `_male_immature`, `_female_mature`, `_female_immature` below.
Type	Estimable vector, estimable vector, estimable, estimable (respectively)
Effects	Parameters of the growth model. *g* and *l* should be 2-vectors, with *g* containing reference growths `g_alpha` and `g_beta` and `l` containing reference sizes `l_alpha` and `l_beta`. The `cv` is the c.v. around the reference line and `minsigma` the lower bound on standard deviation around the reference line.

g_male, g_female **Reference growths for the growth model**

l_male, l_female **Reference sizes for the growth model**

cv_male, cv_female **c.v. for the growth model**

minsigma_male, minsigma_female **Lower bound on sigma for the growth model**

Command	`growth[i]`
Conditions	Only used in a sexed, size-based model. Alternatively use the equivalents suffixed by nothing, `_mature` and `_immature`, or `_male_mature`, `_male_immature`, `_female_mature`, `_female_immature`.
Type	2 x estimable vector, 2 x estimable vector, 2 x estimable, 2 x estimable
Effects	Parameters of the growth model, specified by sex. See the unsexed versions above.

g_mature, g_immature **Reference growths for the growth model**

l_mature, l_immature **Reference sizes for the growth model**

cv_mature, cv_immature **c.v. for the growth model**

minsigma_mature, minsigma_immature **Lower bound on sigma**

Command	`growth[i]`
Conditions	Only used in a size-based model with maturity in the partition. Alternatively use the equivalents suffixed by nothing, `_male` and `_female`, or `_male_mature`, `_male_immature`, `_female_mature`, `_female_immature`.
Type	2 x estimable vector, 2 x estimable vector, 2 x estimable, 2 x estimable
Effects	Parameters of the growth model, specified by maturity. See the other versions above.

g_male_mature, etc. **Reference growths for the growth model**

l_male_mature, etc. **Reference sizes for the growth model**

cv_male_mature, etc. **c.v. for the growth model**

minsigma_male_mature, etc. **Lower bound on sigma**

Command	`growth[i]`
Conditions	Only used in a size-based model with sex and maturity in the partition. Alternatively use the equivalents suffixed by nothing, `_male` and `_female`, or `_mature` and `_immature`.
Type	4 x estimable vector, 4 x estimable vector, 4 x estimable, 4 x estimable
Effects	Parameters of the growth model, specified by sex and maturity. See the other versions above.

[size_class] **Row of the matrix for a user supplied growth transition matrix.**

Command	`growth[i]`
Conditions	Only used in a size-based model, if `type=matrix`.
Type	Constant vector

Effects	Define a row of the growth transition matrix. Enter one row per size class, with each row of length `@n_classes`. The subcommand is the size class, which must be from 1 to `@n_classes`.
Notes	Each row defines the proportion of its size class that moved to the *j*th size class, where *j*=1 … `n_classes`.

Example	With `n_classes=4`, you might enter commands as follows,
	`1 0.2 0.8 0.0 0.0`
	`2 0.0 0.4 0.6 0.0`
	`3 0.0 0.0 0.8 0.2`
	`4 0.0 0.0 0.0 1.0`
	Then, for example, 20% of individuals of size class 1 do not grow and the remaining 80 % grow to be size class 2.

8.6 Defining maturation (when maturity is in the partition)

@maturation	**Maturation block command**
Conditions	Only used in a model where maturity is a partition character.
Effects	Defines any following commands as maturation subcommands for an episode. The *i*th `@maturation` block relates to the *i*th maturation episode in a year.

stock	**Stock that the maturation episode applies to**
Command	`maturation[i]`
Conditions	Only used in a model where maturity is a partition character.
Type	String
Default	All stocks
Effects	If supplied, defines the single stock that the maturation episode applies to. If not supplied, the maturation episode applies to all stocks equally.

area	**Area that the maturation episode applies to**
Command	`maturation[i]`
Conditions	Only used in a model where maturity is a partition character.
Type	String
Default	All areas
Effects	If supplied, defines the single area that the maturation episode applies to. If not supplied, the maturation episode applies to all areas equally.

rates_all	**Rates of maturation by age or size class**
Command	`maturation[i]`
Conditions	Only used in a model where maturity is a partition character. Alternatively use sex-specific rates (`rates_male`, `rates_female`)
Type	ogive
Effects	Defines the rates of maturation by age/size class.

rates_male, rates_female	**Rates of maturation by sex and age or size class**
Command	`maturation[i]`
Conditions	Only used in a model where sex and maturity are partition characters. Alternatively use non-sex-specific rates (`rates_all`)
Type	2 x ogive
Effects	Defines the rates of maturation by sex and age/size class.

8.7 Defining maturity (when maturity is not in the partition)

@maturity_props	**Maturity proportion block command**
Conditions	Only used in a model where maturity is not a partition character.
Effects	Defines any following commands as `@maturity_props` subcommands

all	**Maturity proportions by age/size class**
Command	`maturity_props`
Conditions	Only used in a model where maturity is not a partition character. Alternatively use `male/female`.

Type	ogive
Effects	Defines the proportions mature by age/size class.

male, female	**Maturity proportions by sex and age/size class**
Command	`maturity_props`
Conditions	Only used in a sexed model where maturity is not a partition character. Alternatively use `all`.
Type	2 x ogive
Effects	Defines the proportions mature by sex and age/size class.

8.8 Defining migrations

@migration	**Migration block command**
Label	the name of a migration
Conditions	Only used in a multi-area model.
Effects	Defines any following commands as `@migration` subcommands.

stock	**Stock that migrates**
Command	`migration[label]`
Type	String
Default	All stocks
Effects	If supplied, defines the single stock that migrates. If not supplied, all stocks migrate.

migrators	**Whether mature, immature, or both kinds of fish migrate**
Command	`migration[label]`
Type	String
Default	All
Effects	Defines the fish that can migrate, either `mature`, `immature`, or `all`.

prop	**Proportion of applicable fish that migrate**
Command	`migration[label]`
Conditions	Use one of `prop`, `rates_all`, or `rates_male` and `rates_female`.
Type	Estimable
Default	1, unless one of the other rates parameters is defined.
Effects	Defines the proportion of applicable fish (of the right stock and maturity status) which migrate.

rates_all	**Proportion of applicable fish that migrate, by age/size class**
Command	`migration[label]`
Conditions	Use one of `prop`, `rates_all`, or `rates_male` and `rates_female`.
Type	ogive
Effects	Defines the proportion of applicable fish (of the right stock and maturity status) which migrate, by age/size class.

rates_male, rates_female	**Proportion of applicable fish that migrate, by sex and age/size class**
Command	`migration[label]`
Conditions	Only used in a sexed model. Use one of `prop`, `rates_all`, or `rates_male` and `rates_female`.
Type	2 x ogive
Effects	Defines the proportion of applicable fish (of the right stock and maturity status) which migrate, by sex and age/size class.

annual_variation_years	**What are the years to apply an annual variation to the migration rates?**
Command	`migration[label]`
Type	Constant vector
Default	None

Effects	Defines the years for annual migration values. For each entry of `annual_variation_values` there should be a corresponding year `annual_variation_years` The years should be consecutive, and in the range `initial` to `current`.

annual_variation_values What are the annual variation values to apply to the migration rates?

Command	`migration[label]`
Type	Estimable vector
Default	None
Effects	Defines the annual variation values to apply to a migration rate. For each entry of `annual_variation_years` there should be a corresponding value `annual_variation_values`.

annual_variation_first_free, annual_variation_last_free Range of years to use in initialisation

Command	`migration[label]`
Type	Constant
Default	First and last years of `annual_variation_years`
Effects	The annual variation value used in calculating the initial state will be the average of the values for the years between `annual_variation_first_free` and `annual_variation_last_free`, inclusive. The specified years must be in `annual_variation_years`.

annual_variation_year_range Year range from which randomised annual variation values are resampled

Command	`migration[label]`
Type	Constant vector of length 2
Default	First and last years of `annual_variation_years`
Effects	In each year of projections the annual variation value used will be selected at random from the specified range of years. The specified years must be in `annual_variation_years`.

annual_variation_randomisation_method Randomisation method for migration annual variation in stochastic simulations and projects

Command	`migration[label]`
Type	String
Default	No default
Effects	Defines the randomisation method to use in stochastic simulations or projections. Should be `empirical`, or `none`.
Notes	If randomisation method is defined to be none, the average of annual variation between `annual_variation_first_free` and `annual_variation_last_free` is used in stochastic simulations and projection.

S, D Source and destination density-dependence parameters

Command	`migration[label]`
Conditions	Both these parameters must be provided if density dependence is required.
Type	2 x estimable
Effects	Define the dependence on source and destination abundance.
Notes	Typically `S≤0`, `D≥0`. A zero value means no dependence.

wave Is this one wave of a 2-wave migration. If so, is it the first or second?

Command	`migration[label]`
Conditions	There should be a matching migration for the other wave.
Type	Integer
Default	Not a 2-wave migration.
Effects	If 1, defines this migration event as the first wave of a 2-wave migration. If 2, defines this migration event as the second wave of a 2-wave migration.

pwave	**The proportion of fish in the first wave of a 2-wave migration**
Command	`migration[label]`
Conditions	Only used in a multi-area model. There should be a matching migration for the other wave.
Type	Estimable
Default	Not a 2-wave migration.
Effects	Defines the proportion of fish in the first wave of the 2-wave migration this is part of.
Notes	If you are estimating this parameter, you need to use the `same` subcommand in your `estimation.csl` file, to ensure it takes the same value for both waves of the migration.

8.9 Defining natural mortality

@natural_mortality	**Natural mortality block command**
Label	The name of the stock (if there is more than one `@natural_mortality` command block, i.e., natural mortality is applied to different stock differently).
Effects	Defines any following commands as `@natural_mortality` subcommands.

all	**The overall natural mortality rate**
Command	`natural_mortality`
Conditions	Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	Estimable
Effects	Defines the natural mortality rate.

male, female	**The male and female natural mortality rates**
Command	`natural_mortality`
Conditions	Only used in a sex-based model. Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	2 x estimable
Effects	Defines the natural mortality rate by sex.

avg, diff	**The male/female average and male-female difference in natural mortality rates**
Command	`natural_mortality`
Conditions	Only used in a sex-based model. Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	2 x estimable
Effects	Defines the natural mortality rate by sex. The male rate is (`avg+diff/2`). The female rate is (`avg-diff/2`).

mature, immature	**The mature and immature natural mortality rates**
Command	`natural_mortality`
Conditions	Only used in a model with maturity in the partition. Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	2 x estimable
Effects	Defines the natural mortality rate by maturity.

male_mature, etc.	**Natural mortality rates by sex and maturity**
Command	`natural_mortality`
Conditions	Only used in a model with sex and maturity in the partition. Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	4 x estimable
Effects	Defines the natural mortality rate by sex and maturity.

ogive_all	**The overall natural mortality rate as an ogive**
Command	`natural_mortality`
Conditions	Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	ogive
Effects	Defines the natural mortality rate by age/size class.

ogive_male, ogive_female	**The male and female natural mortality rates as ogives**
Command	`natural_mortality`
Conditions	Only used in a sex-based model. Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	2 x ogive
Effects	Defines the natural mortality rate by sex and age/size class.

ogive_avg, ogive_diff	**The male/female average and male-female difference in natural mortality rates, as ogives**
Command	`natural_mortality`
Conditions	Only used in a sex-based model. Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	2 x ogive
Effects	Defines the natural mortality rate by sex and age/size class. For each age/size class, the male rate is `(avg+diff/2)` and the female rate is `(avg-diff/2).`

ogive_mature, ogive_immature	**The mature and immature natural mortality rates, as ogives**
Command	`natural_mortality`
Conditions	Only used in a model with maturity in the partition. Use one of the following:

	`all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	2 x ogive
Effects	Defines the natural mortality rate by maturity and age/size class.

ogive_male_mature, etc.	**Natural mortality rates by sex and maturity, as ogives**
Command	`natural_mortality`
Conditions	Only used in a model with sex and maturity in the partition. Use one of the following: `all`, `male` and `female`, `avg` and `diff`, `mature` and `immature`, `male_mature` and etc., `ogive_all`, `ogive_male` and `ogive_female`, `ogive_avg` and `ogive_diff`, `ogive_mature` and `ogive_immature`, `ogive_male_mature`, etc.
Type	4 x ogive
Effects	Defines the natural mortality rate by sex, maturity, and age/size class.

allow_negative_M	**Allow values of natural mortality that are negative**
Command	`natural_mortality`
Type	Switch
Default	`False`
Effects	Allows natural mortality to be negative. If `false`, CASAL will error out if it estimates or is supplied with a value for *M* that is negative. Otherwise, CASAL will report an error but continue to run.

8.10 Defining fishing mortality

@fishery	**Fishery block command**
Label	the name of a fishery
Effects	Defines any following commands as `@fishery` subcommands

catches	**Catches by year**
Command	`fishery[fishery_name]`
Type	Constant vector
Effects	Defines the catch for the fishery by year
Notes	Entries should correspond to the years given in *years*.

years	**Years for which catches are provided**
Command	`fishery[fishery_name]`
Type	Constant vector
Effects	Defines the years for which the catches are provided
Notes	For each entry of `catches` there should be a corresponding year in *years*. The years should be consecutive, and in the range `initial` to `current`. If the years start after `initial`, then all catches before the first year supplied are taken to be 0.

selectivity	**Name of the selectivity to use**
Command	`fishery[fishery_name]`
Type	String
Effects	Defines the label of the selectivity to use with this fishery, which should be an entry of `selectivity_names`.

retention_selectivity	**Name of the ogive to use for retention if discards are allowed**
Command	`fishery[fishery_name]`
Type	String
Default	None
Effects	Defines the label of the retention ogive to use with this fishery if discards are allowed. Must be an entry of `selectivity_names`. If this subcommand is omitted no discards are allowed for this fishery (so removals = catch).

F_max	**Maximum fishing pressure (Baranov mortality)**
Command	`fishery[fishery_name]`
Conditions	Only used if the Baranov equation is applied.
Type	Constant
Effects	Defines the maximum possible fishing pressure F_{max}.
U_max	**Maximum fishing pressure (instantaneous mortality)**
Command	`fishery[fishery_name]`
Conditions	Only used if the fishing mortality is applied instantaneously. Must be between 0 and 1, i.e. 0 ≤ `U_max` < 1.
Type	Constant
Effects	Defines the maximum possible fishing pressure U_{max}.
Fs	**Instantaneous mortality *F* by year**
Command	`fishery[fishery_name]`
Condition	Only usable with the Baranov catch equation.
Type	Constant vector
Effects	Defines the *F* for the fishery by year (for years in which catches were not available)
Notes	Entries should correspond to the years given in `Fs_years`. Be clear about the meaning of *F*, it is an instantaneous mortality and is multiplied by the selectivity before it is applied.
Fs_years	**Years for which *F*'s are provided**
Command	`fishery[fishery_name]`
Condition	Only usable with the Baranov catch equation.
Type	Constant vector
Effects	Defines the years for which instantaneous mortalities *F* are provided
Notes	For each entry of *Fs* there should be a corresponding year in `Fs_years`. The years should be consecutive, in the range `initial` to `current`, and non-overlapping with `years` (which is the range of years for which catches are provided).
future_catches	**Catches by year in the projection period**
Command	`fishery[fishery_name]`
Type	Constant vector
Effects	Defines the catch for the fishery by year in the projection period. Ignored unless you are doing projections.
Notes	Entries should correspond to the years given in `future_years`.
future_Fs	**Baranov fishing mortality by year in the projection period**
Command	`fishery[fishery_name]`
Condition	Only useable if the Baranov equation is applied.
Type	Constant vector
Effects	Defines the Baranov mortality rate, *F*, for the fishery by year in the projection period. Ignored unless you are doing projections.
Notes	Entries should correspond to the years given in `future_years`.
future_Us	**Instantaneous fishing mortality by year in the projection period**
Command	`fishery[fishery_name]`
Condition	Do not use if the Baranov catch equation is applied. Must be between 0 and 1 inclusive.
Type	Constant vector
Effects	Defines the instantaneous fishing mortality, *U*, for the fishery by year in the projection period. Ignored unless you are doing projections.
Notes	Entries should correspond to the years given in `future_years`.
future_years	**Years for which catches or mortalities are provided in the projection period**
Command	`fishery[fishery_name]`
Type	Constant vector

Effects	Defines the years for which the catches or future mortalities (`future_catches`, `future_Fs`, or `future_Us`) are provided, in the projection period. Ignored unless you are doing projections.
Notes	For each entry of either `future_catches`, `future_Fs`, or `future_Us` there should be a corresponding year in `future_years`. The years should be consecutive, and in the range `current+1` to `final`.

future_constant_catches **The constant catch applied in every year in the projection period**

Command	`fishery[fishery_name]`
Type	Constant
Effects	Defines the constant catch to be applied in every year in the projection period. Ignored unless you are doing projections.
Notes	Use only one of either `future_catches`, `future_Fs`, `future_Us`, `future_constant_Fs`, `future_constant_Us` or `future_constant_catches`. Do not supply `future_years` as these will be set to the range `current+1` to `final` by default.

future_constant_Fs **The constant Baranov mortality rate applied in every year in the projection period**

Command	`fishery[fishery_name]`
Condition	Only usable with the Baranov catch equation.
Type	Constant
Effects	Defines the constant Baranov mortality rate to be applied in every year in the projection period. Ignored unless you are doing projections.
Notes	Use only one of either `future_catches`, `future_Fs`, `future_Us`, `future_constant_Fs`, `future_constant_Us` or `future_constant_catches`. Do not supply `future_years` as these will be set to the range `current+1` to `final` by default.

future_constant_Us **The constant instantaneous mortality rate applied in every year in the projection period**

Command	`fishery[fishery_name]`
Condition	Do not use if the Baranov catch equation is applied. Must be between 0 and 1 inclusive.
Type	Constant
Effects	Defines the constant instantaneous mortality rate to be applied in every year in the projection period. Ignored unless you are doing projections. Must be between 0 and 1 inclusive.
Notes	Use only one of either `future_catches`, `future_Fs`, `future_Us`, `future_constant_Fs`, `future_constant_Us` or `future_constant_catches`. Do not supply `future_years` as these will be set to the range `current+1` to `final` by default.

8.11 Defining disease mortality

@disease_mortality **Disease mortality block command**

Effects	Defines any following commands as `@disease_mortality` subcommands.

DM **Disease mortality rate**

Command	`disease_mortality`
Conditions	Must be supplied if `@annual cycle.disease_mortality_time` is specified
Type	Estimable
Effects	Defines the disease mortality rate.

selectivity	**The selectivity ogive**
Command	`disease_mortality`
Conditions	Must be supplied if `@annual cycle.disease_mortality_time` is specified
Type	ogive
Effects	Defines the selectivity by age/size class (S_{ij}), i.e., the disease mortality to apply to each age/size class in year k is $index[k] \times DM \times S_{ij}$

years	**Years to apply the disease mortality**
Command	`disease_mortality`
Conditions	Must be supplied if `@annual cycle.disease_mortality_time` is specified
Type	Constant vector
Effects	Defines the years in which to apply the disease mortality.

index	**Relative value of the disease mortality by year**
Command	`disease_mortality`
Conditions	Must be supplied if `@annual cycle.disease_mortality_time` is specified
Type	Estimable vector
Effects	Defines the relative value of DM to apply for each year in `@disease_mortality.years`, i.e., the disease mortality to apply to each age/size class in year k is $index[k] \times DM \times S_{ij}$

future_index	**Relative value of the disease mortality by year in the projection period**
Command	`disease_mortality`
Type	Constant vector
Effects	Defines the relative value of DM to apply for each year in the projection period with years `@disease_mortality.future_years`. Ignored unless you are doing projections.

future_years	**Years for which future disease mortality is provided**
Command	`disease_mortality`
Type	Constant vector
Effects	Defines the years in which to apply the disease mortality in the projection period. Ignored unless you are doing projections.
Notes	For each entry of `future_index` there should be a corresponding year in `future_years`. The years should be consecutive, and in the range `current+1` to `final`.

8.12 Defining tag-release events

@tag	**Label for the tagging event**
Label	The label for the tagging event
Conditions	Supply only if `@n_tags` > 0
Effects	Defines any following commands as `@tag` subcommands. This specifies an actual tagging event (i.e., an event in which a single tag type is applied in a single year and time step).
Notes	The label is arbitrary, but should be unique.

tag_name	**The name of the tagging partition member**
Command	`tag[label]`
Conditions	Should be an element of `@tag_names`.
Type	String
Default	No default
Effects	Name of the tagging partition being used in this event.

release_type	**Method for determining the proportions-at-age release in an age-based model**
Command	`tag[label]`
Conditions	Should be either `deterministic` or `free`.
Type	String
Default	`deterministic` in an age-based model or `free` in a size-based model
Effects	Defines the method for determining the tag-release proportions-at-age in an age-based model, as either calculated deterministically or as free parameters. See Section 5.4.8.

area	**Area that the tagging event applies to**
Command	`tag[label]`
Conditions	This should only be supplied in a model when area is in the partition
Type	String
Default	No default
Effects	Defines the area where the tagging event occurs. Should be an area label as per `@annual_cycle.area_names`.

stock	**Stock that the tagging event applies to**
Command	`tag[label]`
Conditions	This should only be supplied in a model when stock is in the partition
Type	String
Default	All stocks
Effects	Defines the stock that is tagged in this tagging event. Should be a stock label from `@annual_cycle.stock_names`.
Notes	Usually the default will be sensible.

sex	**Sex that the tagging event applies to**
Command	`tag[label]`
Conditions	This should only be supplied in a model when sex is in the partition
Type	String
Default	`both`
Effects	Defines the sex which gets tagged in the tagging event. Should be one of male, female, or both.

year	**Year of the tagging event**
Command	`tag[label]`
Type	Integer
Default	No default
Effects	Defines the year when the tag event is applied. Should be a single number in the range from `initial` to `current` inclusive.

step	**Time step of the tagging event**
Command	`tag[label]`
Type	Integer
Default	No default
Effects	Defines the time step when the tagging event is applied.

mature_only	**Does the tagging event apply to mature or all fish?**
Command	`tag[label]`
Conditions	This should only be supplied in a model when maturity is in the partition
Type	Switch
Default	`False`
Effects	Defines the maturity state of fish which can be tagged. If true, then only mature fish will be tagged, and if false, then both mature and immature will be tagged.

ogive	**Which selectivity ogive should be applied?**
Command	`tag[label]`
Conditions	Age-based model with `release_type=deterministic`
Type	String
Default	No selectivity
Effects	Defines which selectivity ogive should be applied.

number	**The number of fish actually tagged in the tagging event**
Command	`tag[label]`
Type	Integer
Default	No default
Effects	Defines the number of fish that were tagged in the tagging event.

class_mins	**What are the size bins of the observations (in an age-based model)?**
Command	`tag[label]`
Conditions	Age-based model with `release_type=deterministic`
Type	Constant vector
Effects	Defines the lower limits of each of the size classes. If there is no plus group then an additional value defines the upper limit of the last size class.
Notes	If the observations are sexed, both sexes share the same list of size classes.

plus_group	**Is the last age or size class a plus group?**
Command	`tag[label]`
Type	Switch
Default	True (it is a plus group)
Effects	Defines the last age or size class as a plus group.

log_props	**Are the props_all, props_male, or props_female relative rates specified in log space?**
Command	`tag[label]`
Conditions	Age-based model with `release_type=free`
Type	Switch
Default	False
Effects	Defines that the proportions specified in `props_all` or `props_male` and `props_female` are in log space rather than natural space.
Notes	This option may provide better MCMC performance when estimating the proportions-at-age tagged, particularly when only a few of the available age classes are tagged in the tagging event

props_all	**Relative rates of all fish tagged in the tagging event**
Command	`tag[label]`
Conditions	Use either `props_all` or `props_male` and `props_female`, but not both.
Type	Estimable vector
Default	No default
Effects	Defines either the relative proportions at age, or proportions at size that were tagged in the tagging event.
Notes	Provide a vector that has one entry for each of the age classes (in an age-based model) or size classes (in a size-based model).

props_male, props_female	**Relative proportions of male or female fish tagged in the tagging event**
Command	`tag[label]`
Conditions	Only used in a sex based model. Use either `props_all` or `props_male` and `props_female`, but not both.
Type	2 x estimable vector
Default	No default
Effects	Defines either the sex based relative proportions at age, or proportions at size that were tagged in the tagging event.
Notes	Provide a vector that has one entry for each of the age classes (in an age-based model) or size classes (in a size-based model).

mortality	**Proportion of tagged fish that are removed immediately after tagging**
Command	`tag[label]`
Type	Estimable constant
Default	0
Effects	Defines the proportion of tagged fish which are removed from the partition (i.e., die or lose their tags) immediately after the tagging event.
Notes	This is a proportion, between 0 and 1, not an instantaneous mortality rate. The rate is applied equally to all sizes, ages, sexes, etc.

8.13 Defining selectivities

@selectivity_names	**List of selectivity names**
Type	Vector of strings
Conditions	The label `none` cannot be used as a label for a selectivity
Effects	Lists the labels of all the selectivities in the model.

@selectivity	**Selectivity block command**
Label	the name of a selectivity
Effects	Defines any following commands as `@selectivity` subcommands

all	**The selectivity ogive**
Command	`selectivity[name]`
Conditions	Use one of the following: `all`, `male` and `female`, `mature` and `immature`, `male_mature`, etc.
Type	ogive
Effects	Defines the selectivity by age/size class.

male, female	**The selectivity ogives by sex**
Command	`selectivity[name]`
Conditions	Only used in a sex-based model. Use one of the following: `all`, `male` and `female`, `mature` and `immature`, `male_mature`, etc.
Type	2 x ogive
Effects	Defines the selectivity by sex and age/size class.

mature, immature	**The selectivity ogives by maturity**
Command	`selectivity[name]`
Conditions	Only used in a model with maturity in the partition. Use one of the following: `all`, `male` and `female`, `mature` and `immature`, `male_mature`, etc.
Type	2 x ogive
Effects	Defines the selectivity by maturity and age/size class.

male_mature, etc.	**The selectivity ogives by sex and maturity**
Command	`selectivity[name]`
Conditions	Only used in a model with sex and maturity in the partition. Use one of the following: `all`, `male` and `female`, `mature` and `immature`, `male_mature`, etc.
Type	4 x ogive
Effects	Defines the selectivity by sex, maturity, and age/size class.

shift_E	**Exogenous selectivity shift variable E**
Command	`selectivity[name]`
Type	Estimable vector
Effects	Defines the value of the exogenous variable used to shift the fishery selectivity.
Notes	Entries should correspond to the years given in `shift_years`.

shift_years	**Years for which exogenous selectivity shift variable E is provided**
Command	`selectivity[name]`
Type	Constant vector
Effects	Defines the years for which the shift variable E is provided.
Notes	For each entry of `shift_E` there should be a corresponding year in `shift_years`. If there is no entry, no shift is carried out for that year.

shift_a	**Exogenous selectivity shift parameter a**
Command	`selectivity[name]`
Type	Estimable
Effects	Defines the value of the selectivity shift parameter

8.14 Setting the initial state

@n_equilibrium	**Number of years of running the equilibrium model**
Conditions	Only used for a size based model
Type	Integer
Effects	Defines the number of years that the equilibrium model is run when setting the initial state
Notes	Try some different values, if it makes a difference then you are probably too low.

@Rinitial_is_deviate	**Is $R_{initial}$ supplied relative to R_0?**
Conditions	Not used unless $R_{initial}$ is supplied.
Type	Switch
Default	`False` (i.e., $R_{initial}$ is supplied as an absolute number)
Effects	Defines if $R_{initial}$ is supplied relative to R_0, rather than as an absolute number

@initialization	**Initialization block command**
Label	the name of a stock
Effects	Defines any following commands as `@initialization` subcommands for the stock
Notes	Omit the stock label if there is only one stock in the model.

B0	**Equilibrium abundance B_0**
Command	`initialization[stock_name]`
Conditions	Define either `B0` for each stock or `R0` for each stock or, in a two-stock model, ((`B0_total` or `log_B0_total`) and `B0_prop_stock1`) OR ((`R0_total or log_R0_total`) and `R0_prop_stock1`)
Type	Estimable
Effects	Defines the value of B_0
Notes	Not to be used if `use_mean_YCS` is `true`

R0	**Equilibrium recruitment R_0**
Command	`initialization[stock_name]`
Conditions	Define either `B0` for each stock or `R0` for each stock or, in a two-stock model, ((`B0_total` or `log_B0_total`) and `B0_prop_stock1`) OR ((`R0_total` or `log_R0_total`) and `R0_prop_stock1`)
Type	Estimable
Effects	Defines the value of R_0
Notes	Not to be used if `use_mean_YCS` is `true`

Bmean	**Equilibrium abundance B_{mean} corresponding to R_{mean}**
Command	`initialization[stock_name]`
Conditions	Define either B_{mean} for each stock or R_{mean} for each stock or, in a two-stock model, ((`Bmean_total` or `log_Bmean_total`) and `Bmean_prop_stock1`) OR ((`Rmean_total or log_Rmean_total`) and `Rmean_prop_stock1`)
Type	Estimable
Effects	Defines the value of B_{mean}
Notes	Can only be used when `use_mean_YCS` is `true`

Rmean	**Expected recruitment in any year, R_{mean}**
Command	`initialization[stock_name]`
Conditions	Define either `Bmean` for each stock or `Rmean` for each stock or, in a two-stock model, ((`Bmean_total` or `log_Bmean_total`) and `Bmean_prop_stock1`) OR ((`Rmean_total or log_Rmean_total`) and `Rmean_prop_stock1)`
Type	Estimable
Effects	Defines the value of R_{mean}
Notes	Can only be used when `use_mean_YCS` is `true`

Binitial	**Initial abundance $B_{initial}$**
Command	`initialization[stock_name]`
Conditions	Either provide `Binitial` for each stock, or `Rinitial` for each stock, or `Cinitial` for each stock, or `Cinitial_male` and `Cinitial_female` for each stock, or none of them.
Type	Estimable
Effects	Defines the value of $B_{initial}$

Rinitial	**Initial recruitment $R_{initial}$**
Command	`initialization[stock_name]`
Conditions	Either provide `Binitial` for each stock, or `Rinitial` for each stock, or `Cinitial` for each stock, or `Cinitial_male` and `Cinitial_female` for each stock, or none of them.
Type	Estimable
Effects	Defines the value of $R_{initial}$.

Cinitial	**Initial number in each age/size class $C_{initial}$**
Command	`initialization[stock_name]`
Conditions	Either provide `Binitial` for each stock, or `Rinitial` for each stock, or `Cinitial` for each stock, or `Cinitial_male` and `Cinitial_female` for each stock, or none of them.
Type	ogive
Effects	Defines the values of $C_{initial,\ i}$, i.e., the initial number in each age or size class *i*.
Notes	You almost certainly want to use an `allvalues` ogive, as other options make little sense. The $C_{initial}$ value you set for the first age/size class is related to the ordering of ageing and recruitment in your annual cycle (see Section 5.5).

Cinitial_male, Cinitial_female	**Initial number in each age/size class $C_{initial}$ for each sex**
Command	`initialization[stock_name]`
Conditions	Either provide `Binitial` for each stock, or `Rinitial` for each stock, or `Cinitial` for each stock, or `Cinitial_male` and `Cinitial_female` for each stock, or none of them.
Type	2 x ogive
Effects	Defines the values of $C_{initial,\ i}$, i.e., the initial number in each age or size class *i*, for each sex.
Notes	You almost certainly want to use `allvalues` ogives as other options make little sense. The $C_{initial}$ value you set for the first age/size class is related to the ordering of ageing and recruitment in your annual cycle (see Section 5.5).

@B0_total, @log_B0_total, @R0_total, @log_R0_total	
Conditions	In a two-stock model only, you can use one of these parameters instead of supplying `B0` or `R0`. You must supply the appropriate one of `B0_prop_stock1` or `R0_prop_stock1`.
Type	Estimable
Effects	Defines the value of B_0, $\log(B_0)$, R_0, or $\log(R_0)$, summed across the two stocks.

@B0_prop_stock1, @R0_prop_stock1	
Conditions	In a two-stock model only, you can use one of these parameters instead of supplying `B0` or `R0`. You must supply one of `B0_total`, `log_B0_total`, `R0_total`, `log_R0_total`.
Type	Estimable
Effects	Defines the proportion of the equilibrium abundance or recruitment which is of the first stock.

8.15 Defining ogive preferences

@n_quant	**Number of points at which to evaluate size-based ogives in an age-based model**
Conditions	Only used in an age-based model which uses size-based ogives.
Type	Integer
Default	5
Effect	Defines the number of points used in the approximation to the integral of the ogive over the distribution of sizes at age.
Notes	The default should normally be adequate, unless you have ogives which change very steeply.

8.16 Defining size-at-age

@size_at_age_type	**Size-at-age model type**
Conditions	Only used in an age-based model.
Type	String
Effects	Defines the size-at-age model used. So far there is `von_Bert` and `Schnute`, for when growth curves are used, and `data`, for when size-at-age data are provided for one or more years.

@size_at_age_years	**Years for which mean-size-at-age data are provided**
Conditions	Only used in an age-based model with `size_at_age_type=data`.
Type	Constant vector
Effects	Defines the list of years for which mean-size-at-age data are provided.
Notes	Need not be consecutive. Can include any years from *initial* to *final*.

@size_at_age_miss	**Controls the treatment of years for which mean-size-at-age data are not provided**
Conditions	Only used in an age-based model with `size_at_age_type=data`.
Type	String
Default	`mean`
Effects	There are four options: `mean`, `interp`, `interp.mean`, `mean.interp`.
Notes	With `mean`, missing years are always replaced by the mean data for provided years. With `interp`, both internal and external gaps are filled using interpolation. With `interp.mean`, internal gaps are filled using interpolation and external gaps with the mean. With `mean.interp`, internal gaps are filled with the mean and external gaps using interpolation.

@size_at_age_step	**Time step for which size-at-age data are provided**
Conditions	Only used in an age-based model with `size_at_age_type=data`.
Type	Integer
Effects	Defines the time step in which the size-at-age data provided are accurate.
Notes	If any element of `annual_cycle.growth_props` is nonzero, then fish grow between birthdays. In this case, the mean-size-at-age differs between time steps, and mean sizes in time steps other than `size_at_age_step` are filled in by interpolation, based on the two years of data bracketing them.

@size_at_age_dist	**Distribution of sizes-at-age around the mean**
Conditions	Only used in an age-based model.
Type	String
Default	No variation of sizes-at-age around the mean
Effects	Defines the distribution of sizes-at-age around the mean. Can be `normal` or `lognormal`. If you don't choose either, there is no variation of size-at-age around the mean. See `size_at_age[stock_name].by_length` for specifying either age or size interpolation.
Notes	Also used for the likelihood of age-size observations (see Section 6.6). c.v.s are specified within the `@size_at_age` blocks.

@size_at_age	**Size-at-age block command**
Label	the name of a stock (optional)
Conditions	Only used in an age-based model.
Effects	Defines any following commands as `@size_at_age` subcommands for the stock
Notes	Omit the stock label if there is only one stock in the model, or if all the stocks have the same size-at-age.

k, t0, Linf	**von Bertalanffy parameters**
Command	`size_at_age[stock_name]`
Conditions	Only used in an age-based model with `size_at_age_type=von_Bert`. Use either `k`, `t0`, `Linf` or `k_male`, `t0_male`, `Linf_male`, `k_female`, `t0_female`, `Linf_female`.
Type	3 x estimable vector (*NOT* estimable constant)
Effects	Defines the von Bertalanffy parameters. If this is not a growth-path model, each parameter is a 1-vector. If this is a growth-path model, each parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Warning	Make sure that you have specified the relationship in units that are compatible with the catches and the size-weight relationship (see section 5.8 and 5.9)

k_male, t0_male, Linf_male, k_female, t0_female, Linf_female	**von Bertalanffy parameters by sex**
Command	`size_at_age[stock_name]`
Conditions	Only used in an age-based model with `size_at_age_type=von_Bert`. Use either `k`, `t0`, `Linf` or `k_male`, `t0_male`, `Linf_male`, `k_female`, `t0_female`, `Linf_female`.
Type	6 x estimable vector (*NOT* estimable constant)
Effects	Defines the von Bertalanffy parameters by sex. If this is not a growth-path model, each parameter is a 1-vector. If this is a growth-path model, each parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Warning	Make sure that you have specified the relationship in units that are compatible with the catches and the size-weight relationship (see section 5.8 and 5.9)

y1, y2, tau1, tau2, a, b Schnute parameters

Command	`size_at_age[stock_name]`
Conditions	Only used in an age-based model with `size_at_age_type=Schnute`. Alternatively use the versions suffixed `_male`, `_female`.
Type	6 x estimable vector (*NOT* estimable constant)
Effects	Defines the Schnute parameters. If this is not a growth-path model, each parameter is a 1-vector. If this is a growth-path model, each parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Warning	Make sure that you have specified the relationship in units that are compatible with the catches and the size-weight relationship (see section 5.8 and 5.9)

y1_male, y2_male, tau1_male, tau2_male, a_male, b_male, y1_female, y2_female, tau1_female, tau2_female, a_female, b_female Schnute parameters by sex

Command	`size_at_age[stock_name]`
Conditions	Only used in an sex/age model with `size_at_age_type=Schnute`. Alternatively use the versions without suffixes `_male`, `_female`.
Type	12 x estimable vector (*NOT* estimable constant)
Effects	Defines the Schnute parameters by sex. If this is not a growth-path model, each parameter is a 1-vector. If this is a growth-path model, each parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Warning	Make sure that you have specified the relationship in units that are compatible with the catches and the size-weight relationship (see section 5.8 and 5.9)

male_[year], female_[year] Mean-size-at-age of male and female fish in [year]

Command	`size_at_age[stock_name]`
Conditions	Only used with `size_at_age_type=data` in a sexed model. Alternatively use the unsexed version `all_[year]`.
Type	2 x constant vector
Effects	Defines the mean-size-at-age data for males and females in [year]. There should be both male and female data for each year in `size_at_age_years`, and each set of data should have one entry for each age class.
Warning	Make sure that you have specified the relationship in units that are compatible with the catches and the size-weight relationship (see section 5.8 and 5.9)
Example	If you have sexed mean-size-at-age data for 1997, 1998, and 1999, and `min_age=2`, `max_age=6`, you would put, e.g.,

```
@size_at_age_type data
@size_at_age_years 1997 1998 1999
@size_at_age
# age 2 3 4 5 6+
male_1997 20 40 50 55 58
female_1997 40 80 100 110 130
male_1998 18 36 48 52 55
```

… (three more rows)

all_[year] Mean-size-at-age of fish of both sexes in [year]

Command	`size_at_age[stock_name]`
Conditions	Only used with `size_at_age_type=data`. Alternatively use the sexed versions `male_[year]`, `female_[year]`.
Type	Constant vector
Effects	Defines the mean-size-at-age data in [year], applied to both sexes. There should be data for each year in `size_at_age_years`, and each set of data should have one entry for each age class.
Warning	Make sure that you have specified the relationship in units that are compatible with the catches and the size-weight relationship (see section 5.8 and 5.9)
Example	If you have mean-size-at-age data for 1997, 1998, and 1999, and `min_age=2`, `max_age=6`, you would put, e.g.,

```
@size_at_age_type data
@size_at_age_years 1997 1998 1999
@size_at_age
# age 2 3 4 5 6+
all_1997 20 40 50 55 58
all_1998 23 45 52 58 63
all_1999 18 36 48 52 55
```

cv **c.v. of sizes-at-age around the mean**

Command	`size_at_age[stock_name]`
Conditions	Only used in an age-based model where `size_at_age_dist` has been supplied. Alternatively use the versions suffixed `_male`, `_female`, or the alternative parameterisation `cv1` and `cv2`, or `sd1` and `sd2`.
Type	Estimable vector (*NOT* estimable constant)
Effects	Defines the c.v. of sizes around the mean. If this is not a growth-path model, the parameter is a 1-vector. If this is a growth-path model, the parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Notes	Also used for the likelihood of age-size observations (see Section 6.6).

cv_male, cv_female **c.v. of sizes-at-age around the mean, by sex**

Command	`size_at_age[stock_name]`
Conditions	Only used in a sexed, age-based model where `size_at_age_dist` has been supplied. Alternatively use `cv` without suffixes, or the alternative parameterisation `cv1_` and `cv2_`, or `sd1_` and `sd2_`.
Type	2 x estimable vector (*NOT* estimable constant)
Effects	Defines the c.v. of sizes around the mean, by sex. If this is not a growth-path model, each parameter is a 1-vector. If this is a growth-path model, each parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Notes	Also used for the likelihood of age-size observations (see Section 6.6).

cv1, cv2 **c.v. of sizes-at-age around the mean for the minimum and maximum age class in the partition**

Command	`size_at_age[stock_name]`
Conditions	Only used in an age-based model where `size_at_age_dist` has been supplied. Alternatively use the versions suffixed `_male`, `_female`, or the alternative parameterisation `sd1` and `sd2`, or `cv`.
Type	Estimable vector (*NOT* estimable constant)
Effects	Defines the c.v. of sizes around the mean. Here, the c.v. used for age *i* is c.v. that results from a linear interpolation (as a function of either age or mean size) of cv1 at @min_age to cv2 at @max_age. If this is not a growth-path model, the parameter is a 1-vector. If this is a growth-path model, the parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Notes	Also used for the likelihood of age-size observations (see Section 6.6). See `size_at_age[stock_name].by_length` for specifying either age or size interpolation.

cv1_male, cv2_male, cv1_female, cv2_female	**c.v. of sizes-at-age around the mean, by sex, for the minimum and maximum age class in the partition**
Command	`size_at_age[stock_name]`
Conditions	Only used in a sexed, age-based model where `size_at_age_dist` has been supplied. Alternatively use `cv1` and `cv2` without suffixes, or the alternative parameterisation `cv_`, or `sd1_` and `sd_2`.
Type	2 x estimable vector (*NOT* estimable constant)
Effects	Defines the c.v. of sizes around the mean, by sex. Here, the c.v. used for age *i* is c.v. that results from a linear interpolation (as a function of either age or mean size) of cv1 at @min_age to cv2 at @max_age. If this is not a growth-path model, each parameter is a 1-vector. If this is a growth-path model, each parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Notes	Also used for the likelihood of age-size observations (see Section 6.6). See `size_at_age[stock_name].by_length` for specifying either age or size interpolation.

sd1, sd2	**c.v. of sizes-at-age around the mean for the minimum and maximum age class in the partition**
Command	`size_at_age[stock_name]`
Conditions	Only used in an age-based model where `size_at_age_dist` has been supplied. Alternatively use the versions suffixed `_male`, `_female`, or the alternative parameterisation `cv`, or `cv1` and `cv2`.
Type	Estimable vector (*NOT* estimable constant)
Effects	Defines the standard deviation of sizes around the mean. Here, the s.d. used for age *i* is s.d. that results from a linear interpolation (as a function of either age or mean size) of sd1 at @min_age to sd2 at @max_age. If this is not a growth-path model, the parameter is a 1-vector. If this is a growth-path model, the parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Notes	Also used for the likelihood of age-size observations (see Section 6.6). See `size_at_age[stock_name].by_length` for specifying either age or size interpolation.

sd1_male, sd2_male, sd1_female, sd2_female	**s.d. of sizes-at-age around the mean, by sex, for the minimum and maximum age class in the partition**
Command	`size_at_age[stock_name]`
Conditions	Only used in a sexed, age-based model where `size_at_age_dist` has been supplied. Alternatively use `sd1` and `sd2` without suffixes, or the alternative parameterisation `cv`, or `cv1` and `cv2`.
Type	2 x estimable vector (*NOT* estimable constant)
Effects	Defines the standard deviation of sizes around the mean, by sex. Here, the s.d. used for age *i* is s.d. that results from a linear interpolation (as a function of either age or mean size) of sd1_ at @min_age to sd2_ at @max_age. If this is not a growth-path model, each parameter is a 1-vector. If this is a growth-path model, each parameter can either be a 1-vector holding the common value for all paths, or a vector with element *i* holding the value for path *i*.
Notes	Also used for the likelihood of age-size observations (see Section 6.6). See `size_at_age[stock_name].by_length` for specifying either age or size interpolation.

by_length	**Specifies if the linear interpolation of c.v.s or s.d.s is a linear function of mean size or of age.**
Command	`size_at_age[stock_name]`
Conditions	Only used in an age-based model where `size_at_age_dist` has been supplied.
Type	Switch
Default	`False`

Effects	Defines the method of linear interpolation of the c.v.s or standard deviations of sizes around the mean size. If `false`, then the c.v.s or s.d.s applied at age *i* are the results from a linear interpolation of cv1 or sd1 at @min_age to cv2 or sd2 at @max_age. If `true`, then the c.v.s or s.d.s applied at age *i* are the results from a linear interpolation of cv1 or sd1 at the mean size of fish at minimum age to cv2 or sd2 at the mean size of fish at maximum age. Note that minimum and maximum age of a fish will also depend on the amount of growth assigned to individual time steps. (see the `growth_props` subcommand of `@annual_cycle`).
Notes	Also used for the likelihood of age-size observations (see Section 6.6).

@annual_growths	**Use annual growth variation. Amount of an average year's growth that occurs in each year**
Conditions	Only used in an age-based model with `size_at_age_type` not equal to `data`.
Type	Estimable vector
Default	No annual growth variation.
Effects	Defines that annual growth variation is to be used, and defines the proportion of average annual growth that occurs in each year.
Notes	Entries should correspond to the years given in `annual_growth_years`. The first entry must be 1, i.e., a year of ordinary growth. If this is not the case, prefix one more year onto `annual_growth_years`. Warning, watch out for off-by-one errors.

@annual_growth_years	**Years for which annual growths are provided**
Conditions	Only used in an age-based model, where `annual_growths` is set.
Type	Constant vector
Effects	Defines the years for which the annual growth increments are provided
Notes	For each entry of `annual_growths` there should be a corresponding year in `annual_growth_years`. The years should be consecutive, and contained in the range `initial` to `current`. All annual growth increments not supplied are taken to be 1.

8.17 Defining the size-weight relationship

@size_weight	**Size-weight block command**
Label	the name of a stock (optional)
Effects	Defines any following commands as `@size-weight` subcommands for the stock
Notes	Omit the stock label if there is only one stock in the model, or if all the stocks have the same size-weight.

type	**The size-weight relationship function**
Command	`size_weight[stock_name]`
Type	String
Default	`basic`
Effects	Defines the type of size-weight relationship used in the model. The only option available is `basic`.

a, b **The size-weight parameters *a* and *b***

Command	`size_weight[stock_name]`
Conditions	Only used with the `basic` size-weight relationship. Alternatively use the versions suffixed `_male`, `_female`.
Type	2 x constant
Effects	Defines the *a* and *b* parameters of the size-weight relationship
Notes	If you provide your catch in tonnes, and your growth curve in centimetres, then *a* should be on the right scale to convert a length in centimetres to a weight in tonnes.
Warning	Make sure that you have specified the relationship in units that are compatible with the catches and the growth parameters (see `verify_size_weight` and section 5.8 and 5.9)

a_male, b_male, a_female, b_female **The size-weight parameters *a* and *b*, by sex**

Command	`size_weight[stock_name]`
Conditions	Only used with the `basic` size-weight relationship. Alternatively use the unsuffixed versions.
Type	4 x constant
Effects	Defines the *a* and *b* parameters of the size-weight relationship, by sex
Notes	If you provide your catch in tonnes, and your growth curve in centimetres, then *a* should be on the right scale to convert a length in centimetres to a weight in tonnes.
Warning	Make sure that you have specified the relationship in units that are compatible with the catches and the growth parameters (see `verify_size_weight` and section 5.8 and 5.9)

verify_size_weight **Verify the supplied size-weight relationship and units**

Command	`size_weight[stock_name]`
Conditions	Only used with the `basic` size-weight relationship.
Type	Constant vector defining the three parameters (1) fish length (in cm), (2) lower bound on weight at this length (in kg), and (3) upper bound on weight at this length (in kg).
Default	The weight of a fish that is 25 cm is calculated and compared against a lower bound of 0.05 kg and an upper bound of 5 kg. If the weight is outside these bounds then a warning message is given.
Effects	If the fish weight at the given fish length is outside the bounds then the program halts with an error message. Otherwise this fish weight and length are given as part of the output.
Notes	The calculation of the fish weight is done under the assumption that the catch is in tonnes, and the growth curve in centimetres. (see section 5.8 and 5.9)

@weightless_model **Is this a model which does not involve fish weight?**

Type	Switch
Default	`False` (i.e., the model involves fish weight)
Effects	Defines all fish to have a nominal weight of 1.
Notes	As a result, CASAL interprets your catches as numbers of fish rather than tonnes of fish. “Biomass”, including SSB and abundance observations, now also represents numbers of fish.

8.18 Other commands

@comment **User supplied comment**

Type	String
Default	None
Effects	Places a copy of the supplied text into any output files generated
Notes	Allows the user to place a comment into the CASAL output files. See also the `@comment` command in the `estimation.csl` and `output.csl` files

9. THE ESTIMATION.CSL FILE

The estimation parameters are specified in the `estimation.csl` file. See Section 6 for information about the estimation section, and Section 3.4 for instructions on writing a CASAL data file.

9.1 Defining the estimation method

@estimator	**Choice of estimation method**
Type	String
Effects	Defines the estimation method as either `likelihood` or `Bayes`.

9.2 Defining point estimation

@max_iters	**Maximum number of iterations in the minimiser**
Condition	Only used in point estimation.
Type	Integer
Default	300
Effects	Defines the maximum number of quasi-Newton iterations allowed in a minimization.

@max_evals	**Maximum number of evaluations in the minimiser**
Condition	Only used in point estimation.
Type	Integer
Default	1000
Effects	Defines the maximum number of objective function evaluations allowed in a minimization.

@max_iters_intermediate	**Maximum number of iterations in early phases**
Condition	Only used in multi-phase point estimation.
Type	Integer
Default	`max_iters`
Effects	Defines the maximum number of quasi-Newton iterations allowed in all but the last phase of a multi-phase minimization.

@max_evals_intermediate	**Maximum number of evaluations in early phases**
Condition	Only used in multi-phase point estimation.
Type	Integer
Default	`max_evals`
Effects	Defines the maximum number of objective function evaluations allowed in all but the last phase of a multi-phase minimization.

@grad_tol	**Minimiser convergence threshold**
Condition	Only used in point estimation.
Type	Constant
Default	0.002
Effects	Defines the convergence criterion for minimization. The minimiser converges successfully if the maximum absolute gradient, of the objective function with regard to the transformed free parameters, divided by the absolute value of the objective function, is less than `grad_tol`. In other words, make the convergence criterion more severe by decreasing `grad_tol`.
Notes	The minimiser also converges and claims it is successful if the quasi-Newton stepsize becomes very small. We are not sure what the implications of this result are yet. If in doubt, act as if the minimiser had not converged — do more runs from different starting points.

9.3 Defining likelihood or posterior profiling

@profile	**Profile block command**
Conditions	Only used in likelihood or posterior profiling.
Effects	Defines any following commands as `@profile` subcommands
Notes	The *i*th `@profile` block relates to the *i*th parameter to be profiled.

parameter	**Name of the parameter to be profiled**
Command	`profile[i]`
Conditions	Only used in likelihood or posterior profiling.
Type	String
Effects	Defines the name of the *i*th parameter to be profiled

element	**Element of the ogive or vector parameter to be profiled**
Command	`profile[i]`
Conditions	Only used in likelihood or posterior profiling, and only if the parameter being profiled is an ogive or vector parameter.
Type	String
Effects	Defines the *i*th element of the ogive or vector parameter to be profiled

n	**Number of values at which to profile the parameter**
Command	`profile[i]`
Conditions	Only used in likelihood or posterior profiling.
Type	Integer
Default	10
Effects	Defines the number of values at which to profile the parameter. See `l` and `u`.

l, u	**Range of values at which to profile the parameter**
Command	`profile[i]`
Conditions	Only used in likelihood or posterior profiling.
Type	2 x constant
Default	`l` = lower bound on parameter plus (range of parameter/2*n*) `u` = upper bound on parameter less (range of parameter/2*n*)
Effects	Defines the lower and upper values at which to profile the parameter. See `n`.

9.4 Defining MCMC

@MCMC	**MCMC block command**
Conditions	Only used in MCMC.
Effects	Defines any following commands as `@MCMC` subcommands.
Notes	Some of these are only used when running the chain (`casal -m`, `-M`, `-a`). The rest are only used when creating a sub-sample (`casal -C`).

start	**Covariance multiplier for the starting point of the Markov chain**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal –m`, `-M`).
Type	Constant
Effects	If 0, defines the starting point of the chain as the point estimate. If >0, defines the starting point as randomly generated, with covariance matrix equal to the approximate covariance (inverse Hessian) times the value of this `start` parameter.
Notes	This parameter can be overridden by the `casal -i` switch. If the file specified with `-i` contains two parameter vectors, the second is used to start the chain (the first is used to start the initial point estimate).

length	**Length of the Markov chain**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal –m`, `-M`, `-a`).
Type	Integer
Effects	Defines the length of the Markov chain (as a number of iterations)

Notes	(unless you stop the chain first, of course) We recommend an absolute minimum of 100 000 for serious runs. With many parameters, 1 000 000 may not be enough.

keep	**Spacing between recorded values in the chain**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Integer
Default	1 (i.e., all values are recorded)
Effects	Defines the spacing between recorded values in the chain. Samples from the posterior are written to file only if their sample number is evenly divisible by `keep`.

max_cor	**Maximum absolute correlation in the covariance matrix of the proposal distribution**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Constant
Default	0.8
Effects	Defines the maximum correlation in the covariance matrix of the proposal distribution. Correlations greater than `max_cor` are decreased to `max_cor`, and those less than `-max_cor` are increased to `-max_cor` (see Section 6.5 for detail).

covariance_adjustment	**Method for adjusting small variances in the covariance proposal matrix**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	String
Default	`correlation`
Effects	Defines the method (either `correlation` or `covariance`) for the adjusting small variances in the covariance matrix of the proposal distribution (see Section 6.5 for detail).

min_diff	**Minimum nonzero variance times the range of the bounds in the covariance matrix of the proposal distribution**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Constant
Default	0.0001
Effects	Defines the minimum nonzero variance times the difference in the bounds of each parameter in the covariance matrix of the proposal distribution (see Section 6.5 for detail).

stepsize	**Initial stepsize (as a multiplier of the approximate covariance matrix)**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Constant
Default	$2.4d^{-0.5}$ where d is the number of free parameters.
Effects	Defines the stepsize in the Markov chain.
Notes	The covariance of the proposal distribution is the approximate covariance (inverse Hessian) times this `stepsize` parameter. See also `adaptive_stepsize`, `adapt_at`.

adaptive_stepsize	**Should the MCMC stepsize be altered during the chain?**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Switch
Default	`False` (i.e., do not alter stepsize)
Effects	Defines whether the stepsize should be altered adaptively during the chain.

adapt_at	**At which iteration numbers can the MCMC stepsize be altered?**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`). Only used if `adaptive_stepsize` is set.
Type	Constant vector
Effects	Defines the iteration number(s) at which the stepsize is altered adaptively.
Notes	Make sure that the burn-in period is greater than the largest entry of `adapt_at`.

proposal_t	**Should the proposal distribution be multivariate *t*?**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Switch
Default	`False` (i.e., use multivariate normal)
Effects	Defines whether the proposal distribution should be multivariate *t* rather than multivariate normal.

df	**Degrees of freedom of the multivariate *t* proposal distribution.**
Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`). Only used if `proposal_t` is set.
Type	Integer
Default	4
Effects	Defines the degrees of freedom of the multivariate *t* proposal distribution.

burn_in	**Number of samples to be discarded for the burn-in period**
Command	`MCMC`
Conditions	Only used when creating a posterior sub-sample (`casal -C`).
Type	Integer
Effects	Defines the number of samples to be discarded at the start of each chain.
Notes	This is the number of `recorded` samples to be discarded. So, the length of the burn-in period is effectively `burn_in` × `keep`.

subsample_size	**Size of the sub-sample to be generated**
Command	`MCMC`
Conditions	Only used when creating a posterior sub-sample (`casal -C`).
Type	Integer
Default	No random sub-sampling is done.
Effects	Defines the size of the sub-sample to be generated using resampling with replacement.
Notes	This is used to decimate down to a sub-sample of manageable size.

systematic	**Should sub-sampling be systematic?**
Command	`MCMC`
Conditions	Only used when creating a posterior sub-sample (`casal -C`). `subsample_size` must be provided. Cannot be used with `prior_reweighting`.
Type	Switch
Default	`False` (i.e., sub-sample randomly)
Effects	Defines the sub-sampling from the posterior as systematic (i.e., keep every *n*th point) rather than random.

prior_reweighting	**Should the sub-sample be generated using prior reweighting?**
Command	`MCMC`
Conditions	Only used when creating a posterior sub-sample (`casal -C`). `subsample_size` must be provided. `systematic` must not be set.
Type	Switch
Default	`False` (i.e., no prior reweighting)
Effects	Defines that prior reweighting should be carried out, using the current prior.

Notes	The prior commands in `estimation.csl` should have changed since the MCMC run. The sub-sampling is weighted by the ratio of the old prior (which is saved in the `objective` files) to the new prior.

adaptive_covariance Should the MCMC covariance matrix be altered during the chain?

Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Switch
Default	`False` (i.e., do not alter covariance matrix)
Effects	Defines whether the covariance matrix should be altered adaptively during the chain. See also `adaptive_stepsize`.

adapt_covariance_at At which iteration numbers can the MCMC covariance matrix be altered?

Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`). Only used if `adaptive_covariance` is set.
Type	Constant vector
Effects	Defines the iteration number(s) at which the covariance matrix is altered adaptively.
Notes	Make sure that the burn-in period is greater than the largest entry of `adapt_covariance_at`.

adaptive_covariance_discard If the MCMC covariance matrix is altered during the chain, how many observations should be discarded from the start of the chain when taking a subsample for estimating the new covariance matrix?

Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Integer
Effects	Defines the number of observations discarded from the start of the chain. All remaining observations are systematically subsampled, and the (modified) covariance of the subsample is used as the new covariance matrix.
Notes	It is a fatal error if the chain does not move at least once before `adapt_covariance_discard` observations have occurred.

adaptive_covariance_transitions If the MCMC covariance matrix is altered during the chain, how many transitions must occur in the part of the chain used to estimate the new covariance matrix?

Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	integer
Effects	It is a fatal error if the chain does not move at least `adapt_covariance_transitions` times between the end of the discard period (`adapt_covariance_discard`) and the point where the covariance matrix is adapted.

adaptive_covariance_stepsize If the MCMC covariance matrix is altered during the chain, what is the new stepsize value?

Command	`MCMC`
Conditions	Only used when running a Markov chain (`casal -m`, `-M`, `-a`).
Type	Constant
Default	$2.4d^{-0.5}$ where d is the number of free parameters.
Effects	Defines the stepsize in the Markov chain, after the covariance matrix is modified adaptively.
Notes	After modification, the covariance of the proposal distribution is the modified covariance from a sample of the chain times this adaptive_covariance_`stepsize` parameter. See also `stepsize`, `adaptive_stepsize`, `adapt_at`.

@trivariate_normal_test	**Test MCMC algorithm with a simple trivariate normal example**
Conditions	Only used when running a Markov chain (`casal -m`, `-M`).
Type	Integer
Default	`False` (i.e., program runs normally)
Effects	Replace the usual objective function with a trivariate normal density. This is a test of the MCMC algorithm, which should generate a sample from the trivariate normal distribution.

9.5 Defining the free parameters and priors

@estimate	**Free parameter block command**
Effects	Defines any following commands as `estimate` subcommands
Notes	The *i*th `@estimate` block relates to the i^{th} parameter to be estimated.

parameter	**Name of the parameter to be estimated**
Command	`estimate[i]`
Type	String
Effects	Defines the name of the parameter to be estimated.
Notes	See Section 6.2, and Section 3.4 for instructions on generating the parameter names.

Example	`annual_growths`
	`initialization[stock_name].B0`
	`growth[2].g`

same	**Names of the other parameters which are constrained to have the same value**
Command	`estimate[i]`
Type	String
Default	No parameters
Effects	Defines the names of all the other parameters which are constrained to have the same value as this parameter
Notes	Do not give these parameters separate `estimate` blocks. See Section 6.2.

phase	**Phase at which this parameter should be estimated, in point estimation**
Command	`estimate[i]`
Type	Integer
Default	1
Effects	Defines the phase at which this parameter should be freed.
Notes	If no `phase` commands are provided, then estimation is single-phase.

lower_bound, upper_bound	**Bounds on this scalar parameter**
Command	`estimate[i]`
Type	2 x constant
Effects	Defines the lower and upper bounds on this scalar parameter.
Notes	See also the vector versions below.

lower_bound, upper_bound	**Bounds on this vector parameter**
Command	`estimate[i]`
Type	2 x constant vector
Effects	Defines the vectors of lower and upper bounds on this vector parameter.
Notes	See also the scalar versions above.

MCMC_fixed	**Should this parameter be fixed during MCMC?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`
Type	Switch
Default	`False` (i.e., do not fix the parameter during MCMC)
Effects	Define this parameter as fixed during MCMC.

prior	**What type of prior does this parameter have?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`
Type	String
Default	There are no default priors
Effects	Defines the type of prior on this parameter. For scalar parameters, `uniform`, `uniform-log`, `normal`, `normal-by-stdev`, `lognormal`, `normal-log`, `beta`. For vector parameters, all the above plus `normal-AR`, `normal-log-AR`, `normal-log-mean1-AR`.

(Parameters of the prior follow)

mu, cv	**What are the mean and c.v. of this normal or lognormal prior on a scalar parameter?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal` or `lognormal`.
Type	2 x constant
Effects	Defines the prior mean and c.v. (*before* bounds are applied).

mu, stdev	**What are the mean and standard deviation of this normal-by-standard deviation or beta prior on a scalar parameter?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal-by-stdev` or `beta`.
Type	2 x constant
Effects	Defines the prior mean and standard deviation (*before* bounds are applied).

m, s	**What are the mean and standard deviation of the log of this scalar parameter, under the normal-log prior?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal-log`.
Type	2 x constant
Effects	Defines the prior mean and standard deviation of the log-parameter (*before* bounds are applied).

A, B	**What are the lower and upper values for the range parameters of the beta prior?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=beta`.
Type	2 x constant
Default	A=0, and B=1
Effects	Defines the lower and upper range values of the beta prior (*before* bounds are applied).
Warning	Note the bounds must lie *inside* the range parameters.

mu, cv	**What are the mean and c.v. of each element of this normal or lognormal prior on a vector parameter?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal` or `lognormal`.
Type	2 x constant vector
Effects	Defines the prior mean and c.v. of each element of the vector (*before* bounds are applied).

mu, stdev	**What are the mean and standard deviation of each element of this normal-by-standard deviation or beta prior on a vector parameter?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal-by-std.dev` or `beta`.
Type	2 x constant vector
Effects	Defines the prior mean and standard deviation of each element of the vector (*before* bounds are applied).

m, s	**What are the mean and standard deviation of each element of the log of this vector parameter, under the normal-log prior?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal-log`.
Type	2 x constant vector
Effects	Defines the prior mean and standard deviation of each element of the log-parameter (*before* bounds are applied).

A, B	**What are the lower and upper values for each element of the range parameters of the beta prior?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=beta`.
Type	2 x constant
Default	A=0, and B=1
Effects	Defines the lower and upper range values of each element of the beta prior (*before* bounds are applied).
Warning	Note the bounds must lie *inside* the range parameters.

mu, cv, rho	**What are the mean, c.v., and ρ of this normal-AR prior on a vector parameter?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal-AR`.
Type	3 x constant
Effects	Defines the prior mean, c.v., and ρ.
Notes	The single value is used for each element of the parameter.

m, s, r	**What are the log-scale mean, standard deviation, and ρ of this normal-log-AR prior on a vector parameter?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal-log-AR`.
Type	3 x constant
Effects	Defines the prior mean, standard deviation, and ρ of the log-parameter.
Notes	The single value is used for each element of the parameter.

s, r	**What are the log-scale standard deviation and ρ of this normal-log-mean1-AR prior on a vector parameter?**
Command	`estimate[i]`
Conditions	`estimator=Bayes`, `prior=normal-log-mean1-AR`
Type	2 x constant
Effects	Defines the prior standard deviation and ρ of the log-parameter.
Notes	The single value is used for each element of the parameter.

9.6 Defining the catchability coefficients *q*

@q_method	**Method used for catchability coefficients *q***
Type	String
Default	`nuisance`
Effects	Defines the method used to deal with *q*'s as `nuisance` or `free`.

@q_type	**Type of definition for catchability coefficients q**
Type	String
Default	Standard
Effects	Determines whether the expected value of the ith relative abundance index is defined as $qE_i^{1/b}$ (`standard`) or $q(E_i/\max(E))^{1/b}$ (`scaled`). The latter form is not recommended, and is included only for backward compatibility.

@q	**Catchability coefficients *q* block command**
Label	The label of the *q*, as used in the relevant `@observations` block(s)
Effects	Defines any following commands as *q* subcommands
Notes	Only needed if `q_method=free` or you have a curvature parameter *b*

q	**Value of the *q* parameter**
Command	`q[label]`
Conditions	Only used if `q_method=free`.
Type	Estimable
Effects	Defines the starting value of the *q* parameter.

b	**Curvature parameter *b* associated with the *q***
Command	`q[label]`
Type	Estimable
Default	No curvature
Effects	Defines the curvature parameter.
Notes	You also need to set `curvature=true` in the command block for the relative abundance observations.

9.7 Defining the observations

9.7.1 Abundance and relative abundance observations

@abundance	**Absolute abundance block command**
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)
Effects	Defines any following commands as `@abundance` subcommands for the time series.

@relative_abundance	**Relative abundance block command**
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)
Effects	Defines any following commands as `@relative_abundance` subcommands for the time series.

years	**Years of the time series**
Command	`abundance[label], relative_abundance[label]`
Type	Constant vector
Effects	Defines the years for which there are observations. Should be one entry per observation.

step	**Time step in which the observations occur**
Command	`abundance[label], relative_abundance[label]`
Type	Integer
Effects	Defines the time step in which the observations occur.

proportion_mortality	**Proportion of the step's mortality, after which the observations occur**
Command	`abundance[label], relative_abundance[label]`
Type	Constant
Default	0.5
Effects	Defines the proportion of the mortality in the time step after which the observations occur.

area	**Area in which the observations occur**
Command	`abundance[label], relative_abundance[label]`
Condition	`n_areas > 1`
Type	String
Effects	Defines the area in which the observations occur. Use an area label as per `area_names`.

q	**Catchability coefficients *q* to use**
Command	`relative_abundance[label]`
Type	String
Effects	Defines the label of the *q* used by the observations.

curvature	**Should a curvature parameter be used?**
Command	`relative_abundance[label]`
Type	Switch
Default	`False` (i.e., no curvature)
Effects	Defines these observations as using a curvature parameter *b*.
Notes	See Section 6.7.1. Provide the value of the curvature parameter in the *q* command block.

biomass	**Are the observations biomass rather than numbers of fish?**
Command	`abundance[label]`, `relative_abundance[label]`
Type	Switch
Effects	Defines whether the observations are biomass (`biomass=true`) or numbers of fish (`biomass=false`).

ogive	**Which selectivity ogive should be applied?**
Command	`abundance[label]`, `relative_abundance[label]`
Type	String
Default	No selectivity
Effects	Defines which selectivity ogive should be applied when calculating the fits. Use a selectivity label as per `selectivity_names`.

[year]	**Abundance for [year]**
Command	`abundance[label]`, `relative_abundance[label]`
Type	Constant
Effects	Defines the abundance for [year].

Example	If you have observations for 1992, 1995, and 1998, you would put, e.g.,
	`years 1992 1995 1998`
	`1992 100000`
	`1995 3000`
	`1998 12`

mature_only	**Do these observations include mature fish only?**
Command	`abundance[label]`
Type	Switch
Default	`False` (i.e., include both immature and mature fish)
Effects	Defines whether the observations are mature fish only (`mature_only=true`) or both mature and immature fish (`mature_only =false`).
Notes	You will probably only want to use this subcommand when generating pseudo-fits (Section 7.2), to output mature abundance.

stock	**Which stock do these observations relate to?**
Command	`abundance[label]`
Condition	`n_stocks > 1`
Type	String
Default	All stocks
Effects	Defines the name of the stock which is observed. Use a stock label from `stock_names`.
Notes	You will probably only want to use this subcommand when generating pseudo-fits (Section 7.2), to output abundance of a particular stock.

all_areas	**Do these observations cover all areas in the model?**
Command	`abundance[label]`
Condition	`n_areas > 1`
Type	Switch
Default	`False` (i.e., not all areas)
Effects	Defines the observations as covering all areas. The `area` command is superseded by this command.
Notes	You will probably only want to use this subcommand when generating pseudo-fits (Section 7.2), to output abundance over all areas.

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`abundance[label],relative_abundance[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation, if doing parametric bootstraps.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.2 Numbers-at, relative numbers-at, and proportions-at observations

@numbers_at	**Numbers_at block command**
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)
Effects	Defines any following commands as `@numbers_at` subcommands for the time series.

@relative_numbers_at	**Relative numbers-at block command**
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)
Effects	Defines any following commands as `@relative_numbers_at` subcommands for the time series.

@proportions_at	**Proportions-at block command**
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop). Use `catch_at` for commercial catch data.
Effects	Defines any following commands as `@proportions_at` subcommands for the time series.
years	**Years of the time series**
Command	`numbers_at[label],relative_numbers_at[label], proportions_at[label]`
Type	Constant vector
Effects	Defines the years for which there are observations. Should be one entry per row of observations.
step	**Time step in which the observations occur**
Command	`numbers_at[label],relative_numbers_at[label], proportions_at[label]`
Type	Integer
Effects	Defines the time step in which the observations occur.
proportion_mortality	**Proportion of the step's mortality after which the observations occur**
Command	`numbers_at[label],relative_numbers_at[label], proportions_at[label]`
Type	Constant
Default	0.5
Effects	Defines the proportion of the mortality in the time step after which the observations occur.
at_size	**Are the observations by size?**
Command	`numbers_at[label],relative_numbers_at[label], proportions_at[label]`
Type	Switch
Default	`size_based` (i.e., size-based in a size-based model, age-based in an age-based model)
Effects	Defines the observations as size-based (`at_size=true`) or age-based (`at_size=false`).

sexed	**Are the observations sexed?**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Condition	`sex_partition` is set
Type	Switch
Default	`True` (i.e., the observations are sexed)
Effects	Defines the observations as sexed (`sexed=true`) or unsexed (`sexed=false`).

area	**Area in which the observations occur**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Condition	`n_areas > 1`
Type	String
Effects	Defines the area in which the observations occur. Use an area label as per `area_names`.

q	**Catchability coefficients *q* to use**
Command	`relative_numbers_at[label]`
Type	String
Effects	Defines the label of the *q* used by the observations.

ogive	**Which selectivity ogive should be applied?**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Type	String
Default	No selectivity
Effects	Defines which selectivity ogive should be applied when calculating the fits. Use a selectivity label as per `selectivity_names`.

class_mins	**What are the size bins of the observations (in an age-based model)?**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Conditions	Age-based model with `at_size` set, i.e., a size frequency time series in an age-based model
Type	Constant vector
Effects	Defines the lower limits of each of the size classes. If there is no plus group then an additional value defines the upper limit of the last size class.
Notes	If the observations are sexed, both sexes share the same list of size classes (but can use different subsets of them, see `min_class`, `max_class`) — do not put separate size classes for the two sexes here.

class_nums	**What are the class numbers for the observations (in a size-based model)?**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Conditions	Size-based model. The `sexed` switch needs to be set to false.
Type	Constant vector of integers.
Effects	Defines the lower limits of each of the size classes, where size classes are indexed by the size class number (not by size). The size class includes the lower limit. If there is no plus group then the last value defines the upper limit of the last size class (but does not include this value).

class_nums_male, class_nums_female	**What are the class numbers for the sexed observations (in a size-based model)?**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Conditions	Size-based model. The `sexed` switch needs to be set to true.
Type	Constant vector of integers.
Effects	Defines the lower limits of each of the size classes, where size classes are indexed by the size class number (not by size). The size class includes the lower limit. If there is no plus group then the last value defines the upper limit of the last size class (but does not include this value).

Notes	If the observations are not sexed then use `class_nums`. If the observations are sexed then use `class_nums_male` and plus_group

plus_group	**Is the last age or size class a plus group?**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Type	Switch
Default	`True` (i.e., it is a plus group)
Effects	Defines the last age or size class as a plus group.

min_class, max_class	**Which age/size classes are covered by the observations?**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Type	2 x constant vector
Default	All classes are covered
Effects	Defines the first and last age or size classes covered by the observations. If the observations are sexed, each parameter is a 2-vector, for males then females. If unsexed, each parameter is a 1-vector. Age classes are indexed by age. Size classes by size class number (not by size).
Example	Unsexed observations covering 2-year-olds to 6-year-olds would have `min_class=2`, `max_class=6`. Sexed observations covering 2 to 12-year-old males and 2 to 14-year-old females would have `min_class = 2 2`, `max_class = 12 14`. If the model uses size classes of 20-30, 30-40, 40-50, 50+ cm, and the unsexed observations only cover 20-50 cm, then they would have `min_class = 1`, `max_class = 3`.

sum_to_one	**Should the proportions sum to 1?**
Command	`proportions_at[label]`
Type	Switch
Default	`True` (i.e., proportions are expected to sum to 1)
Effect	Defines proportions as summing to 1 for each year of observations.
Notes	This switch is provided for compatibility with previous NIWA software. The default is recommended. See Section 6.6 for details.

ageing_error	**Should ageing error be applied to these observations?**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Conditions	Only used in an age-based model in which ageing error has been specified.
Type	Switch
Default	`True` (i.e., use ageing error as specified)
Effects	Define that these observations use ageing error.
Notes	This is used to turn ageing error off for an individual time series. Ageing error is only applied if it is specified in an `@ageing_error` block (see Section 9.10).

[year]	**Numbers or proportions for [year]**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Type	Constant
Effects	Defines the observations for [year]. If the observations are sexed, put male observations first then female observations (on the same row).
Example	If you have sexed observations of 2, 3, and 4-year-olds for 1992, 1995, and 1998, you would put, e.g.,

```
years 1992 1995 1998
sexed 1
min_class 2 2
max_class 4 4
# M2 M3 M4 F2 F3 F4
1992 0.1 0.2 0.3 0.1 0.2 0.1
1995 0.2 0.3 0.1 0.2 0.1 0.1
1998 0.3 0.1 0.1 0.2 0.2 0.1
```

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`numbers_at[label]`, `relative_numbers_at[label]`, `proportions_at[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.3 Catch-at observations

@catch_at	**Catch_at block command**
Effects	Defines any following commands as `@catch_at` subcommands for the time series.
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)

years	**Years of the time series**
Command	`catch_at[label]`
Type	Constant vector
Effects	Defines the years for which there are observations. Should be one entry per row of observations.

fishery	**Fishery or fisheries covered by the observations**
Command	`catch_at[label]`
Type	Vector of strings
Effects	Defines the fisheries included by the observations. Use fishery labels as in `annual_cycle.fishery_names`.
Notes	Typically a single set of observations will cover only one fishery, but it might be the case that a single administrative fishery might be split into several CASAL fisheries, and the observations are provided for all those fisheries combined.

at_size	**Are the observations by size?**
Command	`catch_at[label]`
Type	Switch
Default	`True` (i.e., observations are size-based in a size-based model, age-based in an age-based model)
Effects	Defines the observations as size-based (`at_size=true`) or age-based (`at_size=false`).

sexed	**Are the observations sexed?**
Command	`catch_at[label]`
Condition	`sex_partition` is set
Type	Switch
Default	`True` (i.e., observations are sexed)
Effects	Defines the observations as sexed (`sexed=true`) or unsexed (`sexed=false`).

class_mins	**What are the size bins of the observations (in an age-based model)?**
Command	`catch_at[label]`
Conditions	Age-based model with `at_size` set, i.e., a size frequency time series in an age-based model
Type	Constant vector
Effects	Defines the lower limits of each of the size classes. If there is no plus group then an additional value defines the upper limit of the last size class.
Notes	If the observations are sexed, both sexes share the same list of size classes (but can use different subsets of them, see `min_class`, `max_class`) — do not put separate size classes for the two sexes here.

class_nums **What are the class numbers for the observations (in a size-based model)?**

Command	`catch_at[label]`
Conditions	Size-based model. The `sexed` switch needs to be set to false.
Type	Constant vector of integers.
Effects	Defines the lower limits of each of the size classes, where size classes are indexed by the size class number (not by size). The size class includes the lower limit. If there is no plus group then the last value defines the upper limit of the last size class (but does not include this value).
Notes	If the observations are sexed then use `class_nums_male` and `class_nums_female`.

class_nums_male, class_nums_female **What are the class numbers for the sexed observations (in a size-based model)?**

Command	`catch_at[label]`
Conditions	Size-based model. The `sexed` switch needs to be set to true.
Type	Constant vector of integers.
Effects	Defines the lower limits of each of the size classes, where size classes are indexed by the size class number (not by size). The size class includes the lower limit. If there is no plus group then the last value defines the upper limit of the last size class (but does not include this value).
Notes	If the observations are not sexed then use `class_nums`.

plus_group **Is the last age or size class a plus group?**

Command	`catch_at[label]`
Type	Switch
Default	`True` (i.e., it is a plus group)
Effects	Defines the last age or size class as a plus group.

min_class, max_class **Which age/size classes are covered by the observations?**

Command	`catch_at[label]`
Type	2 x constant vector
Default	All classes are covered
Effects	Defines the first and last age or size classes covered by the observations. If the observations are sexed, each parameter is a 2-vector, for males then females. If unsexed, each parameter is a 1-vector. Age classes are indexed by age. Size classes by size class number (not by size).
Example	Unsexed observations covering 2-year-olds to 6-year-olds would have `min_class = 2`, `max_class = 6`. Sexed observations covering 2 to 12-year-old males and 2 to 14-year-old females would have `min_class = 2 2`, `max_class = 12 14`. If the model uses size classes of 20-30, 30-40, 40-50, 50+ cm, and the unsexed observations only cover 20-50 cm, then they would have `min_class = 1`, `max_class = 3`.

sum_to_one **Should the proportions sum to 1?**

Command	`catch_at[label]`
Type	Switch
Default	`True` (i.e., proportions are expected to sum to 1)
Effect	Defines proportions as summing to 1 for each year of observations.
Notes	This switch is provided for compatibility with previous NIWA software. The default is recommended. See Section 6.6 for details.

ageing_error **Should ageing error be applied to these observations?**

Command	`catch_at[label]`
Conditions	Only used in an age-based model in which ageing error has been specified.
Type	Switch
Default	`True` (i.e., use ageing error as specified)
Effects	Define that these observations use ageing error.
Notes	This is used to turn ageing error off for an individual time series. Ageing error is only applied if it is specified in an `@ageing_error` block (see Section 9.10).

[year]	**Numbers or proportions for [year]**
Command	`catch_at[label]`
Type	Constant
Effects	Defines the observations for [year]. If the observations are sexed, put male observations first then female observations (on the same row).
Example	If you have sexed observations of 2, 3, and 4-year-olds for 1992, 1995, and 1998, you would put, e.g.,

```
years 1992 1995 1998
sexed 1
min_class 2 2
max_class 4 4
# M2 M3 M4 F2 F3 F4
1992 0.1 0.2 0.3 0.1 0.2 0.1
1995 0.2 0.3 0.1 0.2 0.1 0.1
1998 0.3 0.1 0.1 0.2 0.2 0.1
```

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`catch_at[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.4 Proportions mature observations

@proportions_mature	**Proportions_mature block command**
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)
Effects	Defines any following commands as `@proportions_mature` subcommands for the time series.
years	**Years of the time series**
Command	`proportions_mature[label]`
Type	Constant vector
Effects	Defines the years for which there are observations. There should be one entry per row of observations.
step	**Time step in which the observations occur**
Command	`proportions_mature[label]`
Type	Integer
Effects	Defines the time step in which the observations occur.
proportion_mortality	**Proportion of the step's mortality, prior to when the observations occur**
Command	`proportions_mature[label]`
Type	Constant
Default	0.5
Effects	Defines the proportion of the mortality in the time step, prior to when the observations occur.
sexed	**Are these observations sexed?**
Command	`proportions_mature[label]`
Type	Switch
Default	`True` (for a sex-based model) and `false` otherwise
Effects	Defines these observations as sexed (`sexed=true`) or unsexed (`sexed=false`).
females_only	**Are these observations for females only?**
Command	`proportions_mature[label]`

Type	Switch
Default	`False` (i.e., both males and females)
Effects	Defines these observations as females only (`females_only=true`) or both sexes separately (`females_only=false`).
Notes	Do not use `females_only=true` for `sexed=false` observations or in an unsexed model.

at_size	**Are the observations by size?**
Command	`proportions_mature[label]`
Type	Switch
Default	`size_based` (i.e., size-based in a size-based model, age-based in an age-based model)
Effects	Defines the observations as size-based (`at_size=true`) or age-based (`at_size=false`).

area	**Area in which the observations occur**
Command	`proportions_mature[label]`
Condition	`n_areas > 1`
Type	String
Effects	Defines the area in which the observations occur. Use an area label as per `area_names`.

ogive	**Which selectivity ogive should be applied?**
Command	`proportions_mature[label]`
Type	String
Default	No selectivity
Effects	Defines which selectivity ogive should be applied when calculating the fits. Use a selectivity label as per `selectivity_names`.
Notes	The selectivity ogive only affects the results if it is a size-based ogive in an age-based model.

class_mins	**What are the size bins of the observations (in an age-based model)?**
Command	`proportions_mature[label]`
Conditions	Age-based model with `at_size` set, i.e., a size frequency time series in an age-based model
Type	Constant vector
Effects	Defines the lower limits of each of the size classes. If there is no plus group then an additional value defines the upper limit of the last size class.
Notes	If the observations are sexed, both sexes share the same list of size classes (but can use different subsets of them, see `min_class`, `max_class`) — do not put separate size classes for the two sexes here.

plus_group	**Is the last age or size class a plus group?**
Command	`proportions_mature[label]`
Type	Switch
Default	`True` (i.e., it is a plus group)
Effects	Defines the last age or size class as a plus group.

min_class, max_class	**Which age/size classes are covered by the observations?**
Command	`proportions_mature[label]`
Type	2 x constant vector
Default	All classes are covered
Effects	Defines the first and last age or size classes covered by the observations. If the observations are sexed, each parameter is a 2-vector, for males then females. If unsexed, each parameter is a 1-vector. Age classes are indexed by age. Size classes by size class number (not by size).

Example	Unsexed observations covering 2-year-olds to 6-year-olds would have `min_class = 2`, `max_class = 6`. Sexed observations covering 2 to 12-year-old males and 2 to 14-year-old females would have `min_class = 2 2`, `max_class = 12 14`. If the model uses size classes of 20-30, 30-40, 40-50, 50+ cm, and the unsexed observations only cover 20-50 cm, then they would have `min_class = 1`, `max_class = 3`.

ageing_error	**Should ageing error be applied to these observations?**
Command	`proportions_mature[label]`
Conditions	Only used in an age-based model in which ageing error has been specified.
Type	Switch
Default	`True` (i.e., use ageing error as specified)
Effects	Define that these observations use ageing error.
Notes	This is used to turn ageing error off for an individual time series. Ageing error is only applied if it is specified in an `@ageing_error` block (see Section 9.10).

[year]	**Proportions mature for [year]**
Command	`proportions_mature[label]`
Type	Constant
Effects	Defines the observations for [year]. If the observations are for both sexes, put male observations first then female observations (on the same row).
Example	If you have sexed observations of 2, 3, and 4-year-olds for 1992 and 1995, then:

```
years 1992 1995
min_class 2 2
max_class 4 4
# M2 M3 M4 F2 F3 F4
1992 0.1 0.5 0.8 0.1 0.4 0.9
1995 0.2 0.6 1.0 0.2 0.7 0.9
```

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`proportions_mature[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.5 Proportions migrating observations

@proportions_migrating	**Proportions_migrating block command**
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)
Effects	Defines any following commands as `@proportions_migrating` subcommands for the time series.

years	**Years of the time series**
Command	`proportions_migrating[label]`
Type	Constant vector
Effects	Defines the years for which there are observations. Should be one entry per row of observations.

migration	**Migration to which the observations apply**
Command	`proportions_migrating[label]`
Type	String
Effects	Defines the migration to which the observations apply. Use a label from `annual_cycle.migration_names`.

sex	**Which sex do the observations apply to?**
Command	`proportions_migrating[label]`
Condition	`sex_partition` is set
Type	Integer
Default	Observations apply to both sexes combined
Effects	Defines the observations as male (`sex=1`) or female (`sex=2`).
Notes	If you have separate observations for males and females, enter them as two different time series.

at_size	**Are the observations by size?**
Command	`proportions_migrating[label]`
Type	Switch
Default	`size_based` (i.e., size-based in a size-based model, age-based in an age-based model)
Effects	Defines the observations as size-based (`at_size=true`) or age-based (`at_size=false`).

area	**Area in which the observations occur**
Command	`proportions_migrating[label]`
Condition	`n_areas > 1`
Type	String
Effects	Defines the area in which the observations occur. Use an area label as per `area_names`.

ogive	**Which selectivity ogive should be applied?**
Command	`proportions_migrating[label]`
Type	String
Default	No selectivity
Effects	Defines which selectivity ogive should be applied when calculating the fits. Use a selectivity label as per `selectivity_names`.
Notes	The selectivity ogive only affects the results if it is a size-based ogive in an age-based model. Probably this parameter is unnecessary.

class_mins	**What are the size bins of the observations (in an age-based model)?**
Command	`proportions_migrating[label]`
Conditions	Age-based model with `at_size` set, i.e., a size frequency time series in an age-based model
Type	Constant vector
Effects	Defines the lower limits of each of the size classes. If there is no plus group then an additional value defines the upper limit of the last size class.
Notes	If the observations are sexed, both sexes share the same list of size classes (but can use different subsets of them, see `min_class`, `max_class`) — do not put separate size classes for the two sexes here.

plus_group	**Is the last age or size class a plus group?**
Command	`proportions_migrating[label]`
Type	Switch
Default	`True` (i.e., it is a plus group)
Effects	Defines the last age or size class as a plus group.

min_class, max_class	**Which age/size classes are covered by the observations?**
Command	`proportions_migrating[label]`
Type	2 x constant vector
Default	All classes are covered
Effects	Defines the first and last age or size classes covered by the observations. If the observations are sexed, each parameter is a 2-vector, for males then females. If unsexed, each parameter is a 1-vector. Age classes are indexed by age. Size classes by size class number (not by size).

Example	Unsexed observations covering 2-year-olds to 6-year-olds would have `min_class = 2`, `max_class = 6`. Sexed observations covering 2 to 12-year-old males and 2 to 14-year-old females would have `min_class = 2 2`, `max_class = 12 14`. If the model uses size classes of 20-30, 30-40, 40-50, 50+ cm, and the unsexed observations only cover 20-50 cm, then they would have `min_class = 1`, `max_class = 3`.

ageing_error	**Should ageing error be applied to these observations?**
Command	`proportions_migrating[label]`
Conditions	Only used in an age-based model in which ageing error has been specified.
Type	Switch
Default	`True` (i.e., use ageing error as specified)
Effects	Define that these observations use ageing error.
Notes	This is used to turn ageing error off for an individual time series. Ageing error is only applied if it is specified in an `@ageing_error` block (see Section 9.10).

[year]	**Proportions migrating for [year]**
Command	`proportions_migrating[label]`
Type	Constant
Effects	Defines the observations for [year]. If the observations are for both sexes, put male observations first then female observations (on the same row).

Example	If you have sexed observations of 2, 3, and 4-year-olds for 1992 and 1995, you would put, e.g.,
	`years 1992 1995`
	`min_class 2 2`
	`max_class 4 4`
	`# M2 M3 M4 F2 F3 F4`
	`1992 0.1 0.5 0.8 0.1 0.4 0.9`
	`1995 0.2 0.6 1.0 0.2 0.7 0.9`

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`proportions_migrating[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.6 Age-size observations

@age_size	**Age_size block command**
Label	The text label of the data (should be unique and not contain a full stop).
Conditions	Only used in an age-based model. Do not use age-size data in a growth-path model, nor one where size-at-age depends on maturity.
Effects	Defines any following commands as `@age_size` subcommands for the age-size dataset.

year	**Year in which the data were collected**
Command	`age_size[label]`
Type	Integer
Effects	Defines the year in which the data were collected.

step	**Time step in which the data were collected**
Command	`age_size[label]`
Type	Integer
Effects	Defines the time step in which the data were collected.

proportion_mortality **Proportion of the step's mortality, prior to when the observations occur**

Command	`age_size[label]`
Type	Constant
Default	0.5
Effects	Defines the proportion of the mortality in the time step, prior to when the observations occur.

area **Area in which the observations occur**

Command	`age_size[label]`
Condition	`n_areas > 1`
Type	String
Effects	Defines the area in which the observations occur. Use an area label as per `area_names`.

stock **Stock for which the data were collected**

Command	`age_size[label]`
Conditions	Not used in a single-stock model
Type	String
Default	Data were collected from all stocks in the selected area
Effects	Defines the stock which the fish belong to. Use a text label from `stock_names`.
Notes	This parameter would not normally be necessary.

sample **Sampling method under which the observations were generated**

Command	`age_size[label]`
Type	String
Effects	Defines the sampling method used to generate the age-size observations. Options are `random`, `random_at_sex`, `random_at_age`, `random_at_size`, `random_at_sex_and_size`, and `random_at_sex_and_age`.
Notes	See the main text for an explanation of these options.

ogive **Which selectivity ogive should be applied?**

Command	`age_size[label]`
Type	String
Default	No selectivity
Effects	Defines which selectivity ogive was used when taking this sample. Use a selectivity label as per `selectivity_names`.
Notes	With some sample structures there is no point in specifying a size-based selectivity because it has no effect (and involves more calculations). This is always true with a `random_at_sex_and_size` sample, and it is true with a `random_at_size` sample as long as the size-based selectivity function is not sex dependent. Similarly with some sample structures there is no point in using an age-based selectivity. This is always true with `random_at_sex_and_age`, and with `random_at_age` as long as the age-based selectivity is not sex dependent. See the main text for more explanation.

ageing_error **Should ageing error be applied to these observations?**

Command	`age_size[label]`
Conditions	Only if ageing error has been specified.
Type	Switch
Default	`True` (i.e., use ageing error as specified)
Effects	Define that these observations use ageing error.
Notes	This is used to turn ageing error off for an individual set of observations. Ageing error is only applied if it is specified in an `@ageing_error` block (see Section 9.10).

ages **Age data**

Command	`age_size[label]`
Type	Constant vector

Effects	Defines the age data. For each entry of ‘ages’, there should be a corresponding entry of ‘sizes’, and of ‘sexes’ in a sexed model.
Example	

```
@age_size sizedata
year 1990
step 1
ages 1 1 2 2 2 3 3 3 3 4…
sizes 12 15 23 26 27 36 38 32 40 48…
sexes 1 2 1 2 2 1 1 1 2 1…
```

sizes	**Size data**
Command	`age_size[label]`
Type	Constant vector
Effects	Defines the size data. For each entry of ‘sizes’, there should be a corresponding entry of ‘ages’, and of ‘sexes’ in a sexed model.

sexes	**Sex data**
Command	`age_size[label]`
Conditions	Only used in a sexed model
Type	Constant vector
Effects	Defines the sex data. `1` denotes male, `2` denotes female. For each entry of ‘sexes’, there should be a corresponding entry of ‘ages’ and of ‘sizes’.

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`age_size[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.
Warning	Parametric bootstrapping is not yet implemented for `age_size` observations. If `do_bootstraps` is not set to `False`, then CASAL will report an error.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.7 Age at maturation observations

@age_at_maturation	**Age_at_maturation block command**
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)
Effects	Defines any following commands as `@age_at_maturation` commands.

sexed	**Are these observations sexed?**
Command	`age_at_maturation[label]`
Type	Switch
Default	`True` (for a sex-based model) and `false` otherwise
Effects	Defines these observations as sexed (`sexed=true`) or unsexed (`sexed=false`).

sampled_ages	**What were the estimated ages of these fish at sampling?**
Command	`age_at_maturation[label]`
Type	Constant vector
Effects	Defines the age of each fish at sampling.
Notes	For each entry in ‘sampled_ages’, there should be a matching entry in `maturation_ages`, and in `sexes` if `sexes=T`.

maturation_ages	**What were the estimated ages of these fish at maturation?**
Command	`age_at_maturation[label]`
Type	Constant vector
Effects	Defines the age of each fish at maturation.
Notes	For each entry in `maturation_ages`, there should be a matching entry in `sampled_ages`, and in `sexes` if `sexes=T`.

sexes	**What were the sexes of these fish?**
Command	`age_at_maturation[label]`
Conditions	Only use if sexed=True
Type	Constant vector
Effects	Defines the sex of each fish sampled.
Notes	1=male, 2=female. For each entry in ‘sexes’, there should be a matching entry in `maturation_ages` and `sampled_ages`.

stock	**Which stock do these observations relate to?**
Command	`age_at_maturation[label]`
Condition	`n_stocks > 1`
Type	String
Effects	Defines the name of the stock which is observed. Use a stock label from `stock_names`.
Notes	This parameter must be supplied in a multi-stock model, i.e., there is no default.

ageing_error	**Should ageing error be applied to these observations?**
Command	`age_at_maturation[label]`
Conditions	Only used if ageing error has been specified.
Type	Switch
Default	`True` (i.e., use ageing error as specified)
Effects	Define that the ages at maturation (not at capture!) are subject to ageing error.
Notes	This is used to turn ageing error off for an individual time series. Ageing error is only applied if it is specified in an `@ageing_error` block (see Section 9.10).

k	**After how many years can maturation be detected?**
Command	`age_at_maturation[label]`
Type	Constant
Default	k=0
Effects	Defines the k parameter, e.g., the number of years after maturation during which maturation cannot be detected.

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`age_at_maturation[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.
Warning	Parametric bootstrapping is not yet implemented for `age_at_maturation` observations. If `do_bootstraps` is not set to `False`, then CASAL will report an error.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.8 Selectivity at observations

@selectivity_at	**Selectivity-at block command**
Label	The text label of the pseudo-observation time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop).
Effects	Defines any following commands as `@selectivity_at` subcommands for the time series.
Note	`Selectivity_at` is most commonly used as a pseudo-observation, when its use is to extract the values of a selectivity ogive for a particular year, time step, area, stock. You can also use it for actual observations of selectivity if you have some.

ogive	**Which selectivity should be applied?**
Command	`selectivity_at[label]`
Type	String
Default	None
Effects	Defines which selectivity ogive is being queried. Use a selectivity label as per `selectivity_names`.

years	**Years of the time series**
Command	`selectivity_at[label]`
Type	Constant vector
Effects	Defines the years for which there are observations. Should be one entry for each year in which you want to query the selectivity.

step	**Time step in which the observations occur**
Command	`selectivity_at[label]`
Type	Integer
Effects	Defines the time step in which the observations occur.

proportion_mortality	**Proportion of the step's mortality after which the observations occur**
Command	`selectivity_at[label]`
Type	Constant
Default	0.5
Effects	Defines the proportion of the mortality in the time step after which the observations occur.

sexed	**Are the observations sexed?**
Command	`selectivity_at[label]`
Condition	`sex_partition` is set
Type	Switch
Default	`True` (i.e., observations are sexed)
Effects	Defines the observations as sexed (`sexed=true`) or unsexed (`sexed=false`).

area	**Area in which the observations occur**
Command	`selectivity_at[label]`
Condition	`n_areas > 1`
Type	String
Effects	Defines the area in which the observations occur. Use an area label as per `area_names`.

mature_only	**Do these observations include mature fish only?**
Command	`selectivity_at[label]`
Type	Switch
Default	`False` (i.e., include both immature and mature fish)
Effects	Defines whether the observations are biomass (`biomass=true`) or numbers of fish (`biomass=false`).
Notes	You would only use this subcommand when generating pseudo-fits (Section 7.2) so as to output mature abundance.

stock	**Which stock do these observations relate to?**
Command	`selectivity_at[label]`
Condition	`n_stocks > 1`
Type	String
Default	All stocks
Effects	Defines the name of the stock which is observed. Use a stock label from `stock_names`.
Notes	You would usually only use this subcommand when generating pseudo-fits (see Section 7.2) so as to output abundance of a particular stock.

min_class, max_class	**Which age/size classes are covered by the observations?**
Command	`selectivity_at[label]`
Type	2 x constant vector

Default	All classes are covered
Effects	Defines the first and last age or size classes covered by the observations. If the observations are sexed, each parameter is a 2-vector, for males then females. If unsexed, each parameter is a 1-vector. Age classes are indexed by age. Size classes by size class number (not by size).
Example	Unsexed observations covering 2-year-olds to 6-year-olds would have `min_class=2`, `max_class=6`. Sexed observations covering 2 to 12 year-old males and 2 to 14-year old females would have `min_class=2 2`, `max_class=12 14`. If a size-based model uses size classes of 20-30, 30-40, 40-50, 50+ cm, and the unsexed observations only cover 20-50 cm, then they would have `min_class=1`, `max_class=3`.

[year]	**Numbers or proportions for [year]**
Command	`selectivity_at[label]`
Type	Constant
Effects	Defines the observations for [year].
Notes	This should never be supplied if this is used as a pseudo-observation.

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`selectivity_at[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.
Warning	Parametric bootstrapping is not yet implemented for `selectivity_at` observations. If `do_bootstraps` is not set to `False`, then CASAL will report an error.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.9 Tag release observations

@tag_release	
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)
Conditions	Can only be used in an age-based model with size-based `@tag_release` observations.
Effects	Defines any following commands as `@tag_release` subcommands for the observation.
Notes	This observation type is only available for size-based observations in an age based model.

tag_label	**The label of the tagging event**
Command	`tag_release[label]`
Conditions	Supply only if `@n_tags` > 0. The label must be a label from an `@tag` event
Effects	Defines any following commands as `@tag` subcommands.
Notes	The label for the tagging event to which this `@tag_release` observation applies.

class_mins	**What are the size bins of the observations (in an age-based model)?**
Command	`tag_release[label]`
Type	Constant vector
Effects	Defines the lower limits of each of the size classes. If there is no plus group then an additional value defines the upper limit of the last size class.
Notes	If the observations are sexed, both sexes share the same list of size classes.

plus_group	**Is the last size class a plus group?**
Command	`tag_release[label]`
Type	Switch
Default	`True` (i.e., it is a plus group)

Effects	Defines the last size class as a plus group.

props_all	**Relative rates of all fish tagged in the tagging event**
Command	`tag_release[label]`
Conditions	Use either `props_all` or `props_male` and `props_female`, but not both.
Type	Constant vector
Default	No default
Effects	Defines the size-based proportions that were tagged in the tagging event.

props_male, props_female	**Relative proportions of male or female fish tagged in the tagging event**
Command	`tag_release[label]`
Conditions	Only used in a sex based model. Use either `props_all` or `props_male` and `props_female`, but not both.
Type	2 x constant vector
Default	No default
Effects	Defines the sex and size-based proportions that were tagged in the tagging event.

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`tag_release[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.

(See Section 9.8 for parameters relating to the likelihood weights.)

9.7.10 Tag recapture observations

@tag_recapture	**Tag_recapture block command**
Effects	Defines any following commands as `@tag_recapture` subcommands for the time series.
Label	The text label of the time series (should be unique, not `Bpre` or `Bpost`, and not contain a full stop)

tag_name	**Tag which is being checked for**
Command	`tag_recapture[label]`
Type	String
Effects	Defines the label of the tagging partition event. Should be an entry of `@tag_names`.

sample	**Sampling method for the tag-recapture process**
Command	`tag_recapture[label]`
Type	String
Effects	Defines the sampling method used in the tag-recapture. Options available are `age`, `size`, `age-size`, and `growth`.
Notes	`age` means that age frequencies are available for both scanned fish and tagged fish. `size` the same for size frequencies (usable in either an age- or size-based model). `age-size` means that age frequencies are available for tagged fish with size frequencies for scanned fish. `growth` means that only the size distribution of recaptured fish is available (and can only be used in a size based model). See Section 6.6.5 for details.

detection_probability	**What proportion of tags are detected in the recapture process?**
Command	`tag_recapture[label]`
Type	Estimable
Default	1.0
Effects	Defines the proportion of tagged fish in the sample which are detected.

Notes	Must be between 0 (exclusive) and 1 (inclusive). The expected number of tags detected is calculated by multiplying this number by the number of tagged fish in the sample. Note, this parameter is ignored if `sample=growth.`

years	**Years of the time series**
Command	`tag_recapture[label]`
Type	Constant vector
Effects	Defines the years for which there are observations. Should be one entry per row of observations.

step	**Time step in which the observations occur**
Command	`tag_recapture[label]`
Type	Integer
Effects	Defines the time step in which the observations occur.

proportion_mortality	**Proportion of the step's mortality, prior to when the observations occur**
Command	`tag_recapture[label]`
Type	Constant
Default	0.5
Effects	Defines the proportion of the mortality in the time step, prior to when the observations occur.

area	**Area in which the observations occur**
Command	`tag_recapture[label]`
Condition	`n_areas > 1`
Type	String
Effects	Defines the area in which the observations occur. Use an area label as per `area_names`.

class_mins	**What are the size bins of the observations (in an age-based model)?**
Command	`tag_recapture[label]`
Conditions	Age-based model with `sample=size`
Type	Constant vector
Effects	Defines the lower limits of each of the size classes. If there is no plus group then an additional value defines the upper limit of the last size class.
Notes	If the observations are sexed, both sexes share the same list of size classes.

ogive	**Which selectivity ogive should be applied?**
Command	`tag_recapture[label]`
Conditions	Age-based model with `sample=size`
Type	String
Default	No selectivity
Effects	Defines which selectivity ogive should be applied when calculating the fits. Use a selectivity label as per `selectivity_names`.
Notes	In an age-based model with `sample=age`, or a size-based model with `sample=age`, `size`, or `growth`, an ogive would have no effect on the likelihood. For this reason it has been disallowed.

plus_group	**Is the last age or size class a plus group?**
Command	`tag_recapture[label]`
Type	Switch
Default	True (it is a plus group)
Effects	Defines the last age or size class as a plus group.

ageing_error	**Should ageing error be applied to these observations?**
Command	`tag_recapture[label]`
Conditions	Only used in an age-based model in which ageing error has been specified. Not usable with `sample=size`.
Type	Switch
Default	True (use ageing error as specified)
Effects	Define that these observations use ageing error.
Notes	This is used to turn ageing error off for an individual time series.

Ageing error is applied only if it is specified in an `@ageing_error` block (see Section 9.10).

recaptured_[year] Number of fish recaptured

Command `tag_recapture[label]`

Type Constant vector

Effects Defines the recapture observations for [year]. How many fish in each size or age class were recaptured (including both tagged and untagged fish)?

With `sample=age` or `sample=age-size`, this is a vector of numbers at age (one entry for each age class in the partition). With `sample=size`, this is a vector of numbers at size (one entry for each size class in the partition, in a size-based model, or one entry for each size class defined in `class_mins`, in an age-based model.)

Notes Use either `recaptured_[year]` and `scanned_[year]`, or `recaptured_male_[year]`, `recaptured_female_[year]`, `scanned_male_[year]`, and `scanned_female_[year]`.

recaptured_male[year], recaptured_female[year] Number of fish recaptured

Command `tag_recapture[label]`

Type Constant vector (x2)

Effects Defines the sexed recapture observations for [year]. How many fish in each size or age class were recaptured (including both tagged and untagged fish)?

With `sample=age` or `sample=age-size`, these are vectors of numbers at age (one entry for each age class in the partition). With `sample=size`, these are vectors of numbers at size (one entry for each size class in the partition, in a size-based model, or one entry for each size class defined in `class_mins`, in an age-based model.)

Notes Use either `recaptured_[year]` and `scanned_[year]`, or `recaptured_male_[year]`, `recaptured_female_[year]`, `scanned_male_[year]`, and `scanned_female_[year]`.

scanned_[year] Number of fish recaptured

Command `tag_recapture[label]`

Type Constant vector

Effects Defines the tagged recapture observations for [year]. How many tagged fish in each size or age class were recaptured?

With `sample=age`, this is a vector of numbers at age (one entry for each age class in the partition). With `sample=size` or `sample=age-size`, this is a vector of numbers at size (one entry for each size class in the partition, in a size-based model, or one entry for each size class defined in `class_mins`, in an age-based model.) Not necessary (and not allowed) if `sample=growth`.

Notes Use either `recaptured_[year]` and `scanned_[year]`, or `recaptured_male_[year]`, `recaptured_female_[year]`, `scanned_male_[year]`, and `scanned_female_[year]`.

scanned_male[year], scanned_female[year] Number of fish recaptured

Command `tag_recapture[label]`

Type Constant vector (x2)

Effects Defines the sexed tagged recapture observations for [year]. How many tagged fish in each size or age class were recaptured?

With `sample=age`, these are vectors of numbers at age (one entry for each age class in the partition). With `sample=size` or `sample=age-size`, these are vectors of numbers at size (one entry for each size class in the partition, in a size-based model, or one entry for each size class defined in `class_mins`, in an age-based model.) Not necessary (and not allowed) if `sample=growth`.

Notes Use either `recaptured_[year]` and `scanned_[year]`, or `recaptured_male_[year]`, `recaptured_female_[year]`, `scanned_male_[year]`, and `scanned_female_[year]`.

do_bootstrap	**Print out parametric bootstraps for this observation**
Command	`tag_recapture[label]`
Type	Switch
Default	`True`
Effects	Defines whether the parametric bootstrap observations should be generated for this observation if doing parametric bootstraps.

r	**The robustification parameter used in the binomial likelihood**
Command	`tag_recapture[label]`
Type	Constant
Default	1e-11
Effects	Defines the robustification parameter *r* used in the binomial likelihood. See Section 6.6.5.
Notes	Only relevant when the fitted proportion of tagged, detected fish in a given age or size class is very close to either 0 or 1.

dispersion	**Dispersion term for the tag-recapture likelihood**
Command	`tag_recapture[label]`
Type	Estimable constant
Default	1.0
Effects	Defines the dispersion parameter *ø* used in the tag-recapture likelihood. See Section 6.7.2. The resulting tag-recapture log-likelihood is modified by multiplying by 1/*ø*.

9.8 Defining the objective function associated with the observations

dist	**Likelihood of the observations**
Command	`abundance[label]`, `relative_abundance[label]`, `relative_numbers_at[label]`, `proportions_at[label]`, `catch_at[label]`, `proportions_mature[label]`, `proportions_migrating[label]`, `selectivity_at[label]`
Conditions	Can be `user-supplied` only if you have written a `user.likelihood.cpp` which calculates the likelihood function.
Type	String
Effects	Defines the type of likelihood for this time series. Can be: `normal`, `lognormal`, `normal-log`, or `normal-by-stdev` for `abundance` or `proportions_mature` or `proportions_migrating` or `selectivity_at`; OR `binomial` for `proportions_mature` or `proportions_migrating` or `selectivity_at`; OR `normal`, `lognormal`, `robustified-lognormal`, or `normal-log` for `relative_abundance` or `relative_numbers_at`; OR `multinomial`, `Fournier`, `Coleraine`, `lognormal`, or `robustified-lognormal` for `proportions_at` or `catch_at`; OR user-supplied for any of the above time series types.

r	**Robustifying constant**
Command	`abundance[label]`, `relative_abundance[label]`, `relative_numbers_at[label]`, `proportions_at[label]`, `catch_at[label]`, `proportions_mature[label]`, `proportions_migrating[label]`, `selectivity_at[label]`
Conditions	Only used if `estimator=likelihood` or `Bayes` AND `dist=multinomial` or `robustified-lognormal` or `binomial`
Type	Constant
Default	0 (i.e., no robustification)
Effects	Defines the robustifying constant *r* for this time series. Constants *r* are used for the multinomial, the binomial, and the robustified lognormal — they mean quite different things in the three contexts.

cv	**C.v. for all observations in this time series**
Command	`abundance[label]`,`relative_abundance[label]`, `relative_numbers_at[label]`,`proportions_at[label]`, `catch_at[label]`,`proportions_mature[label]`, `selectivity_at[label]`
Conditions	Only used if `dist=normal`, `lognormal`, `robustified-lognormal`, or `normal-log`.
Type	Constant
Effects	Defines a single c.v. used for every observation in the time series.
Notes	Alternatively supply c.v.s for each year using `cv_[year]` or for each individual observation using `cvs_[year]`.

cv_[year]	**C.v.s by year for this time series**
Command	`abundance[label]`,`relative_abundance[label]`, `relative_numbers_at[label]`,`proportions_at[label]`, `catch_at[label]`,`proportions_mature[label]`, `selectivity_at[label]`
Conditions	Only used if `dist=normal`, `lognormal`, `robustified-lognormal`, or `normal-log`.
Type	Constant (1 per year)
Effects	Defines the c.v.s by year for the time series. One c.v. for each year.
Notes	Alternatively supply c.v.s for each individual observation using `cvs_[year]` or a single c.v. for all years using `cv`.

Example	`years 1992 1993 1994`
	`cv_1992 0.2`
	`cv_1993 0.3`
	`cv_1994 0.4`

cvs_[year]	**C.v.s by observation and year for this time series**
Command	`abundance[label]`,`relative_abundance[label]`, `relative_numbers_at[label]`,`proportions_at[label]`, `catch_at[label]`,`proportions_mature[label]`, `selectivity_at[label]`
Conditions	Only used if `dist=normal`, `lognormal`, `robustified-lognormal`, or `normal-log`.
Type	Constant vector (1 per year)
Effects	Defines the c.v.s by observation and year for the time series. One c.v. for each observation in each year.
Notes	Alternatively supply a single c.v. for each year using `cv_[year]` or a single c.v. for all years using `cv`.

Example	`years 1992 1993 1994`
	`sexed 1`
	`min_class 2 2`
	`max_class 4 4`
	`# M2 M3 M4 F2 F3 F4`
	`cvs_1992 0.40 0.15 0.25 0.55 0.15 0.20`
	`cvs_1993 0.60 0.17 0.12 0.75 0.18 0.14`
	`cvs_1994 0.80 0.19 0.08 0.85 0.31 0.15`

cv_process_error	**Process error c.v. for this time series, used with likelihoods parameterised by the c.v.**
Command	`abundance[label]`,`relative_abundance[label]`, `relative_numbers_at[label]`,`proportions_at[label]`, `catch_at[label]`,`proportions_mature[label]`, `selectivity_at[label]`
Conditions	Only used if `dist=normal`, `lognormal`, `robustified-lognormal`, or `normal-log`.
Type	Estimable
Effects	Defines a c.v. to be ‘added’ to all c.v.s by observation and year for the time series.

stdev **Standard deviation for all observations in this time series, used with normal-by-standard deviation likelihoods**

Command	`abundance[label],proportions_mature[label],` `selectivity_at[label]`
Conditions	Only used if `dist=normal-by-stdev`.
Type	Constant
Effects	Defines a single standard deviation used for every observation in the time series.
Notes	Alternatively supply standard deviation for each year using `stdev_[year]` or for each individual observation using `stdevs_[year]`.

stdev_[year] **Standard deviation by year for this time series, used with normal-by-standard deviation likelihoods**

Command	`abundance[label],proportions_mature[label],` `selectivity_at[label]`
Conditions	Only used if `dist=normal-by-stdev`.
Type	Constant (1 per year)
Effects	Defines the standard deviations by year for the time series. One standard deviation for each year.
Notes	Alternatively supply standard deviation for each individual observation using `stdevs_[year]` or a single std. dev. for all years using `stdev`.

Example	`years 1992 1993 1994`
	`stdev_1992 0.2`
	`stdev_1993 0.3`
	`stdev_1994 0.4`

stdevs_[year] **Standard deviation by observation and year for this time series, used with normal-by-standard deviation likelihoods**

Command	`abundance[label],proportions_mature[label],` `selectivity_at[label]`
Conditions	Only used if `estimator=likelihood` or `Bayes` AND `dist=normal-by-stdev`.
Type	Constant vector (1 per year)
Effects	Defines the standard deviations by observation and year for the time series. One std. dev. for each observation in each year.
Notes	Alternatively supply a single standard deviation for each year using `stdev_[year]` or a single standard deviation for all years using `stdev`.

Example	`years 1992 1993 1994`
	`sexed 1`
	`min_class 2 2`
	`max_class 4 4`
	`# M2 M3 M4 F2 F3 F4`
	`stdev_1992 0.40 0.15 0.25 0.55 0.15 0.20`
	`stdev_1993 0.60 0.17 0.12 0.75 0.18 0.14`
	`stdev_1994 0.80 0.19 0.08 0.85 0.31 0.15`

stdev_process_error **Process error standard deviation for this time series, used with likelihoods parameterised by the standard deviation**

Command	`abundance[label],proportions_mature[label]`
Conditions	Only used if `dist=normal-by-stdev`.
Type	Estimable
Effects	Defines a std. dev. to be 'added' to all standard deviation by observation and year for the time series.

N ***N* for all years in this time series, used with proportions likelihoods**

Command	`proportions_at[label],catch_at[label],`
	`proportions_mature[label],`
	`proportions_migrating[label],selectivity_at[label]`

Conditions	Only used if `estimator=likelihood` or `Bayes` AND `dist=multinomial`, `Fournier`, or `Coleraine`.
Type	Constant

Effects	Defines a single N used for every year in the time series.
Notes	Alternatively supply N’s for each year using `N_[year]` or for each individual observation using `Ns_[year]`.
N_[year]	**N’s by year for this time series, used with proportions likelihoods**
Command	`proportions_at[label]`, `catch_at[label]`, `proportions_mature[label]`, `proportions_migrating[label]`, `selectivity_at[label]`
Conditions	Only used if `dist=multinomial`, `Fournier`, or `Coleraine`.
Type	Constant (1 per year)
Effects	Defines the N’s by year for the time series. One N for each year.
Notes	Alternatively supply N’s for each individual observation using `Ns_[year]` or a single N for all years using `N`.

Example	`years 1992 1993 1994`
	`N_1992 100`
	`N_1993 120`
	`N_1994 150`

Ns_[year]	**N’s by observation and year for this time series, used with proportions likelihoods**
Command	`proportions_at[label]`, `catch_at[label]`, `proportions_mature[label]`, `proportions_migrating[label]`, `selectivity_at[label]`
Conditions	Only used if `estimator=likelihood` or `Bayes` AND `dist=multinomial`, `Fournier`, or `Coleraine`.
Type	Constant vector (1 per year)
Effects	Defines the N’s by observation and year for the time series. One N for each observation in each year.
Notes	Alternatively supply a single N for each year using `N_[year]` or a single N for all years using `N`.

Example	`years 1992 1993 1994`
	`sexed 1`
	`min_class 2 2`
	`max_class 4 4`
	`# M2 M3 M4 F2 F3 F4`
	`Ns_1992 20 100 150 30 90 120`
	`Ns_1993 30 122 156 40 80 125`
	`Ns_1994 40 140 180 50 70 123`

N_process_error	**Process error N for this time series, used with likelihoods parameterised by the effective sample size.**
Command	`proportions_at[label]`, `catch_at[label]`, `proportions_mature[label]`, `proportions_migrating[label]`, `selectivity_at[label]`
Conditions	Only used if `dist=multinomial`, `Fournier`, or `Coleraine`.
Type	Estimable
Effects	Defines an N to be ‘added’ to all N’s by observation and year for the time series.

9.9 Defining the penalties

9.9.1 Ogive smoothing penalty

@ogive_smoothing_penalty	**Ogive smoothing penalty block command**
Effects	Defines any following commands as `@ogive_smoothing_penalty` subcommands
Notes	The ith `@ogive_smoothing_penalty` block relates to the ith penalty. The penalty is on the sum of squares of rth differences in the ogive values, encouraging the ogive to be like a polynomial of degree $r-1$.

label	**The name of the penalty**
Command	`ogive_smoothing_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

ogive	**The name of the ogive parameter to which the penalty is applied**
Command	`ogive_smoothing_penalty[i]`
Type	String
Effects	Defines the name of the ogive parameter to which the penalty is applied.
Notes	Should be an `allvalues` or `allvalues_bounded` ogive. See Section 3.4 for instructions on generating parameter names.

r	**Penalty is applied to *r*th differences**
Command	`ogive_smoothing_penalty[i]`
Type	Integer
Effects	Defines the effect of the penalty, which acts on the sum of squares of *r*th differences (and hence encourages the ogive to be like a polynomial of degree $r-1$).

log_scale	**Should sums of squares be calculated on the log scale?**
Command	`ogive_smoothing_penalty[i]`
Type	Switch
Default	`False`
Effects	Defines sum of squares to be calculated on the log scale (with `log_scale=true`) or the linear scale (with `log_scale=false`).

lower_bound, upper_bound	**Penalty is applied for age or size classes `lower_bound` to `upper_bound`**
Command	`ogive_smoothing_penalty[i]`
Type	2 x integer
Default	`min_age, max_age`
Effects	Defines the effect of the penalty, which acts on ogive elements for age or size classes of `lower_bound` to `upper_bound`. Everything else is dropped off before differencing.

multiplier	**Multiply the penalty by this factor**
Command	`ogive_smoothing_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.2 Catch limit penalty

@catch_limit_penalty	**Catch limit penalty block command**
Effects	Defines any following commands as `@catch_limit_penalty` subcommands
Notes	The *i*th `@catch_limit_penalty` block relates to the *i*th such penalty. The penalty is on the sum of squares of (actual catch less specified catch), optionally on a log scale, for a single fishery.

label	**The name of the penalty**
Command	`catch_limit_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

fishery	**The label of the fishery to which the penalty is applied**
Command	`catch_limit_penalty[i]`
Type	String
Effects	Defines the label of the fishery to which the penalty is applied.

log_scale	**Should sums of squares be calculated on the log scale?**
Command	`catch_limit_penalty[i]`

Type	Switch
Default	`False`
Effects	Defines sum of squares to be calculated on the log scale (with `log_scale=true`) or the linear scale (with `log_scale=false`).

multiplier	**Multiply the penalty by this factor**
Command	`catch_limit_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.3 Vector average penalty

@vector_average_penalty	**Vector average penalty block command**
Effects	Defines any following commands as `@vector_average_penalty` subcommands
Notes	The *i*th `@vector_average_penalty` block relates to the *i*th such penalty. The penalty is on the square of (mean(vector)$-k$), or of (mean(log(vector))$-l$), or of (log(mean(vector)/m)). This encourages the vector to average arithmetically to k or m, or geometrically to exp(l).

label	**The name of the penalty**
Command	`vector_average_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

vector	**The name of the vector parameter to which the penalty is applied**
Command	`vector_average_penalty[i]`
Type	String
Effects	Defines the name of the vector parameter to which the penalty is applied.
Notes	See Section 3.4 for instructions on generating parameter names.

k, l, m	**Vector should average arithmetically to *k* or *m*, or geometrically to *l*.**
Command	`vector_average_penalty[i]`
Conditions	Supply one of k, l, and m.
Type	Constant
Effects	Define the number which the vector should average to. The penalty is on the square of (mean(vector)$-k$), or of (mean(log(vector))$-l$), or of (log(mean(vector)/m)).

lower_bound, upper_bound	**Penalty is applied for vector index classes `lower_bound` to `upper_bound`**
Command	`vector_average_penalty[i]`
Type	2 x integer
Default	1, and the length of vector
Effects	Defines the effect of the penalty, which acts on vector elements of `lower_bound` to `upper_bound`. Everything else is dropped off before averageing.

multiplier	**Multiply the penalty by this factor**
Command	`vector_average_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.4 Vector smoothing penalty

@vector_smoothing_penalty	**Vector smoothing penalty block command**
Effects	Defines any following commands as `@vector_smoothing_penalty` subcommands

Notes	The *i*th `@vector_smoothing_penalty` block relates to the *i*th penalty. The penalty is on the sum of squares of *r*th differences in the vector values, encouraging the vector to be like a polynomial of degree $r-1$.

label	**The name of the penalty**
Command	`vector_smoothing_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

vector	**The name of the vector parameter to which the penalty is applied**
Command	`vector_smoothing_penalty[i]`
Type	String
Effects	Defines the name of the vector parameter to which the penalty is applied.
Notes	See Section 3.4 for instructions on generating parameter names.

r	**Penalty is applied to *r*th differences**
Command	`vector_smoothing_penalty[i]`
Type	Integer
Effects	Defines the effect of the penalty, which acts on the sum of squares of *r*th differences (and hence encourages the vector to be like a polynomial of degree $r-1$).

log_scale	**Should sums of squares be calculated on the log scale?**
Command	`vector_smoothing_penalty[i]`
Type	Switch
Default	`False`
Effects	Defines sum of squares to be calculated on the log scale (with `log_scale=true`) or the linear scale (with `log_scale=false`).

lower_bound, upper_bound	**Penalty is applied for vector index classes `lower_bound` to `upper_bound`**
Command	`vector_smoothing_penalty[i]`
Type	2 x integer
Default	1, and the length of vector
Effects	Defines the effect of the penalty, which acts on vector elements of `lower_bound` to `upper_bound`. Everything else is dropped off before differencing.

multiplier	**Multiply the penalty by this factor**
Command	`vector_smoothing_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.5 Element difference penalty

@element_difference_penalty	**Element difference penalty block command**
Effects	Defines any following commands as `@element_difference_penalty` subcommands
Notes	The *i*th `@element_difference_penalty` block relates to the *i*th such penalty. The penalty is on the square of ($\text{vector}_1[i]-\text{vector}_2[i]$), and is intended to encourage the *i*th elements of the two vectors to be similar.

label	**The name of the penalty**
Command	`element_difference_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

vector1, vector2	**The name of the vector parameters to which the penalty is applied**
Command	`element_difference_penalty[i]`
Type	2 x string
Effects	Defines the names of the vector parameters to which the penalty is applied.

Notes	See Section 3.4 for instructions on generating parameter names.

i	**Penalise differences in the *i*th elements of the two vectors**
Command	`element_difference_penalty[i]`
Type	Integer
Effects	Define which element of the two vectors is to be penalised. The penalty is on the squared difference between the *i*th elements of the vectors.

multiplier	**Multiply the penalty by this factor**
Command	`element_difference_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.6 YCS difference penalty

@YCS_difference_penalty	**YCS difference penalty block command**
Effects	Defines any following commands as `@YCS_difference_penalty` subcommands
Notes	The *i*th `@element_difference_penalty` block relates to the *i*th such penalty. The penalty is on the square of the difference in YCS for a given year for two stocks, and is intended to encourage the two stocks to have the same YCS for that year.

label	**The name of the penalty**
Command	`YCS_difference_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

stock1, stock2	**The names of the stocks to which the penalty is applied**
Command	`YCS_difference_penalty[i]`
Type	2 x string
Effects	Defines the names of the stocks to whose YCS the penalty is applied.

year	**Year for which the penalty is to be applied**
Command	`YCS_difference_penalty[i]`
Type	Integer
Effects	Define which year the penalty is applied to. The penalty is on the squared difference between the YCS of the two stocks for this year.
Notes	This refers to the year in which the year class is spawned.

multiplier	**Multiply the penalty by this factor**
Command	`YCS_difference_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.7 Similar *q*s penalty

@similar_qs_penalty	**Similar *q*'s penalty block command**
Effects	Defines any following commands as `@similar_qs_penalty` subcommands
Notes	The *i*th `@similar_qs_penalty` block relates to the *i*th such penalty. The penalty is on the square of $(\log(q_1)-\log(q_2))$, and is intended to encourage q_1 and q_2 to be similar.

label	**The name of the penalty**
Command	`similar_qs_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

q1, q2	**The names of the two *q*'s to which the penalty is applied**
Command	`similar_qs_penalty[i]`
Type	2 x string
Effects	Defines the names of the *q*'s to which the penalty is applied.
Notes	Use the labels of the *q*'s as specified in the relative observations commands.

multiplier	**Multiply the penalty by this factor**
Command	`similar_qs_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.8 Ratio *q*s penalty

@ratio_qs_penalty	**Ratio *q*'s penalty block command**
Effects	Defines any following commands as `@ratio_qs_penalty` subcommands
Notes	The penalty is a lognormal density function on the ratio q_1/q_2 (i.e., in the case of two *q*'s, this is the ratio of q_B/q_A) When this penalty is used the prior for one or both of the *q*s should be uniform (see Section 6.7.5).

label	**The name of the penalty**
Command	`ratio_qs_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

q1, q2	**The names of the two *q*'s to which the penalty is applied**
Command	`ratio_qs_penalty[i]`
Type	2 x string
Effects	Defines the names of the *q*'s to which the penalty is applied.
Notes	Use the labels of the *q*'s as specified in the relative observations commands.

mu	**Mean of the lognormal for the penalty on q_1/q_2**
Command	`ratio_qs_penalty[i]`
Type	Constant
Effects	Defines the mean of the lognormal .

cv	**C.v. of the lognormal for the penalty on q_1/q_2**
Command	`ratio_qs_penalty[i]`
Type	Constant
Effects	Defines the c.v. of the lognormal.

9.9.9 Ogive comparison penalty

@ogive_comparison_penalty	**Ogive comparison penalty block command**
Effects	Defines any following commands as `@ogive_comparison_penalty` subcommands
Notes	The *i*th `@ogive_comparison_penalty` block relates to the *i*th such penalty. The penalty is on the sum of squares of $\max(\text{ogive}_1-\text{ogive}_2, 0)$, and is intended to encourage ogive_1 to be at or below ogive_2. Cannot be used on size-based ogives in an age-based model.

label	**The name of the penalty**
Command	`ogive_comparison_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

ogive1, ogive2	**The name of the ogive parameters to which the penalty is applied**
Command	`ogive_comparison_penalty[i]`
Type	2 x string
Effects	Defines the names of the vector parameters to which the penalty is applied.
Notes	See Section 3.4 for instructions on generating parameter names.

lower_bound, upper_bound	**Penalty is applied for age or size classes `lower_bound` to `upper_bound`**
Command	`ogive_comparison_penalty[i]`
Type	2 x integer
Default	`min_age`, `max_age`
Effects	Defines the effect of the penalty, which acts on ogive elements for age or size classes of `lower_bound` to `upper_bound`. Everything else is dropped off before comparing.
multiplier	**Multiply the penalty by this factor**
Command	`ogive_comparison_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.10 Ogive difference penalty

@ogive_difference_penalty	**Ogive difference penalty block command**
Effects	Defines any following commands as `@ogive_difference_penalty` subcommands
Notes	The *i*th `@ogive_difference_penalty` block relates to the *i*th such penalty. The penalty is on the square of ($\text{ogive}_1 - \text{ogive}_2$) for a given size or age class, and is intended to encourage the two ogives to have similar values for that size or age class. Cannot be used on size-based ogives in an age-based model.
label	**The name of the penalty**
Command	`ogive_difference_penalty[i]`
Type	String
Effects	Defines a text label for the penalty
ogive1, ogive2	**The name of the ogive parameters to which the penalty is applied**
Command	`ogive_difference_penalty[i]`
Type	2 x string
Effects	Defines the names of the vector parameters to which the penalty is applied.
Notes	See Section 3.4 for instructions on generating parameter names.
class	**The age, or size class number, to which the penalty is applied**
Command	`ogive_difference_penalty[i]`
Type	Integer
Effects	Defines the effect of the penalty, which acts on the specified age, or size class. If an age class, give the age in years. If a size class, give a number from 1, 2, … where 1 is the smallest size class.
multiplier	**Multiply the penalty by this factor**
Command	`ogive_difference_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.9.11 Fish tagged penalty

@fish_tagged_penalty	**'Fish must be tagged' penalty block command**
Effects	Defines any following commands as `@fish_tagged_penalty` subcommands
Notes	The *i*th `@fish_tagged_penalty` block relates to the *i*th such penalty. The penalty is on the square of (number of fish actually tagged less number of fish meant to be tagged), and is intended to discourage parameter values which do not lead to enough fish being present at a tagging episode.

label	**The name of the penalty**
Command	`fish_tagged_penalty[i]`
Type	String
Effects	Defines a text label for the penalty

tagging_episode	**The name of the tagging episode**
Command	`fish_tagged_penalty[i]`
Type	string
Effects	Defines the tagging episode (a single tag applied in a single year) to which the penalty is applied.
Notes	The value of this parameter should be the label of the relevant @tag command in the population parameter file.

multiplier	**Multiply the penalty by this factor**
Command	`fish_tagged_penalty[i]`
Type	Constant
Effects	Defines the factor by which the penalty is multiplied.
Notes	The larger this number, the more severe the penalty.

9.10 Defining the ageing error

@ageing_error	**Ageing error block command**
Conditions	Only used in an age-based model.
Effects	Defines any following commands as `@ageing_error` subcommands.
Notes	Note spelling.

type	**Type of ageing error model**
Command	`ageing_error`
Conditions	Only used in an age-based model.
Type	String
Default	`none` (i.e., no ageing error)
Effects	Defines the ageing error model as `none`, `off_by_one`, `normal`, or `misclassification_matrix`.
Notes	Note punctuation.

p1, p2, k	**Parameters of the `off_by_one` ageing error model**
Command	`ageing_error`
Conditions	Only used in an age-based model. Only used if `type=off_by_one`.
Type	2 x estimable, 1 x integer
Default	The default of `k` is 0, i.e., fish of all ages can be misclassified. There are no defaults for `p1` and `p2`.
Effects	`p1` and `p2` define the proportions of misclassifications down and up by 1 year respectively. `k` defines the minimum age of fish which can be misclassified — fish under age `k` have no ageing error.

c	**Parameter of the `normal` ageing error model**
Command	`ageing_error`
Conditions	Only used in an age-based model. Only used if `type=normal`.
Type	Estimable
Effects	Define the c.v. of misclassification.

[age]	**Row of the misclassification matrix for the `misclassification_matrix` ageing error model.**
Command	`ageing_error`
Conditions	Only used in an age-based model. Only used if `type=misclassification_matrix`.
Type	Constant vector
Effects	Define a row of the misclassification matrix.

Enter one row per age class. The subcommand is the age class, from `min_age` to `max_age`.

Notes: Each row corresponds to a true age, each column to a reported age.

Example: With `min_age=2`, `max_age=4`, you might enter commands as follows,

```
2 0.9 0.1 0.0
3 0.1 0.7 0.2
4 0.0 0.2 0.8
```

Then, for example, 20% of 4-year-olds are misclassified as 3-year-olds.

9.11 CASAL extensions

@user_parameterisation **Is there a reparameterisation of the population section using `user.parameterisation.cpp`?**

Type	Switch
Default	`False` (i.e., no reparameterisation)
Effects	Tells CASAL whether the population parameters are transformed using the user-supplied function `user.parameterisation.cpp`.
Notes	Only available if you have access to the CASAL source code.

@user_components **Lists the names of the user-defined priors or penalties calculated in `user.prior_penalty.cpp`**

Conditions	Only used if you have written code in `user.prior_penalty.cpp`
Type	Vector of strings
Default	No user-defined objective function components
Effects	Gives the labels of each of the objective function components calculated in `user.prior_penalty.cpp`.
Notes	Only available if you have access to the CASAL source code.

9.12 Other commands

@comment **User supplied comment**

Type	String
Default	None
Effects	Places a copy of the supplied text into any output files generated
Notes	Allows the user to place a comment into the CASAL output files. See also the `@comment` command in the `population.csl` and `output.csl` files

10. THE OUTPUT.CSL FILE

The output parameters are specified in the `output.csl` file. See Section 7 for information about the output section, and Section 3.4 for instructions on writing a CASAL data file.

10.1 Defining the printouts

@print **Printouts block command**

Effects	Defines any following commands as `@print` subcommands.

parameters **Print the population, estimation, and output parameters?**

Command	`print`
Type	Switch
Default	`False` (do not print)
Effects	Print the population, estimation, and output parameters to the standard output as they are read in from `population.csl, estimation.csl, output.csl`

unused_parameters **Print a list of the parameters that were never used?**

Command	`print`
Conditions	Only usable with `-r`, `-e`, `-E`.
Type	Switch
Default	`False` (do not print)
Effects	Print a list of the names of parameters which were never accessed by CASAL.
Notes	There are several reasons why a parameter might occur on this list. 1. Because its name was spelt incorrectly 2. Because it was an unnecessary or nonexistent parameter 3. Because the task you were doing does not use that parameter, e.g., `casal -r` does not use `max_iters` 4. Something unexpected happened.

population_section **Print a description of the population section?**

Command	`print`
Conditions	Unless suppressed by `-q`
Type	Switch
Default	`False` (do not print)
Effects	Print a text description of the population section to the standard output.

requests, results **Print a description of the requests sent to the population section and the corresponding results?**

Command	`print`
Conditions	Unless suppressed by `-q`
Type	2 x switch
Default	`False` (do not print)
Effects	Print the requests sent to the population section by the estimation and output sections, and the corresponding results, to the standard output.
Notes	Can produce a lot of output, only intended for use when debugging.

initial_state, state_annually, state_every_step, final_state **Print the state of the population?**

Command	`print`
Conditions	Unless suppressed by `-q`
Type	4 x switch
Default	`False` (do not print)
Effects	Print the initial state, the state after every year, after every step, or the final state, to the standard output.
Notes	Can produce a lot of output, only intended for use when debugging.

estimation_section **Print a description of the estimation section?**

Command	`print`

Type	Switch
Default	`False` (do not print)
Effects	Print a text description of the estimation section to the standard output.

every_mean_size	**Print mean-sizes-at-age for every time step of every year?**
Command	`print`
Conditions	Only used in an age-based model if `@print.population_section` is `True`, unless suppressed by `-q.`
Type	Switch
Default	`False` (do not print)
Effects	Print mean-sizes-at-age for every time step of every year from *initial* to *final*. Used to check that the fish growth model is working correctly.

fits, resids, pearson_resids, normalised_resids	**Print the fits, residuals, and standardised residuals?**
Command	`print`
Type	4 x switch
Default	`False` (do not print)
Effects	At the end of a model run (`-r`) or a parameter estimation (`-e, -E`), print the fits, residuals, Pearson residuals, and/or normalised residuals to the standard output.

covariance	**Print the approximate covariance matrix of the free parameters**
Command	`print`
Type	Switch
Default	`False` (do not print)
Effects	At the end of a parameter estimation (`-e, -E`), print the approximate covariance matrix of the free parameters, i.e., the inverse of the approximation to the Hessian.
Notes	The minimiser needs a number of iterations to build up an accurate Hessian approximation. If few iterations were done, the approximation will be poor. In any case the inverse Hessian is not a good approximation to the covariance of the free parameters, and should not be used for, for example, constructing confidence bounds on parameters.

eigenvalues	**Print the eigenvalues of the Hessian matrix**
Command	`print`
Type	Switch
Default	`False` (do not print)
Effects	At the end of a parameter estimation (`-e, -E`), print out the sorted list of eigenvalues of the Hessian matrix.
Notes	The minimiser needs a number of iterations to build up an accurate Hessian approximation. If few iterations were done, the approximation will be poor.

yields	**Print a description of the yield calculations?**
Command	`print`
Conditions	Only used during yield calculations.
Type	Switch
Default	`True` (print)
Effects	Print a text description of the yield calculations to the standard output.

fits_every_eval, objective_every_eval, parameters_every_eval, parameter_vector_every_eval	**Print the fits, objective function, or parameters at every function evaluation?**
Command	`print`
Type	4 x switch
Default	`False` (do not print)
Effects	During each evaluation of the objective function, print the fits and residuals, objective function and components, free parameters in a verbose format, and/or free parameters as a vector, to the standard output.
Notes	Can produce a lot of output, only intended for use when debugging.

@print_sizebased_ogives_at **Sizes for which size-based ogives should be printed in an age-based model**

Conditions	If printing of size-based ogives in an age-based model is required then the command must be specified. Ignored in a size-based model.
Type	Constant vector
Effects	Defines the sizes for which each size-based ogive will be output, (i) if you ask for a printout of a population section which has size-based selectivity ogives, and (ii) if your output quantities include any of the size-based ogive parameters.

10.2 Defining the output quantities

@quantities **Output quantities block command**

Effects	Defines any following commands as `@quantities` subcommands.

all_free_parameters **Output quantities include all free parameters?**

Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include all free parameters in the output quantities.
Notes	If a parameter is an ogive, then the values of the ogive are printed, not the ogive arguments.

scalar_parameters, vector_parameters, ogive_parameters **Output quantities include these parameters**

Command	`quantities`
Type	3 x vector of strings
Default	`False` (do not print)
Effects	Include some parameters in the output quantities. For each of these three subcommands, give a list of parameter names (which can be constant or free).
Notes	Generate parameter names as described in Section 3.4. If a parameter is an ogive, then the values of the ogive are printed, not the ogive arguments.

ogive_arguments **Output quantities include these ogive arguments**

Command	`quantities`
Type	Vector of strings
Default	`False` (do not print)
Effects	Include the arguments of some ogives in the output quantities. Give a list of ogive parameter names.
Notes	Generate parameter names as described in Section 3.4.

nuisance_qs **Output quantities include nuisance q's?**

Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include nuisance q's in the output quantities.

B0, R0, Bmean, Rmean Binitial, Rinitial **Output quantities include B_0, R_0, B_{mean}, R_{mean}, $B_{initial}$, $R_{initial}$?**

Command	`quantities`
Type	6 x switch
Default	`False` (do not print)
Effects	Include B_0, R_0, B_{mean}, R_{mean}, $B_{initial}$, $R_{initial}$ for each stock in the output quantities.

SSBs **Output quantities include SSBs?**

Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include SSBs for each stock in the output quantities.

actual_catches, actual_catches_by_stock **Output quantities include actual catches (by stock)?**
Command `quantities`
Type 2 x switch
Default `False` (do not print)
Effects Include actual catches or actual catches of each stock in the output quantities.

removals **Output quantities include removals?**
Command `quantities`
Conditions Only available if either `quantities.actual_catches` or `quantities.actual_catches_by_stock` are `True`
Type 2 x switch
Default `False` (do not print)
Effects Include removals (if `quantities.actual_catches)` or removals from each stock (if `quantities.actual_catches_by_stock)` in the output quantities.

discards **Output quantities include discards?**
Command `quantities`
Conditions Only available if either `quantities.actual_catches` or `quantities.actual_catches_by_stock` are `True`
Type 2 x switch
Default `False` (do not print)
Effects Include discards (if `quantities.actual_catches)` or discards from each stock (if `quantities.actual_catches_by_stock)` in the output quantities.

recruitments **Output quantities include recruitments?**
Command `quantities`
Type Switch
Default `False` (do not print)
Effects Include recruitments (as absolute numbers of fish, indexed by the year in which they recruit) in the output quantities.

YCS **Output quantities include YCS?**
Command `quantities`
Type Switch
Default `False` (do not print)
Effects Include YCS (as deviates, indexed by the year in which they are spawned) in the output quantities.
Notes The YCS can be different from the YCS parameter you input for several reasons. First, because if `n_rinitial` > 0, the YCS are prefixed by one or more entries of $R_{initial}$. Second, if using the Haist YCS parameterisation. Third, if you are randomising some YCS in projection, or all YCS in yield simulation.

true_YCS **Output quantities include 'true YCS'?**
Command `quantities`
Type Switch
Default `False` (do not print)
Effects Include `true_YCS` in the output quantities.
Notes See *YCS* above. The 'true YCS' are arguably more informative than the YCS when there is a stock- or climate-recruitment relationship. Multiplying the `true_YCS` in year y by R_0 gives the absolute recruitment in year *y*+*y_enter*.

Ts **Output quantities include climate variable *T*?**
Command `quantities`
Conditions A climate-recruitment relationship is used
Type Switch
Default `False` (do not print)
Effects Include the climate variable T (for each stock) in the output quantities.

migration_annual_variation **Output quantities include migration_annual_variation**

Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include migration annual variation (only if annual variation is defined for any migration episode)
Notes	The annual variation can be different from the `annual_variation` parameter for several reasons. First, annual_variation in years outside the range of annual_variation_years, are prefixed by 1. Second, if you are randomising some annual variations in projection or all annual variations in yield simulations.

fishing_pressures **Output quantities include fishing pressures?**

Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include fishing pressures for each fishery in the output quantities.

stock_crash **Output quantities include ‘stock crash’?**

Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include the ‘stock crash’ quantity for each stock in the output quantities. This is the indicator of the event that SSB falls below 20% B_0 in the projection period.
Notes	For projections only. The stock risk is the projected expectation of this quantity.

disease_biomass_loss **Output quantities include the disease biomass loss resulting from a disease event?**

Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include the effective disease biomass loss quantity for each stock in the output quantities. This is the effective biomass lost from a disease event.
Notes	Can only be used if disease mortality is defined

tagged_age_distribution **Output quantities include the age breakdown of tagged fish in each tagging event?**

Command	`quantities`
Conditions	only produces output for tag events with @tag.`release_type=deterministic`
Type	switch
Default	`False` (do not print)
Effects	Includes the age distribution of tagged fish, over all age classes in the partition, by sex if sex is a partition character, for each tagging event having @tag.`release_type=deterministic`
Warning	Only for use in a model with tagging in the partition that includes a tag-release event.

fits **Output quantities include the ‘fits’ or the expected value for each observation?**

Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include the ‘fits’ quantity for each stock in the output quantities.
Warning	The use of this option can generate a large amount of output, depending on the number of observations in your input data files.

resids **Output quantities include the ‘residuals’ or the observed less expected value for each observation?**

Command	`quantities`

Type	Switch
Default	`False` (do not print)
Effects	Include the 'resids' quantity for each stock in the output quantities.
Warning	The use of this option can generate a large amount of output, depending on the number of observations in your input data files.

pearson_resids	**Output quantities include the 'Pearson residuals' for each observation?**
Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include the 'pearson_resids' quantity for each stock in the output quantities.
Warning	The use of this option can generate a large amount of output, depending on the number of observations in your input data files.

normalised_resids	**Output quantities include the 'normalised residuals' for each observation?**
Command	`quantities`
Type	Switch
Default	`False` (do not print)
Effects	Include the 'normalised_resids' quantity for each stock in the output quantities.
Warning	The use of this option can generate a large amount of output, depending on the number of observations in your input data files.

Pseudo-fits are a special case. Specify the pseudo-observations using the observations parameters in Section 9.7. Do not use relative observation types. Do not provide the observations values, or any of the commands in Section 9.8, which relate to the objective function.

10.3 Defining projections

Most of the projections parameters are in the `population.csl` file. See `@final`, the recruitment variability parameters in Section 8.4, and the future catch parameters in Section 8.10. There is one projections parameter in the `output.csl` file.

@n_projections	**Number of projections to be done (from a point estimate)**
Conditions	Only used when projecting from a point estimate (as opposed to a sample from the posterior)
Type	Integer
Default	300
Effects	Defines the number of projections to be done.
Notes	When projecting from a sample from the posterior, 1 projection is done for each sample point.

See also the output quantity commands in Section 10.2.

10.4 Defining yield calculations

@catch_split	**Catch split used in yield simulations**
Conditions	Only used in yield simulations, if there is more than one fishery.
Type	Constant vector
Effects	Defines a catch split used in yield simulations, as proportions of the total catch by fishery.
Notes	Order the proportions in the same order as the `annual.cycle.fishery_names` parameter in `population.csl`. If they do not sum to 1, they will be rescaled to sum to 1. (Check the output to make sure you specified this parameter correctly.)

@B_pre	**Pre-fishery biomass B_{pre} block command**
Conditions	Only used if the mortality rate in simulations is an exploitation rate (i.e., if the Baranov catch equation is not used).
Effects	Defines any following commands as `B_pre` subcommands.
Notes	B_{pre} is the pre-fishery biomass which is multiplied by the mortality rate to yield the catch, in yield simulations. Note that `B_pre` is probably necessary when estimating CAY in multi-area multi-stock models, see Section 7.5.1 for detail. Note spelling (`B_pre` and not `Bpre`).

mature_only	**Pre-fishery biomass B_{pre} is mature fish only**
Command	`B_pre`
Conditions	Used only if the mortality rate in simulations is an exploitation rate (i.e., if the Baranov catch equation is not used).
Type	Switch
Default	No default
Effects	Defines B_{pre} to include mature fish only.

area	**Area for which pre-fishery biomass B_{pre} is calculated**
Command	`B_pre`
Conditions	Used only if the mortality rate in simulations is an exploitation rate (i.e., if the Baranov catch equation is not used).
Type	String
Default	All areas combined
Effects	Defines B_{pre} to be calculated for a single area. Use an area label as in *area_names*.

step	**Time step in which pre-fishery biomass B_{pre} is calculated**
Command	`B_pre`
Conditions	Used only if the mortality rate in simulations is an exploitation rate (i.e., if the Baranov catch equation is not used).
Type	Integer
Default	The time step of first fishery in the year
Effects	Defines the time step in which B_{pre} is calculated.
Notes	This time step must be earlier than the time step of any fishery, or it can be the time step of the first fishery if `proportion_mortality=0`.

proportion_mortality	**Proportion of the time step's mortality after which pre-fishery biomass B_{pre} is calculated**
Command	`B_pre`
Conditions	Used only if the mortality rate in simulations is an exploitation rate (i.e., if the Baranov catch equation is not used).
Type	Constant
Default	0
Effects	Defines the proportion of the mortality in the time step after which B_{pre} is calculated.
Notes	This has to be 0 unless `step` is earlier than the time step of any fishery.

selectivity	**Selectivity ogive with which pre-fishery biomass B_{pre} is calculated**
Command	`B_pre`
Conditions	Used only if the mortality rate in simulations is an exploitation rate (i.e., if the Baranov catch equation is not used).
Type	String
Default	No default
Effects	Defines the selectivity ogive with which B_{pre} is calculated. Use a selectivity label from `selectivity_names`.

10.5 Defining deterministic yields

@per_recruit	**Per-recruitment block command**
Conditions	Used only in yield calculations.

Effects	Defines any following commands as `@per_recruit` subcommands.

do_YPR_SPR	**Supply data to plot a YPR or SPR curve?**
Command	`per_recruit`
Type	Switch
Default	`False` (do not supply data)
Effects	Print out `F`, `YPR`, and `SPR` for a list of mortality rates you supply through the *F* subcommand.

F	***F*'s at which to calculate YPR & SPR**
Command	`per_recruit`
Conditions	Used only if `do_YPR_SPR` is set.
Type	Constant vector
Effects	Defines the mortality rates *F* for which `YPR`, and `SPR` should be printed. In multi-stock models the same mortality rates are used for each stock.

do_Fmax	**Calculate F_{max}?**
Command	`per_recruit`
Conditions	Cannot be used for multi-stock models.
Type	Switch
Default	`False` (do not calculate F_{max})
Effects	Print out F_{max}

do_F0_1	**Calculate $F_{0.1}$?**
Command	`per_recruit`
Conditions	Cannot be used for multi-stock models.
Type	Switch
Default	`False` (do not calculate $F_{0.1}$)
Effects	Print out $F_{0.1}$
Notes	Note spelling

do_Fx	**Calculate $F_{x\%}$?**
Command	`per_recruit`
Conditions	Cannot be used for multi-stock models.
Type	Switch
Default	`False` (do not calculate $F_{x\%}$)
Effects	Print out $F_{x\%}$
Notes	The value of *x* is supplied through the *x* subcommand.

x	***x* at which to calculate $F_{x\%}$**
Command	`per_recruit`
Conditions	Used only if `do_Fx` is set.
Type	Constant vector
Effects	Defines the percentage *x* for which $F_{x\%}$ should be calculated.
Notes	Supply a percentage not a proportion.

guess	**First guess at F_{max}, $F_{0.1}$, $F_{x\%}$**
Command	`per_recruit`
Conditions	Used only if `do_Fmax`, `do_F0_1`, or `do_Fx` is set.
Type	Constant
Effects	Defines a guesstimate of *F* used by the minimiser in the calculation of F_{max}, $F_{0.1}$ and $F_{x\%}$.

@deterministic_MSY	**Deterministic MSY block command**
Conditions	Used only in yield calculations.
Effects	Defines any following commands as `@deterministic_MSY` subcommands.

do_MSY	**Calculate deterministic MSY?**
Command	`deterministic_MSY`
Type	Switch
Default	`False` (do not calculate MSY)
Effects	Print out deterministic MSY

do_yield_vs_SSB	**Supply data to plot a yield versus SSB curve?**
Command	`deterministic_MSY`
Type	Switch
Default	`False` (do not supply data)
Effects	Print out *F*, yield, and SSB for a list of mortality rates you supply through the `F` subcommand.

F	**F's at which to calculate yield and SSB**
Command	`deterministic_MSY`
Conditions	Used only if `do_yield_vs_SSB` is set.
Type	Constant vector
Effects	Defines the mortality rates *F* for which F, yield, and SSB should be printed.

guess	**First guess at F_{MSYdet}**
Command	`deterministic_MSY`
Conditions	Used only if `do_MSY` is set.
Type	Constant
Effects	Defines a guesstimate of *F* used by the minimiser in the calculation of F_{MSY_det}.

10.6 Defining stochastic yields

@MCY_CAY	**MCY/CAY block command**
Conditions	Used only in yield calculations.
Effects	Defines any following commands as `@MCY_CAY` subcommands.

do_MCY	**Calculate MCY?**
Command	`MCY_CAY`
Type	Switch
Default	`False` (do not calculate MCY)
Effects	Print out MCY

do_CAY	**Calculate CAY?**
Command	`MCY_CAY`
Type	Switch
Default	`False` (do not calculate CAY)
Effects	Print out CAY

interactive	**Should MCY/CAY be calculated interactively?**
Command	`MCY_CAY`
Conditions	Used only if `do_MCY` or `do_CAY` is set.
Type	Switch
Default	`False` (calculate MCY/CAY using the automated minimiser)
Effects	Calculate MCY and CAY interactively, i.e., you will be prompted for values of the harvest rate *H*.

p, q	**Risk constraints in MCY/CAY analysis**
Command	`MCY_CAY`
Conditions	Used only if `do_MCY` or `do_CAY` is set.
Type	2 x constant
Default	0.2, 0.1
Effects	Define the risk constraint parameters *p* and *q*. The stock is at risk if SSB < pB_0, but this can safely happen up to proportion *q* of the time.
Notes	*p* and *q* are proportions not percentages.

MCY_CAY_risk_year	**Specify the year of which the SSB will be used in the MCY/CAY risk constraint**
Command	`MCY_CAY`
Type	Integer
Default	No year: instead use the unfished equilibrium SSB, B_0, as the reference point

Effects	Specify the year of which the SSBs will be used in the risk constraint for all MCY/CAY analysis. The constraint requires that the spawning stock biomass falls below $p \times SSB_{MCY\_CAY\_risk\_year}$ less than $q \times 100\%$ of the time.

n_simulations **Number of simulations for each harvest rate *H* in MCY/CAY analyses (point-based only)**

Command	`MCY_CAY`
Conditions	Used only if `do_MCY` or `do_CAY` is set, and you have supplied exactly one vector of free parameters using `-i`.
Type	Integer
Default	100
Effects	Define the number of point-based simulations to be done for each harvest rate *H*.
Notes	For sample-based simulations, one simulation is done for each sample from the posterior.

n_discard, n_keep **Number of years to discard and to keep in each MCY/CAY simulation.**

Command	`MCY_CAY`
Conditions	Used only if `do_MCY` or `do_CAY` is set.
Type	2 x integer
Effects	Define the numbers of years to be kept and discarded in each MCY/CAY simulation.
Notes	Try several different values and see if it makes a difference.

max_upper_iter **Maximum number of times upper bound on H can be increased when searching for an optimal yield**

Command	`MCY_CAY`
Conditions	Used only if `do_MCY` or `do_CAY` is set.
Type	Integer
Default	40
Effects	Defines the maximum number of times that the optimisation routine can consecutively increase the upper bound on H when searching for an optimum yield.
Notes	Normally the upper bound will be increased repeatedly only when the yield exhibits asymptotic behaviour.

MCY_uncertainty_dist **Distribution of uncertainty in virgin biomass (point-based only)**

Command	`MCY_CAY`
Conditions	Used only if `do_MCY` is set, and you have supplied exactly one vector of free parameters using `-i`.
Type	String
Default	`none`
Effects	Define the distribution of uncertainty in virgin abundance in MCY simulations as `none`, `normal`, or `lognormal`.
Notes	Not used for sample-based simulations, where the posterior distribution should reflect the uncertainty in virgin biomass.

MCY_uncertainty_cv **C.v. of uncertainty in virgin biomass (point-based only)**

Command	`MCY_CAY`
Conditions	Used only if `uncertainty_dist` is set to `normal` or `lognormal`.
Type	Constant
Default	0.2
Effects	Define the c.v. of uncertainty in virgin abundance in MCY simulations.

MCY_guess **First guess at MCY**

Command	`MCY_CAY`
Conditions	Used only if `do_MCY` is set.
Type	Constant
Effects	Defines a guesstimate of MCY used by the minimiser.

CAY_uncertainty_dist **Distribution of annual stock-assessment uncertainty**

Command	`MCY_CAY`

Conditions	Used only if `do_CAY` is set and if the mortality rate is an exploitation rate (i.e., if the Baranov catch equation is not used).
Type	String
Default	`lognormal`
Effects	Define the distribution of annual stock-assessment uncertainty in CAY simulations as `none`, `normal`, or `lognormal`.
Notes	Can be used for both point- and sample-based simulations.

CAY_uncertainty_cv **C.v. of annual stock-assessment uncertainty**

Command	`MCY_CAY`
Conditions	Used only if `do_CAY` is set, if the mortality rate is an exploitation rate (i.e., if the Baranov catch equation is not used), and CAY`_uncertainty_dist` is either `normal` or `lognormal`.
Type	Constant
Default	0.2
Effects	Define the c.v. of annual stock-assessment uncertainty in CAY simulations.

F_CAY_guess **First guess at F_{CAY}**

Command	`MCY_CAY`
Conditions	Used only if `do_CAY` is set.
Type	Constant
Effects	Defines a guesstimate of F_{CAY} used by the minimiser in the calculation of CAY.

@CSP **CSP block command**

Conditions	Used only in yield calculations.
Effects	Defines any following commands as *`@CSP`* subcommands.

do_CSP **Calculate CSP?**

Command	`CSP`
Type	Switch
Default	`False` (do not calculate CSP)
Effects	Print out CSP

individual_stocks **Calculate CSP by individual stock?**

Command	`CSP`
Conditions	Used only if `do_CSP` is set, in a multi-stock model
Type	Switch
Default	`False` (calculate a single overall CSP)
Effects	Print out CSP by individual stock, rather than overall

CSP_guess **First guess at CSP**

Command	`CSP`
Conditions	Used only if `do_CSP` is set.
Type	Constant
Effects	Defines a guesstimate of CSP used by the minimiser.

@B_post **Post-fishery biomass B_{post} block command**

Conditions	Used only if `do_CSP` is set.
Effects	Defines any following commands as `@B_post` subcommands.
Notes	B_{post} is the post-fishery biomass whose expectation is the same in year `current` as in year `current+1`, if the catch in year `current+1` is the CSP. Note spelling (`B_post` and not `Bpost`).

mature_only **Post-fishery biomass B_{post} is mature fish only**

Command	`B_post`
Conditions	Used only if `do_CSP` is set.
Type	Switch
Default	`True` (B_{post} includes mature fish only)
Effects	Defines B_{post} to include mature fish only.

area	**Area for which post-fishery biomass B_{post} is calculated**
Command	`B_post`
Conditions	Used only if `do_CSP` is set.
Type	String
Default	All areas combined
Effects	Defines B_{post} to be calculated for a single area. Use an area label as in `area_names`.

step	**Time step in which post-fishery biomass B_{post} is calculated**
Command	`B_post`
Conditions	Used only if `do_CSP` is set.
Type	Integer
Default	Time step of first fishery in the year
Effects	Defines the time step in which B_{post} is calculated.
Notes	This time step must be earlier than the time step of any fishery, or it can be the time step of the first fishery if `proportion_mortality=0`.

proportion_mortality	**Proportion of the time step's mortality after which post-fishery biomass B_{post} is calculated**
Command	`B_post`
Conditions	Used only if `do_CSP` is set.
Type	Constant
Default	0
Effects	Defines the proportion of the mortality in the time step after which B_{post} is calculated.
Notes	This has to be 0 unless `step` is earlier than the time step of any fishery.

selectivity	**Selectivity ogive with which post-fishery biomass B_{post} is calculated**
Command	`B_post`
Conditions	Used only if `do_CSP` is set.
Type	String
Default	No selectivity
Effects	Defines the selectivity ogive with which B_{post} is calculated. Use a selectivity label from `selectivity_names`.

10.7 Other commands

@comment	**User supplied comment**
Type	String
Default	None
Effects	Places a copy of the supplied text into any output files generated
Notes	Allows the user to place a comment into the CASAL output files. See also the `@comment` command in the `population.csl` and `estimation.csl` files

11. TROUBLESHOOTING

11.1 Typical errors

CASAL is a complex system providing many opportunities for error — either because your parameter files do not correctly specify your model, or because the model you tried to specify does not work. When in doubt, ask an experienced user. Debugging versions of CASAL can be compiled that help to track down cryptic errors.

A short list of common errors that have been seen include;

Misspelt commands

If a command name is misspelt in a parameter file, CASAL will not read it. There are two possible consequences. If the command is compulsory, CASAL will error out, alerting that it failed to find a parameter. If it is optional, CASAL may run but do the wrong thing. One way to diagnose misspelt commands is using `print.unused_parameters` command in the input data file `output.csl` (see Section 10.1). Note that commands are case sensitive.

No carriage return at the end of a parameter file

If you don't put a carriage return after your last command, then that last command will not be read by CASAL. See above.

Misspelt arguments

Probably more likely to be picked up by CASAL than misspelt commands, but still a potential problem. Note that almost all arguments are case sensitive.

Commented out command

If the following lines appear in an input data file, for example,

```
@size_based 0
@fishery trawl
selectivity trawl_sel
…
```

and you comment out the `@fishery` line, then this will affect the `@size_based` command. CASAL will think `@size_based` is the beginning of a command block, because it now appears to have arguments (`selectivity`, etc.). Whereas in fact it is a single standalone command and should be treated as such. So the correct value of `size_based` will not be read. The fix is to comment out or remove the entire `@fishery` block, and not just the first line.

No penalty on exceeding exploitation rate limits

Unless you use penalties, there is nothing to stop CASAL from generating a parameter estimate under which there is not enough fish for the catches to be taken. You probably want to use a catch limit penalty for each fishery (see Section 6.7.6).

No penalty on year class strengths not averaging to 1

If you estimate recruitment with the standard or Haist YCS parameterisation, you may want to force year class strengths to average to 1. You need to use a vector average penalty (see Section 6.7.6) to do this. If you don't, you may find that all your year class strengths are much more than 1 or much less than 1.

Proportions maturing specified as proportions mature

Be clear that the maturation rates supplied to CASAL (when maturity is a partition character) are proportions of fish which *become* mature at each age, not proportions which *are* mature at each age.

11.2 Other errors

When CASAL generates an error and the error message makes no sense, please let the CASAL development team know. Even if you manage to fix the problem yourself, the development team may be able to implement a more helpful error message and make life easier for the next person to encounter the problem. Guidelines for reporting an error are given below in Section 11.4.

Some parameter values of functions or ogives can result in either very large or very small numbers. These can, on occasions, generate internal numeric overflow errors within CASAL. The most common cause of a "Betadiff error" (especially when accompanied by the message "Betadiff returned a NaN gradient for parameters …") is a calculation that has resulted in an overflow error.

CASAL does do some range checking of parameter values before attempting calculations, but there will be some instances where these checks are not adequate. For example, in the logistic ogive, a very small value of $a_{\mathrm{to}95}$ (say, less than 0.2) can result in a very large value of $19^{(a_{50}-x)/a_{to95}}$ for some values of x. Without range checking, this would cause an overflow error. The work around to this type of error is to impose bounds on parameters that exclude the possibility of an overflow error.

However, errors that contain a 'Betadiff error' can also occur in other circumstances and typically mean that an error has been generated that should have been trapped by CASAL (for example, an array indexing error). If you manage to generate a 'betadiff error' please report it to the CASAL development team (see Section 11.3), so that it can be checked and hopefully trapped with a more useful error message.

11.3 Reporting errors

To report a bug or problem with CASAL, please send a bug report to CASAL@niwa.co.nz — after reading the guidelines below.

Use the text "CASAL bug report" as the subject line in the email. Following these guidelines will assist the CASAL development team to identify, reproduce, and hopefully solve any reported bugs. It is helpful to be as specific as possible when describing the problem. But before submitting a bug report, please check that you are using the most recent version of CASAL (see http://www.niwa.co.nz/our-science/fisheries/tools/casal for details about the current release of CASAL).

Note that CASAL is distributed as unsupported software. NIWA does not usually provide help for users of CASAL outside of NIWA. However, the CASAL development team may provide assistance to reasonable requests, but while they would appreciate being notified of any problems or errors in CASAL, they may not be able to provide timely solutions.

11.4 Guidelines for reporting a bug in CASAL

1. Detail the version of CASAL are you using? e.g., “CASAL v2.20-2008/02/14 Microsoft Windows executable”

2. What operating system/environment are you using? e.g., “IBM-PC Intel CPU running Microsoft Windows XP Release 2”.

3. Give a brief one-line description of the problem, e.g., “a segmentation fault was reported” or “a betadiff error was reported”.

4. If the problem is reproducible, please list the *exact* steps required to cause it, remembering to include the relevant CASAL input parameter files. Also specify the exact command line arguments that were used, e.g., “Using the command `casal -e -q -f my- > logfile.out` reports a segmentation fault. The files my-population.csl, my-estimation.csl, and my-output.csl are attached.”

5. If the problem is not reproducible (happened only once, or occasionally for no apparent reason), please describe the circumstances in which it occurred and the symptoms observed (but note it is much harder to reproduce and hence fix non-reproducible bugs, but if several reports are made over time that relate to the same thing, then this may help to track down the problem), e.g., “CASAL crashed, but I cannot reproduce how I did it. It seemed to be related to a local network crash but I cannot be sure.”

6. If the problem causes any error messages to appear, please give the exact text displayed, e.g., “Segmentation fault (core dumped)”

7. Remember to attach all relevant *input* and *output* files so that the problem can be reproduced (it can helpful to compress these into a single file). Without these, it may not be possible to determine the cause of the problem.

12. CASAL EXTENSIONS

12.1 Bootstrap simulations with simCASAL

SimCASAL is a small utility program designed to assist in running operating model/simulation model experiments with CASAL. The program replicates a simple shell scripting process for undertaking operating model/simulation model experiments. Here, the idea is that a number of data sets of observations are simulated from an *operating model*. Then, either MPD or MLE fits are evaluated for these observations sets in an *estimation model*. Note that simCASAL can only conduct MPD or MLE estimation. It cannot be used for MCMC based estimation.

In order to run `simCASAL`, you must have the `CASAL` executable file available either within the same directory as `simCASAL` or in your system path — `simCASAL` just calls on `CASAL` to do the actual work.

SimCASAL requires that you specify the operating model (using the usual three parameter input files `population.csl`, `estimation.csl`, and `output.csl`), a free parameter file (i.e., like that as described in Section 3.3), and an estimation model (again with usual three input parameter files). Typically, the `estimation.csl` file of the estimation model would contain no observations — these are generated from the operating model — but it will contain the usual estimator commands, parameters to estimate, and the definitions of bounds and penalties. SimCASAL simply calls CASAL to generate a number of observation data sets from the operating model (with "true" values as defined in the associated free parameter file), and then, one at a time, pastes these onto the `estimation.csl` file from the estimation model. For each iteration, CASAL undertakes an estimation run (i.e., using `casal -e`), and appends the output to a user-defined file. See Section 6.6 for details about simulating observations from within CASAL.

In order for simCASAL to differentiate between the operating model input files and the estimation model input files, you will need to use a file prefix or file suffix on the estimation model CASAL input parameter files. Optionally, you can also use a file prefix or file suffix on the operating model CASAL input parameter files.

`SimCASAL`, like `CASAL`, is controlled by command line arguments which are used to tell it what task you want to do.

The call to `simCASAL` is of the following form:

```
simCASAL [-i filename] [-s number] [-o outfile] [-e estimation
       model prefix] [-E estimation model suffix]
       [-f operating model prefix] [-F operating model suffix]
       [-l license] [-g RNG_seed] [-q quiet mode] [-Q very
       quiet mode] [-k keep failures] [-c continue previous
       run] [-t temporary filename prefix] [-v alternative
       executable name] [-V]
```

The call should include the following arguments:

`-i [infile]`	Define the *input* parameter file containing the free parameters from which to generate the operating model simulated data.
`-s [number]`	The number of sets of *simulated* data sets to generate (also see the `-s` call to `CASAL` in Section 3.1).

`-o [outfile]`	*Output* file containing free parameter values from the estimation model fits to the simulated data (see the `-o` call to `CASAL` in Section 3.1).

And at least one of the `-e`/`-E` command line arguments for the estimation model definition:

`-e [prefix]`	Use a prefix on the names of the estimation model input parameter files.
`-E [suffix]`	Replace the standard `csl` suffix used on the estimation model input parameter filenames with a user defined suffix.

Optional command line arguments for the operating model definition:

`-f [prefix]`	Use a prefix on the names of the operating model input parameter files.
`-F [suffix]`	Replace the standard `csl` suffix used on the operating model input parameter filenames with a user defined suffix.

In addition, you can use any of the following arguments:

`-l`	Display the `simCASAL` end user *licence*.
`-g [RNG_seed]`	Seed the random number *generator* with this positive (long) integer. value. If this is not specified, then the default is defined as a number based on the computer clock time.
`-q`	Run *quietly*, and suppress all CASAL messages and warnings, i.e., all CASAL standard error output (see the `-Q` call to `CASAL` in Section 3.1).
`-Q`	Run *quietly* (like `-q`), and also delete generated estimation log file output.
`-k`	If `-Q`, then *keep* (i.e., do not delete) any observation data sets and log files where CASAL reported a non-successful exit status.
`-c`	Attempt to continue a previously interrupted run.
`-t [prefix]`	Use an alternative prefix for temporary estimation files and the log file generated for each estimation model fit. The default is `tempSIM`.
`-v`	Defines an alternative name for the CASAL executable (i.e., `casalv207`).
`-V`	Specifies that CASAL use the finite differences minimiser rather than the automatic differentiation minimiser (see `casal -e` and `casal -E`) to fit the estimation models.

12.2 Programmable extensions

Note: this section only applies if you have access to the casal source code. Source code, and hence CASAL programmable extensions, are available on request from NIWA.

It may happen that you want to make a small change to CASAL, but don't want to have to delve into the source code. This section describes relatively simple ways of making the following common changes:

1. Changing the parameterisation used by the CASAL population section.
2. Applying new priors, or new penalties which depend only on the free parameters.
3. Applying new likelihoods.

These changes are implemented by writing snippets of C++ code, using templates supplied with CASAL (files `user.parameterisation.cpp`, `user.prior_penalty.cpp`, `user.likelihood.cpp` respectively). The source code is set up to make this as straightforward as possible. The CASAL source code is required to add such snippets, so that when you recompile CASAL, the new version will incorporate your changes.

To do this, you will need:

1. Access to the CASAL source code.
2. To be able to compile CASAL.
3. The ability to write a (simple) C++ program, with some basic use of Betadiff and of the Standard Template Library (STL) container classes.

If you use these CASAL extensions, remember that for someone else to replicate your results, they need your C++ code as well as your CASAL data files. Also, if you find yourself using a particular CASAL extension frequently, we recommend that you have it implemented and documented in CASAL (or at least document it separately).

12.2.1 New parameterisations (via `user.parameterisation.cpp`)

This section explains how to write your `user.parameterisation.cpp` to implement a different parameterisation in CASAL.

Why use a different parameterisation? It may turn out that the population parameters implemented in CASAL are not suited to the objective function minimisation or MCMC algorithms, and lead to poor convergence. For example, when fitting a von Bertalanffy curve to data, the standard parameterisation of k, t_0, and L_{inf} is not ideal, because of the high correlations between the parameters. You can get better results by changing the parameterisation to t_0, L_1, and L_2, where L_1 and L_2 are the sizes at two reference ages (see Smith et al. 2002). Similarly, the 2002 hoki assessment parameterised initial abundance in terms of *log_B0_total* (the log of the sum of the B_0 values for the two stocks) and *B0_prop_stock1* (the ratio of B_0 for the first of the two stocks, to the sum of the B_0 values for the two stocks), on the basis that this may lead to better convergence in the minimiser and MCMC.

To deal with this situation, we need to make a distinction between the original parameterisation used by the population section, and the new parameterisation which you want to use in the estimation section (and hence in the minimisation and MCMC algorithms). You can do this by writing your data files in terms of the new parameterisation and providing a snippet of code which converts the new parameters back to the original parameters. The CASAL estimation section works with the new parameters, but calls your code whenever it invokes the population section, to translate your parameters back to the original parameterisation. In the first example above, we want the estimation section to work with the new parameterisation of t_0, L_1, and L_2, while leaving the population section in terms of the familiar k, t_0, and L_{inf}. We can do this by writing our data files in terms of t_0, L_1, and L_2 and providing a snippet of code which converts these parameters back to k, t_0, and L_{inf}.

Unfortunately, you cannot use this method to reparameterise ogives. In CASAL, ogives are quite complicated objects (once you take ogive shifts and size-based ogives in age-based models into account): they are too complex to manipulate with this simple technique.

So, your `population.csl` file should include your new parameters, using the standard CASAL parameter syntax. For example,

```
@initialization
log_B0 11
```

The `@estimate` blocks in the `estimation.csl` file should also be in terms of the new parameters,

```
@estimate
parameter initialization.log_B0
lower_bound 8
upper_bound 16
prior uniform
```

You then need to write a `user.parameterisation.cpp` file which converts the new parameters to the old parameters. So, in the example above, convert `initialization.log_B0` to `initialization.B0`. Write the program using the template provided.

1. The inputs are the names and values of the free parameters (under the new parameterisation). Scalar and vector parameters are supplied separately: ogive parameters are not supplied. Bear in mind that parameters which are not free, i.e., not currently being estimated, are not supplied.

2. The outputs are the names and values of the free population parameters (under the old parameterisation). You need to return scalars and vectors separately: you cannot return ogives. Be careful to check the type of each parameter (a 1-vector is different from a scalar).

3. Everything in between is up to you.

More detail is provided in the template file `user.parameterisation.cpp`.

Finally, you need to set `@user_parameterisation T` in your `estimation.csl file`. This lets CASAL know that it should call your `user_parameterisation()` function.

Here is a version of the code to do the log(B_0)−B_0 conversion for a single-stock model (see `user.parameterisation.example.cpp`).

```
########################################################
template<CDVM>
void user_parameterisation(
      std::vector<std::string>& new_scalar_names,
      std::vector<DOUBLE>& new_scalar_vals,
      std::vector<std::string>& new_vector_names,
      std::vector<VECTOR>& new_vector_vals,
      std::vector<std::string>& old_scalar_names,
      std::vector<DOUBLE>& old_scalar_vals,
      std::vector<std::string>& old_vector_names,
      std::vector<VECTOR>& old_vector_vals){
/*
 *** TEMPLATE - BRIAN BULL, 23/5/02 ***

Use this function to implement a new parameterisation in the CASAL
population section.

The function should convert the values of the 'new' parameters (i.e.,
those in your *.csl files) back to the 'old' parameters (i.e., those
in the CASAL manual).
```

```
The inputs to this function are the names and values of the new
scalar and vector parameters, the outputs are the names and values of
the old scalar and vector parameters.

You cannot reparameterise ogives.

You need to use the std::vector and std::string classes(these are in
the STL, see any good C++ book e.g., Stroustrup (2000) The C++
Programming Language, 3rd edition) and DOUBLE and VECTOR, which are
templates either for double and dvector or for dvariable and dvv
(apart from double, these are Betadiff classes: see betadiff.h)

Note that the indices of these VECTORs should always start at 1 (in
this particular function) but the indices of the std::vectors always
start at 0. So, the name of the first new vector parameter is
new_vector_names[0], and its values are new_vector_vals[0][1],
new_vector_vals[0][2]. The output arguments are passed as vectors of
length 0: you need to grow them and fill them in.
*/

/*
*** Brian Bull, 24/5/02. Converts log(B0) to B0 ***

Converts initialization.log_B0 to initialization.B0.

Need to insert the following in estimation.csl:
@user_parameterisation T
*/

DEBUG1("user_parameterisation");

if(!in<std::string>(new_scalar_names,"initialization.log_B0")
      {fatal("You need to estimate initialization.log_B0");}

DOUBLE log_B0 = new_scalar_vals[pos<std::string>
            (new_scalar_names,"initialization.log_B0")];
old_scalar_names.push_back("initialization.B0");
old_scalar_vals.push_back(exp(log_B0));
}
########################################################
```

12.2.2 New priors and penalties (via `user.prior_penalty.cpp`)

This section explains how to write your `user.prior_penalty.cpp` to add new priors, and new penalty functions on the free parameters, to CASAL.

Why would you want to add new priors and penalties on the free parameters? There are several possible reasons:

- You want to use some kind of penalty which is not allowed for in CASAL. For example, you want to use a penalty to encourage the value of q for one set of observations to be twice the value of q for another set of observations.
- You want to use some kind of prior which is not allowed for in CASAL. For example, you want a Student's t prior on M.
- You are using a user-defined parameterisation as per Section 12.2.1, and you want to apply a prior or penalty to the parameters using the old parameterisation. For example, you have reparameterised from B_0 to $\log(B_0)$, but you still want to apply the prior to B_0.

The first two situations could be dealt with by changing the CASAL code, but you may not want to do this, for various reasons, because for example, (i) it would be difficult, (ii) because you think your changes would not be useful to other users, or (iii) because you just want to do a quick fix rather than a proper documented change. (The third would, in general, be genuinely hard to do by changing the main CASAL code.)

To add these objective function components, you need to do two things. First, add the `@user_components` command to your `estimation.csl`: the arguments are the text labels of the priors and penalties that you are going to supply. Second, write a `user.prior_penalty.cpp` which takes the free parameter values and calculates and returns the components. Write the program using the template provided.

- The inputs are:
    (a) the names and values of the free parameters (i.e., those listed in your `estimation.csl`). Scalar, vector, and ogive parameters are supplied separately. (For ogives, only the estimable parameters are supplied, not the non-estimable parameters, like *L* and *H* in an `allvalues_bounded` ogive, and not the ogive values.)
    (b) the names and values of the scalar and vector parameters calculated by `user.parameterisation.cpp` (if any).

- The outputs are the values of the new objective function components, and their labels, which should match those given in the `@user_components` command.

- Everything in between is up to you.

More detail is provided in the template file `user.prior_penalty.cpp`.

Here is a version of the code to implement the three examples above (see `user.prior_penalty.example.cpp`):

```
########################################################
template<CDVM>
void user_prior_penalty(
      std::vector<std::string>& free_scalar_names,
      std::vector<DOUBLE>& free_scalar_vals,
      std::vector<std::string>& free_vector_names,
      std::vector<VECTOR>& free_vector_vals,
      std::vector<std::string>& free_ogive_names,
      std::vector<VECTOR>& free_ogive_arguments,
      std::vector<std::string>& user_scalar_names,
      std::vector<DOUBLE>& user_scalar_vals,
      std::vector<std::string>& user_vector_names,
      std::vector<VECTOR>& user_vector_vals,
      std::vector<std::string>& objective_component_names,
      std::vector<DOUBLE>& objective_component_vals){
/*
*** TEMPLATE - BRIAN BULL, 23/5/02 ***

Use this function to implement new priors, and new penalties on the
free parameters.

This function takes the names and values of the free parameters, and
returns user-defined objective components (i.e., priors and
penalties).
```

```
The inputs to this function are (a) the names and values of the free
parameters (i.e., those in @estimate blocks in your estimation.csl).
If there are free ogive parameters, only their estimable arguments
are supplied - like a1, sL and sR for a double-normal ogive - not the
actual values of the ogives, and (b) the names and values of the
user-defined parameters calculated by user.parameterisation.cpp (if
any) The outputs are the names and values of the user-defined
objective components.

You need to use the std::vector and std::string classes(these are in
the STL, see any good C++ book) and DOUBLE and VECTOR, which are
templates either for double and dvector or for dvariable and dvv
(apart from double, these are Betadiff classes: see betadiff.h)

Note that the indices of these VECTORs should always start at 1 (in
this particular function) but the indices of the std::vectors always
start at 0. So, the name of the first free vector parameter is
free_vector_names[0], and its values are free_vector_vals[0][1],
free_vector_vals[0][2] …

The output arguments are passed as vectors of length 0: you need to
grow them and fill them in.
*/

/*
 *** Brian Bull, 24/5/02. ***

(1) Applies a penalty to encourage q[AEX].q to be approximately 2 x
q[TAN].q.
(2) Applies a t prior to natural_mortality.all
(3) Applies a Cauchy prior to initialization.B0 (which has been
reparameterised as initialization.log_B0).

Notes:
- Both the above q's must be ordinary free parameters (not nuisance
parameters)
- natural_mortality.all must be free
- user.parameterisation.cpp converts initialization.log_B0 to
initialization.B0
- Need to insert the following in estimation.csl:
- @user_components AEX_TAN_penalty prior_on_M prior_on_B0
*/

DEBUG1("user_prior_penalty");

if (!in<std::string>(free_scalar_names,"q[AEX].q")){
      fatal("You need to estimate q[AEX].q");}
if (!in<std::string>(free_scalar_names,"q[TAN].q")){
      fatal("You need to estimate q[TAN].q");}
DOUBLE qAEX = free_scalar_vals[pos<std::string>
       (free_scalar_names,"q[AEX].q")];
DOUBLE qTAN = free_scalar_vals[pos<std::string>
       (free_scalar_names,"q[TAN].q")];
objective_component_vals.push_back(5 * pow(qAEX-2*qTAN, 2));
objective_component_names.push_back("AEX_TAN_penalty");

if (!in<std::string>(free_scalar_names,
 "natural_mortality.all")){
      fatal("You need to estimate natural_mortality.all");}
DOUBLE M = free_scalar_vals[pos<std::string>
      (free_scalar_names,"natural_mortality.all")];
```

```
objective_component_vals.push_back(2.5 * log(1+0.25*
      pow((M-0.25)/0.05,2)));
objective_component_names.push_back("prior_on_M");

if (!in<std::string>(user_scalar_names,"initialization.B0")){
      fatal("You need to estimate initialization.B0 through
      reparameterisation from initialization.log_B0");}
DOUBLE B0 = user_scalar_vals[pos<std::string>
      (user_scalar_names,"initialization.B0")];
objective_component_vals.push_back(log(1+B0*B0));
objective_component_names.push_back("prior_on_B0");
}
#########################################################
```

12.2.3 New likelihoods (via `user.likelihood.cpp`)

This section explains how to write your `user.likelihood.cpp` to add new likelihood functions to CASAL.

Why add new likelihoods? You may want to use some kind of likelihood which is not allowed for in CASAL. For example, you might want a Student's *t* likelihood for an abundance series. This could be done by changing the CASAL code, but you may not want to do this, for various reasons (because it would be difficult, because you think your change would not be useful to other users, because you just want to do a quick fix rather than a proper documented change).

Note that you cannot use nuisance *q*'s for relative time series with user-defined likelihoods. Also, process error is ignored for user-defined likelihoods.

CASAL will not be able to calculate Pearson or normalised residuals for time series for which the likelihood is user-defined (they will be reported as zeros).

To add likelihoods, you need to do two things. First, put `user-supplied` as the argument to the `dist` subcommand in each observations block in `estimation.csl` for which you want to add a new likelihood. This means that CASAL will expect your `user.likelihood.cpp` file to provide a likelihood for that set of observations. Second, write a `user.likelihood.cpp` file which calculates and returns the components. Use the template provided.

- The inputs are the observations and fits (each as a matrix) and the text label of the time series for which the likelihood is to be calculated.
- The output is a negative-log-likelihood for the time series.
- Everything in between is up to you.

More detail is provided in the template file `user.likelihood.cpp`. In particular, the format of the observations and fits matrices is described.

Here is a version of the code to calculate a Student's *t* likelihood for a relative abundance series (see `user.likelihood.example.cpp`).

```
#########################################################
template<CDVM>
DOUBLE user_likelihood(std:: string& label, const MATRIX& fits, const
MATRIX& observations){
```

```
/*
 *** TEMPLATE - BRIAN BULL, 23/5/02 ***

Use this function to implement new likelihood functions.

This function takes the observations and fits, and returns the
negative-log-likelihoods.

You need to set the 'dist' argument for each set of observations for
which you are supplying a likelihood to 'user_supplied'.

You cannot use nuisance q's for relative time series with user-
defined likelihoods (use free q's instead).

The inputs to this function are the text label of a time series and
the observations and fits.

The obs and fits are stored as matrices. There is one row per year.
For abundance data, there is one column. For all other data, there is
one column per age or size class, per sex if they are sexed
observations, with males before females. This is exactly the same as
the format used by CASAL to print the data. The output is the
corresponding negative-log-likelihood. If you want to use user-
defined likelihoods for more than one time series, then you may need
to put in an if-statement based on the 'label' argument, which tells
you which time series is currently being dealt with.

You need to use the DOUBLE, VECTOR, and MATRIX classes which are
templates either for double, dvector and dmatrix or for dvariable,
dvv, and dvm (apart from double, these are Betadiff classes: see
betadiff.h) Note that the indices of these VECTORs and MATRIXs should
always start at 1 (in this particular function)
*/

/*
 *** Brian Bull, 24/5/02. Calculates a Student's T likelihood for a
relative abundance index. ***

Notes: We use a t-likelihood with 4 degrees of freedom and a scale
parameter of 20% of the mean (which means the c.v. is over 20%). The
q for this abundance index must be an ordinary free parameter, not a
nuisance parameter.
*/

DEBUG1("user_likelihood");

if (label! = "acoustics"){
      fatal("user.likelihood.cpp doesn't know how to calculate
      a likelihood for " + label);}

DOUBLE result = 0;
for (int i = 1; i< = observations.rowmax(); i++){
      result += log(fits[i][1])
      + 2.5*log(1+0.25*pow((observations[i][1]-
      fits[i][1])/(0.2*fits[i][1]),2));
 }
 return result;
}
########################################################
```

13. POST-PROCESSING OF CASAL OUTPUT

The R package `casal` contains a set of R functions for reading CASAL output, and is available as a precompiled binary for Microsoft Windows (.zip file) or as a source package (.gz file) for Linux. To check the version number and date of the CASAL R package (useful for checking that you have the most up-to-date version), use the function `casal.version()`.

The CASAL R package includes a range of extract and write functions to aid post-processing of CASAL output. In addition, the package also has a number of undocumented helper functions. The helper functions are usually labelled with the prefix `casal`. These may be of assistance if you need to write your own functions in R, but otherwise they can be ignored.

Most of the main functions are briefly described below, but for more detail, see the CASAL package help page in R (e.g., type `help(casal)` in R after loading the package). Specific help for the R functions is also available.

The key extract functions are;

`extract.header`
: Extracts the header file information from a CASAL standard output file.

`extract.objective.function`
: Extracts the objective function information from a CASAL standard output file.

`extract.free.parameters`
: Extracts free parameters information from a CASAL standard output file.

`extract.fits`
: Extracts the observation and associated fit & residual information from a CASAL standard output file.

`extract.quantities`
: Extracts the estimated quantities from a CASAL standard output file

`extract.parameters.at.bounds`
: Extracts values of free parameters at or near the lower and upper bounds from a CASAL standard output file.

These functions allow the importation of standard output from `casal -r`, `-e` or `-E` into S-Plus for post-processing. When running CASAL, redirect the standard output into a text file and pass the name of the text file to the R functions. These functions are also grouped together into a single function

`extract.mpd`
: Extracts header, objective function, free parameters, fits, quantities, and parameters at bounds (see above) as a list object with elements as for the above extract functions.

: `extract.mpd` also sets the class of the resulting object as `casalMPD`. Specific methods that summarise (e.g., `summary(object)`), print, or plot can be written for objects of class `casalMPD`.

There are two functions that can be used to extract data from table style output, for example, from the output from an MCMC samples/quantities or projection run. These are;

`extract.free.parameters.from.table`

> Extract free parameters from files in the flat format described in Section 3.3 (this is the format used to pass free parameters to CASAL with -i, and is also the output format for samples from a posterior using -m , -M, or -C).

`extract.quantities.from.table`

> Extract output quantities from files in tabular format (this is the output format of `casal -v` and `-P`).

These are also combined into a single function, `extract.mcmc`. This function evaluates if the input is a table of free parameters or quantities in a table, and applies the appropriate extract method. In addition, if an objectives file is also supplied, then the objective log likelihood values are appended to the table.

`extract.mcmc`

> Extracts the output samples from a CASAL MCMC output file or quantities (`casal -v`) into a dataframe object
>
> `extract.mcmc` sets the class of the resulting object as `casalMCMC`. As for objects of class `casalMPD` specific methods that summarise, print, or plot can be written.

Two functions are available that allow you to parse the CASAL input parameter files (i.e., the '*.csl' files);

`extract.csl.file`

> Reads in a '*.csl' file, and sets the resulting object to class `casalCSLfile`.

`write.csl.file`

> Writes a `casalCSLfile` object out as a '*.csl' file.

And a function that can allow you to extract the names of the free parameters and their bounds from an estimation.csl file;

`extract.parameter.bounds`

> Extracts the free parameters and their bounds from a estimation.csl file.

Occasionally, for example from a profile run, you may wish to extract the objective functions from a single output file that contains the details from many consecutive runs. For this purpose, use;

`extract.multiple.objective.functions`

> Read through standard CASAL output file and return all of the objective function summary tables as a matrix (with individual objective function values for each observation, prior, and penalty as rows and the columns representing sequential runs).

Other functions available include `extract.MSY`, `extract.MCYCAY`, `extract.covariance.matrix`, `generate.MVN`, and `generate.MVU`. See the CASAL R package help for details of these and the other available functions.

A graphical representation of the priors used by CASAL can be generated by the R function `plot.priors`. See the CASAL R package help for details.

14. EXAMPLES

14.1 A Bayesian two-sex age-based model

14.1.1 Introduction

The example provided here is a simple Bayesian two-sex age-based model, loosely based on the Chatham Rise hake fishery in New Zealand. (Note that we do not suggest that this is a good model, but rather it is provided as an example of how CASAL may be used.) The biomass observations consist of a CPUE series and a trawl survey biomass series. In addition, the trawl survey series has a series of associated proportions-at-age data. Proportions-at-age data are also available from the commercial catch.

The partition is defined to allow for age-classes in the model from 2–25. Note that there is no plus group, so the model assumes that there are no fish older than 25 in the population. The partition is defined for males and females, but does not include maturity.

The annual cycle is described in the `R1-population.csl` file, and consists of a year round fishery (time step 1), followed by an instantaneous spawning period (time step 2), and then an age increment (time step 3).

An example of the partition at time step 1 (i.e., before the arrival of new recruits) as printed by CASAL might be,

```
Partition:
Sex          2      3      4       5 …      23      24      25
Male         0  915.9  676.2  246.53 …  0.5768  0.4575  0.3662
Female       0  915.9  700.7  250.03 …   1.981   1.618   1.331
```

The remainder of the `population.csl` file defines the recruitment processes, maturation, natural mortality, fishing mortality (and catches), selectivities to be applied to the fishing and trawl surveys, size-at-age, and size-weight relationships. Note that your results will look different, depending model state, as well as the computer and platform used to run CASAL.

14.1.2 The input parameter files

The input parameter files (named with a prefix of `R1-` to identify them as the first 'run') are;

R1-population.csl

```
#INITALISATION (the starting value for B0 is set as 45000 t)
@initialization
B0 45000

# PARTITION
@size_based false   # Define the model as age-based
@min_age 2
@max_age 25         # The partition keeps account of fish aged 2-25
@plus_group false   # and excludes all fish over the age of 25
@sex_partition true # The model is sex-based
@mature_partition false # Maturity is excluded from the partition
@n_areas 1          # Only a single fishing area is defined
@area_names chat    # with the (optional in a single area model) label
   "chat"
@n_stocks 1         # This is a single stock model

# TIME SEQUENCE
@initial 1975       # The model is defined to run from 1975
```

```
@current 2002       # to the current year, 2002
@final 2007         # Projections are run up to the year 2007

@annual_cycle
time_steps 3        # There are three time steps: Oct-Aug, Sep, Sep
recruitment_time 2  # Recruitment occurs in time step 2
recruitment_areas chat # in the area "chat" (the only area defined)
spawning_time 2     # Spawning occurs in time step 2
spawning_part_mort 0.5 # and SSBs are calculated after spawning fish have
   undergone 0.5 of the mortality assigned to this time step
spawning_areas chat # Spawning occurs in the area "chat"
spawning_ps 1       # and all mature fish spawn
aging_time 3        # Age incrementation occurs in time step 3
growth_props 1.00 1.00 0.00 # All fish growth occurs in time step 1
M_props 1.00 0.00 0.00 # All natural mortality occurs in time step 1
baranov false       # The baranov equation is not used
midmortality_partition weighted_sum
fishery_names chatFishery # The fishery has the label "chatFishery
fishery_times 1     # and occurs in the first time step
fishery_areas chat  # in the area labelled "chat"
n_migrations 0      # No migrations are defined

# RECRUITMENT
@y_enter 2          # Recruits enter 2 years after birth
@standardise_YCS true  # Use the "Haist" parameterisation of YCS
@recruitment        # the two following lines define the starting values for
   recruitment for the years 1973-2000
YCS_years 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985
   1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999
   2000
YCS       1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.04 1.21 0.85 1.38 1.03 1.25
   0.89 1.47 1.40 1.84 2.66 2.28 2.17 1.70 1.59 0.71 1.00 1.00 1.00 1.00
   1.00
first_free 1979     # with standardisation occurring over the years
last_free 1998      # 1979-1998
p_male 0.5          # 50% of ‘recruits’ are males
sigma_r 0.6         # Standard deviation of YCS for projections
SR BH               # Use the Beverton-Holt stock-recruit relationship
steepness 0.9       # with a steepness parameter of 0.9

# RECRUITMENT VARIABILITY
@randomisation_method lognormal # Use the lognormal distribution when
   assigning YCS to unknown years during projections
@first_random_year 1999 # Defines the first unknown YCS as 1999

#MATURATION
@maturity_props  # Define the maturity ogive for males and females
male allvalues_bounded   2 10 0.00 0.00 0.02 0.07 0.31 0.78 1.00 1.00 1.00
female allvalues_bounded 2 10 0.00 0.00 0.02 0.04 0.07 0.45 0.86 1.00 1.00

# NATURAL MORTALITY
@natural_mortality
avg 0.20  # Define the average natural mortality of males & females as 0.20
diff 0.02 # and define the difference (male-female) as 0.02, i.e.,
   male_M=0.21 and female_M=0.19

# FISHING
@fishery chatFishery # Define the catch from the chatFishery for years 1975-
   2002
years  1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988
   1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
catches 191  488 1288   34  609  750  997  596  302  344  544  362  509  574
   804  977  991 2454 2775 2898 4094 3760 3761 3673 3524 3700 3700 3700
selectivity chatFsel # Defines that the catch is removed from the population
   using the selectivity defined by the label "chatFsel"
U_max 0.4            # with a maximum possible exploitation rate of 0.4
future_years   2003 2004 2005 2006 2007 # Defines the future years and
future_catches 3700 3700 3700 3700 3700 # catches for use in projections
```

```
# SELECTIVITIES
@selectivity_names chatTANsel chatFsel # Define the two selectivities used
@selectivity chatFsel                  # Fishing selectivity is a logistic
male logistic 9 4                      # curve for males and females, with
female logistic_capped 9 4 0.7         # the female rates relative to males
@selectivity chatTANsel                # And similarly for trawl survey
male logistic 9 4                      # selectivity
female logistic_capped 9 4 0.7

# SIZE AT AGE
@size_at_age_type von_Bert  # Defines that the age-length relationship is
@size_at_age_dist normal    # von-Bertalanffy (defined separately for males
@size_at_age                # and females), with a distribution defined
k_male 0.277                # as normal with c.v.=0.1
t0_male -0.11
Linf_male 90.3
cv_male 0.1
k_female 0.202
t0_female -0.20
Linf_female 113.4
cv_female 0.1

# SIZE-WEIGHT
@size_weight    # Defines the length-weight relationship
a_male 2.49e-9
b_male 3.234
a_female 1.70e-9
b_female 3.328
verify_size_weight 50 0.5 1.5 # Check that these values are correct, by
   confirming that a 50 cm fish has a weight between 0.5 and 1.5 kgs
```

R1-estimation.csl

```
# ESTIMATION
@estimator Bayes # Use the Bayes estimation method
@max_iters 300   # With maximum of 300 iterations for the point estimates
@max_evals 1000  # and 1000 function evaluations
@grad_tol 0.002  # Set the tolerance for the convergence test at 0.002

@MCMC
start 0          # Start the MCMC at 0
length 110000   # and evaluate for 110000 steps
keep 100        # keeping every 100th sample
stepsize 0.02    # with the stepsize for the MCMC set at 0.02
adaptive_stepsize true # but adapt the stepsize during the evaluation
adapt_at 5000  # after the 5000th step
burn_in 100      # The MCMC has a burn-in period of 100*100=10000 steps

# OBSERVATIONS Chatham Rise
@relative_abundance chatCPUE # Define a relative abundance series "chatCPUE"
biomass true    # This time series is an abundance index
q chatCPUEq     # and has a catchability coefficient called "chatCPUEq"
years 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 # index years
step 1          # Occurs in time step 1
proportion_mortality 0.5 # after 0.5 of mortality has been recorded in that
   time step
area chat       # Occurs in the area called "chat"
ogive chatFsel  # and is applied with the selectivity "chatFsel"
1992 1.50       # The values of the index are ...
1993 1.10
1994 0.93
1995 1.33
1996 1.53
1997 0.90
1998 0.68
1999 0.75
2000 0.57
```

```
2001 1.23
cv_1992 0.35    # and each point has the c.v.s ...
cv_1993 0.35
cv_1994 0.35
cv_1995 0.35
cv_1996 0.35
cv_1997 0.35
cv_1998 0.35
cv_1999 0.35
cv_2000 0.35
cv_2001 0.35
dist lognormal       # where the c.v.s have lognormal distribution
cv_process_error 0.0 # and there is no process error applied

@relative_abundance chatTANbiomass # Define a relative abundance series
   "chatTANbiomass"
biomass true    # This time series is an abundance index
q chatTANq      # and has a catchability coefficient called "chatTANq"
years 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 # index years
step 1          # Occurs in time step 1
proportion_mortality 1 # after all mortality has been recorded in that time
   step
area chat       # Occurs in the area "chat"
ogive chatTANsel # and is applied with the selectivity "chatTANsel"
1992 4180       # the values of the index are ...
1993 2950
1994 3353
1995 3303
1996 2457
1997 2811
1998 2873
1999 2302
2000 2090
2001 1589
2002 1567
cv_1992 0.15    # and each point has the c.v.s ...
cv_1993 0.17
cv_1994 0.10
cv_1995 0.23
cv_1996 0.13
cv_1997 0.17
cv_1998 0.18
cv_1999 0.12
cv_2000 0.09
cv_2001 0.13
cv_2002 0.15
dist lognormal       # where the c.v.s have lognormal distribution
cv_process_error 0.0 # and there is no process error applied

@proportions_at chatTANage # Define a series of relative proportions-at-age
years 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 # index years
step 1           # Occurs in time step 1
proportion_mortality 1 # after all mortality has been recorded in that time
   step
area chat        # in area "chat"
sexed true       # The observations are recorded by sex
plus_group true  # The oldest age group is a ‘plus-group’
sum_to_one true  # and the proportions sum to one over each year
min_class 3 3    # The minimum age class is 3 for males and 3 for females
max_class 15 15  # The maximum/plus-group age class is 15 for males and 15
   for females
ogive chatTANsel # And is applied with the selectivity "chatTANsel"
# The proportions-at-age in each column are ...
#         M3   M4      M5     M6     M7     M8     M9    M10    M11    M12
   M13    M14   M15+     F3     F4     F5     F6     F7     F8     F9    F10
   F11    F12    F13    F14   F15+
```

```
1992 0.0186 0.0219 0.0249 0.0390 0.0512 0.0646 0.0422 0.0677 0.0523 0.0687
    0.0299 0.0132 0.0515 0.0055 0.0254 0.0199 0.0320 0.0268 0.0394 0.0250
    0.0536 0.0346 0.0423 0.0489 0.0304 0.0705
1993 0.0449 0.0346 0.0372 0.0112 0.0286 0.0220 0.0279 0.0156 0.0310 0.0300
    0.0690 0.0283 0.0614 0.0552 0.0231 0.0457 0.0301 0.0296 0.0426 0.0122
    0.0415 0.0277 0.0541 0.0697 0.0127 0.1142
1994 0.1063 0.0405 0.0431 0.0274 0.0179 0.0182 0.0170 0.0198 0.0317 0.0252
    0.0127 0.0195 0.0969 0.1176 0.0483 0.0421 0.0513 0.0271 0.0350 0.0095
    0.0154 0.0060 0.0256 0.0083 0.0303 0.1072
1995 0.0933 0.0730 0.0365 0.0123 0.0379 0.0321 0.0195 0.0149 0.0158 0.0314
    0.0322 0.0544 0.0329 0.0788 0.0609 0.0501 0.0368 0.0382 0.0288 0.0034
    0.0476 0.0218 0.0231 0.0197 0.0390 0.0659
1996 0.0528 0.0505 0.0904 0.0485 0.0226 0.0323 0.0064 0.0149 0.0037 0.0039
    0.0046 0.0111 0.0265 0.0689 0.1471 0.1374 0.0786 0.0450 0.0260 0.0135
    0.0282 0.0032 0.0168 0.0286 0.0133 0.0251
1997 0.0942 0.0797 0.0590 0.0498 0.0318 0.0528 0.0045 0.0187 0.0091 0.0152
    0.0093 0.0230 0.0344 0.1124 0.0682 0.0927 0.0814 0.0225 0.0233 0.0180
    0.0073 0.0026 0.0027 0.0297 0.0121 0.0457
1998 0.0397 0.0678 0.0862 0.0457 0.0676 0.0354 0.0201 0.0225 0.0092 0.0176
    0.0066 0.0260 0.0422 0.0195 0.0606 0.0660 0.0831 0.0711 0.0527 0.0291
    0.0170 0.0362 0.0225 0.0095 0.0049 0.0411
1999 0.0683 0.0771 0.0408 0.0364 0.0228 0.0380 0.0148 0.0226 0.0138 0.0109
    0.0045 0.0050 0.0585 0.0628 0.0307 0.0711 0.0411 0.0372 0.0740 0.0521
    0.0465 0.0232 0.0270 0.0180 0.0152 0.0876
2000 0.0623 0.0466 0.0521 0.0292 0.0369 0.0524 0.0508 0.0414 0.0385 0.0138
    0.0120 0.0227 0.0234 0.0131 0.0358 0.0336 0.0433 0.0445 0.0699 0.0413
    0.0265 0.0298 0.0368 0.0187 0.0370 0.0878
2001 0.0033 0.0274 0.0554 0.0259 0.0455 0.0611 0.0413 0.0404 0.0337 0.0204
    0.0124 0.0034 0.0195 0.0064 0.0314 0.0278 0.0364 0.0983 0.0549 0.0798
    0.0681 0.0728 0.0488 0.0076 0.0210 0.0567
2002 0.0173 0.0193 0.0241 0.0346 0.0365 0.0657 0.0427 0.0667 0.0326 0.0307
    0.0272 0.0141 0.0319 0.0353 0.0249 0.0146 0.0133 0.0547 0.0488 0.0745
    0.0660 0.0750 0.0646 0.0304 0.0147 0.0399
# with c.v.s for each observation defined as ...
cvs_1992 0.710 0.469 0.370 0.350 0.419 0.413 0.373 0.383 0.365 0.301 0.393
    0.518 0.302 0.689 0.326 0.394 0.313 0.305 0.322 0.336 0.301 0.308 0.289
    0.276 0.360 0.228
cvs_1993 0.321 0.336 0.361 0.861 0.421 0.532 0.408 0.555 0.409 0.489 0.361
    0.527 0.369 0.346 0.451 0.370 0.428 0.437 0.376 0.780 0.336 0.494 0.366
    0.342 0.628 0.307
cvs_1994 0.268 0.379 0.315 0.492 0.555 0.507 0.587 0.546 0.468 0.510 0.818
    0.547 0.268 0.238 0.290 0.315 0.278 0.405 0.382 0.782 0.640 0.838 0.457
    0.742 0.374 0.188
cvs_1995 0.250 0.367 0.372 0.621 0.655 0.810 0.924 0.772 0.865 0.713 0.632
    0.608 0.612 0.254 0.329 0.340 0.400 0.379 0.587 1.028 0.432 0.542 0.499
    0.506 0.475 0.286
cvs_1996 0.497 0.445 0.353 0.409 0.729 0.560 1.161 0.929 1.481 1.128 1.335
    1.228 0.687 0.392 0.227 0.256 0.317 0.381 0.566 0.716 0.542 3.000 0.881
    0.568 0.930 0.558
cvs_1997 0.298 0.315 0.430 0.441 0.612 0.521 1.504 0.736 0.994 0.867 1.033
    0.668 0.607 0.289 0.313 0.279 0.301 0.631 0.518 0.695 1.059 3.000 1.751
    0.556 1.051 0.458
cvs_1998 0.464 0.403 0.343 0.440 0.423 0.575 0.710 0.703 1.227 0.774 1.503
    0.691 0.445 0.533 0.342 0.341 0.324 0.358 0.357 0.548 0.689 0.511 0.689
    1.097 1.381 0.493
cvs_1999 0.572 0.417 0.454 0.510 0.577 0.521 0.834 0.782 0.808 1.004 1.373
    1.170 0.457 0.458 0.550 0.314 0.440 0.448 0.324 0.335 0.450 0.690 0.588
    0.813 0.749 0.426
cvs_2000 0.414 0.522 0.328 0.399 0.351 0.319 0.320 0.328 0.374 0.631 0.613
    0.589 0.419 1.334 0.569 0.452 0.403 0.371 0.294 0.343 0.427 0.488 0.389
    0.593 0.416 0.229
cvs_2001 1.726 0.527 0.446 0.510 0.510 0.392 0.462 0.442 0.551 0.643 0.761
    1.439 0.637 1.180 0.434 0.552 0.445 0.301 0.429 0.352 0.353 0.368 0.441
    0.767 0.610 0.362
cvs_2002 1.091 0.770 0.539 0.421 0.412 0.297 0.367 0.322 0.391 0.510 0.523
    0.734 0.481 0.612 0.643 0.756 0.772 0.399 0.369 0.331 0.306 0.304 0.309
    0.461 0.752 0.423
dist lognormal       # where the c.v.s have lognormal distribution
```

```
cv_process_error 0.0 # and there is no additional process error applied
ageing_error true    # apply an ageing error model to the observations

@catch_at chatOBS    # Define a series of proportions-at-age from the catch
years 1998 1999 2000 2001  # for the years 1998-2001
fishery chatFishery  # that occur in the fishery "chatFishery"
sexed true       # The observations are recorded by sex
plus_group true  # The oldest age group is a ‘plus-group’
sum_to_one true  # and the proportions sum to one over each year
min_class 3 3    # The minimum age class is 3 for males and 3 for females
max_class 15 15  # The maximim/plus-group age class is 15 for males and 15
   for females
# The proportions-at-age in each column are ...
#         M3    M4     M5     M6     M7     M8     M9    M10    M11    M12
   M13    M14   M15+     F3     F4     F5     F6     F7     F8     F9    F10
   F11    F12    F13    F14   F15+
1998 0.1079 0.0696 0.0580 0.0607 0.0865 0.0706 0.0288 0.0247 0.0062 0.0077
   0.0076 0.0070 0.0115 0.0905 0.0581 0.0608 0.0373 0.0427 0.0548 0.0241
   0.0247 0.0245 0.0105 0.0063 0.0036 0.0152
1999 0.0264 0.0641 0.0445 0.0714 0.0413 0.0516 0.0329 0.0271 0.0270 0.0117
   0.0023 0.0021 0.0209 0.0229 0.0690 0.0485 0.0913 0.0563 0.0537 0.0594
   0.0517 0.0412 0.0133 0.0157 0.0137 0.0401
2000 0.0161 0.0441 0.0605 0.0509 0.0658 0.0590 0.0715 0.0432 0.0291 0.0154
   0.0116 0.0051 0.0180 0.0129 0.0405 0.0315 0.0428 0.0766 0.1011 0.0573
   0.0309 0.0436 0.0248 0.0071 0.0060 0.0346
2001 0.0087 0.0280 0.0422 0.0427 0.0849 0.0887 0.0788 0.0711 0.0566 0.0275
   0.0162 0.0166 0.0507 0.0019 0.0383 0.0246 0.0332 0.0786 0.0594 0.0345
   0.0295 0.0240 0.0219 0.0120 0.0062 0.0233
# with c.v.s for each observation defined as ...
cvs_1998 0.175 0.232 0.227 0.207 0.173 0.172 0.278 0.299 0.515 0.487 0.440
   0.494 0.307 0.183 0.209 0.228 0.269 0.234 0.184 0.296 0.282 0.295 0.459
   0.402 0.746 0.275
cvs_1999 0.328 0.221 0.313 0.250 0.269 0.283 0.270 0.361 0.380 0.481 1.162
   1.189 0.409 0.332 0.217 0.329 0.196 0.257 0.235 0.222 0.244 0.250 0.421
   0.583 0.512 0.259
cvs_2000 0.495 0.264 0.237 0.233 0.196 0.219 0.203 0.238 0.349 0.426 0.458
   0.714 0.343 0.424 0.299 0.404 0.252 0.188 0.146 0.208 0.275 0.217 0.294
   0.424 0.514 0.184
cvs_2001 0.383 0.332 0.311 0.300 0.192 0.200 0.227 0.245 0.268 0.372 0.551
   0.602 0.257 1.445 0.257 0.333 0.305 0.191 0.195 0.211 0.255 0.265 0.295
   0.356 0.481 0.239
dist lognormal       # where the c.v.s have lognormal distribution
cv_process_error 0.2 # and there is c.v.=0.2 process error applied
ageing_error true    # Apply an ageing error model to the observations

# CATCHABILITY COEFFICIENTS
@q_method nuisance   # Use the "nuisance" method for estimating q

@estimate
parameter q[chatCPUEq].q  # Estimate the parameter q[chatCPUEq].q when
   fitting the model
lower_bound 1e-6          # with a lower bound
upper_bound 10            # and upper bound
prior uniform-log         # and use a uniform-log prior

@estimate
parameter q[chatTANq].q   # Estimate the parameter q[chatTANq].q when
   fitting the model
lower_bound 1e-8
upper_bound 1
prior uniform-log

#FREE PARAMETERS
@estimate
parameter initialization.B0 # Estimate B0
phase 2                     # Use two-phase estimation, and only try to
   "fit" this parameter after fitting all other parameters first
lower_bound 2500            # Define the lower bound
```

```
upper_bound 150000          # Define the upper bound
prior uniform-log           # and use a uniform-log prior

@estimate
parameter recruitment.YCS   # Estimate YCS when fitting the model
#the YCSyears are 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983
   1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997
   1998 1999 2000
lower_bound          1    1    1    1    1    1 0.01 0.01 0.01 0.01 0.01
   0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
   0.01    1    1 # lower bounds
upper_bound          1    1    1    1    1    1  100  100  100  100  100
   100  100  100  100  100  100  100  100  100  100  100  100  100  100  100
   1    1 # upper bounds. Note that some YCS are constrained to be equal to
   one
prior lognormal  # Use a lognormal prior, with parameters
mu                   1    1    1    1    1    1    1    1    1    1    1
   1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
   1    1 # with μ=1, and c.v.=1.1
cv                 1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1
   1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1  1.1
   1.1  1.1

@estimate
parameter selectivity[chatTANsel].male # Estimate the "chatTANsel.male"
   ogive.
lower_bound 0 0     # the two logistic parameters have lower and
upper_bound 30 30   # upper bounds
prior uniform       # and they have uniform priors

@estimate
parameter selectivity[chatTANsel].female # Estimate the "chatTANsel.female"
   ogive.
lower_bound 0 0 0.2 # The three logistic parameters have lower and
upper_bound 30 30 5 # upper bounds
prior uniform       # And they have uniform priors

@estimate
parameter selectivity[chatFsel].male # Estimate the "chatFsel.male" ogive.
lower_bound 0 0     # The two logistic parameters have lower and
upper_bound 30 30   # upper bounds
prior uniform       # And they have uniform priors

@estimate
parameter selectivity[chatFsel].female # Estimate the "chatFsel.female"
   ogive.
lower_bound 0 0 0.2 # The three logistic parameters have lower and
upper_bound 30 30 5 # upper bounds
prior uniform       # And they have uniform priors

{
# This is a comment block, commenting out the request to estimate the two
# parameters natural_mortality.avg and natural_mortality.diff
@estimate
parameter natural_mortality.avg
phase 3
prior lognormal
mu 0.20
cv 0.20
lower_bound 0.10
upper_bound 0.30

@estimate
parameter natural_mortality.diff
phase 3
prior normal-by-stdev
mu 0
stdev 0.05
```

```
lower_bound -0.20
upper_bound 0.20
}

# PENALTIES
@catch_limit_penalty  # This specifies that the model must attempt to have a
   biomass large enough so that the catch is takable from the population
label chatCatchMustBeTaken
fishery chatFishery
log_scale true
multiplier 1000       # The penalty has a high "multiplier"
@ageing_error         # Specify the ageing error model used
type normal           # Ageing error is of type "normal"
c 0.08                # with a c.v.=0.08

@vector_average_penalty
label meanYCS_1
vector recruitment.YCS
k 1
multiplier 6
```

R1-output.csl

```
@print # Specifies the outputs that CASAL should generate
# estimation section
parameters false
fits_every_eval false
objective_every_eval false
parameters_every_eval false
parameter_vector_every_eval false
fits true
resids false
pearson_resids false
normalised_resids true
estimation_section false
covariance True
# population section
requests true
initial_state false
state_annually false
state_every_step false
final_state true
results false
#output section
yields true
unused_parameters true

@quantities
all_free_parameters true
fishing_pressures true
nuisance_qs true
true_YCS true
B0 true
R0 true
SSBs true
YCS true
actual_catches false
ogive_parameters  selectivity[chatTANsel].male  selectivity[chatTANsel].female
   selectivity[chatFsel].male selectivity[chatFsel].female

@B_pre
selectivity none
mature_only true

@MCY_CAY
do_MCY true
MCY_guess 10000
```

```
n_discard 100
n_keep 100
n_simulations 100
do_CAY true
F_CAY_guess 0.2
interactive false

@abundance MyAbundance
biomass True
years 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
ogive chatFsel
step 2
proportion_mortality 0.5

@proportions_at MyNumbersAtAge
years 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
sexed True
ogive chatFsel
step 2
proportion_mortality 0.5
```

14.1.3 CASAL output

The call `casal -e -q –g 0 -f R1- > R1-estimate.log` with the above parameter files generates an output file. The first few lines of the output file, `R1-estimate.log`, are;

```
CASAL (C++ algorithmic stock assessment laboratory)
Call: casal -e -q -g 0 -O MPD.dat -f R1-
Date: Wed Mar 21 13:29:22 2012
v2.30-2012-03-21 00:22:59 UTC (rev.4648)  (c) Copyright 2002-2012, NIWA
User name: adunn
Machine name: thotter (Linux 2.6.16.60-0.93.1-smp x86_64, PID=7251)

Prefix for the input parameter filenames : R1-

Random number seed : 0

A male fish of size 50 cm has a weight of 0.777443 kg in your model. This
     size-weight scale check assumes that the growth curve is in
     centimetres, and that the catch is in tonnes.

A female fish of size 50 cm has a weight of 0.766692 kg in your model. This
     size-weight scale check assumes that the growth curve is in
     centimetres, and that the catch is in tonnes.

A male fish of size 50 cm has a weight of 0.777443 kg in your model. This
     size-weight scale check assumes that the growth curve is in
     centimetres, and that the catch is in tonnes.

A female fish of size 50 cm has a weight of 0.766692 kg in your model. This
     size-weight scale check assumes that the growth curve is in
     centimetres, and that the catch is in tonnes.

In phase 1 Minimiser achieved convergence after 55 quasi-Newton iterations
     using 88 objective function evaluations

In phase 2 Minimiser achieved convergence after 38 quasi-Newton iterations
     using 100 objective function evaluations


The following parameters were estimated at or near a bound:
Parameter        Estimate  lower_bound  upper_bound
recruitment.YCS[26]  0.0520493  0.01  100
```

The first set of lines give information on the CASAL version that was run, the date, and machine, file prefix, random number generator seed, length weight validation output, and the convergence summary from the minimiser. Note that the estimation carried out here is two phase estimation, with the first phase holding B_0 constant.

The remainder of the output file gives parameters estimated near or at a bound, fits, residuals, and other requested output information, as defined in the `R1-output.csl` file. Some of this output is reproduced here.

```
Start extracting output from here

Point estimate:
initialization.B0
current value: 26991.3

recruitment.YCS
current values:
1 1 1 1 1 1 0.190797 0.933788 0.358616 0.436584 0.267153 0.344418 0.268054
  0.322199 0.21143 0.423725 0.342837 0.499052 0.72796 0.575916 0.460651
  0.441897 0.336943 0.195483 0.146892 0.0520493 1 1

selectivity[chatTANsel].male (parameters)
current values:
11.8155 11.2515

selectivity[chatTANsel].female (parameters)
current values:
9.9213 10.853 0.691629

selectivity[chatFsel].male (parameters)
current values:
8.35602 11.7958

selectivity[chatFsel].female (parameters)
current values:
6.34521 10.8179 0.626889


In a format suitable for -i:

initialization.B0 recruitment.YCS 28 selectivity[chatTANsel].male 2
  selectivity[chatTANsel].female 3 selectivity[chatFsel].male 2
  selectivity[chatFsel].female 3
26991.3 1 1 1 1 1 1 0.190797 0.933788 0.358616 0.436584 0.267153 0.344418
  0.268054 0.322199 0.21143 0.423725 0.342837 0.499052 0.72796 0.575916
  0.460651 0.441897 0.336943 0.195483 0.146892 0.0520493 1 1 11.8155 11.2515
  9.9213 10.853 0.691629 8.35602 11.7958 6.34521 10.8179 0.626889

Objective function : -149.628

Components :
                                          chatCPUE -7.02643
                                    chatTANbiomass -18.4818
                                        chatTANage -74.8265
                                           chatOBS -39.6556
                       prior_on_initialization.B0 10.2033
                         prior_on_recruitment.YCS -10.3265
             prior_on_selectivity[chatTANsel].male 0
           prior_on_selectivity[chatTANsel].female 0
               prior_on_selectivity[chatFsel].male 0
             prior_on_selectivity[chatFsel].female 0
                              prior_on_q_chatCPUEq -9.53375
                               prior_on_q_chatTANq -1.16944
                              chatCatchMustBeTaken 0
                                         meanYCS_1 1.18883
…
```

The output then continues with fits (if requested), other output quantities, unused parameters, and other requested quantities.

These include the following lines …

```
Output quantities start here

* Scalar parameter values
initialization.B0 26991.3

* Vector parameter values
recruitment.YCS 1 1 1 1 1 1 0.190797 0.933788 0.358616 0.436584 0.267153
  0.344418 0.268054 0.322199 0.21143 0.423725 0.342837 0.499052 0.72796
  0.575916 0.460651 0.441897 0.336943 0.195483 0.146892 0.0520493 1 1

* Ogive parameter values
selectivity[chatTANsel].male 0.0711831 0.0905479 0.114531 0.143862 0.179184
  0.220941 0.269237 0.323703 0.383407 0.446847 0.512066 0.576877 0.639146
  0.697064 0.749331 0.795229 0.834579 0.867626 0.894902 0.917095 0.934942
  0.94916 0.960403 0.96924
selectivity[chatTANsel].female 0.0722185 0.0917414 0.115557 0.144071
  0.177453 0.215525 0.257672 0.30282 0.349506 0.396058 0.440822 0.482387
  0.51975 0.552368 0.580123 0.603232 0.622126 0.637345 0.649457 0.659005
  0.666475 0.672285 0.676783 0.680252
selectivity[chatFsel].male 0.169866 0.208011 0.252119 0.302015 0.35707
  0.416177 0.477797 0.540101 0.601176 0.659257 0.712919 0.761191 0.803582
  0.84003 0.870802 0.896385 0.917383 0.934437 0.948169 0.959151 0.967885
  0.9748 0.980257 0.984551
selectivity[chatFsel].female 0.147049 0.179852 0.216669 0.256694 0.29873
  0.341302 0.382863 0.422007 0.457647 0.489111 0.516142 0.538824 0.557485
  0.57259 0.584657 0.594195 0.601672 0.607495 0.612007 0.615488 0.618167
  0.620223 0.621798 0.623004

* Nuisance q's
chatCPUEq 7.23679e-005
chatTANq 0.310541

* B0
26991.3

* R0
2.08052e+006

* SSBs
SSB 26856.7 26521.7 25647.5 25714.4 25387.1 24996.2 24479.7 24295.6 24324.5
  24417 24334.9 24317 25985.1 27347.5 27260 26173.3 24891.9 22535.8 20209.9
  18103.3 16144.7 15101.5 14808.1 15386.2 15808 15212.1 13877.8 11984.6
year 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988
  1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
…
```

The file `R1-estimate.log` gives the estimated point estimates, in this case, for the MPD fit using Bayesian estimation. Values for the requested pseudo-observations (`@abundance MyAbundance` and `@proportions_at MyNumbersAtAge` in the `R1-output.csl` file) are also printed, e.g.,

```
* MyAbundance
1990  17325.4
1991  16808.8
1992  15768.2
1993  14995.4
…
```

And

```
* MyNumbersAtAge
1990  0.0581263 0.0284441 0.0420489 0.03362 0.0409341 0.0296477 0.0444192
  0.0329543 0.0762843 0.0135908 0.0229504 0.0193782 0.0161948 0.0134595
  0.0109508 0.00890025 0.00720975 0.00582549 0.00469327 0.00377397
  0.00303162 0.00243447 0.00195532 0.00157135 0.0503185 0.0251267 0.0377231
  0.0304816 0.0373439 0.0271131 0.0406257 0.030095 0.0695491 0.012388
  0.0209675 0.0177865 0.0149759 0.0125684 0.0103734 0.0085689 0.00706996
  0.00582922 0.00480031 0.00395067 0.00325104 0.00267588 0.0022034
  0.00181531
1991  0.0480715 0.0583304 0.0281859 0.0410706 0.032314 0.0386628 0.0274909
  0.040417 0.0294277 0.0668999 0.011719 0.0194872 0.0162308 0.0134046
  0.0110288 0.00889773 0.0071814 0.00578441 0.00465239 0.00373438 0.00299409
  0.00239955 0.00192336 0.00154256 0.0416143 0.0515305 0.0252956 0.037267
  0.0295133 0.0354163 0.0251903 0.037009 0.0269215 0.0612099 0.0107493
  0.0179764 0.0150978 0.0126091 0.0105131 0.00863212 0.00710145 0.00584053
  0.00480363 0.00394817 0.00324457 0.00266695 0.00219322 0.00180476
…
```

MCMC output can be generated by the call (directing the output to `R1-mcmc.log`). In this example, the random number seed has been specified to be zero (using the switch `-g 0`), rather than allowing CASAL to generate it from the local computer time.

```
> casal -m -q -g 0 -f R1- > R1-mcmc.log.
```

This generates two additional output files `objectives.1` and `samples.1` (assuming that the directory contains no other objectives or samples output). The `objectives.1` file lists the model call and header, the estimated covariance matrix from the MPD fitting, and the MCMC sampling diagnostics for each requested step (in this case, every 100th step), i.e.,

```
…
Main table:
sample posterior prior likelihood penalties stepsize acceptance_rate
stepsize_changes
100 -137.744 1.11438 -139.735 0.876353 0.02 0.51 0
200 -129.725 1.02938 -131.46 0.706078 0.02 0.53 0
300 -128.338 0.817104 -130.05 0.895355 0.02 0.533333 0
…
```

`samples.1` contains the MCMC output (in a format compatible with `casal -i [`*`filename`*`]`) i.e.,

```
initialization.B0 recruitment.YCS 28 selectivity[chatTANsel].male 2
  selectivity[chatTANsel].female 3 selectivity[chatFsel].male 2
  selectivity[chatFsel].female 3
28195.1 1 1 1 1 1 1 0.188928 1.27394 0.407116 0.569972 0.25885 0.465744
  0.393874 0.335951 0.222171 0.487427 0.368807 0.592477 0.9533 0.602292
  0.621707 0.551611 0.422651 0.298872 0.225235 0.0581383 1 1 11.7844 10.7146
  10.3776 11.3544 0.699292 9.32381 13.0633 5.98178 6.37918 0.484272
28972 1 1 1 1 1 1 0.276769 1.306 0.747165 0.324584 0.491915 0.513114
  0.356683 0.45288 0.238011 0.5314 0.414419 0.656852 1.02117 0.609924
  0.760073 0.533547 0.52732 0.292659 0.232141 0.108128 1 1 9.73686 9.10515
  10.0051 10.4983 0.850674 5.71922 9.2195 5.25037 6.53829 0.699299
…
```

The MCMC output contains estimates of the parameters defined in the `estimation.csl` file. To generate a file of quantities, CASAL must be run with the command

```
>casal -i samples.1 -v quantities.1 -f R1- > R1-quantities.log
```

`quantities.1` contains the MCMC quantities output. The file contains all of the estimated quantities at each point in the `samples.1` output file. For example,

```
Quantity values :
initialization.B0 recruitment.YCS[1] recruitment.YCS[2] recruitment.YCS[3]
   recruitment.YCS[4] recruitment.YCS[5] recruitment.YCS[6]
   recruitment.YCS[7] recruitment.YCS[8] recruitment.YCS[9] ...
28195.1 1 1 1 1 1 1 0.188928 1.27394 0.407116 0.569972 0.25885 0.465744
   0.393874 0.335951 0.222171 0.487427 0.368807 0.592477 0.9533 0.602292
   0.621707 0.551611 0.422651 0.298872 0.225235 0.0581383 1 1 0.0636366 ...
```

The output, `quantities.1`, is then available, in tabular format, for importing into another package (e.g., Microsoft Excel, S-Plus, or R) for plotting and summarising.

The call `casal -i samples.1 -Y -q -g 0 -f R1- > R1-yields.log` with the above parameter files generates an output file detailing the estimated yields. Note that this may take some time. The requested results (MCY and CAY) in the output file `R1-yields.log` are,

```
MCY      = 2569.02
Avg catch under MCY = 2543.07
B_MCY               = 10968.3
Prisk               = 0.1
'current' MCY = 2569.02
```

and

```
F_CAY    = 0.280151
MAY      = 2924.26
B_MAY    = 7749.12
Prisk    = 0.1
CAY      = 3681.86
```

14.1.4 Generating simulated output

The call `casal -e –O MPD.dat -q –g 0 -f R1-` with the above parameter files generates an additional output file, MPD.dat. This file contains only two lines, a header, and the free parameter values at the fitted MPD, e.g.,

```
initialization.B0 recruitment.YCS 28 selectivity[chatTANsel].male 2
     selectivity[chatTANsel].female 3 selectivity[chatFsel].male 2
     selectivity[chatFsel].female 3
26991.3 1 1 1 1 1 1 0.190797 0.933788 0.358616 0.436584 0.267153 0.344418
     0.268054 0.322199 0.21143 0.423725 0.342837 0.499052 0.72796 0.575916
     0.460651 0.441897 0.336943 0.195483 0.146892 0.0520493 1 1 11.8155
     11.2515 9.9213 10.853 0.691629 8.35602 11.7958 6.34521 10.8179 0.626889
```

The call `casal -s 10 simulated -i MPD.dat -g 0 -q -f R1-` generates 10 sets of files with simulated observations, derived from the fit specified by MPD.dat. These files are named `simulated.par1.sim[n]`, where [*n*] is a number from 1 to 10.

These can be appended (one at a time) to another estimation.csl file (that has no observations) called, say, `R1-estimation.stub`, the result renamed to `Sim-estimation.csl`, and CASAL called sequentially on each of these to parametric bootstrap around the MPD, i.e., assuming that the files `Sim-population.csl` and `Sim-ouput.csl` are copies of the corresponding `R1-` files, then repeating the call

```
casal -e -i MPD.dat -o simulations.dat -f Sim-
```

for each set of simulated observations will generate the file `simulations.dat` that contains 10 bootstrap estimates around the MPD, i.e.,

```
initialization.B0 recruitment.YCS 28 selectivity[chatTANsel].male 2
   selectivity[chatTANsel].female 3 selectivity[chatFsel].male 2
   selectivity[chatFsel].female 3
28226.2 1 1 1 1 1 1 0.611182 0.739155 0.358737 0.485705 0.385997 0.322078
   0.216983 0.35426 0.148324 0.362803 0.310595 0.564815 0.673322 0.596395
   0.349376 0.45873 0.264275 0.189889 0.116135 0.0517188 1 1 14.5405 16.088
   7.75714 8.71783 0.495174 7.10632 7.11254 4.87845 8.1634 0.654546
29354.6 1 1 1 1 1 1 0.269912 0.787098 0.631726 0.226458 0.415245 0.219591
   0.129776 0.316059 0.161407 0.33608 0.393575 0.476906 0.677873 0.593494
   0.41574 0.430715 0.372423 0.242873 0.198932 0.0957925 1 1 10.1676 10.2208
   10.1976 10.2057 0.991846 10.4214 12.9909 6.50454 9.74551 0.482515
…
```

This file can then be used to generate bootstrap quantities (using `casal -i simulations.dat -v`, as earlier with the MCMC output)

14.2 A Bayesian single-sex size-based model

14.2.1 Introduction

The example provided here is a slightly more complex Bayesian single-sex size-based model, incorporating disease mortality and multiple fisheries. This model is loosely based on the Foveaux Strait dredge oyster fishery in southern New Zealand. The abundance data consist of observations CPUE, and absolute dredge survey estimates. In addition, the survey series has a series of associated proportions-at-size data. Proportions-at-size data are also available from the commercial catch. This model is also a "numbers" model (i.e., a weightless model), and hence the biomasses reported by CASAL should be read as numbers. CASAL warns the user about this at the beginning of each output file with the message,

```
Note, this is a weightless model. 'Biomass' and 'weight in tons' should be
read as numbers of fish throughout.
```

The partition is defined to allow for size-classes in the model from 2–98+. Note that the last group is a plus group, and the mean size of fish in the plus group is assumed to be 100 mm. The partition ignores sex and does not include maturity.

The annual cycle is described in the `R2-population.csl` file, and consists of two time steps. The first represents a summer period (limited summer commercial fishing, growth, and disease mortality). The second time step represents the winter period (and includes the main commercial fishery, along with lesser recreational, customary, and an estimate of illegal fishing, and recruitment).

The remainder of the `population.csl` file defines the recruitment processes, maturation, natural mortality, fishing mortality (and catches), selectivities to be applied to the fishing and surveys, and growth increment parameters. Note that your results may look slightly different, depending on the computer and platform used to run CASAL.

14.2.2 The input parameter files

R2-population.csl

```
@initialization
B0 5000        # All biomass are recorded in NUMBERS (in millions)
@n_equilibrium 120

@weightless_model True # This model reports all outputs as numbers of fish,
   not biomass
@size_based True # Defines the model as a size-based model
@n_classes 49 # 49 size classes (labelled 1-49), each with width 2
   millimetres
# class nos. 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
   22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
   46 47 48 49
@class_mins  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42
   44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90
   92 94 96 98
@plus_group_size 100 # And the size of fish in the plus group is assumed to
   be 100 mm
@plus_group True
@sex_partition False # No sex distinction in the partition, i.e., this is a
   single sex model
@mature_partition False
@n_areas 1
@n_stocks 1

@initial 1985
@current 2003
@final 2008

@annual_cycle
time_steps 2 # 1=Summer (growth+spawn+fishery); 2=Winter (+fishery)
disease_mortality_time 1 # This model incorporates (and estimates) disease
   mortality from Bonamia exitiosa
recruitment_time 2
spawning_time 1
spawning_part_mort 0.5
spawning_p 1
n_growths 1
growth_times 1
M_props 0.5 0.5
baranov False
fishery_names SummerFishery WinterFishery Recreational Customary Illegal
fishery_times 1 2 2 2 2

@y_enter 1
@standardise_YCS True

@recruitment
n_rinitial 1
YCS_years 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997
   1998 1999 2000 2001 2002
YCS       1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
   1.00 1.00 1.00 1.00 1.00
SR BH
steepness 0.9
initial_size_mean 15.5
initial_size_cv 0.40
first_free 1985
last_free 2002
year_range 1985 2002

@randomisation_method lognormal-empirical
@first_random_year 2000
```

```
@growth
type basic
l 30 55
g 11.91 3.61
cv 0.31
minsigma 4.45

@maturity_props
all logistic 58 18

@natural_mortality
all 0.1

@disease_mortality
years 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998
   1999 2000 2001 2002 2003
index 0.00 0.18 0.03 0.05 0.11 0.39 0.38 0.20 0.00 0.01 0.17 0.00 0.00 0.00
   0.00 0.35 0.27 0.67 0.50
DM 1.0
selectivity DiseaseSel
future_years 2004 2005 2006 2007 2008
future_index  0.2  0.2  0.2  0.2  0.2

@fishery SummerFishery
years    1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997
   1998 1999 2000 2001 2002 2003
catches  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 2.43 3.09 3.03 0.93 0.20
   0.72 0.00 1.00 0.00 0.00 0.00
selectivity FishingSel
U_max 0.5

@fishery WinterFishery
years    1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995
   1996  1997  1998  1999  2000  2001  2002 2003
catches 81.79 60.22 47.64 67.81 65.81 35.69 41.80  4.51  0.00  0.00  0.00
   13.41 14.82 14.85 14.94 14.43 15.11 14.45 7.40
future_years    2004  2005  2006  2007  2008
future_catches 15.00 15.00 15.00 15.00 15.00
selectivity FishingSel
U_max 0.5

@fishery Recreational
years    1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995
   1996  1997  1998  1999  2000  2001  2002  2003
catches  0.38  0.38  0.38  0.39  0.39  0.39  0.39  0.40  0.00  0.00  0.00
   0.41  0.41  0.42  0.42  0.42  0.42  0.43  0.43
future_years   2004  2005  2006  2007  2008
future_catches 0.43  0.43  0.43  0.43  0.43
selectivity FishingSel
U_max 0.5

@fishery Customary
years    1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995
   1996  1997  1998  1999  2000  2001  2002  2003
catches  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
   0.00  0.00  0.14  0.18  0.22  0.26  0.18  0.20
future_years   2004  2005  2006  2007  2008
future_catches 0.20  0.20  0.20  0.20  0.20
selectivity FishingSel
U_max 0.5

@fishery Illegal
years    1985  1986  1987  1988  1989  1990  1991  1992  1993  1994  1995
   1996  1997  1998  1999  2000  2001  2002  2003
catches  0.04  0.04  0.04  0.04  0.04  0.04  0.04  0.04  0.00  0.00  0.00
   0.04  0.04  0.06  0.06  0.06  0.07  0.06  0.06
future_years   2004  2005  2006  2007  2008
```

```
future_catches 0.04  0.04  0.04  0.04  0.04
selectivity FishingSel
U_max 0.5

@selectivity_names FishingSel DiseaseSel DredgeSurveySelLegal
   DredgeSurveySelSubLegal DredgeSurveySelSmall LegalSized
@selectivity FishingSel
all logistic 60 3.5
@selectivity DiseaseSel
all logistic 58 18
@selectivity DredgeSurveySelLegal
all logistic 58 18
@selectivity DredgeSurveySelSubLegal
all double_logistic 50 5 8 5 0.8
@selectivity DredgeSurveySelSmall
all double_logistic 25 20 25 5 0.6
@selectivity LegalSized
all knife_edge 58
```

R2-estimation.csl

```
# ESTIMATION
@estimator Bayes # Use the Bayes estimation method
@max_iters 300   # With maximum of 300 iterations for the point estimates
@max_evals 1000  # and 1000 function evaluations
@grad_tol 0.002  # Set the tolerance for the convergence test at 0.002

@MCMC
start 0          # Start the MCMC at 0
length 110000   # and evaluate for 110000 steps
keep 100        # keeping every 100th sample
stepsize 0.02    # with the stepsize for the MCMC set at 0.02
adaptive_stepsize true # but adapt the stepsize during the evaluation
adapt_at 5000  # after the 5000th step
burn_in 100      # The MCMC has a burn-in period of 100*100=10000 steps

# OBSERVATIONS: Biomass CPUE (catch per hour)
@relative_abundance CPUE-C
years 1985 1986 1987 1988 1990 1991 1992 1996 1997 1998 1999 2000 2001 2002
   2003
step 2
proportion_mortality 0.5
q CPUE-Cq
biomass True
ogive FishingSel
1985 12.13
1986 10.51
1987  9.13
1988 10.00
1990  9.65
1991  5.82
1992  3.16
1996  5.16
1997  6.43
1998  6.27
1999  6.37
2000  6.56
2001  6.50
2002  3.18
2003  2.32
cv 0.25
dist lognormal

# OBSERVATIONS: Biomass Dredge surveys (Legal Sized)
@relative_abundance OctSurveyLegal
years 1990 1993 1995 1997 1999 2001 2002
step 2
proportion_mortality 1.0
```

```
q DredgeSurveyq
biomass True
ogive DredgeSurveySelLegal
1990  623
1993  397
1995  782
1997  660
1999 1453
2001  995
2002  502
cvs_1990 0.12
cvs_1993 0.10
cvs_1995 0.11
cvs_1997 0.14
cvs_1999 0.16
cvs_2001 0.11
cvs_2002 0.14
dist lognormal

@relative_abundance JulSurveyLegal
years 1990
step 2
proportion_mortality 0.8
q DredgeSurveyq
biomass True
ogive DredgeSurveySelLegal
1990 707
cvs_1990 0.11
dist lognormal

@relative_abundance MarSurveyLegal
years 1992 1995
step 2
proportion_mortality 0.0
q DredgeSurveyq
biomass True
ogive DredgeSurveySelLegal
1992  285
1995  576
cvs_1992 0.11
cvs_1995 0.25
dist lognormal

# OBSERVATIONS: Biomass Dredge survey (sub-legal sized)
@relative_abundance OctSurveySubLegal
years 1993 1995 1997 1999 2001 2002
step 2
proportion_mortality 1.0
q DredgeSurveyq
biomass True
ogive DredgeSurveySelSubLegal
1993  383
1995  380
1997  727
1999  896
2001  872
2002  520
cvs_1993  0.11
cvs_1995  0.10
cvs_1997  0.14
cvs_1999  0.12
cvs_2001  0.12
cvs_2002  0.11
dist lognormal

@relative_abundance MarSurveySubLegal
years 1995
step 2
```

```
proportion_mortality 0.0
q DredgeSurveyq
biomass True
ogive DredgeSurveySelSubLegal
1995  401
cvs_1995 0.28
dist lognormal

# OBSERVATIONS: Biomass Dredge survey (Smalls)
@relative_abundance OctSurveySmall
years 1993 1995 1997 1999 2001 2002
step 2
proportion_mortality 1.0
q DredgeSurveyq
biomass True
ogive DredgeSurveySelSmall
1993 1004
1995  718
1997  918
1999 1364
2001 1410
2002 1243
cvs_1993  0.10
cvs_1995  0.21
cvs_1997  0.14
cvs_1999  0.11
cvs_2001  0.12
cvs_2002  0.10
dist lognormal

@relative_abundance MarSurveySmall
years 1995
step 2
proportion_mortality 0.0
q DredgeSurveyq
biomass True
ogive DredgeSurveySelSmall
1995  402
cvs_1995 0.25
dist lognormal

# OBSERVATIONS: proportions-at-length
@proportions_at OctSurveyLegalLength
years 1999 2001
step 2
proportion_mortality 1.0
sexed false
plus_group true
sum_to_one true
min_class 5
max_class 45
ogive DredgeSurveySelLegal
# class      10     12     14     16     18     20     22     24     26
    28     30     32     34     36     38     40     42     44     46     48
    50     52     54     56     58     60     62     64     66     68     70
    72     74     76     78     80     82     84     86     88     90
# class nos.  5      6      7      8      9     10     11     12     13
    14     15     16     17     18     19     20     21     22     23     24
    25     26     27     28     29     30     31     32     33     34     35
    36     37     38     39     40     41     42     43     44     45
1999     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0.0000 0.0003 0.0001 0.0012 0.0043 0.0351 0.1382 0.1121 0.1338 0.0979
    0.0931 0.1199 0.0598 0.0645 0.0418 0.0224 0.0285 0.0145 0.0113 0.0050
    0.0059 0.0103
2001     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
    0.0005 0.0004 0.0036 0.0070 0.0139 0.0833 0.1803 0.1474 0.1637 0.1144
```

```
   0.0904 0.0683 0.0446 0.0262 0.0158 0.0153 0.0091 0.0052 0.0065 0.0014
   0.0014 0.0012
N_1999 1277
N_2001  887
dist multinomial
r 0.00001

@proportions_at OctSurveySubLegalLength
years 1999 2001
step 2
proportion_mortality 1.0
sexed false
plus_group true
sum_to_one true
min_class 5
max_class 45
ogive DredgeSurveySelSubLegal
# class      10     12     14     16     18     20     22     24     26
   28     30     32     34     36     38     40     42     44     46     48
   50     52     54     56     58     60     62     64     66     68     70
   72     74     76     78     80     82     84     86     88     90
# class nos.  5      6      7      8      9     10     11     12     13
   14     15     16     17     18     19     20     21     22     23     24
   25     26     27     28     29     30     31     32     33     34     35
   36     37     38     39     40     41     42     43     44     45
1999     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001 0.0016 0.0016 0.0024
   0.0058 0.1087 0.1708 0.2673 0.2322 0.1220 0.0537 0.0142 0.0114 0.0057
   0.0025 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000
2001     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0010 0.0043 0.0084 0.0099
   0.0260 0.1329 0.1924 0.2075 0.2203 0.1348 0.0389 0.0129 0.0064 0.0026
   0.0016 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000
N_1999 953
N_2001 544
dist multinomial
r 0.00001

@proportions_at OctSurveySmallLength
years 1999 2001
step 2
proportion_mortality 1.0
sexed false
plus_group true
sum_to_one true
min_class 5
max_class 45
ogive DredgeSurveySelSmall
# class      10     12     14     16     18     20     22     24     26
   28     30     32     34     36     38     40     42     44     46     48
   50     52     54     56     58     60     62     64     66     68     70
   72     74     76     78     80     82     84     86     88     90
# class nos.  5      6      7      8      9     10     11     12     13
   14     15     16     17     18     19     20     21     22     23     24
   25     26     27     28     29     30     31     32     33     34     35
   36     37     38     39     40     41     42     43     44     45
1999     0.0190 0.0265 0.0350 0.0301 0.0442 0.0437 0.0404 0.0492 0.0387
   0.0416 0.0571 0.0504 0.0571 0.0514 0.0459 0.0598 0.0567 0.0725 0.0638
   0.0749 0.0422 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000
2001     0.0220 0.0341 0.0366 0.0355 0.0365 0.0501 0.0460 0.0429 0.0372
   0.0354 0.0407 0.0335 0.0386 0.0410 0.0335 0.0652 0.0654 0.0682 0.0916
   0.0756 0.0423 0.0140 0.0083 0.0047 0.0010 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000
```

```
N_1999 1273
N_2001 1074
dist multinomial
r 0.00001

@proportions_at OctDive1990
years 1990
step 2
proportion_mortality 1.0
sexed false
plus_group false
sum_to_one true
min_class 5
max_class 45
# class      10     12     14     16     18     20     22     24     26
   28     30     32     34     36     38     40     42     44     46     48
   50     52     54     56     58     60     62     64     66     68     70
   72     74     76     78     80     82     84     86     88     90
# class nos.  5      6      7      8      9     10     11     12     13
   14     15     16     17     18     19     20     21     22     23     24
   25     26     27     28     29     30     31     32     33     34     35
   36     37     38     39     40     41     42     43     44     45
1990     0.0437 0.0694 0.0549 0.0768 0.0297 0.0362 0.0326 0.0281 0.0190
   0.0108 0.0181 0.0060 0.0131 0.0122 0.0100 0.0182 0.0226 0.0218 0.0251
   0.0182 0.0307 0.0494 0.0388 0.0680 0.0340 0.0811 0.0333 0.0324 0.0299
   0.0178 0.0069 0.0077 0.0013 0.0010 0.0009 0.0000 0.0004 0.0000 0.0000
   0.0000 0.0000
N_1990 461
dist multinomial
r 0.00001

# OBSERVATIONS: Commercial catch-at-length
@catch_at CommercialCatchLength
years 2002 2003
fishery WinterFishery
sexed false
plus_group true
sum_to_one true
min_class 5
max_class 45
# class      10     12     14     16     18     20     22     24     26
   28     30     32     34     36     38     40     42     44     46     48
   50     52     54     56     58     60     62     64     66     68     70
   72     74     76     78     80     82     84     86     88     90
# class nos.  5      6      7      8      9     10     11     12     13
   14     15     16     17     18     19     20     21     22     23     24
   25     26     27     28     29     30     31     32     33     34     35
   36     37     38     39     40     41     42     43     44     45
2002     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0002 0.0015 0.0042 0.0138 0.0436 0.1523 0.1721 0.1538 0.1336
   0.1055 0.0768 0.0567 0.0374 0.0207 0.0137 0.0069 0.0039 0.0015 0.0010
   0.0005 0.0000
2003     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
   0.0000 0.0001 0.0001 0.0077 0.0255 0.0548 0.1666 0.1843 0.1699 0.1255
   0.0965 0.0694 0.0440 0.0245 0.0131 0.0093 0.0044 0.0023 0.0011 0.0005
   0.0001 0.0002
N_2002 10932
N_2003 15254
dist multinomial
r 0.00001

# OBSERVATIONS: Maturity observations from paper by Jeffs & Hickman (2000)
@proportions_mature JeffsHickman
years 2000
step 1
proportion_mortality 0.5
```

```
sexed false
plus_group false
min_class  7
max_class 42
#class mins 14   16   18   20   22   24   26   28   30   32   34   36   38
   40   42   44   46   48   50   52   54   56   58   60   62   64   66   68
   70   72   74   76   78   80   82   84
#class nos   7    8    9   10   11   12   13   14   15   16   17   18   19
   20   21   22   23   24   25   26   27   28   29   30   31   32   33   34
   35   36   37   38   39   40   41   42
2000      0.00 0.00 0.14 0.00 0.00 0.04 0.00 0.03 0.00 0.00 0.00 0.00 0.00
   0.33 0.25 0.15 0.26 0.40 0.49 0.62 0.61 0.50 0.64 0.77 0.66 0.79 0.79
   0.86 0.90 0.83 0.87 0.87 0.91 0.94 0.88 0.83
Ns_2000      1    5    7   10   14   25   29   37   26   23   23   14    7
   3    4   13   43   58   76   34   18   26   25   44   65   81   62   28
   30   40   60   45   22   18    8    6
dist binomial
r 0.00001

# CATCHABILITY COEFFICIENTS
@q_method nuisance

@estimate
parameter q[CPUE-Cq].q
lower_bound 0.00000001
upper_bound 1.0
prior uniform-log

@estimate
parameter q[DredgeSurveyq].q
lower_bound 1.00
upper_bound 1.00
prior lognormal
mu 1.0
cv 0.1

#FREE PARAMETERS
@estimate
parameter initialization.B0
lower_bound    100
upper_bound  10000
prior uniform

@estimate
parameter recruitment.YCS
#YCS_years    1985  1986  1987  1988  1989  1990  1991  1992  1993  1994
   1995  1996  1997  1998  1999  2000  2001  2002
lower_bound  0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001
   0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001
upper_bound  100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0
   100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0
prior lognormal
mu               1     1     1     1     1     1     1     1     1     1
   1     1     1     1     1     1     1     1
cv             0.2   0.2   0.2   0.2   0.2   0.2   0.2   0.2   0.2   0.2
   0.2   0.2   0.2   0.2   0.2   0.2   0.2   0.2

@estimate
parameter disease_mortality.index
#index.years 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996
   1997 1998 1999 2000 2001 2002 2003
lower_bound  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
   0.00 0.00 0.00 0.00 0.00 0.00 0.00
upper_bound  0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.80 0.00
   0.00 0.00 0.00 0.80 0.80 0.80 0.80
prior normal-by-stdev
mu           -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -
   0.2 -0.2 -0.2 -0.2 -0.2 -0.2 -0.2
```

```
stdev        0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20 0.20
   0.20 0.20 0.20 0.20 0.20 0.20 0.20

@estimate
parameter maturity_props.all
same selectivity[DiseaseSel].all
lower_bound  10 0.1
upper_bound 100 80
prior uniform

@estimate
parameter selectivity[FishingSel].all
same selectivity[DredgeSurveySelLegal].all
lower_bound  50 0.1
upper_bound  75  45
prior uniform

@estimate
parameter selectivity[DredgeSurveySelSubLegal].all
lower_bound  35  1  1  1 0.2
upper_bound  65 10 20 10 1.5
prior uniform

@estimate
parameter selectivity[DredgeSurveySelSmall].all
lower_bound   5  1  1  1 0.2
upper_bound  55 50 50 50 1.5
prior uniform

# PENALTIES
@catch_limit_penalty
# Penalise model fits that do not allow the Commercial Fishing catch to be
   taken
label WinterCatchMustBeTaken
fishery WinterFishery
log_scale true
multiplier 1000

@catch_limit_penalty
# Penalise model fits that do not allow the Summer special permit catch to
   be taken
label SummerCatchMustBeTaken
fishery SummerFishery
log_scale true
multiplier 1000

@catch_limit_penalty
# Penalise model fits that do not allow the Recreational catch to be taken
label RecreationalCatchMustBeTaken
fishery Recreational
log_scale true
multiplier 1000

@catch_limit_penalty
# Penalise model fits that do not allow the Customary catch to be taken
label CustomaryCatchMustBeTaken
fishery Customary
log_scale true
multiplier 1000

@catch_limit_penalty
# Penalise model fits that do not allow the Illegal catch to be taken
label IllegalCatchMustBeTaken
fishery Illegal
log_scale true
multiplier 1000

@vector_average_penalty
```

```
label meanYCS_1
vector recruitment.YCS
k 1
multiplier 6
```

R2-output.csl

```
@print
parameters False
unused_parameters True
population_section True
requests True
results False
initial_state False
state_annually False
state_every_step False
final_state True
estimation_section True
fits True
resids True
pearson_resids False
normalised_resids True
covariance False
yields True
fits_every_eval False
objective_every_eval False
parameters_every_eval False
parameter_vector_every_eval False

@quantities
all_free_parameters True
fishing_pressures True
nuisance_qs True
true_YCS True
B0 True
R0 True
SSBs True
YCS True
actual_catches False
ogive_parameters selectivity[FishingSel].all selectivity[DiseaseSel].all
   selectivity[DredgeSurveySelLegal].all
   selectivity[DredgeSurveySelSubLegal].all
   selectivity[DredgeSurveySelSmall].all

@catch_split 1 1 1 1 1

@B_pre
step 1
proportion_mortality 0
selectivity none
mature_only true

@MCY_CAY
do_MCY True
MCY_guess 500
n_discard 100
n_keep 100
n_simulations 100
do_CAY True
F_CAY_guess 0.2
interactive False

@abundance FishingVulnerableBiomass
biomass true
step 2
proportion_mortality 0.5
ogive FishingSel
```

```
years 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998
   1999 2000 2001 2002 2003 2004 2005 2006 2007 2008

@abundance LegalSizedBiomass
biomass true
step 2
proportion_mortality 0.0
ogive LegalSized
years 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998
   1999 2000 2001 2002 2003 2004 2005 2006 2007 2008
```

14.2.3 CASAL output

The call `casal -e -q –g 0 -f R2- > R2-estimate.log` with the above parameter files generates an output file. The first few lines of the output file, `R2-estimate.log`, are;

```
CASAL (C++ algorithmic stock assessment laboratory)
Call: c:\Windows\Batch\casal.exe -e -q -g 0 -O MPD.dat -f R2-
Date: Wed Mar 21 13:30:09 2012
v2.30-2012-03-21 00:22:59 UTC (rev.4648)  (c) Copyright 2002-2012, NIWA
User name: dunn
Machine name: DUNNA (Microsoft Windows, PID=3836)

Prefix for the input parameter filenames : R2-

Random number seed : 0

Note, this is a weightless model. 'Biomass' and 'weight in tons' should be
   read as numbers of fish throughout.
Note, this is a weightless model. 'Biomass' and 'weight in tons' should be
   read as numbers of fish throughout.
Estimation section :

Objective function: Bayes

Observations:

The observations are:
CPUE-C
JulSurveyLegal
MarSurveyLegal
MarSurveySmall
MarSurveySubLegal
OctSurveyLegal
OctSurveySmall
OctSurveySubLegal
OctDive1990
OctSurveyLegalLength
OctSurveySmallLength
OctSurveySubLegalLength
CommercialCatchLength
JeffsHickman
…
Minimiser achieved convergence after 74 quasi-Newton iterations using 117
   objective function evaluations
…
```

These lines give information on the CASAL version that was run, the date, and machine, file prefix, random number generator seed, and the convergence summary from the minimiser.

The remainder of the output file gives fits, residuals, and other requested output information, as defined in the `R2-output.csl` file. Some of this output is reproduced here.

```
Start extracting output from here

Point estimate:
initialization.B0
current value: 5003.3

recruitment.YCS
current values:
0.842139 0.620656 0.470037 0.443903 1.23306 0.604053 0.667887 0.790149
   0.804799 1.09531 2.18669 1.17794 1.96644 1.41552 1.01876 1.41475 0.981823
   0.889131

disease_mortality.index
current values:
9.57954e-008 0.560928 0.145464 0.0449698 0.0940256 0.484967 0.28497 0.207733
   1.77539e-010 3.65729e-006 0.211501 0 0 0 0 0.720868 2.5918e-006 0.636595
   0.699337

maturity_props.all (parameters)
current values:
56.7198 23.1345

selectivity[FishingSel].all (parameters)
current values:
60.5598 3.7443

selectivity[DredgeSurveySelSubLegal].all (parameters)
current values:
52.5115 4.65798 6.07433 5.51786 1.02564

selectivity[DredgeSurveySelSmall].all (parameters)
current values:
10.4203 10.3425 40.5668 3.60959 0.521079


In a format suitable for -i:

initialization.B0 recruitment.YCS 18 disease_mortality.index 19
   maturity_props.all 2 selectivity[FishingSel].all 2
   selectivity[DredgeSurveySelSubLegal].all 5
   selectivity[DredgeSurveySelSmall].all 5
5003.3 0.842139 0.620656 0.470037 0.443903 1.23306 0.604053 0.667887
   0.790149 0.804799 1.09531 2.18669 1.17794 1.96644 1.41552 1.01876 1.41475
   0.981823 0.889131 9.57954e-008 0.560928 0.145464 0.0449698 0.0940256
   0.484967 0.28497 0.207733 1.77539e-010 3.65729e-006 0.211501 0 0 0 0
   0.720868 2.5918e-006 0.636595 0.699337 56.7198 23.1345 60.5598 3.7443
   52.5115 4.65798 6.07433 5.51786 1.02564 10.4203 10.3425 40.5668 3.60959
   0.521079

Objective function : 995.643

Components :
                                            CPUE-C -4.59804
                                    JulSurveyLegal -2.09795
                                    MarSurveyLegal 6.69786
                                    MarSurveySmall 2.99077
                                 MarSurveySubLegal -1.01242
                                    OctSurveyLegal 0.139132
                                    OctSurveySmall -0.537178
                                 OctSurveySubLegal -8.50333
                                       OctDive1990 125.741
                              OctSurveyLegalLength 211.905
                              OctSurveySmallLength 183.532
                           OctSurveySubLegalLength 99.5975
                             CommercialCatchLength 219.106
                                      JeffsHickman 68.6688
                        prior_on_initialization.B0 0
```

```
                           prior_on_recruitment.YCS 41.9491
                   prior_on_disease_mortality.index 57.0184
                        prior_on_maturity_props.all 0
               prior_on_selectivity[FishingSel].all 0
 prior_on_selectivity[DredgeSurveySelSubLegal].all 0
    prior_on_selectivity[DredgeSurveySelSmall].all 0
                                 prior_on_q_CPUE-Cq -4.96185
                           prior_on_q_DredgeSurveyq 0.00124379
                             WinterCatchMustBeTaken 0
                             SummerCatchMustBeTaken 0
                       RecreationalCatchMustBeTaken 0
                          CustomaryCatchMustBeTaken 0
                            IllegalCatchMustBeTaken 0
                                          meanYCS_1 0.00718855
…
```

The output then continues with fits (if requested), other output quantities, unused parameters, and other requested quantities.

These include the following lines …

```
Output quantities start here

* Scalar parameter values
initialization.B0 5003.3

* Vector parameter values
recruitment.YCS 0.842139 0.620656 0.470037 0.443903 1.23306 0.604053
   0.667887 0.790149 0.804799 1.09531 2.18669 1.17794 1.96644 1.41552
   1.01876 1.41475 0.981823 0.889131
disease_mortality.index 9.57954e-008 0.560928 0.145464 0.0449698 0.0940256
   0.484967 0.28497 0.207733 1.77539e-010 3.65729e-006 0.211501 0 0 0 0
   0.720868 2.5918e-006 0.636595 0.699337

* Ogive parameter values
maturity_props.all 0.00107196 0.00138228 0.00178226 0.00229772 0.00296181
   0.0038171 0.00491817 0.00633482 0.00815619 0.0104957 0.0134971 0.0173419
   0.022257 0.0285249 0.036492 0.0465776 0.0592792 0.0751714 0.0948943
   0.119125 0.148529 0.183676 0.224943 0.272389 0.325637 0.383803 0.445496
   0.508915 0.572048 0.63292 0.689828 0.741516 0.787248 0.826778 0.860268
   0.888158 0.911057 0.929639 0.944576 0.956489 0.965935 0.973387 0.979243
   0.983833 0.98742 0.99022 0.992401 0.994099 0.995964
selectivity[FishingSel].all 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   1.00687e-006 4.85301e-006 2.33907e-005 0.000112732 0.000543125 0.00261241
   0.0124672 0.0573594 0.226779 0.585689 0.872019 0.97045 0.993722 0.998691
   0.999728 0.999944 0.999988 0.999998 0.999999 1 1 1 1 1 1 1 1 1 1
selectivity[DredgeSurveySelSubLegal].all 3.0741e-014 1.08837e-013 3.85333e-
   013 1.36425e-012 4.83007e-012 1.71006e-011 6.0544e-011 2.14353e-010
   7.58907e-010 2.68688e-009 9.51276e-009 3.36795e-008 1.19241e-007
   4.22166e-007 1.49466e-006 5.29174e-006 1.8735e-005 6.63276e-005
   0.000234797 0.000830875 0.00293652 0.0103325 0.0358003 0.117837 0.334039
   0.693354 0.995943 0.841498 0.535092 0.259927 0.104176 0.0379914 0.013344
   0.0046234 0.00159427 0.000548837 0.000188833 6.49569e-005 2.23432e-005
   7.68517e-006 2.64338e-006 9.09208e-007 3.12728e-007 1.07565e-007
   3.69977e-008 1.27256e-008 4.37705e-009 1.50551e-009 3.03697e-010
selectivity[DredgeSurveySelSmall].all 0.0562282 0.09177 0.142874 0.208611
   0.282045 0.3522 0.409895 0.451773 0.479494 0.496742 0.507064 0.513096
   0.516574 0.518563 0.519696 0.520339 0.520703 0.52091 0.521026 0.521028
   0.520408 0.517264 0.50177 0.435146 0.259221 0.0845345 0.0190203
   0.00383376 0.000754521 0.000147791 2.89211e-005 5.65853e-006 1.10707e-006
   2.16594e-007 4.23756e-008 8.29059e-009 1.62201e-009 3.17339e-010
   6.20859e-011 1.21468e-011 2.37646e-012 4.64944e-013 9.09641e-014
   1.77967e-014 3.48184e-015 6.81204e-016 1.33274e-016 2.60745e-017
   2.25642e-018
selectivity[DiseaseSel].all 0.00107196 0.00138228 0.00178226 0.00229772
   0.00296181 0.0038171 0.00491817 0.00633482 0.00815619 0.0104957 0.0134971
```

```
   0.0173419 0.022257 0.0285249 0.036492 0.0465776 0.0592792 0.0751714
   0.0948943 0.119125 0.148529 0.183676 0.224943 0.272389 0.325637 0.383803
   0.445496 0.508915 0.572048 0.63292 0.689828 0.741516 0.787248 0.826778
   0.860268 0.888158 0.911057 0.929639 0.944576 0.956489 0.965935 0.973387
   0.979243 0.983833 0.98742 0.99022 0.992401 0.994099 0.995964
selectivity[DredgeSurveySelLegal].all 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   0 1.00687e-006 4.85301e-006 2.33907e-005 0.000112732 0.000543125
   0.00261241 0.0124672 0.0573594 0.226779 0.585689 0.872019 0.97045
   0.993722 0.998691 0.999728 0.999944 0.999988 0.999998 0.999999 1 1 1 1 1
   1 1 1 1 1

* Nuisance q's
CPUE-Cq 0.00699994
DredgeSurveyq 1

* B0
5003.3

* R0
802.15
…
```

The file `R2-estimate.log` gives the estimated point estimates, in this case, for the MPD fit using Bayesian estimation. MCMC output can be generated by the call (directing the output to `R2-mcmc.log`). In this example, the random number seed has been specified to be zero (using the switch `-g 0`), rather than allowing CASAL to generate it from the local computer time.

```
> casal -m -q -g 0 -f R2- > R2-mcmc.log.
```

This generates two additional output files `objectives.1` and `samples.1` (assuming that the directory contains no other objectives or samples output). The `objectives.1` file lists the model call and header, the estimated covariance matrix from the MPD fitting, and the MCMC sampling diagnostics for each requested step (in this case, every 100th step), i.e.,

```
…
Main table:
sample posterior prior likelihood penalties stepsize acceptance_rate
   stepsize_changes
100 1009.06 107.394 901.645 0.0212105 0.02 0.45 0
200 1009.67 106.965 902.639 0.0613163 0.02 0.5 0
300 1015.1 98.9947 916.024 0.0811597 0.02 0.506667 0
…
```

`samples.1` contains the MCMC output (in a format compatible with `casal -i [filename]`) i.e.,

```
initialization.B0 recruitment.YCS 18 disease_mortality.index 19
   maturity_props.all 2 selectivity[FishingSel].all 2
   selectivity[DredgeSurveySelSubLegal].all 5
   selectivity[DredgeSurveySelSmall].all 5
4988.03 0.8903 0.541577 0.464004 0.334931 1.15734 0.678958 0.690249 0.914572
   0.841393 1.25355 2.03705 1.4419 2.06978 1.40167 1.05352 1.47004 1.11363
   0.715744 0.00901196 0.545628 0.168589 0.043636 0.103411 0.478688 0.270416
   0.197135 0.0213912 0.0141107 0.177349 0 0 0 0 0.667133 0.018037 0.687701
   0.675338 56.3313 21.6523 60.5883 3.78719 52.5086 4.78339 6.11415 5.20456
   1.02133 9.78621 11.3041 41.3282 3.52578 0.499744
5227.69 0.783381 0.599342 0.419466 0.381306 1.25542 0.672009 0.677731
   0.961751 0.878782 1.16736 1.84341 1.82665 2.10989 1.51046 1.11757 1.435
   1.20196 0.978154 0.00329859 0.568461 0.156018 0.0410332 0.108869 0.463687
   0.271007 0.20154 0.0201264 0.00254885 0.181828 0 0 0 0 0.673671 0.0231019
```

```
   0.659933 0.668149 56.3034 23.013 60.5304 3.74548 52.566 4.74327 6.12708
   5.17516 0.966804 9.16697 12.0507 41.9043 3.23227 0.492871
…
```

The MCMC output contains estimates of the parameters defined in the `estimation.csl` file. To generate a file of quantities, CASAL must be run with the command

```
>casal -i samples.1 -v quantities.1 -f R2- > R2-quantities.log
```

`quantities.1` contains the MCMC quantities output. The file contains all of the estimated quantities at each point in the `samples.1` output file. For example,

```
Quantity values :
initialization.B0 recruitment.YCS[1] recruitment.YCS[2] recruitment.YCS[3]
   recruitment.YCS[4] recruitment.YCS[5] recruitment.YCS[6]
   recruitment.YCS[7] recruitment.YCS[8] recruitment.YCS[9] ...
4988.03 0.8903 0.541577 0.464004 0.334931 1.15734 0.678958 0.690249 0.914572
   0.841393 1.25355 2.03705 1.4419 2.06978 1.40167 1.05352 1.47004 1.11363
   0.715744 0.00901196 0.545628 0.168589 0.043636 0.103411 0.478688 ...
```

The output, `quantities.1`, is then available, in tabular format, for importing into another package (e.g., Microsoft Excel, S-Plus, or R) for plotting and summarising.

The call `casal -i samples.1 -Y -q -g 0 -f R2- > R2-yields.log` with the above parameter files generates an output file detailing the estimated yields. Again, as for the age-based example, yield calculations can take some time. The requested results (MCY and CAY) in the output file `R2-yields.log` are,

```
MCY      = 374.889
Avg catch under MCY = 363.711
B_MCY               = 1629.62
Prisk               = 0.1
'current' MCY = 332.603
```

and

```
F_CAY    = 0.301389
MAY      = 392.55
B_MAY    = 1318.77
Prisk    = 0.1
CAY      = 187.731
```

15. CHANGES AND ENHANCEMENTS FROM PREVIOUS VERSIONS

15.1 Introduction

The changes and enhancements between published versions CASAL are detailed below. These include any incremental changes for unpublished versions that may have been distributed in-between published versions. References for the published versions of CASAL are Bull et al. (2002) for CASAL v1.02-2002/10/21, Bull et al. (2003) for CASAL v2.01-2003/08/01, Bull et al. (2004) for CASAL v2.06-2004/09/26, Bull at al. (2005) for v2.07-2005/08/21, and Bull et al. (2008) for v2.20-2008/02/14.

15.2 Changes and enhancements from v1.02-2002/10/21 to v2.01-2003/08/01

1. A wide range of additional error checking has been implemented. CASAL now does much more extensive checks to ensure that the commands, sub-commands, and parameters for the input parameter files are valid.

2. In certain circumstances, simulations (i.e., MCY/CAY calculations) produced erroneous results. These calculations have been repaired.

3. The burn-in period specified for Bayesian models in the `estimation.csl` file has been repaired so that it now acts as specified in the manual.

4. A bug relating to the application of size-based fishery selectivity ogives in age-based models with no ogive shift has been repaired.

5. The application of mature and immature selectivities has been swapped, so it is now acting as it should.

6. Some minor bugs relating to the implementation of size based models have been repaired.

7. Natural mortality can now be estimated for independently for different stocks in a multi-stock model.

8. A bug in `@profile` has been repaired so the output when profiling more than one parameter is as expected.

9. If a logical (switch) argument is supplied that is not of the form true/false, CASAL will now report an error.

10. A bug in the output of quantities when `Cinitial` was specified has been corrected.

11. A problem with determining the initial state (`Cinitial`) that occurred if ageing is before recruitment in the annual cycle has been identified. CASAL now reports an error message if it cannot determine the initial state under these circumstances.

12. Overflow errors that resulted from some calculations are identified before being attempted. CASAL now handles most (but not all) situations where parameter values would lead to an overflow using improved algorithms.

13. An error in the `logistic_producing` ogive has been repaired, so it is now acting as it should.

14. The `@estimate.same` subcommand has been repaired so that it now works as it should with estimable estimation parameters (i.e., catchability coefficients, *q*).

15. A small number of other problems in lesser used functions and print functions have also been corrected.

16. A new command line switch `casal -l` has been added to display the end-user licence.

17. New command line switches (`-o` and `-O`) have been added that dump the free parameters (from `casal -r`, `-e`, or `-E`) to a text file, in a form suitable for use with `casal -i`.

18. An additional transition process, disease mortality, has been added to the annual cycle. See Section 5 for details.

19. Additional output for size based selectivities in an age-based model have been added. These allow the user to request that CASAL output size based selectivities as age based selectivities, given the partition, year, and time step.

20. An additional penalty, `vector_smoothing_penalty` has been introduced. This behaves in a manner similar to the `ogive_smoothing_penalty`.

21. Additional ogives have been added; `double_normal_plateau`, `double_logistic`, and `logistic_product`.

22. Fits, residuals, Pearson residuals, and normalised residuals can now be output as quantities.

23. The `pseudo`, `vector`, and `ogive` sub-commands have been classified as obsolete. These are no longer necessary.

24. The constraint that did not allow migrations to be both a source and a destination in the same time step has been removed. Migrations are now applied in the order specified in the input parameter files.

25. The beta prior has been enhanced to include scale and shift parameters. It is now specified in terms of a standard deviation rather than a c.v.

26. Multiple copies of single warning messages are now (mostly) suppressed, so that each warning message is only reported once.

27. The behaviour of the random number seed has been changed so that the default behaviour (when no seed is supplied) is to create a seed from the computer clock. Use the command `casal -g 0` option to replicate previous behaviour.

28. Two new command line switches allow the user to modify the names of the parameter input files (`casal -f` to add a prefix and `casal -F` to modify the `csl` suffix)

29. A new form of parameterising YCS has been added for Bayesian estimation (the Francis parameterisation). See Section 5.4.2.

30. The `@recruitment` subcommand, `standardise_YCS`, has been made a command. Now the form of YCS parameterisation must be specified for the entire model with a single command, rather than separately by stock (i.e., by using the command `@standardise_YCS true).`

31. The `@proportions_mature` observation type has been enhanced to allow for (a) size-based models and (b) unsexed models.

32. The `@age_size` observation type has been rewritten and considerably enhanced. See Section 6.6 for details.

33. The default method for generating recruitments in MCY/CAY simulations has been changed so that there is no default. This must now be specified by the user.

34. Alternative reference years for defining risk in the MCY/CAY simulations can now be specified.

35. CASAL has been ported from `gcc` version 2.95/2.96 to `gcc` version 3.2.3. CASAL has also been implemented as a native Microsoft Windows application, removing the need for the Cygwin on Microsoft Windows platforms.

36. Simulations with a growth curve of type `@size_at_age = data` are now implemented for multi-stock models. The `@size_at_age.simulation_male`, `simulation_female`, and `simulation_all` subcommands are now obsolete. CASAL uses the mean size of all supplied observations in simulations instead.

37. An additional error check on values of user supplied units of the size-weight parameters has been implemented. CASAL will now either warn if the values of the size-weight parameters supplied are not within the default range, or will error out if the size-weight parameters are outside a user supplied range.

38. A new observation type, `@age-at-maturation` has been added. See Section 6.6.3 for details.

39. Size based observations in a size-based model can now be integer combinations (i.e., pooled combinations) of the size-classes defined by the partition. This allows size observations with measurements made at a coarser scale than that defined by the partition to be used in the model.

40. CASAL can now generate simulated observations, i.e., generate observed values with random error that are based on a supplied "fit". See Section 6.9.

41. CASAL can now, optionally, print the eigenvalues of the Hessian.

42. CASAL can now, optionally, update the Hessian matrix during the burn-in phase of an MCMC.

43. The binomial likelihood has been added for use with proportions mature and proportions migrating observations. See Section 6.7.1.

15.3 Changes and enhancements from v2.01-2003/08/01 to v2.06-2004/09/26

1. The header lines associated with all CASAL output have been simplified, and the date associated with the CASAL version number has been supplemented with a UTC time stamp.

2. The `.C` filename suffix for user extensions has been renamed `.cpp` to comply with standard C++ programming practise.

3. The subcommands `@disease_mortality.future_years` and `@disease_mortality.future_index` have been added to allow projections to incorporate deterministic future disease mortality.

4. A new subcommand `@print.eigenvalues` has been added.

5. New subcommands `@fishery.future_Fs` and `@fishery.future_Us` of have been added to allow projections with a fishing mortality rate rather than a defined catch.

6. CASAL now insists that file of free parameters is supplied when requesting yields (i.e., if the option `-Y` is chosen on the CASAL command line, then a file of free parameters must also be supplied using `-i [`*`filename`*`]`).

7. A new command line switch (`-Q`) has been added that suppresses all error messages and warnings, i.e., all standard error output.

8. Additional ogives have been added; `Richards` and `Richards_capped`.

9. CASAL now allows the use of the `growth_props` argument with mean size at age data. If fish growth occurs between birthdays (i.e., some element of `growth_props` is nonzero), then CASAL uses interpolation to fill in mean fish sizes in other time steps.

10. An error in CASAL that occurred when doing yield estimates with a disease mortality transition present has been repaired.

11. An error in combining MCMC results into a subsample with the command line –C has been repaired.

12. `@selectivity_at` observations can now be implemented as actual observations, not just pseudo observations.

13. An error in generating parametric bootstrap observations for non-sequential data series has been repaired. Plus, a new subcommand (`do_bootstrap`) has been added to each observation type that allows the user to specify what sets of observations to include/exclude when generating bootstraps. Note that bootstrapping has not been implemented for some observation types.

14. Error checks to ensure that the parameters for priors have been correctly specified have been implemented.

15. New partition members for tagging data have been added, along with associated parameters and the observation types `@tag_release` and `@tag_recapture`.

16. Occasionally the MCMC proposal distribution fails to find an acceptable starting point after many attempts (due, mostly, to scenarios where the set of parameter estimates are very close to bounds relative to their standard deviation). In this situation CASAL would continuously generate new candidates, but without success. CASAL now only makes 1000 attempts before returning an error message that it failed to find an acceptable starting point.

17. An error in the calculation of the lognormal prior (when applied to vectors) has been corrected.

18. CASAL can now incorporate annual stock assessment uncertainty in the calculation of CAY (see `@MCY_CAY.CAY_uncertainty_dist` on page 193). This is now the default behaviour.

15.4 Changes and enhancements from v2.06-2004/09/26 to v2.07-2005/08/21

1. An option has been introduced to choose the method of adjusting small variances in the proposal matrix for MCMCs. The new option allows changes to small variances in the covariance matrix, but without changing correlations between parameters (see Section 6.5).

2. The `@tag-recapture` observation has been modified to include the sample type `growth`.

3. Minor bug-fixes have been made to allow specification of multiple growth episodes in multi-stock size-based models.

4. CASAL now returns an error if you attempt to provide an observation with a size class outside the range of size classes defined in the partition.

5. CASAL now returns an error if you attempt to define the minimum age in the partition as age 0 and have an annual cycle that puts recruitment before spawning.

6. The add-on utility simCASAL has been created to assist with simple operating model/estimation model simulation experiments (but note that SimCASAL will only work with versions of CASAL later than v2.07-2004/12/06).

7. The methods for describing variability about the mean growth curve in an age based model have been made more flexible.

8. A new growth increment model has been added for use with size-based models (‘exponential’). See Section 5.4.5.

9. An error in the calculation of the initial biomasses when applying density dependent migrations has been repaired.

10. The definition of the exogenous shift parameter for selectivities has been modified to be estimable. In addition, migration events now allow optionally estimable annual variations.

11. Additional ogives have been added; `Hillary` and `Hillary_capped`.

12. An option for setting constant future catches, *F*s, or *U*s for projections has been added, allowing a single value of future catch, *F*, or *U* to be applied in all future years (i.e., `current+1` to `final`) in the projection period.

15.5 Changes and enhancements from v2.07-2005/08/21 to v2.20-2008/02/14

1. CASAL now does not rescale the curvature parameter, *b*, when calculating catchability coefficients (*q*'s) by max(*E*). However, for backwards compatibility, rescaling of the curvature parameter can still be achieved with the command `@q_type=scaled`. In addition, an error that occurred when attempting to estimate two *q*'s (with type='free') as the same has been fixed.

2. Additional 'U' shaped ogives have been added; `cosh` and `double_exponential`.

3. The growth transition matrix in a size based model can now be supplied directly, instead of using one of the growth equations. See Section 5.4.5 for detail.

4. An option to denote that the proportions-at-age or -size for tag release events where `release_type=free` are given as `log(props)` has been added. See Section 5.4.8.

5. An additional penalty, `ratio_qs_penalty`, has been introduced.

6. Variation in the variability about the mean growth curve in an age based model can now, optionally, be a function of the mean size at age, instead of a function of age.

7. An error in the calculation of the age-size likelihoods has been repaired, and the methods for calculating the 'random_at_size' and 'random_at_sex_and_size' likelihoods have been made more efficient in some instances.

8. CASAL's memory management has been tweaked to give better performance when minimising large memory models, and a number of additional error messages have been added to assist debugging.

9. A bug that caused CASAL to crash in size-based models with growth defined by sex and maturity status has been fixed.

10. CASAL can now output free parameter values to a text file (with the command line options `-O` and `-o`) when doing a likelihood profile. Note that if you use this option, CASAL will also carry out a `-r` run on the profile point estimates, and append that output to the standard output. In addition, profiling of an element of a vector or ogive parameter is now permitted.

11. A selectivity ogive can now be used with tag-release events where `release_type=deterministic`.

12. The compiler used to compile the Microsoft Windows version of CASAL has been changed to Microsoft Visual C++ (Microsoft Visual Studio 2005), resulting in a smaller executable and faster run times. The speed of CASAL on Microsoft Windows and Linux platforms (with similar hardware) is now equivalent. CASAL is also available as a 64-bit application on Linux platforms.

13. An option to allow semelparous mortality within models with maturity in the partition has been added.

14. An option to calculate SSBs from either females or males only within sexed models that have maturity in the partition has been added.

15. New options to allow modelling of discards have been added by the use of a retention ogive associated with a fishery. However, these have not yet been implemented for Baranov fisheries.

16. The bug that did not allow the option to report 'sexed' proportions mature as a pseudo observation has been fixed.

17. Some of the functions in CASAL have been modified to trap 'divide by zero' errors, including a possible divide by zero error that could occur with the multinomial likelihood.

18. A number of modifications have been made to the manual, including the descriptions of `y_enter`, `YCS`, and `true_YCS`, and the location of the index within the document. The changes should make these parts of the manual clearer.

19. It is now a fatal error to use the wrong value of `y_enter` in an age-based model.

20. The option to specify a different stock recruitment function (Section 5.4.2) and steepness parameter *h* in yield estimates from the estimation model have been removed (i.e., the recruitment subcommands in earlier versions of CASAL labelled `simulation_SR` and `simulation_steepness` have been deprecated).

21. An option to modify the randomised YCS used in a simulations and projections by an arbitrary multiplier has been added.

22. CASAL can no longer do least squares estimation (i.e., `estimator=least_squares` is now deprecated). Least squares estimation had not been implemented for all observations types and was rarely, if ever, used.

23. The CASAL end-user licence has been revised. CASAL is now freely available and the source code may be obtained on request.

24. When using the Francis parameterisation and where only a subset of YCS years were specified explicitly, CASAL could, in some circumstances, apply an incorrect value of the mean YCS to the YCS not specified. The error has been corrected.

25. CASAL now carries out better checks on the number of values in an input file (i.e., the file supplied with `casal -i`) to ensure the correct number of values on each line have been supplied.

15.6 Changes and enhancements from v2.20-2008/02/14 to v2.30-2012/03/21

1. Use of the command `@stock_names` is now deprecated for single stock models.

2. MCMC chains can now, optionally, be initialised using finite differences minimisation (casal –M) to generate the MCMC starting point and the covariance matrix for the proposal distribution.

3. A bug in the CASAL that caused intermittent crashes when simulating some data (i.e., when using `casal -s`) has been fixed.

4. The CASAL version number has been slightly modified to include the source code revision identification number in the output header line. The revision number always increases with changes to the underlying code and has a unique relationship with the date/time stamp. Note, however, that it will not necessarily be a sequential number.

5. A bug in the 32-bit Linux version of CASAL that resulted in the random number seed being reset for every set of draws when generating recruitments in projections has been repaired. Microsoft Windows, 32-bit and 64-bit Linux versions should now produce the same results when using the same random number seed.

6. The calculation of deterministic yields for weightless models has been deprecated.

7. Fixed an error where CASAL crashed with c.v.s of the growth curve were a function of length in models where the first age in the partition was not 1.

8. Added an option to allow annually varying migrations to be randomised in projections.

9. Added an option for size-based maturity ogives to be used in an age-based model where the age-size relationship does vary between years. Note that size-based maturity ogives cannot be used in age-based models where maturity is not in the partition.

10. In deterministic yields for non-Baranov fisheries, CASAL default values for `B_pre.selectivity` and `B_pre.mature_only` have been deprecated. These previously defaulted to a selectivity of constant value=1 over all ages and `mature_only=true`. The choice of `selectivity` and `mature_only` now need to be specified by the user.

11. Added an error check to ensure *q*'s have valid labels in the ratio *q*s penalty.

12. Added optional index range to the vector average penalty.

13. Added an optional command to exclude years from the range of free YCS that are used to average to one in the Haist parameterisation.

14. CASAL will now correctly report an error if using Baranov with multiple fisheries in the same time step in a single area model.

15. Corrected an error where growth was not being applied to all stocks in size-based models with multiple stocks.

16. Added optional arguments for the ogive smoothing and vector smoothing penalties to be calculated as the log of differences.

ACKNOWLEDGMENTS

We thank Andreas Griewank and the other creators of the ADOL-C package — the basis of the `Betadiff` minimisation software underlying CASAL — for freely supplying this product. Thanks also to Robert Davies for providing the `newran` random number generation library.

We thank Ian Doonan and Allan Hicks for help in the planning stages of the development of CASAL. Ralph Coburn, Neville Phillips, Peter Horn, Stuart Hanchet, Susan Kim, Paul Breen, and Rubén Roa-Ureta provided helpful discussions with the authors and constructive comments on the contents of the manual. Gavin Macaulay provided invaluable advice on the technical aspects of the C++ implementation as well as assisting in porting CASAL to MinGW. The authors would also like to thank Andrew Constable and Steve Candy (Australian Antarctic Division), and David Agnew and Richard Hillary (Imperial College) for constructive comments and suggestions for developments of CASAL. The CASAL logo and icon were designed by Erika Mackay. The development of CASAL was funded by NIWA.

REFERENCES

Bull, B.; Francis, R.I.C.C.; Dunn, A.; Gilbert, D.J. (2002). CASAL (C++ algorithmic stock assessment laboratory): CASAL user manual v1.02.2002/10/21. *NIWA Technical Report. 117.* 199 p.

Bull, B.; Francis, R.I.C.C.; Dunn, A.; McKenzie, A.; Gilbert, D.J.; Smith, M.H. (2003). CASAL (C++ algorithmic stock assessment laboratory): CASAL user manual v2.01-2003/08/01. *NIWA Technical Report. 124.* 223 p.

Bull, B.; Francis, R.I.C.C.; Dunn, A.; McKenzie, A.; Gilbert, D.J.; Smith, M.H. (2004). CASAL (C++ algorithmic stock assessment laboratory): CASAL user manual v2.06-2004/09/26. *NIWA Technical Report. 126.* 261 p.

Bull, B.; Francis, R.I.C.C.; Dunn, A.; McKenzie, A.; Gilbert, D.J.; Smith, M.H. (2005). CASAL (C++ algorithmic stock assessment laboratory): CASAL user manual v2.07-2005/08/21. *NIWA Technical Report. 127.* 272 p.

Bull, B.; Francis, R.I.C.C.; Dunn, A.; McKenzie, A.; Gilbert, D.J.; Smith, M.H.; Bian, R. (2008). CASAL (C++ algorithmic stock assessment laboratory): CASAL user manual v2.20-2008/02/14. *NIWA Technical Report. 130.* 276 p.

Bull, B.; Livingston, M.E. (2001). Links between climate variation and the year class strength of New Zealand hoki (*Macruronus novaezelandiae*): an update. *New Zealand Journal of Marine and Freshwater Research 35(5)*: 871–880.

Clark, W.G. (1991). Groundfish exploitation rates based on life history parameters. *Canadian Journal of Fisheries and Aquatic Sciences 48*: 734–750.

Davies, R.B. (1998). Newran02A. A random number generator library. URL http://www.robertnz.net/index.html.

Dennis Jr, J.E.; Schnabel, R.B. (1996). Numerical methods for unconstrained optimisation and nonlinear equations. Prentice Hall. 378 p.

Fournier, D.A. (1994). AUTODIF. A C++ array language extension with automatic differentiation for use in nonlinear modeling and statistics. Otter Research Ltd., Sidney, Canada. 123 p.

Fournier, D.A.; Sibert, J.R.; Majkowski, J.; Hampton, J. (1990). MULTIFAN: a likelihood-based method for estimating growth parameters and age composition from multiple length frequency data sets illustrated using data for southern bluefin tuna. *Canadian Journal of Fisheries and Aquatic Sciences 47*: 301–317.

Francis, R.I.C.C. (1988). Maximum likelihood estimation of growth and growth variability from tagging data. *New Zealand Journal of Marine and Freshwater Research 22(1)*: 43–51.

Francis, R.I.C.C. (1992). Recommendations concerning the calculation of Maximum Constant Yield (MCY) and Current Annual Yield (CAY). New Zealand Fisheries Assessment Research Document. 92/8. 23 p. MAF (Fisheries), Wellington. (Unpublished report held in NIWA library, Wellington.)

Francis, R.I.C.C.; Haist, V.; Bull, B. (2003). Assessment of hoki (*Macruronus novaezelandiae*) in 2002 using a new model. *New Zealand Fisheries Assessment Report. 2003/6.* 69 p.

Francis, R.I.C.C.; Horn, P.L. (1997). Transition zone in otoliths of orange roughy (*Hoplostethus atlanticus*) and its relationship to the onset of maturity. *Marine Biology 129*: 681–687.

Gelman, A.B.; Carlin, J.S.; Stern, H.S.; Rubin, D.B. (1995). Bayesian data analysis. Chapman and Hall, London. 526 p.

Gilks, W.R.; Richardson, S.; Spiegelhalter, D.J. (eds.) (1998). Markov chain Monte Carlo in practice. *Interdisciplinary statistics*. 399 p. Chapman and Hall/CRC Press, Boca Raton, Florida.

Griewank, A.; Juedes, D.; Mitev, H.; Utke, J.; Vogel, O.; Walther, A. (1996). ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software 22(2)*: 131–167. Algorithm 755.

Gulland, J.A.; Boerema, L.K. (1973). Scientific advice on catch levels. *Fishery Bulletin 71*: 325–335.

Harley, S.J.; Myers, R.A.; Dunn, A. (2001). Is catch-per-unit-effort proportional to abundance? *Canadian Journal of Fisheries and Aquatic Sciences 58(9)*: 1760–1772.

Hilborn, R.; Maunder, M.; Parma, A.; Ernst, B.; Payne, J.; Starr, P.J. (2001). Coleraine: A generalized age-structured stock assessment model. Users manual Version 2.0. School of Aquatic & Fishery Sciences, University of Washington. FRI-UW Report Series 0116. 58 p. University of Washington, Seattle WA

Otter Research Limited (2000). An introduction to AD model builder, version 4: For use in nonlinear modelling and statistics. 127 p. Sydney, B.C., Canada.

Payne, A.; Kirkwood, G.P.; Hillary, R.; Agnew, D.J. (2005). Age structured production model for toothfish at South Georgia. WG-FSA-SAM-05/18. 10 p. CCAMLR, Hobart. Unpublished manuscript presented to the Stock Assessment Methods Subgroup of the Fish Stock Assessment Working Group of CCAMLR

Punt, A.E.; Hilborn, R. (2001). BAYES-SA. Bayesian stock assessment methods in fisheries. User's manual. *FAO Computerized information series (fisheries) 12*. Food and Agriculture Organisation of the United Nations, Rome (Italy). 56 p.

R Development Core Team (2007). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Schnute, J. (1981). A versatile growth model with statistically stable parameters. *Canadian Journal of Fisheries and Aquatic Sciences 38(9)*: 1128–1140.

Smith, A.D.M.; Punt, A.E.; Wayte, S.E.; Starr, P.J.; Francis, R.I.C.C.; Stokes, T.K.; Hilborn, R.; Langley, A. (2002). Stock assessment of the northeast Chatham Rise orange roughy for 2001. *New Zealand Fisheries Assessment Report*. *2002/25*. 30 p.

Smith, B.J. (2001). Bayesian output analysis program. Version 1.00 user's manual. Unpublished manuscript. 45 p. University of Iowa College of Public Health. (*see* http://www.public-health.uiowa.edu/boa)

Starr, P.J.; Bentley, N.; Maunder, M. (1999). Assessment of the NSN and NSS stocks of red rock lobster (*Jasus edwardsii*) for 1998. New Zealand Fisheries Assessment Research

Document. 99/34. 45 p. Ministry of Fisheries, Wellington. (Unpublished report held by the Ministry of Fisheries, Wellington.)

Walters, C.J.; Ludwig, D. (1994). Calculation of Bayes posterior probability distributions for key population parameters. *Canadian Journal of Fisheries and Aquatic Sciences 51*: 713–722.

QUICK REFERENCE

Command line arguments

```
casal [-l] [-r] [-e] [-E] [-p] [-m] [-M] [-a number] [-C
      filelist
      -S outfile] [-s number prefix] [-v outfile] [-P outfile]
      [-Y] [-f prefix] [-F suffix] [-q] [-Q] [-i infile]
      [-I infile] [-O outfile] [-o outfile] [-g RNG_seed]
      [-n name]
```

`-l`	Display the CASAL end user licence.
`-r`	Run the population section and calculate the objective function.
`-e`	Calculate the point estimate of the parameters.
`-E`	Using finite differences instead of automatic differentiation.
`-p`	Calculate likelihood or posterior profiles.
`-m`	Use MCMC.
`-m`	Use MCMC using finite differences.
`-a [number]`	Recover MCMC run; continue; and append further results.
`-C [filelist]`	Concatenate the MCMC results files for the specified files
`-s [no. prefix]`	Generate simulated observations, with number of simulations and outfile prefix
`-v [outfile]`	Output the values of the output quantities.
`-P [outfile]`	Calculate projected outputs (use with `-i`).
`-Y`	Calculate yields (use with `-i`).

Optional command line arguments

`-f [prefix]`	Use a prefix on the names of the three input parameter files.
`-F [suffix]`	Replace the standard “csl” suffix used on the input parameter filenames with a user defined suffix.
`-q`	Run quietly, i.e., suppress output from population section.
`-Q`	Suppress all messages and warnings.
`-i [infile]`	Input free parameter values (use with `-r`, `-e`, `-p`, `-m`, `-M`, `-s`, `-v`, `-P`, or `-Y`).
`-I [infile]`	with `-m -i`, or `-M -i` input the covariance matrix used for MCMC from file.
`-O [outfile]`	Output a set of free parameter values to a text file (use with `-r`, `-e`, `-E`, or `-p`).
`-o [outfile]`	Output (with append) a set of free parameter values to a text file (use with `-r`, `-e`, `-E`, or `-p`).
`-S [outfile]`	(With `-C`) dump the posterior sub-sample into `outfile`.
`-g [RNG_seed]`	Set the random number generator seed.
`-n [name]`	(With `-m`, `-M` or `-a`) specify the name of the current machine.

Order of transitions within a time step

1. Ageing (in an age-based model)
2. Recruitment
3. Maturation
4. Migration
5. Growth
6. Mortality (natural and fishing)
7. Disease mortality

8. Tag release events
9. Tag shedding rate
10. Semelparous mortality

Available ogives

`constant`: Has the estimable parameter C; can be shifted; and can be used as a size-based ogive in an age-based model.

`knife_edge`: Has the non-estimable parameter E; cannot be shifted; and can be used as a size-based ogive in an age-based model.

`allvalues`: Has estimable parameters V_{low} V_{low+1} … V_{high}; cannot be shifted; and cannot be used as a size-based ogive in an age-based model.

`allvalues_bounded`: Has non-estimable parameters L and H, and estimable parameters are V_L V_{L+1} … V_H; cannot be shifted; and cannot be used as a size-based ogive in an age-based model.

`logistic`: Has estimable parameters a_{50} and a_{to95}; can be shifted; and can be used as a size-based ogive in an age-based model.

`logistic_capped`: Has estimable parameters a_{50}, a_{to95}, and $a_{\max}$,; can be shifted; and can be used as a size-based ogive in an age-based model.

`logistic_bounded`: Has estimable parameters a_{50} and a_{to95}; can be shifted; and can be used as a size-based ogive in an age-based model.

`double_logistic`: Has estimable parameters a_{50}, $a_{\mathrm{to}95}$, b_{50}, b_{to95}, and $a_{\max}$; can be shifted; and can be used as a size-based ogive in an age-based model.

`logistic_product`: Has estimable parameters a_{50}, $a_{\mathrm{to}95}$, b_{50}, b_{to95}, and $a_{\max}$; can be shifted; and can be used as a size-based ogive in an age-based model.

`Richards`: Has estimable parameters a_{50}, a_{to95}, and asymmetry parameter δ; can be shifted; and can be used as a size-based ogive in an age-based model.

`Richards_capped`: Has estimable parameters a_{50}, a_{to95}, asymmetry parameter δ, and a_{max}; can be shifted; and can be used as a size-based ogive in an age-based model.

`double_normal`: Has estimable parameters a_1, s_L, and s_R; can be shifted; and can be used as a size-based ogive in an age-based model.

`double_normal_capped`: Has estimable parameters a_1, s_L, s_R, and $a_{\max}$; can be shifted; and can be used as a size-based ogive in an age-based model.

`double_normal_plateau`: Has estimable parameters a_1, a_2, s_L, s_R, and $a_{\max}$; can be shifted; and can be used as a size-based ogive in an age-based model.

`double_normal_coleraine`: Has estimable parameters a_1, σ_L^2, and σ_R^2; can be shifted; and can be used as a size-based ogive in an age-based model

`logistic_producing`: Has the non-estimable parameters L and H, and has estimable parameters a_{50} and a_{to95}; cannot be shifted; and cannot be size-based.

`increasing`: Has non-estimable parameters L and H, and estimable parameters π_L π_{L+1} … π_H; cannot be shifted; and cannot be used as a size-based ogive in an age-based model.

`increasing_capped`: Has non-estimable parameters L, H, and C, and estimable parameters π_L π_{L+1} … π_{H-1}; cannot be shifted; and cannot be used as a size-based ogive in an age-based model.

`Hillary`: Has estimable parameters α, ψ, and γ; can be shifted; and can be used as a size-based ogive in an age-based model.

`Hillary_capped`: Has estimable parameters α, ψ, γ, and $a_{\max}$; can be shifted; and can be used as a size-based ogive in an age-based model.

`cosh`: Has estimable parameters y, $x_{\min}$, α, and β; can be shifted; and can be used as a size-based ogive in an age-based model.

`double_exponential`: Has non-estimable parameters x_1 and x_2, and estimable parameters x_0, y_0, y_1, and y_2. It cannot be shifted, but can be used as a size-based ogive in an age-based model.

Available penalties

`ogive_smoothing_penalty`: Sum of squares of rth differences applied to an `allvalues` or `allvalues_bounded` ogive parameter.

`catch_limit_penalty`: Sum of squares of (actual catch less specified catch), optionally on a log scale, for a single fishery.

`vector_average_penalty`: Square of (mean(vector)−k), or of (mean(log(vector))−l), or of (log(mean(vector)/m)) applied to a vector parameter

`vector_smoothing_penalty`: Sum of squares of rth differences applied to elements of a vector.

`element_difference_penalty`: Square of (vector_1[i]−vector_2[i]) applied to two vector parameters.

`YCS_difference_penalty`: Squared difference between the YCS values for a given year in the two stocks (for a two stock model).

`similar_qs_penalty`: Square of ($\log(q_i)-\log(q_j)$) applied to two catchability coefficients q

`similar_qs_penalty`: Lognormal density penalty applied to the ratio of two catchability coefficients q.

`ogive_comparison_penalty`: Sum of squares of max($\text{ogive}_1-\text{ogive}_2$, 0) applied to two ogive parameters.

`ogive_difference_penalty`: Square of ($\text{ogive}_1-\text{ogive}_2$) for a single size or age class, applied to two ogive parameters.

`fish_tagged_penalty`: Square of (number of fish actually tagged less number of fish meant to be tagged).

List of commands and sub-commands in the `population.csl` data input file

Defining the partition

@size_based	Should the model be size-based rather than age-based?
@n_classes	Number of size classes
@class_mins	Size class lower limits (plus the upper limit of the last class if it is not a plus group)
@min_age, @max_age	Minimum and maximum age limits
@plus_group	Should a plus age or size group be used?
@plus_group_size	Mean size of plus group
@sex_partition	Is the partition sex-structured?
@mature_partition	Is the partition structured by maturity?
@n_areas	Number of areas in the partition
@area_names	Area names
@n_stocks	Number of stocks in the partition
@stock_names	Stock names
@n_growthpaths	Number of growth paths in the partition
@exclusions_char1	Partition exclusion term 1
@exclusions_val1	Partition exclusion value 1
@exclusions_char2	Partition exclusion term 2
@exclusions_val2	Partition exclusion value 2
@n_tags	Number of tagging partitions to include in the model
@tag_names	Names of the tagging partition members
@tag_shedding_rate	The tag shedding rate to apply to the tagging partitions in the model
@tag_loss_props	Proportion of tag loss that has occurred by each time step

Defining the annual cycle and the time sequence

@initial	Initial assessment year
@current	Current assessment year
@final	Final projection year
@annual_cycle	Annual cycle block command
time_steps	Number of time steps
recruitment_time	Time step in which recruitment occurs
recruitment_areas	Area in which where recruitment occurs, for each stock
spawning_time	Time step for recording SSB
spawning_part_mort	Proportion of mortality in the time step before recording SSB
spawning_areas	Area for recording SSB, for each stock
spawning_all_areas	Is SSB recorded for all areas combined?
spawning_ps	Spawning proportions by age/size class
spawning_p	Spawning proportion
spawning_sex	The sexes to include in the calculation of spawning stock biomass
spawning_use_total_B	Should SSB be defined as total biomass rather than mature biomass?
semelparous_time	Time step to apply semelparous mortality
semelparous_mortality	Proportion of mature fish assumed to die when applying semelparous mortality
n_growths	Number of growth episodes per year
growth_times	Time step in which each growth episode occurs
ageing_time	Time step when age is incremented
growth_props	Proportion of growth that has occurred by each time step
M_props	Proportion of natural mortality that occurs in each time step

baranov	Should fishing mortality be applied simultaneously with natural mortality using the Baranov equation, rather than instantaneously?
midmortality_partition	Method to calculate mortality within the time step
fishery_names	Names of the fishery
fishery_times	Time step when each fishery occurs
fishery_areas	Area where each fishery occurs
n_migrations	Number of migrations in each year
migration_names	Names of the fishery
migration_times	Time step of each migration
migrate_from	Area from which each migration departs
migrate_to	Area where each migration arrives
n_maturations	Number of maturation episodes in each year
maturation_times	Time step of each maturation episode
disease_mortality_time	Time step to apply disease mortality

Defining recruitment

@y_enter	Number of years after its birth that a year class enters the partition
@standardise_YCS	Use the Haist parameterisation for YCS?
@use_mean_YCS	Use the Francis parameterisation for YCS?
@recruitment	Recruitment block command
YCS	Year class strengths for the stock
YCS_years	Years for which YCS are provided
n_rinitial	Number of years for which Rinitial is to be used as the YCS
SR	Stock-recruitment relationship
steepness	Steepness parameter of the stock-recruitment relationship
CR	Climate-recruitment relationship
Ts	Climate variable T
Ts_years	Years for which the climate variable T is provided
CR_alpha, CR_beta	Climate-recruitment parameters alpha and beta
CR_beta2	Climate-recruitment parameter beta2
CR_p	Climate-recruitment parameter p
initial_size_mean	Mean size at recruitment (both sexes)
initial_size_cv	C.v. of size at recruitment (both sexes)
initial_size_mean_male	Mean size at recruitment (male)
initial_size_mean_female	Mean size at recruitment (female)
initial_size_cv_male	C.v. of size at recruitment (male)
initial_size_cv_female	C.v. of size at recruitment (female)
p_male	Proportion of recruits that are male
growthpaths	Proportion of recruits on each growth path
first_free, last_free	Range of YCS defining R_0, with the Haist or Francis YCS parameterisation
exclude_free	Exclude years from the range of free YCS defining R0 with the Haist YCS parameterisation

Defining recruitment variability

@randomisation_method	Randomisation method for recruitment variability in stochastic simulations and projections
@first_random_year	For projections, the first year for which YCS are randomised
@recruitment sub-commands that define recruitment variability	
sigma_r	Standard deviation on the log scale of randomised YCS
T_sigma_r	Standard deviation on the log scale of randomised climate data T
rho	Lag-1 log-scale autocorrelation of randomised YCS
T_rho	Lag-1 log-scale autocorrelation of randomised climate data T
year_range	Year range from which randomised YCS are resampled
T_year_range	Year range from which randomised climate data T are resampled
T_mean	Mean on the linear scale of randomised climate data T
recruitment_multipler	Multiply the randomised YCS by an arbitrary multiplier

Defining growth (in a size based model)

@growth	Growth block command
stock	Stock that the growth episode applies to
type	Growth model used by the growth episode
g	Reference growths for the growth model
l	Reference sizes for the growth model
cv	C.v. for the growth model
minsigma	Lower bound on sigma for the growth model
g_male, g_female	Reference growths for the growth model
l_male, l_female	Reference sizes for the growth model
cv_male, cv_female	C.v. for the growth model
minsigma_male, minsigma_female	Lower bound on sigma for the growth model
g_mature, g_immature	Reference growths for the growth model
l_mature, l_immature	Reference sizes for the growth model
cv_mature, cv_immature	C.v. for the growth model
minsigma_mature	
minsigma_immature	Lower bound on sigma for mature/immature
g_male_mature, etc.	Reference growths for the growth model
l_male_mature, etc.	Reference sizes for the growth model
cv_male_mature, etc.	C.v. for the growth model
minsigma_male_mature, etc.	Lower bound on sigma
[size_class]	Row of the matrix for a user supplied growth transition matrix.

Defining maturation (when maturity is in the partition)

@maturation	Maturation block command
stock	Stock that the maturation episode applies to
area	Area that the maturation episode applies to
rates_all	Rates of maturation by age or size class
rates_male, rates_female	Rates of maturation by sex and age or size class

Defining maturity (when maturity is not in the partition)

@maturity_props	Maturity proportion block command
all	Maturity proportions by age/size class
male, female	Maturity proportions by sex and age/size class

Defining migrations

@migration	Migration block command
stock	Stock that migrates
migrators	Whether mature, immature, or both kinds of fish migrate
prop	Proportion of applicable fish that migrate
rates_all	Proportion of applicable fish that migrate, by age/size class
rates_male, rates_female	Proportion of applicable fish that migrate, by sex and age/size class
annual_variation_years	What are the years to apply an annual variation to the migration rates?
annual_variation_values	What are the annual variation values to apply to the migration rates?
annual_variation_year_range	Year range from which randomised annual variation values are resampled
annual_variation_randomisation_method	Randomisation method for migration annual variation in stochastic simulations and projects
S, D	Source and destination density-dependence parameters
wave	Is this one wave of a 2-wave migration. If so, is it the first or second?
pwave	The proportion of fish in the first wave of a 2-wave migration

Defining natural mortality

@natural_mortality	Natural mortality block command
all	The overall natural mortality rate
male, female	The male and female natural mortality rates

avg, diff	The male/female average and male-female difference in natural mortality rates
mature, immature	The mature and immature natural mortality rates
male_mature, etc.	Natural mortality rates by sex and maturity
ogive_all	The overall natural mortality rate as an ogive
ogive_male, ogive_female	The male and female natural mortality rates as ogives
ogive_avg, ogive_diff	The male/female average and male-female difference in natural mortality rates, as ogives
ogive_mature, ogive_immature	The mature and immature natural mortality rates, as ogives
ogive_male_mature, etc.	Natural mortality rates by sex and maturity, as ogives
Allow_negative_M	Allow values of natural mortality that are negative

Defining fishing mortality

@fishery	Fishery block command
catches	Catches by year
years	Years for which catches are provided
selectivity	Name of the selectivity to use
retention_selectivity	Name of the ogive to use for retention if discards are allowed
F_max	Maximum fishing pressure (Baranov mortality)
U_max	Maximum fishing pressure (instantaneous mortality)
Fs	Instantaneous mortality *F* by year
Fs_years	Years for which *F*'s are provided
future_catches	Catches by year in the projection period
future_Fs	Baranov fishing mortality by year in the projection period
future_Us	Instantaneous fishing mortality by year in the projection period
future_years	Years for which catches or mortalities are provided in the projection period
future_constant_catches	The constant catch applied in every year in the projection period
future_constant_Fs	The constant Baranov mortality rate applied in every year in the projection period
future_constant_Us	The constant instantaneous mortality rate applied in every year in the projection period

Defining disease mortality

@disease_mortality	Disease mortality block command
DM	Disease mortality rate
selectivity	The selectivity ogive
years	Years to apply the disease mortality
index	Relative value of the disease mortality by year
future_index	Relative value of the disease mortality in the projection period
future_years	Years for which future disease mortality is provided

Defining tag-release events

@tag	Label for the tagging event
tag_name	The name of the tagging partition member
release_type	Method for determining the proportions-at-age release in an age-based model
area	Area that the tagging event applies to
stock	Stock that the tagging event applies to
sex	Sex that the tagging event applies to
year	Year of the tagging event
step	Time step of the tagging event
mature_only	Does the tagging event apply to mature or all fish?
ogive	Which selectivity ogive should be applied?
number	The number of fish actually tagged in the tagging event
class_mins	What are the size bins of the observations (in an age-based model)?
plus_group	Is the last age or size class a plus group?
log_props	Are the `props_all`, `props_male`, or `props_female` relative rates specified in log space?

props_all	Relative rates of all fish tagged in the tagging event
props_male, props_female	Relative proportions of male or female fish tagged in the tagging event
mortality	Proportion of tagged fish that are removed immediately after tagging

Defining selectivities

@selectivity_names	List of selectivity names
@selectivity	Selectivity block command
all	The selectivity ogive
male, female	The selectivity ogives by sex
mature, immature	The selectivity ogives by maturity
male_mature, etc.	The selectivity ogives by sex and maturity
shift_E	Exogenous selectivity shift variable E
shift_years	Years for which exogenous selectivity shift variable E is provided
shift_a	Exogenous selectivity shift parameter a

Setting the initial state

@n_equilibrium	Number of years of running the equilibrium model
@Rinitial_is_deviate	Is $R_{initial}$ supplied relative to R_0?
@initialization	Initialization block command
B0	Equilibrium abundance B_0
R0	Equilibrium recruitment R_0
Bmean	Equilibrium abundance B_{mean} corresponding to R_{mean}
Rmean	Expected recruitment in any year R_{mean}
Binitial	Initial abundance $B_{initial}$
Rinitial	Initial recruitment $R_{initial}$
Cinitial	Initial number in each age/size class $C_{initial}$
Cinitial_male, Cinitial_female	Initial number in each age/size class $C_{initial}$ for each sex
@B0_total, @log_B0_total, @R0_total, @log_R0_total, @B0_prop_stock1, @R0_prop_stock1	Two stock model options for specifying initial state

Defining ogive preferences

@n_quant	Number of points at which to evaluate size-based ogives in an age-based model

Defining size-at-age

@size_at_age_type	Size-at-age model type
@size_at_age_years	Years for which mean-size-at-age data are provided
@size_at_age_miss	Controls the treatment of years for which mean-size-at-age data are not provided
@size_at_age_step	Time step for which size-at-age data are provided
@size_at_age_dist	Distribution of sizes-at-age around the mean
@size_at_age	Size-at-age block command
k, t0, Linf	von Bertalanffy parameters
k_male, t0_male, Linf_male, k_female, t0_female, Linf_female	von Bertalanffy parameters by sex
y1, y2, tau1, tau2, a, b	Schnute parameters
y1_male, y2_male, tau1_male, tau2_male, a_male, b_male, y1_female, y2_female, tau1_female, tau2_female, a_female, b_female	Schnute parameters by sex
male_[year], female_[year]	Mean-size-at-age of male and female fish in [year]
all_[year]	Mean-size-at-age of fish of both sexes in [year]
cv	c.v. of sizes-at-age around the mean
cv_male, cv_female	c.v. of sizes-at-age around the mean, by sex
cv1, cv2	c.v. of sizes-at-age around the mean for the minimum and maximum age class in the partition
cv1_male, cv2_male, cv1_female, cv2_female	c.v. of sizes-at-age around the mean, by sex, for the minimum and maximum age class in the partition

sd1, sd2	c.v. of sizes-at-age around the mean for the minimum and maximum age class in the partition
sd1_male, sd2_male, sd1_female, sd2_female	s.d. of sizes-at-age around the mean, by sex, for the minimum and maximum age class in the partition
by_length	Specifies if the linear interpolation of c.v.s or s.d.s is a linear function of mean size or of age.
@annual_growths	Use annual growth variation: amount of an average year's growth that occurs in each year
@annual_growth_years	Years for which annual growths are provided

Defining the size-weight relationship

@size_weight	Size-weight block command
type	The size-weight relationship function
a, b	The size-weight parameters *a* and *b*
a_male, b_male, a_female, b_female	The size-weight parameters *a* and *b*, by sex
verify_size_weight	Verify the supplied size-weight relationship and units
@weightless_model	Is this a model which does not involve fish weight?

Other commands

@comment	User supplied comment

List of commands and sub-commands in the `estimation.csl` data input file

Defining the estimation method

@estimator	Choice of estimation method

Defining point estimation

@max_iters	Maximum number of iterations in the minimiser
@max_evals	Maximum number of evaluations in the minimiser
@max_iters_intermediate	Maximum number of iterations in early phases
@max_evals_intermediate	Maximum number of evaluations in early phases
@grad_tol	Minimiser convergence threshold

Defining likelihood or posterior profiling

@profile	Profile block command
element	Element of the ogive or vector parameter to be profiled
parameter	Name of the parameter to be profiled
n	Number of values at which to profile the parameter
l, u	Range of values at which to profile the parameter

Defining MCMC

@MCMC	MCMC block command
start	Covariance multiplier for the starting point of the Markov chain
length	Length of the Markov chain
keep	Spacing between recorded values in the chain
max_cor	Maximum absolute correlation in the covariance matrix of the proposal distribution
covariance_adjustment	Method for adjusting small nonzero variances in the covariance matrix of the proposal distribution
min_diff	Minimum nonzero variance times the range of the bounds in the covariance matrix of the proposal distribution
stepsize	Initial stepsize (as a multiplier of the approximate covariance matrix)
adaptive_stepsize	Should the MCMC stepsize be altered during the chain?
adapt_at	At which iteration numbers can the MCMC stepsize be altered?
proposal_t	Should the proposal distribution be multivariate *t*?
df	Degrees of freedom of the multivariate *t* proposal distribution.
burn_in	Number of samples to be discarded for the burn-in period

subsample_size	Size of the sub-sample to be generated
systematic	Should sub-sampling be systematic?
prior_reweighting	Should the sub-sample be generated using prior reweighting?
adaptive_covariance	Should the MCMC covariance matrix be altered during the chain?
adapt_covariance_at	At which iteration numbers can the MCMC covariance matrix be altered?
adaptive_covariance_discard	If the MCMC covariance matrix is altered during the chain, how many observations should be discarded from the start of the chain when taking a subsample for estimating the new covariance matrix?
adaptive_covariance_transitions	If the MCMC covariance matrix is altered during the chain, how many transitions must occur in the part of the chain used to estimate the new covariance matrix?
adaptive_covariance_stepsize	If the MCMC covariance matrix is altered during the chain, what is the new stepsize value?
@trivariate_normal_test	Test MCMC algorithm with a simple trivariate normal example

Defining the free parameters and priors

@estimate	Free parameter block command
parameter	Name of the parameter to be estimated
same	Names of the other parameters which are constrained to have the same value
phase	Phase at which this parameter should be estimated, in point estimation
lower_bound, upper_bound	Bounds on this scalar parameter
lower_bound, upper_bound	Bounds on this vector parameter
MCMC_fixed	Should this parameter be fixed during MCMC?
prior	What type of prior does this parameter have?
mu, cv	What are the mean and c.v. of this normal or lognormal prior on a scalar parameter?
mu, stdev	What are the mean and standard deviation of this normal-by-standard deviation or beta prior on a scalar parameter?
A, B	What are the lower and upper values for the range parameters of the beta prior?
m, s	What are the mean and standard deviation of the log of this scalar parameter, under the normal-log prior?
mu, cv	What are the mean and c.v. of each element of this normal or lognormal prior on a vector parameter?
mu, stdev	What are the mean and standard deviation of each element of this normal-by-standard deviation or beta prior on a vector parameter?
A, B	What are the lower and upper values for each element of the range parameters of the beta prior?
m, s	What are the mean and standard deviation of each element of the log of this vector parameter, under the normal-log prior?
mu, cv, rho	What are the mean, c.v., and ρ of this normal-AR prior on a vector parameter?
m, s, r	What are the log-scale mean, standard deviation, and ρ of this normal-log-AR prior on a vector parameter?
s, r	What are the log-scale standard deviation and ρ of this normal-log-mean1-AR prior on a vector parameter?

Defining the catchability coefficients *q*

@q_method	Method used for catchability coefficients q
@q_type	Type of definition for catchability coefficient q
@q	Catchability coefficients q block command
q	Value of the *q* parameter
b	Curvature parameter b associated with the *q*

Defining the observations

@abundance	Absolute abundance block command
@relative_abundance	Relative abundance block command
years	Years of the time series
step	Time step in which the observations occur
proportion_mortality	Proportion of the step's mortality, after which the observations occur
area	Area in which the observations occur
q	Catchability coefficients *q* to use
curvature	Should a curvature parameter be used?
biomass	Are the observations biomass rather than numbers of fish?
ogive	Which selectivity ogive should be applied?
[year]	Abundance for [year]
mature_only	Do these observations include mature fish only?
stock	Which stock do these observations relate to?
all_areas	Do these observations cover all areas in the model?
do_bootstrap	Print out parametric bootstraps for this observation
@numbers_at	Numbers_at block command
@relative_numbers_at	Relative numbers-at block command
@proportions_at	Proportions-at block command
years	Years of the time series
step	Time step in which the observations occur
proportion_mortality	Proportion of the step's mortality after which the observations occur
at_size	Are the observations by size?
sexed	Are the observations sexed?
area	Area in which the observations occur
q	Catchability coefficients q to use
ogive	Which selectivity ogive should be applied?
class_mins	What are the size bins of the observations (in an age-based model)?
class_nums	What are the class numbers for the observations (in a size-based model)?
class_nums_male, class_nums_female	What are the class numbers for the sexed observations (in a size-based model)?
plus_group	Is the last age or size class a plus group?
min_class, max_class	Which age/size classes are covered by the observations?
sum_to_one	Should the proportions sum to 1?
ageing_error	Should ageing error be applied to these observations?
[year]	Numbers or proportions for [year]
do_bootstrap	Print out parametric bootstraps for this observation
@catch_at	Catch_at block command
years	Years of the time series
fishery	Fishery or fisheries covered by the observations
at_size	Are the observations by size?
sexed	Are the observations sexed?
class_mins	What are the size bins of the observations (in an age-based model)?
class_nums	What are the class numbers of the observations (in a size-based model)?
class_nums_male, class_nums_female	What are the class numbers of the sexed observations (in a size-based model)?
plus_group	Is the last age or size class a plus group?
min_class, max_class	Which age/size classes are covered by the observations?
sum_to_one	Should the proportions sum to 1?
ageing_error	Should ageing error be applied to these observations?
[year]	Numbers or proportions for [year]
do_bootstrap	Print out parametric bootstraps for this observation

@proportions_mature	Proportions_mature block command
years	Years of the time series
step	Time step in which the observations occur
proportion_mortality	Proportion of the step's mortality, prior to when the observations occur
sexed	Are these observations sexed?
females_only	Are these observations for females only?
at_size	Are the observations by size?
area	Area in which the observations occur
ogive	Which selectivity ogive should be applied?
class_mins	What are the size bins of the observations (in an age-based model)?
plus_group	Is the last age or size class a plus group?
min_class, max_class	Which age/size classes are covered by the observations?
ageing_error	Should ageing error be applied to these observations?
[year]	Proportions mature for [year]
do_bootstrap	Print out parametric bootstraps for this observation
@proportions_migrating	Proportions_migrating block command
years	Years of the time series
migration	Migration to which the observations apply
sex	Which sex do the observations apply to?
at_size	Are the observations by size?
area	Area in which the observations occur
ogive	Which selectivity ogive should be applied?
class_mins	What are the size bins of the observations (in an age-based model)?
plus_group	Is the last age or size class a plus group?
min_class, max_class	Which age/size classes are covered by the observations?
ageing_error	Should ageing error be applied to these observations?
[year]	Proportions migrating for [year]
do_bootstrap	Print out parametric bootstraps for this observation
@age_size	Age_size block command
year	Year in which the data were collected
step	Time step in which the data were collected
proportion_mortality	Proportion of the step's mortality, prior to when the observations occur
area	Area in which the observations occur
stock	Stock for which the data were collected
sample	Sampling method under which the observations were generated
ogive	Which selectivity ogive should be applied?
ageing_error	Should ageing error be applied to these observations?
ages	Age data
sizes	Size data
sexes	Sex data
do_bootstrap	Print out parametric bootstraps for this observation
@age_at_maturation	Age_at_maturation block command
sexed	Are these observations sexed?
sampled_ages	What were the estimated ages of these fish at sampling?
maturation_ages	What were the estimated ages of these fish at maturation?
sexes	What were the sexes of these fish?
stock	Which stock do these observations relate to?
ageing_error	Should ageing error be applied to these observations?
k	After how many years can maturation be detected?
do_bootstrap	Print out parametric bootstraps for this observation
@selectivity_at	Selectivity-at block command
ogive	Which selectivity should be applied?
years	Years of the time series
step	Time step in which the observations occur
proportion_mortality	Proportion of the step's mortality after which the observations occur

sexed	Are the observations sexed?
area	Area in which the observations occur
mature_only	Do these observations include mature fish only?
stock	Which stock do these observations relate to?
min_class, max_class	Which age/size classes are covered by the observations?
[year]	Numbers or proportions for [year]
do_bootstrap	Print out parametric bootstraps for this observation
@tag_release	
tag_label	The label of the tagging event
class_mins	What are the size bins of the observations (in an age-based model)?
plus_group	Is the last size class a plus group?
props_all	Relative rates of all fish tagged in the tagging event
props_male, props_female	Relative proportions of male or female fish tagged in the tagging event
do_bootstrap	Print out parametric bootstraps for this observation
@tag_recapture	Tag_recapture block command
tag_name	Tag which is being checked for
sample	Sampling method for the tag-recapture process
detection_probability	What proportion of tags are detected in the recapture process?
years	Years of the time series
step	Time step in which the observations occur
proportion_mortality	Proportion of the step's mortality, prior to when the observations occur
area	Area in which the observations occur
class_mins	What are the size bins of the observations (in an age-based model)?
ogive	Which selectivity ogive should be applied?
plus_group	Is the last age or size class a plus group?
ageing_error	Should ageing error be applied to these observations?
recaptured_[year]	Number of fish recaptured
recaptured_male[year], recaptured_female[year]	Number of fish recaptured
scanned_[year]	Number of fish recaptured
scanned_male[year], scanned_female[year]	Number of fish recaptured
do_bootstrap	Print out parametric bootstraps for this observation
r	The robustification parameter used in the binomial likelihood
dispersion	Dispersion term for the tag-recapture likelihood

Defining the objective function associated with the observations

dist	Likelihood of the observations
r	Robustifying constant
cv	C.v. for all observations in this time series, used with likelihoods
cv_[year]	C.v.s by year for this time series, used with likelihoods
cvs_[year]	C.v.s by observation and year for this time series, used with likelihoods
cv_process_error	Process error c.v. for this time series, used with likelihoods parameterised by the c.v.
stdev	Standard deviation for all observations in this time series, used with normal-by-standard deviation likelihoods
stdev_[year]	Standard deviation by year for this time series, used with normal-by-standard deviation likelihoods
stdevs_[year]	Standard deviation by observation and year for this time series, used with normal-by-standard deviation likelihoods
stdev_process_error	Process error standard deviation for this time series, used with likelihoods parameterised by the standard deviation
N	*N* for all years in this time series, used with proportions likelihoods

N_[year]	*N*'s by year for this time series, used with proportions likelihoods
Ns_[year]	*N*'s by observation and year for this time series, used with proportions likelihoods
N_process_error	Process error *N* for this time series, used with likelihoods parameterised by the effective sample size.

Defining the penalties

@ogive_smoothing_penalty	Ogive smoothing penalty block command
label	The name of the penalty
ogive	The name of the ogive parameter to which the penalty is applied
r	Penalty is applied to rth differences
log_scale	Should sums of squares be calculated on the log scale?
lower_bound, upper_bound	Penalty is applied for age or size classes lower_bound to upper_bound
multiplier	Multiply the penalty by this factor
@catch_limit_penalty	Catch limit penalty block command
label	The name of the penalty
fishery	The label of the fishery to which the penalty is applied
log_scale	Should sums of squares be calculated on the log scale?
multiplier	Multiply the penalty by this factor
@vector_average_penalty	Vector average penalty block command
label	The name of the penalty
vector	The name of the vector parameter to which the penalty is applied
k, l, m	Vector should average arithmetically to *k* or *m*, or geometrically to l.
multiplier	Multiply the penalty by this factor
@vector_smoothing_penalty	Vector smoothing penalty block command
label	The name of the penalty
ogive	The name of the vector parameter to which the penalty is applied
r	Penalty is applied to rth differences
log_scale	Should sums of squares be calculated on the log scale?
lower_bound, upper_bound	Penalty is applied for index lower_bound to upper_bound
multiplier	Multiply the penalty by this factor
@element_difference_penalty	Element difference penalty block command
label	The name of the penalty
vector1, vector2	The name of the vector parameters to which the penalty is applied
i	Penalise differences in the ith elements of the two vectors
multiplier	Multiply the penalty by this factor
@YCS_difference_penalty	YCS difference penalty block command
label	The name of the penalty
stock1, stock2	The names of the stocks to which the penalty is applied
year	Year for which the penalty is to be applied
multiplier	Multiply the penalty by this factor
@similar_qs_penalty	Similar *q*'s penalty block command
label	The name of the penalty
q1, q2	The names of the two *q*'s to which the penalty is applied
multiplier	Multiply the penalty by this factor
@ratio_qs_penalty	Ratio *q*'s penalty block command
label	The name of the penalty
q1, q2	The names of the two *q*'s to which the penalty is applied
mu	Mean of the lognormal for the penalty on q_1/q_2
cv	C.v. of the lognormal for the penalty on q_1/q_2
@ogive_comparison_penalty	Ogive comparison penalty block command
label	The name of the penalty
ogive1, ogive2	The name of the ogive parameters to which the penalty is applied

lower_bound, upper_bound	Penalty is applied for age or size classes lower_bound to upper_bound
multiplier	Multiply the penalty by this factor
@ogive_difference_penalty	Ogive difference penalty block command
label	The name of the penalty
ogive1, ogive2	The name of the ogive parameters to which the penalty is applied
class	The age, or size class number, to which the penalty is applied
multiplier	Multiply the penalty by this factor
@fish_tagged_penalty	'Fish must be tagged' penalty block command
label	The name of the penalty
tagging_episode	The name of the tagging episode
multiplier	Multiply the penalty by this factor

Defining the ageing error

@ageing_error	Ageing error block command
type	Type of ageing error model
p1, p2, k	Parameters of the "off_by_one" ageing error model
c	Parameter of the "normal" ageing error model
[age]	Row of the misclassification matrix for the "misclassification_matrix" ageing error model.

CASAL extensions

@user_parameterisation	Is there a reparameterisation of the population section using `user.parameterisation.cpp`?
@user_components	Lists the names of the user-defined priors or penalties calculated in `user.prior_penalty.cpp`

Other commands

@comment	User supplied comment

List of commands and sub-commands in the `output.csl` data input file

Defining the printouts

@print	Printouts block command
parameters	Print the population, estimation, and output parameters?
unused_parameters	Print a list of the parameters that were never used?
population_section	Print a description of the population section?
requests, results	Print a description of the requests sent to the population section and the corresponding results?
initial_state, state_annually, state_every_step, final_state	Print the state of the population?
estimation_section	Print a description of the estimation section?
every_mean_size	Print mean-sizes-at-age for every time step of every year?
fits, resids, pearson_resids, normalised_resids	Print the fits, residuals, and standardised residuals?
covariance	Print the approximate covariance matrix of the free parameters
yields	Print a description of the yield calculations?
fits_every_eval, objective_every_eval, parameters_every_eval, parameter_vector_every_eval	Print the fits, objective function, or parameters at every function evaluation?
@print_sizebased_ogives_at	Sizes for which size-based ogives should be printed in an age-based model

Defining the output quantities

@quantities	Output quantities block command
all_free_parameters	Output quantities include all free parameters?

scalar_parameters, vector_parameters, ogive_parameters	Output quantities include these parameters
ogive_arguments	Output quantities include these ogive arguments
nuisance_qs	Output quantities include nuisance q's?
B0, R0, Bmean, Rmean Binitial, Rinitial	Output quantities include B0, R0, Bmean, Rmean, Binitial, Rinitial?
SSBs	Output quantities include SSBs?
actual_catches, actual_catches_by_stock	Output quantities include actual catches (by stock)?
removals	Output quantities include removals?
discards	Output quantities include discards?
recruitments	Output quantities include recruitments?
YCS	Output quantities include YCS?
true_YCS	Output quantities include 'true YCS'?
Ts	Output quantities include climate variable T?
migration_annual_variation	Output quantities include migration_annual_variation
fishing_pressures	Output quantities include fishing pressures?
stock_crash	Output quantities include 'stock crash'?
disease_biomass_loss	Output quantities include the disease biomass loss resulting from a disease event?
tagged_age_distribution	Output quantities include the age breakdown of tagged fish in each tagging event?
fits	Output quantities include the 'fits' or the expected value for each observation?
resids	Output quantities include the 'residuals' or the observed less expected value for each observation?
pearson_resids	Output quantities include the 'Pearson residuals' for each observation?
normalised_resids	Output quantities include the 'normalised residuals' for each observation?

Defining projections

@n_projections	Number of projections to be done (from a point estimate)

Defining yield calculations

@catch_split	Catch split used in yield simulations
@B_pre	Pre-fishery biomass `Bpre` block command
mature_only	Pre-fishery biomass `Bpre` is mature fish only?
area	Area for which pre-fishery biomass `Bpre` is calculated
step	Time step in which pre-fishery biomass `Bpre` is calculated
proportion_mortality	Proportion of the time step's mortality after which pre-fishery biomass Bpre is calculated
selectivity	Selectivity ogive with which pre-fishery biomass `Bpre` is calculated

Defining deterministic yields

@per_recruit	Per-recruit block command
do_YPR_SPR	Supply data to plot a YPR or SPR curve?
F	F's at which to calculate YPR & SPR
do_Fmax	Calculate F_{max}?
do_F0_1	Calculate $F_{0.1}$?
do_Fx	Calculate $F_{x\%}$?
x	x at which to calculate $F_{x\%}$
guess	First guess at F_{max}, $F_{0.1}$, $F_{x\%}$
@deterministic_MSY	Deterministic MSY block command
do_MSY	Calculate deterministic MSY?
do_yield_vs_SSB	Supply data to plot a yield versus SSB curve?
F	F's at which to calculate yield and SSB
guess	First guess at F_{MSYdet}

Defining stochastic yields

@MCY_CAY	MCY/CAY block command
do_MCY	Calculate MCY?
do_CAY	Calculate CAY?
interactive	Should MCY/CAY be calculated interactively?
p, q	Risk constraints in MCY/CAY analysis
n_simulations	Number of simulations for each harvest rate *H* in MCY/CAY analyses (point-based only)
n_discard, n_keep	Number of years to discard and to keep in each MCY/CAY simulation.
max_upper_iter	Maximum number of times upper bound on H can be increased when searching for an optimal yield.
MCY_uncertainty_dist	Distribution of uncertainty in virgin biomass (point-based only)
MCY_uncertainty_cv	C.v. of uncertainty in virgin biomass (point-based only)
MCY_guess	First guess at MCY
CAY_uncertainty_dist	Distribution of annual stock-assessment uncertainty
CAY_uncertainty_cv	C.v. of annual stock-assessment uncertainty
F_CAY_guess	First guess at F_{CAY}
@CSP	CSP block command
do_CSP	Calculate CSP?
individual_stocks	Calculate CSP by individual stock?
CSP_guess	First guess at CSP
@B_post	Post-fishery biomass `B_post` block command
mature_only	Post-fishery biomass `B_post` is mature fish only?
area	Area for which post-fishery biomass `B_post` is calculated
step	Time step in which post-fishery biomass `B_post` is calculated
proportion_mortality	Proportion of the time step's mortality after which post-fishery biomass `B_post` is calculated
selectivity	Selectivity ogive with which post-fishery biomass `B_post` is calculated

Other commands

@comment	User supplied comment

INDEX