

# CASAL2 User Manual

I. Doonan, A. Dunn, K. Large, C. Marsh, S. Rasmussen, S.  
Mormede



NIWA Technical Report xxx  
ISSN 1174-2631  
2015

CASAL2 User Manual (modified: 2015-10-20)  
for use with `casal2` 2015-10-20 (rev. f610208)



CASAL2 User Manual (modified: 2015-10-20)  
for use with casal2 2015-10-20 (rev. f610208)

Citation: I. Doonan, A. Dunn, K. Large, C. Marsh, S. Rasmussen, S. Mormede (2015) CASAL2 User Manual, 2015-10-20 (rev. f610208). National Institute of Water & Atmospheric Research Ltd. *NIWA Technical Report*. 156 p.



## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Version . . . . .	1
1.2	Citing CASAL2 . . . . .	1
1.3	Software license . . . . .	1
1.4	System requirements . . . . .	1
1.5	Necessary files . . . . .	2
1.6	Getting help . . . . .	2
1.7	Technical details . . . . .	2
<b>2</b>	<b>Model overview</b>	<b>5</b>
2.1	Introduction . . . . .	5
2.2	The population section . . . . .	6
2.3	The estimation section . . . . .	6
2.4	The observation section . . . . .	6
2.5	The report section . . . . .	7
<b>3</b>	<b>Running CASAL2</b>	<b>9</b>
3.1	Using CASAL2 . . . . .	9
3.2	The input configuration file . . . . .	9
3.3	Redirecting standard output . . . . .	10
3.4	Command line arguments . . . . .	10
3.5	Constructing an CASAL2 input configuration file . . . . .	11
3.5.1	Commands . . . . .	11
3.5.2	Subcommands . . . . .	12
3.5.3	The command-block format . . . . .	12
3.5.4	Commenting out lines . . . . .	13
3.5.5	Determining parameter names . . . . .	13
3.6	CASAL2 exit status values . . . . .	14
<b>4</b>	<b>The population section</b>	<b>15</b>
4.1	Introduction . . . . .	15
4.2	Population structure . . . . .	15
4.3	The state object and the partition . . . . .	17
4.4	Time sequences . . . . .	0
4.5	Time sequences . . . . .	2
4.5.1	Annual cycle . . . . .	2
4.5.2	Initialisation . . . . .	3
4.5.3	Model years . . . . .	4
4.5.4	Projections . . . . .	4
4.6	Population processes . . . . .	4
4.6.1	Recruitment . . . . .	4

4.6.2	Ageing . . . . .	6
4.6.3	Mortality . . . . .	7
4.6.4	Maturation . . . . .	9
4.6.5	Migration . . . . .	9
4.6.5.1	Tag release events . . . . .	11
4.6.5.2	Tag shedding rate . . . . .	11
4.7	Derived quantities . . . . .	11
4.8	Age-length relationship . . . . .	11
4.9	Weightless model . . . . .	13
4.10	Maturity, in models without maturing in the partition . . . . .	13
4.11	Selectivities . . . . .	14
4.11.1	Constant . . . . .	14
4.11.2	Knife-edge . . . . .	15
4.11.3	All-values . . . . .	15
4.11.4	All-values-bounded . . . . .	15
4.11.5	Increasing . . . . .	15
4.11.6	Logistic . . . . .	15
4.11.7	Inverse logistic . . . . .	15
4.11.8	Logistic producing . . . . .	16
4.11.9	Double-normal . . . . .	16
4.11.10	Double-exponential . . . . .	16
4.11.11	Spline . . . . .	16
<b>5</b>	<b>The estimation section</b>	<b>17</b>
5.1	Role of the estimation section . . . . .	17
5.2	The objective function . . . . .	17
5.3	Specifying the parameters to be estimated . . . . .	18
5.4	Point estimation . . . . .	18
5.4.1	The numerical differences minimiser . . . . .	18
5.4.2	The differential evolution minimiser . . . . .	19
5.4.3	The Betadiff minimiser . . . . .	20
5.5	Posterior profiles . . . . .	20
5.6	Bayesian estimation . . . . .	20
5.7	Priors . . . . .	24
5.8	Penalties . . . . .	25
<b>6</b>	<b>The observation section</b>	<b>27</b>
6.1	Observations and likelihoods . . . . .	27
6.2	Proportions-at-age observations . . . . .	27
6.2.1	Likelihoods for proportions-at-age observations . . . . .	30
6.3	Proportions-by-category observations . . . . .	31
6.3.1	Likelihoods for proportions-by-category observations . . . . .	32

6.4	Abundance or biomass observations . . . . .	33
6.4.1	Likelihoods for abundance observations . . . . .	35
6.5	Process error . . . . .	35
6.6	Ageing error . . . . .	36
6.7	Simulating observations . . . . .	36
6.8	Pseudo-observations . . . . .	37
<b>7</b>	<b>The report section</b>	<b>39</b>
<b>8</b>	<b>Population command and subcommand syntax</b>	<b>41</b>
8.1	Model structure . . . . .	41
8.2	Initialisation . . . . .	42
8.2.1	Derived . . . . .	42
8.2.2	Iterative . . . . .	42
8.2.3	State Category By Age . . . . .	43
8.3	Categories . . . . .	43
8.4	Time-steps . . . . .	43
8.5	Processes . . . . .	44
8.5.1	Ageing . . . . .	44
8.5.2	Growth . . . . .	44
8.5.3	Maturation . . . . .	44
8.5.4	Mortality Constant Rate . . . . .	45
8.5.5	Mortality Event . . . . .	45
8.5.6	Mortality Event Biomass . . . . .	46
8.5.7	Mortality Instantaneous . . . . .	46
8.5.8	Nop . . . . .	47
8.5.9	Recruitment Beverton Holt . . . . .	47
8.5.10	Recruitment Constant . . . . .	48
8.5.11	Tag By Age . . . . .	48
8.5.12	Tag By Length . . . . .	50
8.5.13	Transition Category . . . . .	51
8.5.14	Transition Category By Age . . . . .	51
8.6	Time varying parameters . . . . .	52
8.6.1	Annual Shift . . . . .	52
8.6.2	Constant . . . . .	53
8.6.3	Exogenous . . . . .	53
8.7	Derived quantities . . . . .	53
8.7.1	Abundance . . . . .	54
8.7.2	Biomass . . . . .	54
8.8	Age-length relationship . . . . .	54
8.8.1	Data . . . . .	55
8.8.2	None . . . . .	55

8.8.3	Schnute . . . . .	55
8.8.4	Von Bertalanffy . . . . .	56
8.9	Length-weight . . . . .	57
8.9.1	Basic . . . . .	57
8.9.2	None . . . . .	58
8.10	Selectivities . . . . .	58
8.10.1	All Values . . . . .	58
8.10.2	All Values Bounded . . . . .	58
8.10.3	Constant . . . . .	58
8.10.4	Double Exponential . . . . .	58
8.10.5	Double Normal . . . . .	59
8.10.6	Increasing . . . . .	59
8.10.7	Inverse Logistic . . . . .	59
8.10.8	Knife Edge . . . . .	60
8.10.9	Logistic . . . . .	60
8.10.10	Logistic Producing . . . . .	60
<b>9</b>	<b>Estimation command and subcommand syntax</b>	<b>60</b>
9.1	Estimation methods . . . . .	60
9.1.1	Beta . . . . .	61
9.1.2	Lognormal . . . . .	62
9.1.3	Normal . . . . .	62
9.1.4	Normal By Stdev . . . . .	62
9.1.5	Normal Log . . . . .	63
9.1.6	Uniform . . . . .	63
9.1.7	Uniform Log . . . . .	63
9.2	Point estimation . . . . .	63
9.2.1	A D O L C . . . . .	63
9.2.2	Beta Diff . . . . .	64
9.2.3	C P P A D . . . . .	64
9.2.4	D E Solver . . . . .	64
9.2.5	D Lib . . . . .	65
9.2.6	Gamma Diff . . . . .	65
9.3	Monte Carlo Markov Chain (MCMC) . . . . .	65
9.3.1	Independence Metropolis . . . . .	67
9.4	Profiles . . . . .	67
9.5	Defining catchability constants . . . . .	67
9.5.1	Free . . . . .	67
9.6	Defining penalties . . . . .	67
9.6.1	Process . . . . .	68
9.6.2	Vector Average . . . . .	68
9.6.3	Vector Smoothing . . . . .	68



9.7	Defining priors on parameter ratios, differences and means . . . . .	69
9.7.1	Beta . . . . .	69
<b>10</b>	<b>Observation command and subcommand syntax</b>	<b>70</b>
10.1	Observation types . . . . .	70
10.1.1	Process Abundance . . . . .	71
10.1.2	Process Abundance . . . . .	71
10.1.3	Process Biomass . . . . .	72
10.1.4	Process Biomass . . . . .	72
10.1.5	Process Proportions At Age . . . . .	73
10.1.6	Process Proportions At Age . . . . .	74
10.1.7	Process Proportions At Age For Fishery . . . . .	75
10.1.8	Process Proportions At Age For Fishery . . . . .	75
10.1.9	Process Proportions At Length . . . . .	76
10.1.10	Process Proportions At Length . . . . .	77
10.1.11	Process Proportions At Length For Fishery . . . . .	78
10.1.12	Process Proportions At Length For Fishery . . . . .	78
10.1.13	Process Proportions By Category . . . . .	79
10.1.14	Process Proportions By Category . . . . .	80
10.1.15	Process Proportions Migrating . . . . .	81
10.1.16	Process Proportions Migrating . . . . .	81
10.1.17	Time Step Abundance . . . . .	82
10.1.18	Time Step Abundance . . . . .	82
10.1.19	Time Step Biomass . . . . .	83
10.1.20	Time Step Biomass . . . . .	83
10.1.21	Time Step Proportions At Age . . . . .	84
10.1.22	Time Step Proportions At Age . . . . .	85
10.1.23	Time Step Proportions At Length . . . . .	86
10.1.24	Time Step Proportions At Length . . . . .	86
10.1.25	Time Step Proportions By Category . . . . .	87
10.1.26	Time Step Proportions By Category . . . . .	88
10.2	Likelihoods . . . . .	88
10.2.1	Binomial . . . . .	89
10.2.2	Binomial Approx . . . . .	89
10.2.3	Dirichlet . . . . .	89
10.2.4	Log Normal . . . . .	89
10.2.5	Log Normal With Q . . . . .	89
10.2.6	Multinomial . . . . .	89
10.2.7	Normal . . . . .	89
10.2.8	Pseudo . . . . .	89
10.3	Defining ageing error . . . . .	89
10.3.1	Data . . . . .	89

10.3.2	Normal . . . . .	89
10.3.3	Off By One . . . . .	90
<b>11</b>	<b>Report command and subcommand syntax</b>	<b>90</b>
11.1	Available reports . . . . .	90
11.2	Report commands and subcommands . . . . .	90
11.2.1	Ageing Error Matrix . . . . .	91
11.2.2	Category Info . . . . .	91
11.2.3	Category List . . . . .	91
11.2.4	Covariance Matrix . . . . .	91
11.2.5	Derived Quantity . . . . .	91
11.2.6	Estimable . . . . .	91
11.2.7	Estimate Summary . . . . .	92
11.2.8	Estimate Value . . . . .	92
11.2.9	Initialisation Partition . . . . .	92
11.2.10	M C M C Chain . . . . .	92
11.2.11	M C M C Covariance . . . . .	92
11.2.12	Objective Function . . . . .	92
11.2.13	Observation . . . . .	92
11.2.14	Partition . . . . .	92
11.2.15	Partition Biomass . . . . .	92
11.2.16	Partition Mean Weight . . . . .	93
11.2.17	Random Number Seed . . . . .	93
11.2.18	Selectivity . . . . .	93
11.2.19	Simulated Observation . . . . .	93
11.2.20	Standard Header . . . . .	93
<b>12</b>	<b>Other commands and subcommands</b>	<b>93</b>
<b>13</b>	<b>Examples</b>	<b>95</b>
<b>14</b>	<b>Post processing output using R</b>	<b>97</b>
<b>15</b>	<b>Troubleshooting</b>	<b>99</b>
15.1	Introduction . . . . .	99
15.2	Reporting errors . . . . .	99
15.3	Guidelines for reporting a problem with CASAL2 . . . . .	99
<b>16</b>	<b>Acknowledgements</b>	<b>101</b>
<b>17</b>	<b>Quick reference</b>	<b>103</b>
<b>18</b>	<b>References</b>	<b>119</b>
<b>19</b>		<b>121</b>





---

## 1. Introduction

*check and confirm text*

*KL comment: Needs rewriting*

CASAL2 (`casal2`) is a generalised age- or size-structured fish stock assessment model that allows flexibility in specifying population dynamics, parameter estimation and model outputs. CASAL2 can model population dynamics for an age-structured population using a range of observations, including tagging, relative abundance, and age frequency data. CASAL2 implements an age-structured population which can have user defined categories (e.g., immature, mature, male, female, etc.), and age range.

This manual describes how to use CASAL2, including how to run CASAL2, how to set up an input configuration file. Further, we describe the population dynamics and estimation methods, and describe how to specify and interpret output.

### 1.1. Version

*check and confirm text*

This document (last modified 2015-10-20) describes CASAL2 2015-10-20 (rev. f610208). The CASAL2 version number is suffixed with a date/time (yyyy-mm-dd) and revision number, giving the revision control system UTC date and revision number for the most recent modification of the source files. User manual updates will usually be issued for each minor version or date release of CASAL2, and can be obtained, on request, from the authors.

### 1.2. Citing CASAL2

*check and confirm text*

A suitable reference for CASAL2 and this document is:

I. Doonan, A. Dunn, K. Large, C. Marsh, S. Rasmussen, S. Mormede (2015) CASAL2 User Manual, 2015-10-20 (rev. f610208). National Institute of Water & Atmospheric Research Ltd. *NIWA Technical Report*. 156 p.

### 1.3. Software license

*check and confirm text*

This program and the accompanying materials are made available under the terms of the Common Public License v1.0 which accompanies this software (see Section 19).

Copyright ©2008-2015, National Institute of Water & Atmospheric Research Ltd. and the New Zealand Ministry for Primary Industries. All rights reserved.

### 1.4. System requirements

*check and confirm text*

CASAL2 is available for most IBM compatible machines running 64-bit Linux and Microsoft Windows operating systems.

Several of CASAL2s tasks are highly computer intensive and a fast processor is recommended. Depending on the model implemented, some of CASAL2s tasks can take a considerable amount of

time (minutes to hours), and in extreme cases can even take several days to estimate a model fit. Some of CASAL2s tasks can be multi-threaded, and hence multi-core machines may perform some tasks quicker than single core processors.

The program itself requires only a few megabytes of hard-disk space but output files can consume large amounts of disk space. Depending on number and type of user output requests, the output could range from a few hundred kilobytes to several hundred megabytes. When estimating model fits, several hundred megabytes of RAM may be required, depending on the spatial size of the model, number of categories, and complexity of processes and observations. For extremely large models, several gigabytes of RAM may be required.

### 1.5. Necessary files

#### *check and confirm text*

For both 64-bit Linux and Microsoft Windows, only the binary file `spm` or `spm.exe` is required to run CASAL2. No other software is required. We do not compile a version for 32-bit operating systems.

CASAL2 offers little in the way of post-processing of the output, and a package available that allows tabulation and graphing of model outputs is recommended. We suggest software such as **R** (R Development Core Team 2007) to assist in the post processing of CASAL2 output. We provide the `sam` **R** package for importing the CASAL2 output into **R** (see Section 14).

### 1.6. Getting help

#### *check and confirm text*

CASAL2 is distributed as unsupported software, however we would appreciate being notified of any problems or errors in CASAL2. See Section 15.2 for how to report errors to the I. Doonan, A. Dunn, K. Large, C. Marsh, S. Rasmussen, S. Mormede. Further information about CASAL2 can be obtained by contacting the I. Doonan, A. Dunn, K. Large, C. Marsh, S. Rasmussen, S. Mormede.

### 1.7. Technical details

#### *check and confirm text*

CASAL2 was compiled on Linux using `gcc`, the C/C++ compiler developed by the GNU Project. The 64-bit Linux version was compiled using `gcc` version 4.8.1 20130909 (SUSE Linux). Note that CASAL2 is not supported for Linux kernel versions prior to 2.6. The Microsoft Windows version was compiled using Mingw32 `gcc` (tdm64-1) 4.7.1. The Microsoft Windows installer was built using the Nullsoft Scriptable Install System.

CASAL2 uses two minimisers — the first is closely based on the main algorithm of Dennis Jr and Schnabel (1996), and which which uses finite difference gradients, and the second is an implementation of the differential evolution solver (Storn and Price, 1995), and based on code by Lester E. Godwin of PushCorp, Inc.

The random number generator used by CASAL2 uses an implementation of the Mersenne twister random number generator (Matsumoto and Nishimura, 1998). This, the command line functionality, matrix operations, and a number of other functions use the BOOST C++ library (Version 1.54.0).

Note that the output from CASAL2 may differ slightly on the different platforms due to different precision arithmetic or other platform dependent implementation issues. The source code for CASAL2 is available either as a part of the installation, or on request from the I. Doonan, A. Dunn,

K. Large, C. Marsh, S. Rasmussen, S. Mormede.

Unit tests of the underlying CASAL2 code are carried out at build time, using the BOOST unit testing framework. The unit test framework aims to cover a significant proportion of the key functions and processes within the CASAL2 code base. The unit test code for CASAL2 is available as a part of the underlying source code.





---

## 2. Model overview

### 2.1. Introduction

*check and confirm text*

CASAL2 is an age-structured population dynamics model. It implements a statistical catch-at-age population dynamics, using a discrete time-step state-space model that represents a cohort-based population age structure .

CASAL2 is run from the console window on Microsoft Windows or from a terminal window on Linux. CASAL2 gets its information from input data files, the main one of which is the *input configuration file*. Commands and subcommands in the input configuration file are used to define the model structure, provide observations, define parameters, and define the outputs (reports) for CASAL2. Command line switches tell CASAL2 the run mode and where to direct its output. See Section ?? for the details.

We define the model in terms of the *state*. The state consists of two parts, the *partition*, and any *derived quantities* or *derived quantities by cell*. The state will typically change one or more times in every *time-step* of every year, depending on the *processes* defined for each model.

The partition is a representation of the population at an instance in time, and is a matrix of the numbers of individuals within each age, and category. A derived quantity is a cumulative summary of the partition (over all cells) at some point in time. A derived quantity by cell is a cumulative summary of the partition in each of the cells at some point in time. Unlike the partition (which is updated as each new process is applied), each derived quantity records a single value for each year of the model run, and each derived quantity by cell records a layer of values for each year of the model run. Hence, derived quantities build up a vector of values over the model run years. For example, the total number of individuals in a category labelled mature at some point in the annual cycle may be a derived quantity and the total number of individuals in a category labelled mature in each cell of the model at some point in the annual cycle may be a derived quantity by cell. The state is the combination of the partition and any derived quantities or derived quantities by cell at some instance in time. Changes to the state occur by the application of processes. Additions to the vectors of derived quantities occur when a model is requested to add a value to each derived quantity vector.

Running of the model consists of two main parts — first the model state is initialised for a number of iterations (years), then the model runs over a range of predefined years.

The application of processes within each year is controlled by the *annual cycle*. This defines what processes happen in each model year, and in what sequence. Initialisation can be phased, and for each phase, the user need to define the processes that occur in each year, and the order in which they are applied.

For the run years, each year is split up into one or more time-steps (with at least one process occurring in each time-step). You can think of each time-step as representing a particular part of the calendar year, or you can just treat them as an abstract sequence of events.

The division of the year into an arbitrary number of time-steps allows the user to specify the exact order in which processes occur and when observations are evaluated. The user specifies the time-steps, their order, and the processes within each time-step. If more than one process occurs in the same time-step, then they occur in the order that they are specified. Observations are always evaluated at the end of the time-step in which they occur. Hence, time-steps can be used to break processes into groups, and assist in defining the timing of the observations within the annual cycle.

The population structure of CASAL2 follows the usual population modelling conventions and is similar to those implemented in other population models, for example CASAL (Bull et al., 2012). The model records the numbers of individuals by age and category (e.g., male, female). In general, cohorts are added via a recruitment event, are aged annually, and are removed from the population via various forms of mortality. The population is assumed to be closed (i.e., no immigration or emigration from the modelled area)

A model is implemented in CASAL2 using an input configuration file, which is a complete description of the model structure (i.e., spatial and population processes), observations, estimation methods, and reports (outputs) requested. CASAL2 runs from a console window on Microsoft Windows or from a text terminal on Linux. A model can be either *run*, estimable parameters can be *estimated* or *profiled*, *MCMC* distributions calculated, and these estimates can be used by CASAL2 as parameters of an operating model to *simulate* observations.

A model in CASAL2 is specified by an input configuration file, and comprises of four main components. These are the population section (model structure, population dynamics, etc.), the estimation section (methods of estimation and the parameters to be estimated), the observation section (observational data and associated likelihoods), and the report section (printouts and reports from the model). The input configuration file completely describes a model implemented in CASAL2. See Sections 8, 9, 10, and 11 for details and specification of CASAL2s command and subcommand syntax within the input configuration file.

### 2.2. The population section

*check and confirm text*

The population section (Section 4) defines the model of the population dynamics. It describes the model structure (i.e. the population structure), initialisation and run years (model period), population processes (for example, recruitment, migration, and mortality), selectivities, and key population parameters.

### 2.3. The estimation section

*check and confirm text*

The estimation section (Section 5) specifies the parameters to be estimated, estimation methods, penalties and priors. Estimation is based on an objective function (e.g., negative log posterior). Depending on the run mode, the estimation section is used to specify the methods for finding a point estimate (i.e., the set of parameter values that minimizes the objective function), doing profiles, or MCMC methods and options, etc.

Further, the estimation section specifies the parameters to be estimated within each model run and the estimation methods. The estimation section specifies the choice of estimation method, which model parameters are to be estimated, priors, starting values, and minimiser control values.

Penalties and priors act as constraints on the estimation. They can either encourage or discourage (depending on the specific implementation) parameter estimates that are ‘near’ some value, and hence influence the estimation process. For example, a penalty can be included in the objective function to discourage parameter estimates that lead to models where the recorded catch was unable to be fully taken.

### 2.4. The observation section

*check and confirm text*

Types of observations, their values, and the associated error structures are defined in the observation section (Section 6). Observations are data which allow us to make inferences about unknown parameters. The observation section specifies the observations, their errors, likelihoods, and when the observations occur. Examples include relative or absolute abundance indices, proportions-at-age frequencies, etc. Estimation uses the observations to find values for each of the estimated parameters so that each observation is ‘close’ (in some mathematical sense) to a corresponding expected value.

**2.5. The report section***check and confirm text*

The report section (Section 7) specifies the model outputs. It defines the quantities and model summaries to be output to external files or to the standard output. While CASAL2 will provide informational messages to the screen, CASAL2 will only produce model estimates, population states, and other data as requested by the report section. Note that if no reports are specified, then no output will be produced.



---

### 3. Running CASAL2

#### *check and confirm text*

CASAL2 is run from the console window (i.e., the DOS command line) on Microsoft Windows or from a terminal window on Linux. CASAL2 gets its information from input data files, the key one of which is the input configuration file.

The input configuration file is compulsory and defines the model structure, processes, observations, parameters (both the fixed parameters and the parameters to be estimated), and the reports (outputs) requested. The following sections describe how to construct the CASAL2 configuration file. By convention, the name of the input configuration file ends with the suffix `.csl2`, however, any file name is acceptable.

Other input files can, in some circumstances, be supplied to define the starting point for an estimation or as a point estimate from which to simulate observations.

Simple command line arguments are used to determine the actions or *tasks* of CASAL2, i.e., to run a model with a set of parameter values, estimate parameter values (either point estimates or MCMC), project quantities into the future, simulate observations, etc.,. Hence, the *command line arguments* define the *task*. For example, `-r` is the *run*, `-e` is the *estimation*, and `-m` is the *MCMC* task. The *command line arguments* are described in Section 3.4.

#### 3.1. Using CASAL2

##### *check and confirm text*

To use CASAL2, open a console (i.e. the command prompt) window (Microsoft Windows) or a terminal window (Linux). Navigate to a directory of your choice, where your input configuration files are located. Then type `casal2` with any arguments (see Section 3.4 for the the list of possible arguments). CASAL2 will print output to the screen and return you to the command prompt when it completes its task. Note that the CASAL2 executable (binary) must be either in the directory where you run it or somewhere in your `PATH`. Note that an automated installer is available for CASAL2 on Microsoft Windows. If you use the installer, then it will give you the option of modifying your `PATH` for you (as well a a number of other options to make using the program a little easier). Otherwise, see your operating system documentation for help on identifying or modifying your `PATH`.

[need an example here...](#) a small example of directory navigation and `casal2` with arguments as would be typed in the command prompt window

#### 3.2. The input configuration file

##### *check and confirm text*

The input configuration file is made up of four broad sections; the description of the population structure and parameters (the population section), the estimation methods and variables (the estimation section), the observations and their associated likelihoods (the observation section), and the outputs and reports that CASAL2 will return (the report section). The input configuration file is made up of a number of commands (many with subcommands) which specify various options for each of these components.

The command and subcommand definitions in the input configuration file can be extensive (especially when you have a model that has many observations), and can result in a input configuration file that is long and difficult to navigate. To aid readability and flexibility, we can use the input configuration file command `!include file`. The command causes an external

file, *file*, to be read and processed, exactly as if its contents had been inserted in the main input configuration file at that point. The file name must be a complete file name with extension, but can use either a relative or absolute path as part of its name. Note that included files can also contain *!include* commands — but be careful that you do not set up a recursive state. See Section 12 for more detail.

### 3.3. Redirecting standard output

*check and confirm text*

CASAL2 uses the standard output stream `standard output` to display run-time information. The standard error stream is used by CASAL2 to output the program exit status and run-time errors. We suggest redirecting both the standard output and standard error into files. With the bash shell (on Linux systems), you can do this using the command structure,

```
(\cname\ [arguments] > out) >& err &
```

It may be useful to redirect the standard input, especially if you're using CASAL2 inside a batch job software, i.e.

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

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

```
\cname\ [arguments] > out
```

And, on some Microsoft Windows systems (e.g., Windows7), you can redirect to both standard output and standard error, using the syntax,

```
\cname\ [arguments] > out 2> err
```

Note that CASAL2 outputs a few lines of header information to the output. The header consists of the program name and version, the arguments passed to CASAL2 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). These can be used to track outputs as well as identifying the version of CASAL2 used to run the model.

### 3.4. Command line arguments

*check and confirm text*

The call to CASAL2 is of the following form.:

```
casal2[-c config_file] [task] [options]
```

**-c *config\_file*** Define the input configuration file for CASAL2. If omitted, then CASAL2 looks for a file named `config.csl2`.

and where *task* is one of;

**-h** Display help (this page).

- l** Display the reference for the software license (CPLv1.0).
- v** Display the CASAL2 version number.
- r** Run the model once using the parameter values in the input configuration file, or optionally, with the values from the file denoted with the command line argument *-i file*.
- e** Do a point *estimate* using the values in the input configuration file as the starting point for the parameters to be estimated, or optionally, with the start values from the file denoted with the command line argument *-i file*.
- p** Do a likelihood *profile* using the parameter values in the input configuration file as the starting point, or optionally, with the start values from the file denoted with the command line argument *-i file*.
- m** Do an *MCMC* estimate using the values in the input configuration file as the starting point for the parameters to be estimated, or optionally, with the start values from the file denoted with the command line argument *-i file*.
- s number** Simulate the *number* of observation sets using values in the input configuration file as the parameter values, or optionally, with the values for the parameters denoted as estimated from the file with the command line argument *-i file*.

In addition, the following are optional arguments [*options*],

- i file** Input one or more sets of free (estimated) parameter values from *file*. See Section ?? for details about the format of *file*.
- o file** Output a report of the free (estimated) parameter values in a format suitable for *-i file*. See Section ?? for details about the format of *file*.
- t number** Maximum number of *threads* to use when the model includes multi-threaded process.
- q** Run *quietly*, i.e., suppress verbose printing of CASAL2.
- g seed** Seed the random number *generator* with *seed*, a positive (long) integer value. Note, if *-g* is not specified, then CASAL2 will generate a random number seed based on the computer clock time.

### 3.5. Constructing an CASAL2 input configuration file

*check and confirm text*

The model definition, parameters, observations, and reports are specified in an input configuration file. The population section is described in Section 4 and the population commands in Section 8. Similarly, the estimation section is described in Section 5 and its commands in Section 9, and in Section 7 and Section 11 for the report and report commands.

#### 3.5.1. Commands

*check and confirm text*

CASAL2 has a range of commands that define the model structure, processes, observations, and how tasks are carried out. There are three types of commands,

1. Commands that have an argument and do not have subcommands (for example, *!include file*)

2. Commands that have a label and subcommands (for example `@process` must have a label, and has subcommands)
3. Commands that do not have either a label or argument, but have subcommands (for example `@model`)

Commands that have a label must have a unique label, i.e., the label cannot be used on more than one command of that type. The labels must start with a letter or underscore, can contain letters, underscores, or numbers. Labels must not contain white-space, a full-point ('.'), or other characters that are not letters, numbers, or an underscore.

*need an example here...*

### 3.5.2. Subcommands

#### *check and confirm text*

Subcommands in CASAL2 are for defining options and parameter values for commands. They always take an argument which is one of a specific *type*. The types acceptable for each subcommand are defined in Section 12, and are summarised below.

Like commands (`@command`), subcommands and their arguments are not order specific — except that that all subcommands of a given command must appear before the next `@command` block. CASAL2 may report an error if they are not supplied in this way, however, in some circumstances a different order may result in a valid, but unintended set of actions, leading to possible errors in your expected results.

The arguments for a subcommand are either:

<b>switch</b>	true/false
<b>integer</b>	an integer number,
<b>integer vector</b>	a vector of integer numbers,
<b>integer range</b>	a range of integer numbers separated by a hyphen (-), e.g. 1994-1996 2000 is expanded to an integer vector of values 1994 1995 1996 2000),
<b>constant</b>	a real number (i.e. double),
<b>constant vector</b>	a vector of real numbers (i.e. vector of doubles),
<b>estimable</b>	a real number that can be estimated (i.e. estimable double),
<b>estimable vector</b>	a vector of real numbers that can be estimated (i.e. vector of estimable doubles),
<b>string</b>	a categorical (string) value, or
<b>string vector</b>	a vector of categorical values.

Switches are parameters which are either true or false. Enter *true* as `true` or `t`, and *false* as `false` or `f`.

Integers must be entered as integers (i.e., if `year` is an integer then use 2008, not 2008.0)

Arguments of type integer vector, integer range, constant vector, estimable vector, or categorical vector contain one or more entries on a row, separated by white space (tabs or spaces).

*Estimable* parameters are those parameters that CASAL2 can estimate, if requested. If a particular parameter is not being estimated in a particular model run, then it acts as a constant. Within CASAL2 only estimable parameters can be estimated. And, you have to tell CASAL2 those that are to be estimated in any particular model. Estimable parameters that are being estimated within a particular model run are called the *estimated parameters*.

### 3.5.3. The command-block format



*check and confirm text*

Each command-block either consists of a single command (starting with the symbol @) and, for most commands, a label or an argument. Each command is then followed by its subcommands and their arguments, e.g.,

```
@command, or
@command argument, or
@command label
```

**and then**

```
subcommand argument
subcommand argument
etc.,
```

Blank lines are ignored, as is extra white space (i.e., tabs and spaces) between arguments. But don't put extra white space before a @ character (which must also be the first character on the line), and make sure the file ends with a carriage return.

There is no need to mark the end of a command block. This is automatically recognized by either the end of the file, section, or the start of the next command block (which is marked by the @ on the first character of a line). Note, however, that the *!include* is the only exception to this rule. See Section 12) for details of the use of *!include*.

Note that in the input configuration file, commands, sub-commands, and arguments are not case sensitive. However, labels and variable values are case sensitive. Also note that if you are on a Linux system then external calls to files are case sensitive (i.e., when using *!include file*, the argument *file* will be case sensitive).

Characters used in labels must be alphanumeric and can include underscores (\_). Other characters will result in an error.

**3.5.4. Commenting out lines***check and confirm text*

Text that follows a # on a line are considered to be comments and are ignored. If you want to remove a group of commands or subcommands using #, then comment out all lines in the block, not just the first line.

Alternatively, you can comment out an entire block or section by placing curly brackets around the text that you want to comment 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. (These should ideally be the first character on a line. But if not, then the entire line will be treated as part of the comment block.)

*need an example here.....*[short example of comment syntax](#)

**3.5.5. Determining parameter names***check and confirm text*

When CASAL2 processes a input configuration file, it translates each command and each subcommand into a parameter with a unique name. For commands, this parameter name is simply the command name. For subcommands, the parameter name format is either

`command[label].subcommand` if the command has a label, or  
`command.subcommand` if the command has no label, or  
`command[label].subcommand(i)` if the command has a label and the subcommand arguments are a vector, and we are accessing the  $i$ th element of that vector.

The unique parameter name is used to reference the parameter when estimating, applying a penalty, or applying a profile. For example, the parameter name of subcommand `r0` of the command `@process` with the label `MyRecruitment` is

```
process[MyRecruitment].r0
```

#### 3.6. CASAL2 exit status values

*check and confirm text*

When CASAL2 completes its task successfully or errors out gracefully, it returns a single exit status value (0) to the operating system. The operating system will return (−1) if CASAL2 terminates unexpectedly. To determine if CASAL2 has completed its task successfully, check the standard output for error and information messages.

---

## 4. The population section

### 4.1. Introduction

The population section specifies the model structure, population dynamics, and other associated parameters. It describes the model structure (population structure), defines the population (e.g., recruitment, migration, and mortality), selectivities, and model parameters.

The population section consists of several components, including;

- The population structure;
- Model initialisation (i.e., the state of the model at the start of the first year);
- The years over which the model runs (i.e., the start and end years of the model)
- The annual cycle (time-steps and processes that are applied in each time-step);
- The specifications and parameters of the population processes (i.e. processes that add, remove individuals to or from the partition, or shift numbers between ages and categories in the partition);
- Selectivities;
- Parameter values and their definitions;
- Derived quantities, required as parameters for some processes (e.g. spawning stock biomass to resolve the spawner-recruit relationship in a recruitment process).

### 4.2. Population structure

The basic structure of a CASAL2 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, CASAL2 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. **In every time step, there exists a mortality block, this is a block where individuals are removed from the partition. If there are no mortality processes then the mortality block is empty (nothing happens) and occurs at the end of a time step .**

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: length class or age class, sex, maturity, area, stock, tag, and growth-path. The user chooses:

- whether the partition is subdivided by length 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 length 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.

#### Craigs attempt

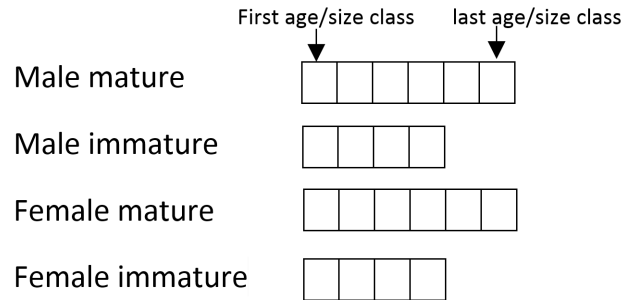
The resulting partition can be conceptualised as a group of vectors, where each category is represented as a vector and the size of the vector is the number of age or length classes, shown in Figure 4.1. Each element in a vector represents the number of individuals for that category in that age or length bin. splitting the partition up into separate categories allows categories to have different age or length structures.

The names and number of categories are user defined, but there must be at least one category defined for a model. The ages are defined as a sequence from  $age_{min}$  to  $age_{max}$ , with the last age optionally a plus group. In order to calculate biomass, the age-length relationship for each category must also be defined. An example of how this is specified for four categories based on sex and area is as follows,

```
@categories
format spawn.sex
names spawn.male spawn.female nonspawn.male nonspawn.female
age_lengths male_AL female_AL male_AL female_AL
```

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.

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.



**Figure 4.1: A visual representation of a partition**

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.

An example, to specify a model with 2 categories (male and female) with ages 1-20 (with the last age a plus group) and an age-length relationship defined with the label `male-growth` and `female-growth`, then the `@model` example from above becomes,

```
@model
start_year
final_year
min_age 1
max_age 20
age_plus_group True
initialisation_phases iphase
time_steps step1 step2
```

### 4.3. 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 length classes, the rows are combinations of the following characters:

The key component of the state object is the partition, a group of vectors that store numbers of fish at age or length for a specific category. A category represents a group of fish that have specific attributes, examples of such attributes include life histories and growth paths. Characters in

a population that display different attributes and that can make up a category or separate categories are:

- 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).

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 length at age in an age-based model that uses some length/size data. Each growth-path has its own growth curve, and the length-based model features will consequently have different effects on different growth-paths. So, you need to tell CASAL2 the following:

- Whether the model is age- or length-based.
- The number and nature of length classes in a length-based model.
- The minimum and maximum age classes in an age-based model.
- Whether there is a plus group.
- The names of all categories and their corresponding growth path labels.
- 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. Length classes are defined by the user, and you need to specify how many length classes there are, the lower bound of each length class, and whether the last length 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, length 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`.

#### **Pretty sure, CASAL 2 doesn't support this concept, can be deleted**

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.

Another important component of the state object in CASAL is a vector of spawning stock biomasses (SSB's, mid-spawning season biomasses of spawning fish) for each stock. CASAL needs to include this in the stat object to calculate future recruitments, if there is a stock-recruitment relationship.

Another important component of the state object in `casal`<sup>2</sup> are spawning stock biomasses (SSBs, mid-spawning season biomasses of spawning fish) for each stock. `Casal`<sup>2</sup> derives through the command `@derived_quantity`, this is needed if there is a stock-recruitment relationship.

#### 4.4. Time sequences

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 length-based model);
- Natural and fishing mortality;
- Disease mortality; (Not yet implemented)
- Tag release events;
- Tag shedding rate;
- Semelparous mortality.(Not yet implemented)

If two or more processes are specified for the same time step CASAL2 will apply the processes in sequential order, that is the order the user has specified. For example, in an annual cycle that has a single time step labelled `step1`,

```
@model
time_steps step1
and in step1 there are three processes that occurred in the following order, Recruitment,
Natural_mortality and Ageing, you would specify that time step as,
@time_step step1
processes Recruitment Natural_mortality Ageing
```

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 basic unit of fishing mortality is a fishery, defined as fishing mortality on group of individuals in the partition in a single time step. 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 spawning stock, recruitment is handled separately for each stock. 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 CASAL2 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 <sup>1</sup>;
- 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 configuration file;
- 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 length-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 years growth which has occurred by the start of each time step <sup>2</sup>;

<sup>1</sup>The SSB (spawning stock biomass) is a common model output and is also the measure of abundance used in stock-recruitment relationships in CASAL2 (where applicable). Different models define SSB in quite different ways so we allow several options in CASAL2 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. There is no default for calculating an SSB. When defining SSB you need, which categories contribute to the SSB, how different age or length classes contribute to SSB through selectivities (i.e. maturity being age or length based), whether SSB happens part way through natural and fishing mortality. 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 a category 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, through the use of a constant selectivity of one.

<sup>2</sup>Fish growth in an age-based model is handled quite differently from a length-based model. The simplest option is to assume that the mean length 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 length 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_proportions` parameter. This is a vector with one entry per time step. The mean length of fish of age  $a$  years



- The proportion of the years 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 <sup>3</sup>
- 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 length class, in which member of the partition to put the tagged fish, and the tag shedding rates, if defined.

You then need to provide CASAL2 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 that can be made. Some of the less obvious ones are listed here. It is an error if:

- the sum of the proportions of the years 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 length-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.

## 4.5. Time sequences

The time sequence of the model is defined in two parts;

- Initialisation
- Run years
- Projection years

### 4.5.1. Annual cycle

The annual cycle is implemented as a set of processes that occur, in a user-defined order, within each year. Time-steps are used to break the annual cycle into separate components, and allow observations to be associated with different sets of processes. Any number of processes can occur within each time-step, in any order and can occur multiple times within each time-step. Note that time-steps are not implemented during the initialisation phases (effectively, there is only one time-step), and that

---

(rounded down) in the  $i$ th time step is calculated as if their age was  $(a + \text{growth\_proportions}[i])$ . So, if the first entry of `growth_proportions` is 0.5, then, in time step 1, the mean length of 2-year-old fish is calculated as if they were age 2.5. The default is `growth_proportions = 0` (i.e., no growth between birthdays).

<sup>3</sup> 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 **Not implemented**. 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.

the annual cycle in the initialisation phases can be different from that which is applied during the model years.

#### 4.5.2. Initialisation

There are multiple methods to initialise a partition in CASAL2. These methods are: iterative, fixed, and analytical. Model initialisation can occur in several phases, each of which iterates through a number of years carrying out the population processes defined for that phase. At the end of the initialisation step, CASAL2 runs through the model years carrying out processes in the order defined in the annual cycle, and can evaluate expected values of observations in order to calculate likelihoods, project forward to determine future states, or simulate observations from the current state.

One of CASAL2 methods for initialising the initial equilibrium state as an iterative process: a general solution that initialises complex structured models can be difficult to implement using analytic techniques. However, initialising via iteration for a long-lived species with complex transitions can take many iterations and be slow to run. In CASAL2, we allow for user-defined multi-phased initialisation using iteration to allow the user to optimize models for speed. Each phase of the initialisation can involve any number of processes. Note that the length of the initialisation period may affect the model outputs, and that a period should be chosen to allow the population state to converge.

In addition, each initialisation process can optionally be stopped early if a user defined convergence criteria is met. For a set of user defined years in the initialisation phase, convergence is defined as met if the proportional absolute summed difference between the the state in year  $t - 1$  and the state in year  $t$  ( $\hat{\lambda}$ ) is less than  $\lambda$  where,

$$\hat{\lambda} = \frac{\sum_i \sum_j |\text{element}(i, j)_t - \text{element}(i, j)_{t-1}|}{\sum_i \sum_j \text{element}(i, j)_t} \quad (4.1)$$

In each initialisation phase, the processes defined for that phase are carried out and used as the starting point for the following phase or, if it is the last phase, then the years that the model is run over. The first phase is always initialised with each element (i.e., each age and category) set at zero. Note that this means that recruitment processes where the numbers of recruits is based on a stock recruitment or density dependant relationship will likely fail if used in the first phase of an initialisation.

The multi-phase iteration allows the user to determine if the initialisation has converged in a particular model run. Here, add an additional initialisation phase for, say, 1 year as the last initialisation phase (with the same processes applied). Then, using the initialisation reports (`@report[label].type=initialisation_partition`), print a copy of the partition just before and just after that phase. If the initialisation has converged to an equilibrium state, then the partition at both these time intervals will be the same.

Hence, for an iterative initialisation you need to define;

- The initialisation phases.
- The number of years in each phase and the processes to apply in each (default is the annual cycle).

### 4.5.3. Model years

Following initialisation, the model then runs over a number of user-defined years. For this part of the model, the annual cycle can be broken into separate time-steps, and observations can be associated with the state of the model at the end of any time-step, i.e., likelihoods for particular observations are evaluated, if required, at the end of each time-step.

Processes are carried out in the order specified within each time-step, and can be the same or different to processes in other initialisation phases of the model. The run years define the years over which the model is to run and the annual cycle within each year. The model runs from the start of year `initial` and runs to the end of year `current`. The projection part then extends the run time up to the end of year `final`.

- The time-steps and the processes applied in each
- The initial year (i.e., the model start year)
- The current year (i.e., the model end year)
- The final year (i.e., the model projection end year)

### 4.5.4. Projections

CASAL2 can project, from a set of parameter estimates, the state of the model into the future. In a projection run, the model is initialised and run through the model years from `initial` to the `current`. Then, the model is run from `current` to `final`.

## 4.6. Population processes

*check and confirm text*

Population processes are those processes that change the population state of individuals. Processes produce changes in the model partition, by adding, removing or moving individuals between ages or categories. The population processes include recruitment, ageing, mortality events (e.g., natural and exploitation) and category transition processes (i.e., processes that move individuals between categories, while preserving their age structure). See Section 4 for a complete list of available processes.

There are two types of processes, processes that occur across multiple time steps in the annual cycle e.g Natural Mortality and Instantaneous Mortality. There are also processes that only occur within the time step they are defined. Each of these processes is carried out in the user-defined prescribed order when initialising the model, and then for a user-defined order in each year in the annual cycle.

### 4.6.1. Recruitment

*check and confirm text*

Recruitment processes are defined as process that introduces new individuals into the model. CASAL2 implements two types of recruitment process, constant recruitment and Beverton-Holt recruitment (Beverton and Holt, 1957).

In the recruitment processes, the number of individuals are added to a single age class within the partition, with the amount defined by the type of recruitment process and its function. If more than one category is defined, then the proportion of recruiting individuals to be added to each category

is specified by the `proportions` parameter. For example, if recruiting to categories labelled male and female, then you might set the proportions as 0.5 and 0.5 respectively to denote that half of the recruits recruit to the male category and the remaining half to the female category.

For the constant and Beverton-Holt recruitment processes, the number of individuals following recruitment in year  $y$  is,

$$\text{element}(i, j) \leftarrow \text{element}(i, j) + p_j(R_y) \quad (4.2)$$

where  $\text{age}$  is the age defined as the recruitment age,  $p_j$  is the proportion to category  $k$  defined to have recruitment,  $n$  is the number of spatial locations where recruitment occurs, and the recruitment to each cell is scaled to be proportional to the value of the layer in that cell. See below for how  $R_y$  is determined in each of these cases.

**Craig's attempt at re writing the above**

$$N_{a,k} \leftarrow N_{a,k} + p_k(R_y) \quad (4.3)$$

where  $a$  is the age defined as the recruitment age,  $p_k$  is the proportion to category  $k$ , and  $R_y$  is the number of recruits for year  $y$ . See below for how  $R_y$  is determined in each of these cases.

### Constant Recruitment

*check and confirm text*

In the constant recruitment process the total number of recruits added each year is  $R_y$ , and is simply  $R_0$ , i.e.

$$R_y = R_0 \quad (4.4)$$

It is equivalent to a Beverton-Holt recruitment process where steepness is set equal to one ( $h = 1$ ).

For example, to specify a constant recruitment process, where individuals are added to the category 'immature' at  $\text{age} = 1$ , and the number to add is  $R_0 = 5 \times 10^5$ , then the syntax is

```
@process Recruitment
type constant_recruitment
categories immature
proportions 1.0
r0 500000
age 1
```

### Beverton-Holt recruitment

*check and confirm text*

In the Beverton-Holt recruitment process the total number of recruits added each year is  $R_y$ , and is the product of the average recruitment  $R_0$ , the annual year class strength multiplier,  $YCS$ , and the stock-recruit relationship i.e.,

$$R_y = R_0 \times YCS_{y-\text{offset}} \times SR(SSB_{y-\text{offset}}) \quad (4.5)$$

where  $\text{offset}$  is the number of years offset to link the year class with the year of spawning  $y$ , and  $SR$  is the Beverton-Holt stock-recruit relationship parametrised by the steepness  $h$ ,

$$SR(SSB_y) = \frac{SSB_y}{B_0} / \left( 1 - \frac{5h-1}{4h} \left( 1 - \frac{SSB_y}{B_0} \right) \right) \quad (4.6)$$

Note that the Beverton-Holt recruitment process requires a value for  $B_0$  and  $SSB_y$  to resolve the stock-recruitment relationship. Here, a derived quantity (see Section 4.7) must be defined that provides the annual  $SSB_y$  for the recruitment process.  $B_0$  is then defined as the value of the  $SSB$  at the end of one of the initialisation phases. During initialisation the  $YCS$  multipliers are assumed to be equal to one, and recruitment that happens in the initialisation phases that occur before and during the phase when  $B_0$  is determined is assumed to have steepness  $h = 1$  (i.e. in those initialisation phases, recruitment is simply equal to  $R_0$ ). Recruitment in the initialisation phases after the phase where  $B_0$  was determined follow the Beverton-Holt stock-recruit relationship defined above. Recruits are then distributed across cells in proportion to the values in a numeric layer.

For example, assume a Beverton-Holt recruitment process, where individuals are added to the category ‘immature’ at  $age = 1$ , the number to add is  $R_0 = 5 \times 10^5$ . Then `SSB_Biomass` is a derived quantity that specifies the total spawning stock biomass, with  $B_0$  the value of the derived quantity at the end of the initialisation phase labelled `phase1`. The  $YCS$  are standardised to have mean one in the period 1994 to 2004, and recruits enter into the model two years following spawning. Then the command specification is

```
@process Recruitment
type recruitment_beverton_holt
categories immature
proportions 1.0
r0 500000
steepness 0.75
age 1
ssb SSB_Biomass
standardise_ycs_years 1994-2004
ycs_years 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006
YCS_values 1 1 1 1 1 1 1 1 1 1 1 1 1
ssb_offset 2
```

### Check

Note that if not specified, `ssb_offset` is set at the value of `age`. This corresponds to cases where recruitment happens after spawning. `SSB_offset` can be user-defined to a different value if needed — for example, if recruitment happens in a time-step before spawning, the offset will need to be specified as `age + 1`.

### 4.6.2. Ageing

#### *check and confirm text*

The ageing process simply moves all individuals in the named categories to the next age class. The ageing process is defined as,

$$\text{element}(i, j) \leftarrow \text{element}(i, j - 1) \quad (4.7)$$

except that in the case of the plus group (if defined),

$$\text{element}(i, \text{age}_{\max}) \leftarrow \text{element}(i, \text{age}_{\max}) + \text{element}(i, \text{age}_{\max-1}). \quad (4.8)$$

For example, to apply ageing to the categories `immature` and `mature`, then the syntax is,

```
@process Ageing
type ageing
categories immature mature
```

Note that ageing is *not* applied by CASAL2 by default. As with other processes, CASAL2 will not apply a process unless its defined and specified as a process within the annual cycle. Hence, it is possible to specify a model where a category is not aged. CASAL2 will not check or otherwise warn if there is a category defined where ageing is not applied.

### 4.6.3. Mortality

#### *check and confirm text*

Four types of mortality processes are permissible in CASAL2, constant rate, event, biomass-event and instantaneous. These processes remove individuals from the partition, either as a rate, as a total number (abundance), as a biomass of individuals or as a mixture of these. CASAL2 does not implement the Baranov catch equation or any other process where both natural and event mortality are applied simultaneously. To apply both natural and biomass-event mortality, the population processes must be defined to remove some natural mortality (e.g. as a constant), then some event mortality (e.g. fishing) in sequence. It is up to the user to specify how this happens.

#### Constant mortality rate

To specify a constant annual mortality rate ( $M = 0.2$ ) for categories ‘male’ and ‘female’, then,

```
@process NaturalMortality
type mortality_constant_rate
categories male female
selectivities One One
M 0.2 0.2
```

Note that the mortality rate process requires a selectivity. To apply the same mortality rate over all age classes, use a selectivity defined as  $S_i = 1.0$  for all ages  $i$ , e.g.

```
@selectivity One
type constant
c 1
```

#### Event and biomass-event mortality

The event mortality process and biomass mortality processes act in a similar manner, except that they remove a specified abundance (number of individuals) or biomass respectively, rather than applying mortality as a rate. However, the maximum abundance or biomass to remove is constrained by a maximum exploitation rate.

CASAL2 removes as many individuals or as much biomass as it can while not exceeding the maximum exploitation rate. Event mortality processes require a penalty function to discourage parameter values that do not allow the defined number of individuals to be removed. Here, the model penalises those parameter estimates that result in an insufficient number of individuals in defined categories (after applying selectivities). See Section 5.8 for more information on specifying penalties.

For example, the event mortality applied to user-defined categories  $i$ , with the numbers removed at age  $j$  determined by a selectivity-at-age  $S_j$  is applied as follows:

First, calculate the vulnerable abundance for each category  $i$  in  $1 \dots I$  for ages  $j = 1 \dots J$  that are subject to event mortality,

$$V(i, j) = S(j)N(i, j) \quad (4.9)$$

And hence define the total vulnerable abundance  $V_{total}$  as,

$$V_{total} = \sum_i \sum_j V(i, j) \quad (4.10)$$

Hence the exploitation rate to apply is

$$U = \begin{cases} C/V_{total}, & \text{if } C/V_{total} \leq U_{max} \\ U_{max}, & \text{otherwise} \end{cases} \quad (4.11)$$

And the number removed  $R$  from each age  $j$  in category  $i$  is,

$$R(i, j) = UV(i, j) \quad (4.12)$$

For example, to specify fishing mortality based on catches given for each year, over categories ‘immature’ and ‘mature’, with selectivity ‘FishingSel’ and assuming a maximum possible exploitation rate of 0.7, then the syntax is

```
@process Fishing
type event_mortality
categories immature mature
years 2000 2001 2002 2003
U_max 0.70
selectivities FishingSel FishingSel
penalty event_mortality_penalty
```

### Instantaneous mortality

The instantaneous mortality process is a combination of natural mortality and event biomass mortality that occurs across multiple time steps. This process applies half the natural mortality, then to apply the mortalities from all the fisheries instantaneously, then to apply the remaining half of the natural mortality.

When Instantaneous mortality is applied the following equations are used.

**Steal the rest from CASAL manual**

```
@process instant_mort
type mortality_instantaneous
m 0.19
time_step_ratio 0.42 0.25 0.33
selectivities One
categories stock
table catches
year FishingWest FishingEast
1975 80000 111000
1976 152000 336000
1977 74000 1214000
end table
```

```

table fisheries
fishery      category  selectivity u_max  time_step penalty
FishingWest  stock      westFSel   0.7   step1    CatchPenalty
FishingEast  stock      eastFSel   0.7   step1    CatchPenalty
end_table

```

#### 4.6.4. Maturation

*KL comment: Text from CASAL, need to rewrite as relevant to SAM*

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.

#### 4.6.5. Migration

*KL comment: text from CASAL. Rewrites required?*

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.

Migration is the process of moving fish from one area to another. For this to be sensibly applied in CASAL2 there needs to be a category for the source area and a category for the target area. 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 length classes), or provide an ogive of proportions migrating by age or length class.

A migration can involve any number of categories as long as there is one to one category relationship. That is for every source category there is one target category. You can state that a given proportion of these fish migrate (constant across all age or length classes), or provide a selectivity of proportions migrating by age or length class.

An example, to specify a simple spawning migration of mature males and females moving from a western area to an eastern (spawning) area, then the syntax is

```
@process Spawning_migration
```



```

type category_transition
from West.males West.females
to East.males West.females
selectivities MatureSel matureSel

```

Where `MatureSel` is a selectivity that describes the proportion of age or length classes that are mature and move to the eastern area.

**Have not implemented** 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  $pi$  is the specified proportion of fish migrating from the  $i$ th partition element, proportion  $(pwave\ pi)$  will migrate in wave 1 and proportion  $(1 - pwave)pi / (1 - (pwave\ pi))$  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  $P_{a,b}^i(y)$  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

$$O_{a,b}^i(y) = \frac{P_{a,b}^i(y)}{1 - P_{a,b}^i(y)} \quad (4.13)$$

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

$$\vartheta_{a,b}^i(y) = O_{a,b}^i(y) \times F_{a,b}(y) \quad (4.14)$$

and hence the proportion of fish migrating to

$$\Pi_{a,b}^i(y) = \frac{\vartheta_{a,b}^i(y)}{1 - \vartheta_{a,b}^i(y)} \quad (4.15)$$

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  $b$

$$F_{a,b}(y) = \exp \left( -S \left( \frac{A_{a,y} - A_{a,0}}{A_{a,0}} \right) - D \left( \frac{A_{b,y} - A_{b,0}}{A_{b,0}} \right) \right) \quad (4.16)$$

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 length classes), what is it? Or, if fish migrate according to an ogive across age or length 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 ?

Two-wave migrations require more details – see earlier.

#### 4.6.5.1. Tag release events

#### 4.6.5.2. Tag shedding rate

### 4.7. Derived quantities

Derived quantities represent a snap shot of abundance or biomass of the partition at a specified point in time. The most common use of derived quantities is to calculate spawning stock biomasses. Derived quantities are associated with the **mortality block** of that time step. This allows you to specify whether the quantity should be at the beginning of mortality, at the end or part-way through a mortality event.

### 4.8. Age-length relationship

The age-length relationship defines the length at age (and the weight at length, see Section 4.8) of individuals at age/category within the model. There are three length-age relationships available in CASAL2. The first is the naive no relationship (where each individual has length 1 irrespective of age). The second and third are the von-Bertalanffy and Schnute relationships respectively. The length-at-age relationship is used to determine the length frequency, given age, and then with the length-weight relationship, a weight-at-age of individuals within an age/category.

The three age-length relationships are,

None: where the length of each individual is exactly 1 for all ages, in which case the none length-weight relationship must also be used.

von Bertalanffy: where length at age is defined as,

$$\bar{s}(\text{age}) = L_{\infty} (1 - \exp(-k(\text{age} - t_0))) \quad (4.17)$$

Schnute: where length at age is defined as,

$$\bar{s}(\text{age}) = \begin{cases} \left[ y_1^b + (y_2^b - y_1^b) \frac{1 - \exp(-a(\text{age} - \tau_1))}{1 - \exp(-a(\tau_2 - \tau_1))} \right]^{1/b}, & \text{if } a \neq 0 \text{ and } b \neq 0 \\ y_1 \exp \left[ \ln(y_2/y_1) \frac{1 - \exp(-a(\text{age} - \tau_1))}{1 - \exp(-a(\tau_2 - \tau_1))} \right], & \text{if } a \neq 0 \text{ and } b = 0 \\ \left[ y_1^b + (y_2^b - y_1^b) \frac{\text{age} - \tau_1}{\tau_2 - \tau_1} \right]^{1/b}, & \text{if } a = 0 \text{ and } b \neq 0 \\ y_1 \exp \left[ \ln(y_2/y_1) \frac{\text{age} - \tau_1}{\tau_2 - \tau_1} \right], & \text{if } a = 0 \text{ and } b = 0 \end{cases} \quad (4.18)$$

The von Bertalanffy curve is parameterised by  $L_{\infty}$ ,  $k$ , and  $t_0$ ; the Schnute curve (Schnute, 1981) by  $y_1$  and  $y_2$ , which are the mean lengths at reference ages  $\tau_1$  and  $\tau_2$ , and  $a$  and  $b$  (when  $b = 1$ , this reduces to the von Bertalanffy with  $k = a$ ).

When defining length-at-age in CASAL2, you must also define a length-weight relationship (see Section 4.8 below).

## Calculation of length-at-age (in an age-based model)

### Interpolation of length-at-age

### Size-weight relationship

There are two length-weight relationship,s available in CASAL2. The first is the naive no relationship. Here, the weight of an individual, regardless of length, is always 1. The second is the basic relationship.

The two length-weight relationships are,

- None: The length-weight relationship where

$$\text{mean weight} = 1 \quad (4.19)$$

- Basic: The length-weight relationship where the mean weight  $w$  of an individual of length  $l$  is

$$w = al^b \quad (4.20)$$

Note that if a distribution of length-at-age is specified, then the mean weight is calculated over the distribution of lengths, and is

$$w = (al^b)(1 + cv^2)^{\frac{b(b-1)}{2}} \quad (4.21)$$

where the  $cv$  is the c.v. of lengths-at-age. This adjustment is exact for lognormal distributions, and a close approximation for normal distributions if the c.v. is not large (Bull et al., 2012).

Be careful about the scale of  $a$  — this can easily be 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. Note that there are reports available that can be used to help check that the units specified are plausible (see Section 7).

### **Calculation of mean weight**

#### **4.9. Weightless model**

#### **4.10. Maturity, in models without maturing in the partition**

### 4.11. Selectivities

A selectivity is a function that can have a different value for each age class. Selectivities are used throughout CASAL2 to interpret observations (Section 5) or to modify the effects of processes on each age class (Section 4). CASAL2 implements a number of different parametric forms, including logistic, knife edge, and double normal selectivities.

**CASAL concept that has been removed can be removed?** A selectivity is always defined to apply just to one category of the partition. To apply the same selectivity to more than one category, then just repeat the selectivity for each category that it is applied to.

Note that selectivities are indexed by age, with indices from `min_age` to `max_age`. For example, you might have an age-based selectivity that was logistic with 50% selected at age 5 and 95% selected at age 7. This would be defined by the `type=logistic` with parameters  $a_{50} = 5$  and  $a_{95} = (7 - 5) = 2$ . Then the value of the selectivity at age  $x = 7$  is 0.95 and the selectivity at  $x = 3$  is 0.05.

Note that the function values for some choices of parameters for some selectivities 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). CASAL2 implements range checks on some parameters to test for a possible numeric overflow error before attempting to calculate function values. For example, the logistic selectivity is implemented such that if  $(a_{50} - x)/a_{95} > 5$  then the value of the selectivity at  $x = 0$ , i.e., for  $a_{50} = 5$ ,  $a_{95} = 0.1$ , then the value of the selectivity at  $x = 1$ , without range checking would be  $7.1 \times 10^{-52}$ . With range checking, that value is 0 (as  $(a_{50}x)/a_{95} = 40 > 5$ ).

The available selectivities are;

- Constant
- Knife-edge
- All values
- All values bounded
- Increasing
- Logistic
- Inverse logistic
- Logistic producing
- Double normal
- Double exponential
- Cubic spline

The available selectivities are described below.

#### 4.11.1. constant

$$f(x) = C \tag{4.22}$$

The constant selectivity has the estimable parameter  $C$ .

**4.11.2. knife\_edge**

$$f(x) = \begin{cases} 0, & \text{if } x < E \\ \alpha, & \text{if } x \geq E \end{cases} \quad (4.23)$$

The knife-edge ogive has the estimable parameter  $E$  and a scaling parameter  $\alpha$ , where the default value of  $\alpha = 1$

**4.11.3. all\_values**

$$f(x) = V_x \quad (4.24)$$

The all-values selectivity has estimable parameters  $V_{low}, V_{low+1} \dots V_{high}$ . Here, you need to provide the selectivity value for each age class.

**4.11.4. all\_values\_bounded**

$$f(x) = \begin{cases} 0, & \text{if } x < L \\ V_x, & \text{if } L \leq x \leq H \\ V_H, & \text{if } x > H \end{cases} \quad (4.25)$$

The all-values-bounded selectivity has non-estimable parameters  $L$  and  $H$ . The estimable parameters are  $V_L, V_{L+1} \dots V_H$ . Here, you need to provide an selectivity value for each age class from  $L \dots H$ .

**4.11.5. increasing**

$$f(x) = \begin{cases} 0, & \text{if } x < L \\ f(x-1) + \pi_x(\alpha - f(x-1)), & \text{if } L \leq x \leq H \\ f(\alpha), & \text{if } x \geq H \end{cases} \quad (4.26)$$

The increasing ogive has non-estimable parameters  $L$  and  $H$ . The estimable parameters are  $\pi_L, \pi_{L+1} \dots \pi_H$  (but if these are estimated, they should always be constrained to be between 0 and 1).  $\alpha$  is a scaling parameter, with default value of  $\alpha = 1$ . Note that the increasing ogive is similar to the all-values-bounded ogive, but is constrained to be non-decreasing.

**4.11.6. logistic**

$$f(x) = \alpha / [1 + 19^{(a_{50}-x)/a_{t095}}] \quad (4.27)$$

The logistic selectivity has estimable parameters  $a_{50}$  and  $a_{t095}$ .  $\alpha$  is a scaling parameter, with default value of  $\alpha = 1$ . The logistic selectivity takes values  $0.5\alpha$  at  $x = a_{50}$  and  $0.95\alpha$  at  $x = a_{50} + a_{t095}$ .

**4.11.7. inverse\_logistic**

$$f(x) = \alpha - \alpha / [1 + 19^{(a_{50}-x)/a_{t095}}] \quad (4.28)$$

The inverse logistic selectivity has estimable parameters  $a_{50}$  and  $a_{t095}$ .  $\alpha$  is a scaling parameter, with default value of  $\alpha = 1$ . The logistic selectivity takes values  $0.5\alpha$  at  $x = a_{50}$  and  $0.95\alpha$  at  $x = a_{50} - a_{t095}$ .

**4.11.8. logistic\_producing**

$$f(x) = \begin{cases} 0, & \text{if } x < L \\ \lambda(L), & \text{if } x = L \\ (\lambda(x) - \lambda(x-1)) / (1 - \lambda(x-1)), & \text{if } L < x < H \\ 1, & \text{if } x \geq H \end{cases} \quad (4.29)$$

The logistic-producing selectivity has the non-estimable parameters  $L$  and  $H$ , and has estimable parameters  $a_{50}$  and  $a_{t095}$ .  $\alpha$  is a scaling parameter, with default value of  $\alpha = 1$ . For category transitions,  $f(x)$  represents the proportion moving, not the proportion that have moved. This selectivity was designed for use in an age-based model to model maturity. In such a model, a logistic-producing maturation selectivity will (in the absence of other influences) make the proportions mature follow a logistic curve with parameters  $a_{50}$ ,  $a_{t095}$ .

**4.11.9. double\_normal**

$$f(x) = \begin{cases} \alpha 2^{-[(x-\mu)/\sigma_L]^2}, & \text{if } x \leq \mu \\ \alpha 2^{-[(x-\mu)/\sigma_R]^2}, & \text{if } x \geq \mu \end{cases} \quad (4.30)$$

The double-normal selectivity has estimable parameters  $a_1$ ,  $s_L$ , and  $s_R$ .  $\alpha$  is a scaling parameter, with default value of  $\alpha = 1$ . It has values  $\alpha$  at  $x = a_1$ , and  $0.5\alpha$  at  $x = a_1 - s_L$  and  $x = a_1 + s_R$ .

**4.11.10. double\_exponential**

$$f(x) = \begin{cases} \alpha y_0 (y_1/y_0)^{(x-x_0)/(x_1-x_0)}, & \text{if } x \leq x_0 \\ \alpha y_0 (y_2/y_0)^{(x-x_0)/(x_2-x_0)}, & \text{if } x > x_0 \end{cases} \quad (4.31)$$

The double-exponential selectivity has non-estimable parameters  $x_1$  and  $x_2$ , and estimable parameters  $x_0$ ,  $y_0$ ,  $y_1$ , and  $y_2$ .  $\alpha$  is a scaling parameter, with default value of  $\alpha = 1$ . It can be ‘U-shaped’. Bounds for  $x_0$  must be such that  $x_1 < x_0 < x_2$ . With  $\alpha = 1$ , the selectivity 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 selectivity is ‘U-shaped’ with minimum at  $(x_0, y_0)$ .

**4.11.11. spline**

The spline selectivity implements a cubic spline that has non-estimable knots, and an estimable value for each knot. The cubic spline is either (i) a natural splines where the second derivatives are set to 0 at the boundaries, i.e., the values at the boundaries are horizontal, (ii) a spline with a fixed first derivative at the boundaries (linear, but not necessarily horizontal) and (iii) spline which turns into a parabola at the boundaries.

---

## 5. The estimation section

### 5.1. Role of the estimation section

*check and confirm text*

*KL comment: This text from SPM, a rewrite of CASAL. OK to use?*

The role of the estimation section is to define the tasks carried out by CASAL2:

1. Define the objective function (see Section 5.2)
2. Define the parameters to be estimated (see Section 5.3)
3. Calculate a point estimate, i.e., the maximum posterior density estimate (MPD) (see Section 5.4).
4. Calculate a posterior profile selected parameters, i.e., find, for each of a series of values of a parameter, allowing the other estimated parameters to vary, the minimum value of the objective function (see Section 5.5).
5. Generate an MCMC sample from the posterior distribution (see Section 5.6).
6. Calculate the approximate covariance matrix of the parameters as the inverse of the minimizer's approximation to the Hessian, and the corresponding correlation matrix (see Section 5.4).

The estimation section defines the objective function, parameters of the model, and the method of estimation (point estimates, Bayesian posteriors, profiles, etc.). The objective function is based on a goodness-of-fit measure of the model to observations, priors and penalties. See the observation section for a description of the observations, likelihoods, priors and penalties.

### 5.2. The objective function

*check and confirm text*

*KL comment: This text from SPM. Does not cover all text in S6.7 of CASAL, e.g. max likelihood. OK to use? Additions required?*

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

$$Objective(p) = -\sum_i \log [L(\mathbf{p}|O_i)] - \log [\pi(\mathbf{p})] \quad (5.1)$$

where  $\pi$  is the joint prior density of the parameters  $p$ .

The contribution to the objective function from the likelihoods are defined in Section 6.1. In addition to likelihoods, priors (see Section 5.7) and penalties (see Section 5.8) are components of the objective function.

Penalties can be used to ensure that the exploitation rate constraints on mortality events (i.e., fisheries) are not breached (otherwise there is nothing to prevent the model from having abundances so low that the recorded mortalities could not have been taken), penalties on category transitions (to ensure there are enough individuals to move), and possibly penalties to encourage estimated values to be similar or smooth, etc.



### 5.3. Specifying the parameters to be estimated

*check and confirm text*

*KL comment: This text from SPM. Does not cover all the text in CASAL, e.g. catchabilities. OK to use? Additions required?*

The estimable parameters that will be estimated are defined using `@estimate` commands (see Section 9). An `@estimate` command-block looks like,

```
@estimate process[MyRecruitment].r0
lower_bound 1000
upper_bound 100000
prior uniform
```

See Section 3.5.5 for instructions on how to generate the parameter name. At least one parameter to be estimated if doing an estimation, profile, or MCMC run. Initial values for the parameters to be estimated will still need to be provided, and these are used as the starting values for the minimiser. However, these may be overwritten if you provide a set of alternative starting values (i.e., using `casal2 -i`, see Section 3.4).

All parameters are estimated within bounds. For each parameter to be estimated, you need to specify the bounds and the prior (Section 5.7). Note that the bounds and prior for each parameter refer to the values of the parameters, not the actual values resulting from the application of the parameter to an equation. **Bounds should be carefully chosen as they effect the space in which the minimisers search over. Minimisers convert lower and upper bound to -1,1 space. Needs to be referenced.** If estimating only some elements of a vector, either define the elements of the vector to be estimated (see 3.5.5) or fix the others by setting the bounds equal.

The estimation of parameters can be phased. Here, some of the estimated parameters are initially held fixed, and a minimisation is carried out. Next, some or all of the remaining parameters that were initially held fixed are freed, and another minimisation is carried out. This process continues until all phases have been carried out.

### 5.4. Point estimation

*check and confirm text*

*KL comment: This text from SPM. A rewrite of what's in CASAL and set out differntly. OK to use? Additions required?*

Point estimation is invoked with `casal2 -e`. Mathematically, it is an attempt to find a minimum of the objective function. SPM has two algorithms for solving (minimising) the optimisation problem. The first uses a quasi-Newton minimiser built which is a slightly modified implementation of the main algorithm of Dennis Jr. & Schnabel (Dennis Jr and Schnabel, 1996), while the second uses a genetic algorithm developed by Storn & Price (Storn and Price, 1995), the differential evolution minimiser. However, the second minimiser does not produce an estimate of the covariance matrix, and hence cannot be used to start an MCMC.

#### 5.4.1. The numerical differences minimiser

*check and confirm text*

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

1. Successful convergence (suggests you have found a local minimum, at least).
2. Convergence failure (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 halted but was unable to determine if convergence occurred. You may be at a local minimum, although you should check by restarting the minimiser at the final values of the estimated 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 recommend 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 an alternative starting point of the minimiser using `casal2 -i`.

We want to stress that this is a local optimisation algorithm trying to solve a global optimisation problem. What this means is that, even if you get a ‘successful convergence’ message, your solution may be only a local minimum, not a global one. To diagnose this problem, try doing multiple runs from different starting points and comparing the results, or doing profiles of one or more key parameters and seeing if any of the profiled estimates finds a better optimum than the original point estimate.

The approximate covariance matrix of the estimated parameters can be calculated as the inverse of the minimiser’s approximation to the Hessian, and the corresponding correlation matrix is also calculated. Be aware 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
- the inverse Hessian is not a good approximation to the covariance matrix of the estimated parameters, and may not be useful to construct, for example, confidence intervals.

Also note that if an estimated 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.

#### 5.4.2. The differential evolution minimiser

##### *check and confirm text*

The differential evolution minimiser is a simple population based, stochastic function minimizer, but is claimed to be quite powerful in solving minimisation problems. It is a method of mathematical optimization of multidimensional functions and belongs to the class of evolution strategy optimizers. Initially, the procedure randomly generates and evaluates a number of solution vectors (the population size), each with  $p$  parameters. Then, for each generation (iteration), the algorithm creates a candidate solution for each existing solution by random mutation and uniform crossover. The random mutation generates a new solution by multiplying the difference between two randomly selected solution vectors by some scale factor, then adding the result to a third vector. Then an element-wise crossover takes place with probability  $P_{cr}$ , to generate a potential candidate solution. If this is better than the initial solution vector, it replaces it, otherwise the original solution is retained. The algorithm is terminated after either a predefined number of generations (`max_generations`) or when the maximum difference between the scaled individual parameters from the candidate solutions from all populations is less than some predefined amount `tolerance`.

The differential evolution minimiser can be good at finding global minimums in surfaces that may

have local minima. However, the speed of the minimiser, and the ability to find a good minima depend on the number of initial ‘populations’. Some authors recommend that the number of populations be set at about  $10 * p$ , where  $p$  is the number of free parameters. However, depending on your problem, you may find that you may need more, or that less will suffice.

We note that there is no proof of convergence for the differential evolution solver, but several papers have found it to be an efficient method of solving multidimensional problems. Our (limited) experience suggests that it can often find a better minima and may be faster or longer (depending on the actual model specification) at finding a solution when compared with the numerical differences minimiser. Comparisons with auto-differentiation minimisers or other more sophisticated algorithms have not been made.

### 5.4.3. The Betadiff minimiser

An auto-differentiable minimiser for non-linear models.

## 5.5. Posterior profiles

*check and confirm text*

*KL comment:* This text from SPM. A rewrite of what’s in CASAL, except for par. 5 and 7. OK to use? Additions required?

If profiles are requested `casal2 -p`, CASAL2 will first calculate a point estimate. For each scalar parameter or, in the case of vectors or selectivities, the element of the parameter to be profiled, CASAL2 will fix its value at a sequence of  $n$  evenly spaced numbers (*step*) between a specified lower and upper bounds  $l$  and  $u$ , and calculate a point estimate at each value.

By default  $step = 10$ , and  $(l, u) = (\text{lower bound on parameter plus } (range/(2n)), \text{upper bound on parameter less } (range/(2n)))$ . Each minimisation starts at the final parameter values from the previous resulting value of the parameter being profiled. CASAL2 will report the objective function for each parameter value. Note that an initial point estimate should be compared with the profile, not least to 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 the number of steps, lower bound, and upper bound for each. In the case of vector parameters, you will also need to specify the element of the vector being profiled.

You can also supply the initial starting point for the estimation using `casal2 -i file` — this may improve the minimiser performance for the profiles.

If you get an implausible profile, it may be a result of not using enough iterations in the minimiser or a poor choice of minimiser control variables (e.g., the minimiser tolerance). It also may be useful to try both if the minimisers in CASAL2 and compare the results.

## 5.6. Bayesian estimation

*check and confirm text*

*KL comment:* This text is from SPM and is nearly verbatim S6.5 in CASASL, but two large sections excluded: ...request covariate matrix change adaptively... and from ...multivariate t dist... onwards. OK to use/ Additions required?

CASAL2 can use a Monte Carlo Markov Chain (MCMC) to generate a sample from the posterior distribution of the estimated parameters `casal2 -m` and output the sampled values to a file (optionally keeping only every  $n$ th set of values).

As CASAL2 has no post-processing capabilities. CASAL2 cannot produce MCMC convergence diagnostics (use a package such as BOA) or 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. (1994) 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 CASAL2. See Section 9.3 for a better description of the sequence of CASAL2 commands used in a full Bayesian analysis.

CASAL2 uses a straightforward implementation of the Metropolis-Hastings algorithm (Gelman et al., 1995, Gilks et al., 1994). The Metropolis-Hastings algorithm attempts to draw a sample from a Bayesian posterior distribution, and calculates the posterior density  $\pi$ , scaled by an unknown constant. The algorithm generates a ‘chain’ or sequence of values. Typically the beginning of the chain is discarded and every  $N$ th 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:

- Draw a candidate step  $s$  from a proposal distribution  $J$ , which should be symmetric i.e.,  $J(-s) = J(s)$ .
- Calculate  $r = \min(\pi(x_i + s)/\pi(x_i), 1)$ .
- 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 estimated 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 `casal2 -i`. Don’t start it too close to the actual estimate (either by using `casal2 -i`, or by changing the initial parameter values in input configuration file) as it takes a few iterations to form a reasonable approximation to the Hessian.

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

- Start from the point estimate.
- 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 may be useful if the chain gets ‘stuck’ at the point estimate, or if you wish to generate multiple chains from for later MCMC diagnostic tests.
- Start from a point specified by the user with `casal2 -i` (*was NYI, to be included?*)

The chain moves in natural space, i.e., no transformations are applied to the estimated parameters. The default proposal distribution is a multivariate  $t$  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:

- The covariance matrix is taken as the inverse of the approximate Hessian from the quasi-Newton minimiser.
- The covariance matrix is modified so as to decrease all correlations greater than `@mcmc.max_correlation` down to `@mcmc.max_correlation`, and similarly to increase all correlations less than `-@mcmc.max_correlation` up to `-@mcmc.max_correlation` (the `@mcmc.max_correlation` parameter defaults to 0.8). This should help to avoid getting ‘stuck’ in a lower-dimensional subspace.
- The covariance matrix is then modified either by,
  - if `@mcmc.adjustment_method=covariance`: that if the variance of the  $i$ th parameter is non-zero and less than `@mcmc.min_difference` 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

$$\text{Cov}(i, j)' = \sqrt{k} \text{Cov}(i, j) / \text{sd}(i)$$

for  $i \neq j$ , and  $\text{var}(i)' = k$

- if `@mcmc.adjustment_method=correlation`: that if the variance of the  $i$ th parameter is non-zero and less than `@mcmc.min_difference` times the difference between the parameters’ lower and upper bound, then its variance is changed to  $k = \min\_diff(upper\_bound_i - lower\_bound_i)$ . 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 each estimated parameter to move in the MCMC even if its variance is very small according to the inverse Hessian. In both cases, the `@mcmc.min_difference` parameter defaults to 0.0001.

- The `@mcmc.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 estimated parameters, as recommended by Gelman et al. (Gelman et al., 1995). However, you may find that a smaller value may often be better.

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. The adaptive mechanisms are as follows:

1. You can request that the stepsize change adaptively at one or more sample numbers. At each adaptation, the stepsize 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. (Gelman et al., 1995) for justification.)
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. *(was NYI, to be included?)*

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 CASAL2 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). *(was NYI, to be included?)*

The variance-covariance matrix of this sub-sample of chain is calculated. As above, correlations greater than `@mcmc.max_correlation` are reduced to `@mcmc.max_correlation`, correlations less than `@mcmc.max_correlation` are increased to `@mcmc.max_correlation`, and very small non-zero variances are increased (`@mcmc.covariance_adjustment` and `@mcmc.min_difference`). The result is the new variance-covariance matrix of the proposal distribution. *(was NYI, to be included?)*

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

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. The number of covariance modifications by each iteration is recorded as a column on the objectives file. *(was NYI, to be included?)*

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 recorded.

You have the option to specify that some of the estimated parameters are fixed during the 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 `casal2 -i`, these fixed parameters are set to the values in the file. *(was NYI, to be included?)*

The posterior sample can be used for (projections (Section 4.5.4)(*was NYI, to be included?*)) or simulations (Section 6.7) with the values supplied using `casal2 -i file`.

*(following from CASAL, to be included?)*

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 `nth` 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.

## 5.7. Priors

In a Bayesian analysis, you need to give a prior for every parameter that is being estimated. There are no default priors.

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.

CASAL2 has the following priors (expressed in terms of their contribution to the objective function):

1. Uniform

$$-\log(\pi(p)) = 0 \tag{5.2}$$

2. Uniform-log (i.e.,  $\log(p) \sim \text{uniform}$ )

$$-\log(\pi(p)) = \log(p) \tag{5.3}$$

3. Normal with mean  $\mu$  and c.v.  $c$

$$-\log(\pi(p)) = 0.5 \left( \frac{p - \mu}{c\mu} \right)^2 \tag{5.4}$$

4. Normal with mean  $\mu$  and standard deviation  $\sigma$

$$-\log(\pi(p)) = 0.5 \left( \frac{p - \mu}{\sigma} \right)^2 \tag{5.5}$$

5. Lognormal with mean  $\mu$  and c.v.  $c$

$$-\log(\pi(p)) = \log(p) + 0.5 \left( \frac{\log(p/\mu)}{s} + \frac{s}{2} \right)^2 \tag{5.6}$$

where  $s$  is the standard deviation of  $\log(p)$  and  $s = \sqrt{\log(1+c^2)}$ .

(following from CASAL, to be included?)

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

6. Beta with mean  $\mu$  and standard deviation  $\sigma$ , and range parameters  $A$  and  $B$

$$-\log(\pi(p)) = (1-m)\log(p-A) + (1-n)\log(B-p) \quad (5.7)$$

where  $v = \frac{\mu-A}{B-A}$ , and  $\tau = \frac{(\mu-A)(B-\mu)}{\sigma^2} - 1$  and then  $\mu = \tau v$  and  $n = \tau(1-v)$ . Note that the beta prior is undefined when  $\tau \leq 0$ .

(following from CASAL, to be included?)

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

## 5.8. Penalties

*check and confirm text*

*KL comment:* This text from SPM, currently iSAM coded as described here. CASAL documentation (S6.7.6) includes 11 penalties. Will this be included?

Penalties can be used to encourage or discourage parameter values or model outputs that are unlikely to be sensible, by adding a penalty to the objective function. For example, parameter estimates that do not allow a known mortality event to remove enough individuals from the population can be discouraged with an event mortality penalty. CASAL2 requires penalty functions for processes that move or shift a *number* of individuals between categories or from the partition.

For most penalties, you need to specify a multiplier, and the objective function is increased by this multiplier times the penalty value as described below. In some cases you will need to make the multiplier quite large to prohibit some model behaviour.

Currently, the penalties for the processes `@process[label].type=event_mortality` and `@process[label].type=category_transition` are the only penalties implemented.



For both of these processes, two types of penalty can be defined, natural scale (the default) and log scale. Both of these types add a penalty value of the squared difference between the observed value (i.e., the actual number of individuals to be removed in an event mortality process or the actual number of individuals to shift in a category transition process), and the number that were moved (if less than or equal), times the penalty multiplier.

The natural scale penalty just uses at the squared difference on a natural scale, while the log scale penalty uses the squared difference of the logged values.

---

## 6. The observation section

### 6.1. Observations and likelihoods

Observations are typically supplied as observations at an instance in time, over some spatially aggregated area. Time series of observations can be supplied as separate observations for each year or point in time.

CASAL2 allows the following types of observations;

- Observations of proportions by age class within categories
- Observations of proportions between categories within age classes
- Relative and absolute abundance/biomass observations

The definitions for each type of observation are described below, including how the observed values should be supplied, how CASAL2 calculates the expected values, and the likelihoods that are available for each type of observation.

CASAL2 evaluates the observations at the end of a time-step (i.e., after all of the processes for that time-step have been applied). However, the observation can be applied to the abundance at the start of a time-step or part-way through a time-step by the use of the `proportion_time_step` subcommand.

By default (i.e., if `proportion_method = mean`), the partition at some point  $p$  during the time-step is then evaluated as the weighted sum between the start and end of the time-step, i.e, for any element  $i$  in the partition,  $n_i = (1 - p)n_i^{start} + pn_i^{end}$ . Note that it may not be sensible to use a value other than one, depending on the processes that happen during the time-step (for example, if the time-step contains an ageing process).

If the `proportion_method = difference`, then the observation is of the *difference* between the population state at the start of the time-step and the end. This can be used to generate expected values for observations of, for example removals due to a mortality event, by only having a single process in the time-step. In this case, the `proportion_time_step` is simply a multiplier of the population state.

### 6.2. Proportions-at-age observations

Proportions-at-age observations are observations of either the relative number of individuals at age or relative biomass at age, via some selectivity.

The observation is supplied for a given year and time-step, for some selected age classes of the population (i.e., for a range of ages multiplied by a selectivity), for categories aggregated over a set of spatial cells. Note that the categories defined in the observations must have an associated selectivity, defined by `selectivities`.

The age range must be ages defined in the partition (i.e., between `@model.min_age` and `@model.max_age` inclusive), but the upper end of the age range can optionally be a plus group — which must be either the same or less than the plus group defined for the partition.

Proportions-at-age observations can be supplied as;

1. a set of proportions for a single category,
2. a set of proportions for multiple categories, or
3. a set of proportions across aggregated categories.

For example, for a model with the two categories *male* and *female*, we might supply either (i) a set of proportions for a single category (i.e., males) within each age class; (ii) a set of proportions describing the proportions of individuals within each age class across multiple categories (i.e., males and females) simultaneously, or (iii) a set of proportions for the total number of individuals over the aggregated categories (i.e., males + females) combined, within each age class.

The way the categories of the observation are defined specifies which of these alternatives are used. It is also possible to have an observation with multiple and aggregated categories simultaneously.

### Proportions-at-age for a single category

This form of defining the observation is the simplest, and is used to model a set of proportions of a single category by age class. For example, to specify that the observations are of the proportions of male within each age class, then the subcommand `categories` for the `@observation[label].type=proportion.by_age` command is,

```
categories male
```

CASAL2 then expects that there will be a single vector of proportions supplied, with one proportion for each age class within the defined age range, and that these proportions sum to one.

For example, if the age range was 3 to 10, then 8 proportions should be supplied (one proportion for each of the ages 3, 4, 5, 6, 7, 8, 9, and 10). The expected values will be the expected proportions of males within each of these age classes (after ignoring any males aged less than 3 or older than 10), after applying a selectivity at the year and time-step specified. The supplied vector of proportions (i.e., in this example, the 8 proportions) must sum to one, which is evaluated with a default tolerance of 0.001.

The observations must be also supplied using all or some of the values of defined by some *categorical* layer. CASAL2 calculates the expected values by summing over the defined ages (via the age range and selectivity) and categories for those spatial cells where the categorical layer has the same value as defined for each vector of observations.

For example, in a  $2 \times 2$  spatial model a categorical layer (e.g., with label *Area*) may define that cells (1, 1) and (1, 2) have value *A* and cells (2, 1) and (2, 2) have value *B*, i.e.,

```
@layer Area
type categorical
data A A
data B B
```

The observations for those spatial cells where the categorical layer has value *A* would be,

```
@observation MyProportions
type proportions_at_age
layer Area
...
categories male
min_age 1
max_age 5
obs A 0.01 0.09 0.20 0.30 0.40
...
```

Or, for both *A* and *B* as,

```
@observation MyProportions
type proportions_at_age
layer Area
...
categories male
min_age 1
max_age 5
obs A 0.01 0.09 0.20 0.30 0.40
obs B 0.02 0.06 0.12 0.25 0.55
...
```

Note that to have an observation for each individual spatial cell in a model, then define a categorical layer that has a single, unique value for each spatial cell for use in the observation.

### Proportions-at-age for multiple categories

This form of the observation extends the idea above for multiple categories. It is used to model a set of proportions over several categories by age class. For example, to specify that the observations are of the proportions of male or females within each age class, then the subcommand `categories` for the `@observation[label].type=proportion_by_age` command is,

```
categories male female
```

CASAL2 then expects that there will be a single vector of proportions supplied, with one proportion for each category and age class combination, and that these proportions sum to one.

For example, if there were two categories and the age range was 3 to 10, then 16 proportions should be supplied (one proportion for each of the the ages 3, 4, 5, 6, 7, 8, 9, and 10, for each category male and female). The expected values will be the expected proportions of males and within each of these age classes (after ignoring those aged less than 3 or older than 10), after applying a selectivity at the year and time-step specified. The supplied vector of proportions (i.e., in this example, the 16 proportions) must sum to one, which is evaluated with a default tolerance of 0.001.

For example, using the earlier spatial model with a categorical layer that has label *Area*, the observations for those spatial cells where the categorical layer has value *A* would be,

```
@observation MyProportions
type proportions_at_age
layer Area
...
categories male female
min_age 1
max_age 5
obs A 0.01 0.05 0.10 0.20 0.20 0.01 0.05 0.15 0.20 0.03
obs B 0.02 0.06 0.10 0.21 0.18 0.02 0.05 0.15 0.20 0.01
...
```

### Proportions-at-age across aggregated categories

This form of the observation extends the idea above, but allows categories to be aggregated before the proportions are calculated. It is used to model a set of proportions from several categories that have been combined by age class. To indicate that two (or more) categories are to be aggregated,

separate them with a '+' symbol. For example, to specify that the observations are of the proportions of male and females combined within each age class, then the subcommand `categories` for the `@observation[label].type=proportion.by-age` command is,

```
categories male + female
```

CASAL2 then expects that there will be a single vector of proportions supplied, with one proportion for each age class, and that these proportions sum to one.

For example, if there were two categories and the age range was 3 to 10, then 8 proportions should be supplied (one proportion for each of the ages 3, 4, 5, 6, 7, 8, 9, and 10, for the sum of males and females within each age class). The expected values will be the expected proportions of males + females within each of these age classes (after ignoring those aged less than 3 or older than 10), after applying a selectivity at the year and time-step specified. The supplied vector of proportions (i.e., in this example, the 16 proportions) must sum to one, which is evaluated with a default tolerance of 0.001.

For example, using the earlier spatial model with a categorical layer that has label `Area`, the observations for those spatial cells where the categorical layer has value `A` would be,

```
@observation MyProportions
type proportions_at_age
layer Area
...
categories male + female
min_age 1
max_age 5
obs A 0.02 0.13 0.25 0.30 0.30
obs B 0.02 0.06 0.18 0.35 0.39
...
```

The later form can then be extended to include multiple categories, or multiple aggregated categories. For example, to describe proportions for the three groups: immature males, mature males, and all females (immature and mature females added together) for ages 1–4, a total of 12 proportions are required

```
@observation MyProportions
type proportions_at_age
layer Area
...
categories male_immature male_mature female_immature + female_mature
min_age 1
max_age 4
obs A 0.05 0.15 0.15 0.05 0.02 0.03 0.08 0.04 0.05 0.15 0.15 0.08
...
```

### 6.2.1. Likelihoods for proportions-at-age observations

CASAL2 implements two likelihoods for proportions-at-age observations, the multinomial likelihood and the lognormal likelihood.

#### The multinomial likelihood

For the observed proportions at age  $O_i$  for age classes  $i$ , with sample size  $N$ , and the expected proportions at the same age classes  $E_i$ , the negative log-likelihood is defined as;

$$-\log(L) = -\log(N!) + \sum_i \log((NO_i)!) - NO_i \log(Z(E_i, \delta)) \quad (6.1)$$

where  $\sum_i O_i = 1$  and  $\sum_i E_i = 1$ .  $Z(\theta, \delta)$  is a robustifying function to prevent division by zero errors, with parameter  $\delta > 0$ .  $Z(\theta, \delta)$  is defined as,

$$Z(\theta, \delta) = \begin{cases} \theta, & \text{where } \theta \geq r \\ \delta / (2 - \theta / \delta), & \text{otherwise} \end{cases} \quad (6.2)$$

The default value of  $\delta$  is  $1 \times 10^{-11}$ .

### The lognormal likelihood

For the observed proportions at age  $O_i$  for age classes  $i$ , with c.v.  $c_i$ , and the expected proportions at the same age classes  $E_i$ , the negative log-likelihood is defined as;

$$-\log(L) = \sum_i \left( \log(\sigma_i) + 0.5 \left( \frac{\log(O_i / Z(E_i, \delta))}{\sigma_i} + 0.5 \sigma_i \right)^2 \right) \quad (6.3)$$

where

$$\sigma_i = \sqrt{\log(1 + c_i^2)} \quad (6.4)$$

and the  $c_i$ 's are the c.v.s for each age class  $i$ , and  $Z(\theta, \delta)$  is a robustifying function to prevent division by zero errors, with parameter  $\delta > 0$ .  $Z(\theta, \delta)$  is defined as,

$$Z(\theta, \delta) = \begin{cases} \theta, & \text{where } \theta \geq r \\ \delta / (2 - \theta / \delta), & \text{otherwise} \end{cases} \quad (6.5)$$

The default value of  $\delta$  is  $1 \times 10^{-11}$ .

## 6.3. Proportions-by-category observations

Proportions-by-category observations are observations of either the relative number of individuals between categories within age classes, or relative biomass between categories within age classes.

The observation is supplied for a given year and time-step, for some selected age classes of the population (i.e., for a range of ages multiplied by a selectivity), for categories aggregated over a set of spatial cells.

The age range must be ages defined in the partition (i.e., between `@model.min_age` and `@model.max_age` inclusive), but the upper end of the age range can optionally be a plus group — which may or may not be the same as the plus group defined for the partition.

Proportions-by-category observations can be supplied for any set of categories as a proportion of themselves and any set of additional categories. For example, for a model with the two categories *male* and *female*, we might supply observations of the proportions of males in the population at each age class. The subcommand `categories` defines the categories for the numerator in the calculation of the proportion, and the subcommand `categories2` supplies the additional categories to be used in the denominator of the calculation. In addition, each category must have an associated selectivity, defined by `selectivities` for the numerator categories and `selectivities2` for the additional categories used in the denominator, e.g.,

```
categories male
```

```
categories2 female
selectivities male-selectivity
selectivities2 female-selectivity
```

defines that the proportion of males in each age class as a proportion of males + females. CASAL2 then expects that there will be a vector of proportions supplied, with one proportion for each age class within the defined age range, i.e., if the age range was 3 to 10, then 8 proportions should be supplied (one proportion for each of the ages 3, 4, 5, 6, 7, 8, 9, and 10). The expected values will be the expected proportions of male to male + female within each of these age classes, after applying the selectivities at the year and time-step specified.

The observations must be supplied using all or some of the values defined by a categorical layer. CASAL2 calculates the expected values by summing over the ages (via the age range and selectivity) and categories for those spatial cells where the categorical layer has the same value as defined for each vector of observations.

For example, in a  $2 \times 2$  spatial model a categorical layer (e.g., with label Area) may define that cells (1,1) and (1,2) have value *A* and cells (2,1) and (2,2) have value *B*, i.e.,

```
@layer Area
type categorical
data A A
data B B
```

Here we supply observations for those spatial cells where the categorical layer has value *A* as,

```
@observation MyProportions
type proportions_by_category
layer Area
...
categories male
categories2 female
min_age 1
max_age 5
obs A 0.01 0.05 0.10 0.20 0.20
...
```

Or, for both *A* and *B* as,

```
@observation MyProportions
type proportions_by_category
layer Area
...
categories male
categories2 female
min_age 1
max_age 5
obs A 0.01 0.05 0.10 0.20 0.20
obs B 0.02 0.06 0.10 0.21 0.18
...
```

To supply an observation for individual spatial cells, then you will need to define a categorical layer with a single, unique value for each spatial cell.

### 6.3.1. Likelihoods for proportions-by-category observations

CASAL2 implements two likelihoods for proportions-by-category observations, the binomial likelihood, and the normal approximation to the binomial (binomial-approx).

### The binomial likelihood

For observed proportions  $O_i$  for age class  $i$ , where  $E_i$  are the expected proportions for age class  $i$ , and  $N_i$  is the effective sample size for age class  $i$ , then the negative log-likelihood is defined as;

$$-\log(L) = -\sum_i [\log(N_i!) - \log((N_i(1 - O_i))!) - \log((N_i O_i)!) + N_i O_i \log(Z(E_i, \delta)) + N_i(1 - O_i) \log(Z(1 - E_i, \delta))] \quad (6.6)$$

where  $Z(\theta, \delta)$  is a robustifying function to prevent division by zero errors, with parameter  $\delta > 0$ .  $Z(\theta, \delta)$  is defined as,

$$Z(\theta, \delta) = \begin{cases} \theta, & \text{where } \theta \geq r \\ \delta / (2 - \theta / \delta), & \text{otherwise} \end{cases} \quad (6.7)$$

The default value of  $\delta$  is  $1 \times 10^{-11}$ .

### The normal approximation to the binomial likelihood

For observed proportions  $O_i$  for age class  $i$ , where  $E_i$  are the expected proportions for age class  $i$ , and  $N_i$  is the effective sample size for age class  $i$ , then the negative log-likelihood is defined as;

$$-\log(L) = \sum_i \log \left( \sqrt{Z(E_i, \delta) Z(1 - E_i, \delta) / N_i} \right) + \frac{1}{2} \left( \frac{O_i - E_i}{\sqrt{Z(E_i, \delta) Z(1 - E_i, \delta) / N_i}} \right)^2 \quad (6.8)$$

where  $Z(\theta, \delta)$  is a robustifying function to prevent division by zero errors, with parameter  $\delta > 0$ .  $Z(\theta, \delta)$  is defined as,

$$Z(\theta, \delta) = \begin{cases} \theta, & \text{where } \theta \geq r \\ \delta / (2 - \theta / \delta), & \text{otherwise} \end{cases} \quad (6.9)$$

The default value of  $\delta$  is  $1 \times 10^{-11}$ .

## 6.4. Abundance or biomass observations

Abundance (or biomass) observations are observations of either a relative or absolute number (or biomass) of individuals from a set of categories after applying a selectivity. The observations classes are the same, except that a biomass observation will use the biomass as the observed (and expected) value (calculated from mean weight of individuals within each age and category) while an abundance observation is just the number of individuals.

Each observation is for a given year and time-step, for some selected age classes of the population (i.e., for a range of ages multiplied by a selectivity), for categories aggregated over a set of spatial cells. Further, you need to provide the label of the catchability coefficient  $q$ , which can either be estimated or fixed. For absolute abundance or absolute biomass observations, define a catchability where  $q = 1$ .

The observations can be supplied for any set of categories. For example, for a model with the two categories *male* and *female*, we might supply an observation of the total abundance/biomass (male + female) or just male abundance/biomass. The subcommand `categories` defines the categories used to aggregate the abundance/biomass. In addition, each category must have an associated selectivity, defined by `selectivities`. For example,



```
categories male
selectivities male-selectivity
```

defines an observation for males after applying the selectivity male-selectivity. CASAL2 then expects that there will be a single observation supplied. The expected values for the observations will be the expected abundance (or biomass) of males, after applying the selectivities, at the year and time-step specified.

The observations must be supplied using all or some of the values of defined by a categorical layer. CASAL2 calculates the expected values by summing over the defined ages (via the age range and selectivity) and categories for those spatial cells where the categorical layer has the same value as defined for each vector of observations.

For example, in a  $2 \times 2$  spatial model a categorical layer (e.g., with label Area) may define that cells (1,1) and (1,2) have value *A* and cells (2,1) and (2,2) have value *B*, i.e.,

```
@layer Area
type categorical
data A A
data B B
```

Here we supply abundance observations for those spatial cells where the categorical layer has value *A* as,

```
@observation MyAbundance
type abundance
layer Area
...
categories male
obs A 1000
...
```

Or, for both *A* and *B* as,

```
@observation MyAbundance
type abundance
layer Area
...
categories male
obs A 1000
obs B 1200
...
```

To supply an observation for individual spatial cells, then you will need to define a categorical layer with a single, unique value for each spatial cell.

Note that, to define a biomass observation instead of an abundance observation, use

```
@observation MyBiomass
type biomass
...
```

### 6.4.1. Likelihoods for abundance observations

#### The lognormal likelihood

For observations  $O_i$ , c.v.  $c_i$ , and expected values  $qE_i$ , the negative log-likelihood is defined as;

$$-\log(L) = \sum_i \left( \log(\sigma_i) + 0.5 \left( \frac{\log(O_i/qZ(E_i, \delta))}{\sigma_i} + 0.5\sigma_i \right)^2 \right) \quad (6.10)$$

where

$$\sigma_i = \sqrt{\log(1 + c_i^2)} \quad (6.11)$$

and  $Z(\theta, \delta)$  is a robustifying function to prevent division by zero errors, with parameter  $\delta > 0$ .  $Z(\theta, \delta)$  is defined as,

$$Z(\theta, \delta) = \begin{cases} \theta, & \text{where } \theta \geq r \\ \delta / (2 - \theta/\delta), & \text{otherwise} \end{cases} \quad (6.12)$$

The default value of  $\delta$  is  $1 \times 10^{-11}$ .

#### The normal likelihood

For observations  $O_i$ , c.v.  $c_i$ , and expected values  $qE_i$ , the negative log-likelihood is defined as;

$$-\log(L) = \sum_i \left( \log(c_i E_i) + 0.5 \left( \frac{O_i - E_i}{Z(c_i E_i, \delta)} \right)^2 \right) \quad (6.13)$$

and  $Z(\theta, \delta)$  is a robustifying function to prevent division by zero errors, with parameter  $\delta > 0$ .  $Z(\theta, \delta)$  is defined as,

$$Z(\theta, \delta) = \begin{cases} \theta, & \text{where } \theta \geq r \\ \delta / (2 - \theta/\delta), & \text{otherwise} \end{cases} \quad (6.14)$$

The default value of  $\delta$  is  $1 \times 10^{-11}$ .

### 6.5. Process error

Additional ‘process error’ can be defined for each set of observations. Additional process error has the effect of increasing the observation error in the data, and hence of decreasing the relative weight given to the data in the fitting process.

For observations 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

$$c'_i = \sqrt{c_i^2 + c_{process\_error}^2} \quad (6.15)$$

Note that  $c_{process\_error} \geq 0$ , and that  $c_{process\_error} = 0$  is equivalent to no process error.

Similarly, if the likelihood is parameterised by the effective sample size  $N$ ,

$$N'_i = \frac{1}{1/N_i + 1/N_{process\_error}} \quad (6.16)$$

Note that this requires that  $N_{process\_error} > 0$ , but we allow the special case of  $N_{process\_error} = 0$ , and define  $N_{process\_error} = 0$  as no process error (i.e., defined to be equivalent to  $N_{process\_error} = \infty$ ).

For both the c.v. and  $N$  process errors, the process error has more effect on small errors than on large ones. Be clear that a large value for the  $N$  process error means a small process error.

## 6.6. Ageing error

CASAL2 can apply ageing error age frequency observations. Ageing error is applied to the expected values for proportions-at-age observations. The ageing error is applied as a misclassification matrix, which has the effect of 'smearing' the age frequencies. These are used in calculating the fits to the observed values, and hence the contribution to the total objective function.

Ageing error is optional, and if it is used, it may be omitted for any individual time series. Different ageing error models may be applied for different observation commands. See Section ?? for reporting the misclassification matrix.

The ageing error models implemented are,

1. None: The default model is to apply no ageing error.
2. Off by one: Proportion  $p_1$  of individuals of each age  $a$  are misclassified as age  $a - 1$  and proportion  $p_2$  are misclassified as age  $a + 1$ . Individuals of age  $a < k$  are not misclassified. If there is no plus group in the population model, then proportion  $p_2$  of the oldest age class will 'fall off the edge' and disappear.
3. Normal: Individuals 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 individuals of the older age classes may disappear. If  $c$  is high enough, some of the younger age classes may 'fall off the other edge'. Individuals of age  $a < k$  are not misclassified.

Note that the expected values (fits) reported by CASAL2 for observations with ageing error will have had the ageing error applied.

## 6.7. Simulating observations

CASAL2 can generate simulated observations for a given model with given parameter values (using `spm -s`). Simulated observations are randomly distributed values, generated according to the error assumptions defined for each observation, around fits calculated from one or more sets of the 'true' parameter values. Simulating from a set of parameters can be used to generate observations from an operating model or as a form of parametric bootstrap.

The procedure CASAL2 uses for simulating observations is to first run using the 'true' parameter values and generate the expected values. Then, if a set of observations uses ageing error, ageing error is applied. Finally a random value for each observed value is generated based on (i) the expected values, (ii) the type of likelihood specified, and (iii) the variability parameters (e.g., `error_value` and `process_error`).

Methods for generating the random error, and hence simulated values, depend on the specific likelihood type of each observation.

1. Normal likelihood parameterised by c.v.: Let  $E_i$  be the fitted value for observation  $i$ , and  $c_i$  be the corresponding c.v. (adjusted by the process error if applicable). Each simulated observation value  $S_i$  is generated as an independent normal deviate with mean  $E_i$  and standard deviation  $E_i c_i$ .

2. Log-normal likelihood: Let  $E_i$  be the fitted value for observation  $i$  and  $c_i$  be the corresponding c.v. (adjusted by the process error if applicable). Each simulated observation value  $S_i$  is generated as an independent lognormal deviate with mean and standard deviation (on the natural scale, not the log-scale) of  $E_i$  and  $E_i c_i$  respectively. The robustification parameter  $\delta$  is ignored.
3. Multinomial likelihood: Let  $E_i$  be the fitted value for observation  $i$ , for  $i$  between 1 and  $n$ , and let  $N$  be the sample size (adjusted by process error if applicable, and then rounded up to the next whole number). The robustification parameter  $\delta$  is ignored. Then,
  - (a) A sample of  $N$  values from 1 to  $n$  is generated using the multinomial distribution, using sample probabilities proportional to the values of  $E_i$ .
  - (b) Each simulated observation value  $S_i$  is calculated as the proportion of the  $N$  sampled values equalling  $i$
  - (c) The simulated observation values  $S_i$  are then rescaled so that their sum is equal to 1
4. Binomial and the normal approximation to the binomial likelihoods: Let  $E_i$  be the fitted value for observation  $i$ , for  $i$  between 1 and  $n$ , and  $N_i$  the corresponding equivalent sample size (adjusted by process error if applicable, and then rounded up to the next whole number). The robustification parameter  $\delta$  is ignored. Then,
  - (a) A sample of  $N_i$  independent binary variates is generated, equalling 1 with probability  $E_i$
  - (b) The simulated observation value  $S_i$  is calculated as the sum of these binary variates divided by  $N_i$

Note that CASAL2 will report simulated observations using the usual observation report (`@report[label].type=observation`). The report `@report[label].type=simulated_observation` will generate simulated observations in a form suitable for use as input within a CASAL2 input configuration file. See Section 7 for more detail.

## 6.8. Pseudo-observations

CASAL2 can generate expected values for observations without them contributing to the total objective function. These are called pseudo-observations, and can be used to either generate the expected values from CASAL2 for reporting or diagnostic purposes. To define an observation as a pseudo-observation, use the command `@observation[label].likelihood=none`. Any observation type can be used as a pseudo-observation. CASAL2 can also generate simulated observations from pseudo-observations. Note that;

- Output will only be generated if a report command `@report[label].type=observation` is specified.
- The observed values should be supplied (even if they are ‘dummy’ observation). These will be processed by CASAL2 as if they were actual observation values, and must conform to the validations carried out for the other types of likelihood.
- The subcommands `likelihood`, `obs`, `error_value` and `process_error` have no effect when generating the expected values for the pseudo-observation.
- When simulating observations, CASAL2 needs the subcommand `simulation_likelihood` to tell it what sort of likelihood to use. In this case, the `obs`, `error_value` and `process_error` are used to determine the appropriate terms to use for the likelihood when simulating.



---

## **7. The report section**



---

## 8. Population command and subcommand syntax

### 8.1. Model structure

**@model** *label*     Define an object type Model

*initialisation\_phases*     Define the labels of the phases of the initialisation

    Type: string vector

    Default: true

    Value: A list of valid labels defined by @initialisation\_phase

*start\_year*     Define the first year of the model, immediately following initialisation

    Type: non-negative integer

    Default: No default

    Value: Defines the first year of the model,  $\geq 1$ , e.g. 1990

*age\_plus*     Define the oldest age as a plus group

    Type: boolean

    Default: false

    Value: true, false

*min\_age*     Minimum age of individuals in the population

    Type: non-negative integer

    Default: 0

    Value:  $0 \leq \text{age}_{\min} \leq \text{age}_{\max}$

*length\_bins*

    Type: constant vector

    Default: true

*type*     Type of model (the partition structure). Either age, length or hybrid

    Type: string

    Default: age

*max\_age*     Maximum age of individuals in the population

    Type: non-negative integer

    Default: 0

    Value:  $0 \leq \text{age}_{\min} \leq \text{age}_{\max}$

*final\_year*     Define the final year of the model, excluding years in the projection period

    Type: non-negative integer

    Default: No default

    Value: Defines the last year of the model, i.e., the model is run from *start\_year* to *final\_year*

*time\_steps*     Define the labels of the time steps, in the order that they are applied, to form the



annual cycle

Type: string vector

Default: No default

Value: A list of valid labels defined by @time\_step

projection\_final\_year      Define the final year of the model in projection mode

Type: non-negative integer

Default: 0

Value: Defines the last year of the projection period, i.e., the projection period runs from final\_year+1 to projection\_final\_year. For the default, 0, no projections are run.

## 8.2. Initialisation

**@initialisation\_phase label**      Define an object type Initialisation.Phase

label      Label

Type: string

Default: No default

type      Type

Type: string

Default: iterative

### 8.2.1. @initialisation\_phase[label].type=derived

exclude\_processes      The processes to exclude from all time steps

Type: string vector

Default: true

### 8.2.2. @initialisation\_phase[label].type=iterative

insert\_processes      The processes to insert in to target time steps

Type: string vector

Default: true

lambda      Lambda

Type: constant

Default: Double(0.0)

exclude\_processes      The processes to exclude from all time steps

Type: string vector

Default: true

convergence\_years      The years to test for convergence

Type: non-negative integer vector

Default: true

### 8.2.3. `@initialisation_phase[label].type=state_category_by_age`

`min_age` Minimum age to use for this process

Type: non-negative integer

Default: No default

`max_age` Maximum age to use for this process

Type: non-negative integer

Default: No default

`categories` List of categories to use

Type: string vector

Default: No default

## 8.3. Categories

`@categories label` Define an object type Categories

`age_lengths` The labels of `age_length` objects that are assigned to categories

Type: string vector

Default: true

`names` The names of the categories to be used in the model

Type: string vector

Default: No default

`years` The years that individual categories will be active for. This overrides the model values

Type: string vector

Default: true

`ages` The ages that individual categories support. This overrides the model values

Type: string vector

Default: true

`format` The format that the category names should adhere too

Type: string

Default: No default

## 8.4. Time-steps

`@time_step label` Define an object type Time\_Step

processes      Processes  
Type: string vector  
Default: No default

label      Label  
Type: string  
Default: No default

## 8.5. Processes

**@process** *label*      Define an object type Process

label      Label  
Type: string  
Default: No default

type      Type  
Type: string  
Default: ""

print\_report      Print parameter report  
Type: boolean  
Default: false

### 8.5.1. @process[label].type=ageing

categories      Categories  
Type: string vector  
Default: No default

### 8.5.2. @process[label].type=growth

### 8.5.3. @process[label].type=maturation

rates      The rates to mature for each year  
Type: estimable vector  
Default: No default

selectivities      List of selectivities to use for maturation  
Type: string vector  
Default: No default

to      List of categories to mature too

Type: string vector  
Default: No default

years      The years to be associated with rates  
Type: non-negative integer vector  
Default: No default

#### **8.5.4. @process[label].type=mortality\_constant\_rate**

m      Mortality rates  
Type: estimable vector  
Default: No default

selectivities      Selectivities  
Type: string vector  
Default: No default

categories      List of categories  
Type: string vector  
Default: No default

time\_step\_ratio      Time step ratios for M  
Type: constant vector  
Default: true

#### **8.5.5. @process[label].type=mortality\_event**

penalty      Penalty label  
Type: string  
Default: ""

selectivities      List of selectivities  
Type: string vector  
Default: No default

u\_max      U Max  
Type: estimable  
Default: 0.99

catches      Catches  
Type: estimable vector  
Default: No default

years      Years

Type: non-negative integer vector  
Default: No default

### 8.5.6. `@process[label].type=mortality_event_biomass`

penalty      Penalty label  
Type: string  
Default: ""

selectivities      Selectivity labels  
Type: string vector  
Default: No default

u\_max      U Max  
Type: estimable  
Default: 0.99

catches      Catches for each year  
Type: estimable vector  
Default: No default

years      Years to apply mortality  
Type: non-negative integer vector  
Default: No default

### 8.5.7. `@process[label].type=mortality_instantaneous`

penalty      Label of penalty to be applied  
Type: string  
Default: ""

selectivities      Selectivities for Natural Mortality  
Type: string vector  
Default: No default

difference      The difference between, two categories natural mortality around the average  
Type: estimable  
Default: Double(-999.0)

m      Mortality rates  
Type: estimable vector  
Default: No default  
Lower Bound: 0.0 (inclusive)  
Upper Bound: 1.0 (inclusive)

`categories`      Categories for natural mortality  
Type: string vector  
Default: No default

`time_step_ratio`      Time step ratios for M  
Type: constant vector  
Default: true  
Lower Bound: 0.0 (inclusive)  
Upper Bound: 1.0 (inclusive)

`average`      The average mortality rate  
Type: estimable  
Default: Double(-999.0)

### **8.5.8. `@process[label].type=nop`**

### **8.5.9. `@process[label].type=recruitment_beverton_holt`**

`r0`      R0  
Type: estimable  
Default: No default

`ssb_offset`      SSB Offset (year offset)  
Type: int  
Default: 0

`standardise_ycs_years`  
Type: non-negative integer vector  
Default: true

`ycs_values`      YCS Values  
Type: estimable vector  
Default: No default

`age`      Age to recruit at  
Type: non-negative integer  
Default: true

`b0`      B0 Label  
Type: string  
Default: ""

`steepness`      Steepness

Type: estimable  
Default: 1.0

ssb      SSB Label (derived quantity)  
Type: string  
Default: No default

prior\_standardised\_ycs      Priors for year class strength on ycs values (not standardised ycs values)  
Type: boolean  
Default: true

categories      Category labels  
Type: string vector  
Default: No default

proportions      Proportions  
Type: estimable vector  
Default: No default

#### **8.5.10. @process[label].type=recruitment\_constant**

r0      R0  
Type: estimable  
Default: No default  
Lower Bound: 0.0 (exclusive)

age      Age  
Type: non-negative integer  
Default: No default

proportions      Proportions  
Type: estimable vector  
Default: true

categories      Categories  
Type: string vector  
Default: No default

#### **8.5.11. @process[label].type=tag\_by\_age**

penalty      Penalty label  
Type: string  
Default: ""

u\_max      U Max

Type: constant

Default: 0.99

loss\_rate\_selectivities

Type: string vector

Default: true

selectivities

Type: string vector

Default: No default

min\_age      Minimum age to transition

Type: non-negative integer

Default: No default

to      Categories to transition to

Type: string vector

Default: No default

years      Years to execute the transition in

Type: non-negative integer vector

Default: No default

from      Categories to transition from

Type: string vector

Default: No default

initial\_mortality\_selectivity

Type: string

Default: ""

loss\_rate

Type: constant vector

Default: No default

max\_age      Maximum age to transition

Type: non-negative integer

Default: No default

initial\_mortality

Type: constant

Default: Double(0)



n

Type: constant vector

Default: true

### 8.5.12. @process[label].type=tag\_by\_length

penalty      Penalty label

Type: string

Default: ""

u\_max      U Max

Type: constant

Default: 0.99

loss\_rate\_selectivities

Type: string vector

Default: true

selectivities

Type: string vector

Default: No default

min\_age      Minimum age to transition

Type: non-negative integer

Default: No default

to      Categories to transition to

Type: string vector

Default: No default

years      Years to execute the transition in

Type: non-negative integer vector

Default: No default

from      Categories to transition from

Type: string vector

Default: No default

initial\_mortality\_selectivity

Type: string

Default: ""

loss\_rate

Type: constant vector

Default: No default

`max_age`      Maximum age to transition  
Type: non-negative integer  
Default: No default

`initial_mortality`  
Type: constant  
Default: Double(0)

`n`  
Type: constant vector  
Default: true

### **8.5.13. @process[label].type=transition\_category**

`from`      From  
Type: string vector  
Default: No default

`selectivities`      Selectivity names  
Type: string vector  
Default: No default

`to`      To  
Type: string vector  
Default: No default

`proportions`      Proportions  
Type: estimable vector  
Default: No default

### **8.5.14. @process[label].type=transition\_category\_by\_age**

`penalty`      Penalty label  
Type: string  
Default: ""

`u_max`      U Max  
Type: constant  
Default: 0.99

`min_age`      Minimum age to transition  
Type: non-negative integer  
Default: No default

to      Categories to transition to

Type: string vector

Default: No default

years      Years to execute the transition in

Type: non-negative integer vector

Default: No default

from      Categories to transition from

Type: string vector

Default: No default

max\_age      Maximum age to transition

Type: non-negative integer

Default: No default

## 8.6. Time varying parameters

**@time\_varying** *label*      Define an object type Time\_Varying

parameter      Parameter to vary

Type: string

Default: No default

label      Label

Type: string

Default: No default

type      Type

Type: string

Default: ""

years      Years to recalculate the values

Type: non-negative integer vector

Default: No default

### 8.6.1. @time\_varying[label].type=annual\_shift

scaling\_years

Type: non-negative integer vector

Default: true

years

Type: non-negative integer vector  
 Default: No default

d

Type: constant  
 Default: No default

a

Type: constant  
 Default: No default

c

Type: constant  
 Default: No default

b

Type: constant  
 Default: No default

### 8.6.2. `@time_varying[label].type=constant`

value      Value to assign to estimable  
 Type: constant  
 Default: No default

### 8.6.3. `@time_varying[label].type=exogenous`

exogeneous\_variable      Values of exogeneous variable for each year  
 Type: constant vector  
 Default: No default

a      Shift parameter  
 Type: estimable  
 Default: No default

## 8.7. Derived quantities

`@derived_quantity label`      Define an object type `Derived_Quantity`

time\_step      The time step to calculate the derived quantity after  
 Type: string  
 Default: No default

time\_step\_proportion\_method

Type: string

Default: weighted\_sum

Allowed Values: weighted\_sum, weighted\_product

selectivities      The list of selectivities to use when calculating the derived quantity. 1 per category

Type: string vector

Default: No default

type      Type

Type: string

Default: No default

categories      The list of categories to use when calculating the derived quantity

Type: string vector

Default: No default

time\_step\_proportion

Type: constant

Default: Double(1.0)

**8.7.1. @derived\_quantity[label].type=abundance**

**8.7.2. @derived\_quantity[label].type=biomass**

## **8.8. Age-length relationship**

**@age\_length** *label*      Define an object type Age.Length

distribution      TBA

Type: string

Default: normal

label      Label

Type: string

Default: No default

type      Type

Type: string

Default: No default

time\_step\_proportions

Type: constant vector

Default: true

**8.8.1. @age\_length[label].type=data**

cv\_first      CV for the first age class

Type: constant

Default: Double(0.0

Lower Bound: 0.0 (inclusive)

internal\_gaps

Type: string

Default: mean

Allowed Values: mean, nearest\_neighbour, interpolate

length\_weight      TBA

Type: string

Default: No default

external\_gaps

Type: string

Default: mean

Allowed Values: mean, nearest\_neighbour

cv\_last      CV for maximum age

Type: constant

Default: Double(0.0

Lower Bound: 0.0 (inclusive)

**8.8.2. @age\_length[label].type=none****8.8.3. @age\_length[label].type=schnute**

tau2      TBA

Type: estimable

Default: No default

length\_weight      TBA

Type: string

Default: No default

tau1      TBA

Type: estimable

Default: No default

cv\_first      CV for the first age class

Type: estimable

Default: Double(0.0

Lower Bound: 0.0 (inclusive)

y1      TBA

Type: estimable

Default: No default

y2      TBA

Type: estimable

Default: No default

cv\_last      CV for maximum age

Type: estimable

Default: Double(0.0

Lower Bound: 0.0 (inclusive)

by\_length      TBA

Type: boolean

Default: true

a      TBA

Type: estimable

Default: No default

Lower Bound: 0.0 (inclusive)

b      TBA

Type: estimable

Default: No default

Lower Bound: 0.0 (exclusive)

#### **8.8.4. @age\_length[label].type=von\_bertalanffy**

linf      TBA

Type: estimable

Default: No default

Lower Bound: 0.0 (inclusive)

length\_weight      TBA

Type: string

Default: No default

cv\_first      CV for the first age class

Type: estimable

Default: Double(0.0

Lower Bound: 0.0 (inclusive)

t0      TBA

Type: estimable  
Default: No default

`cv_last` CV for last age class

Type: estimable  
Default: Double(0.0  
Lower Bound: 0.0 (inclusive)

`k` TBA

Type: estimable  
Default: No default  
Lower Bound: 0.0 (inclusive)

`by_length` Specifies if the linear interpolation of CV's is a linear function of mean length at age. Default is just by age

Type: boolean  
Default: true

## 8.9. Length-weight

**@length.weight** *label* Define an object type Length\_Weight

`label` Label

Type: string  
Default: No default

`type` Type

Type: string  
Default: No default

### 8.9.1. @length.weight[label].type=basic

`units` Units of measure (tonnes, kgs, grams)

Type: string  
Default: No default

`a` A

Type: constant  
Default: No default

`b` B

Type: constant  
Default: No default



### 8.9.2. @length\_weight[label].type=none

## 8.10. Selectivities

**@selectivity** *label*     Define an object type Selectivity

type     Type  
           Type: string  
           Default: No default

### 8.10.1. @selectivity[label].type=all\_values

### 8.10.2. @selectivity[label].type=all\_values\_bounded

v     V  
       Type: estimable vector  
       Default: No default

h     H  
       Type: non-negative integer  
       Default: No default

### 8.10.3. @selectivity[label].type=constant

### 8.10.4. @selectivity[label].type=double\_exponential

y1     Y1  
       Type: estimable  
       Default: No default

y0     Y0  
       Type: estimable  
       Default: No default

y2     Y2  
       Type: estimable  
       Default: No default

x2     X2  
       Type: constant  
       Default: No default

alpha     Alpha  
           Type: estimable  
           Default: 1.0

x1      X1  
Type: constant  
Default: No default

#### **8.10.5. @selectivity[label].type=double normal**

sigma\_l      Sigma L  
Type: estimable  
Default: No default

sigma\_r      Sigma R  
Type: estimable  
Default: No default

alpha      Alpha  
Type: estimable  
Default: 1.0

#### **8.10.6. @selectivity[label].type=increasing**

v      V  
Type: estimable vector  
Default: No default

h      High  
Type: non-negative integer  
Default: No default

alpha      Alpha  
Type: constant  
Default: 1.0

#### **8.10.7. @selectivity[label].type=inverse\_logistic**

ato95      aTo95  
Type: estimable  
Default: No default

alpha      Alpha  
Type: estimable  
Default: 1.0

#### 8.10.8. @selectivity[label].type=knife\_edge

alpha     Alpha  
Type: estimable  
Default: 1.0

#### 8.10.9. @selectivity[label].type=logistic

ato95     Ato95  
Type: estimable  
Default: No default

alpha     Alpha  
Type: estimable  
Default: 1.0

#### 8.10.10. @selectivity[label].type=logistic\_producing

ato95     Ato95  
Type: estimable  
Default: No default

alpha     Alpha  
Type: estimable  
Default: 1.0

h          High  
Type: non-negative integer  
Default: No default

a50        A50  
Type: estimable  
Default: No default

### 9. Estimation command and subcommand syntax

#### 9.1. Estimation methods

**@estimate** *label*     Define an object type Estimate

lower\_bound     The lowest value the parameter is allowed to have  
Type: constant  
Default: No default

transformation

Type: string vector

Default: ""

same      A list of parameters that are bound to the value of this estimate

Type: string vector

Default: ""

prior      The name of the prior to use for the parameter

Type: string

Default: ""

upper\_bound      The highest value the parameter is allowed to have

Type: constant

Default: No default

label      Label

Type: string

Default: ""

parameter      The name of the variable to estimate in the model

Type: string

Default: No default

type      Type

Type: string

Default: No default

estimation\_phase      TBA

Type: non-negative integer

Default: 1u

mcmc      TBA

Type: boolean

Default: false

### 9.1.1. @estimate[label].type=beta

mu      Mu

Type: estimable

Default: No default

sigma      Sigma

Type: estimable  
Default: No default  
Lower Bound: 0.0 (exclusive)

a     A  
Type: constant  
Default: No default

b     B  
Type: constant  
Default: No default

### 9.1.2. @estimate[label].type=lognormal

mu     Mu  
Type: estimable  
Default: No default  
Lower Bound: 0.0 (exclusive)

cv     Cv  
Type: estimable  
Default: No default  
Lower Bound: 0.0 (exclusive)

### 9.1.3. @estimate[label].type=normal

mu     Mu  
Type: estimable  
Default: No default

cv     Cv  
Type: estimable  
Default: No default  
Lower Bound: 0.0 (exclusive)

### 9.1.4. @estimate[label].type=normal\_by\_stdev

mu     Mu  
Type: estimable  
Default: No default

sigma   Sigma

Type: estimable  
Default: No default  
Lower Bound: 0.0 (exclusive)

### 9.1.5. `@estimate[label].type=normal_log`

`mu`      `Mu`  
Type: estimable  
Default: No default

`sigma`    `Sigma`  
Type: estimable  
Default: No default  
Lower Bound: 0.0 (exclusive)

### 9.1.6. `@estimate[label].type=uniform`

### 9.1.7. `@estimate[label].type=uniform_log`

## 9.2. Point estimation

`@minimiser label`    Define an object type Minimiser

`active`    True if this minimiser is active  
Type: boolean  
Default: false

`label`    `Label`  
Type: string  
Default: No default

`type`    Type of minimiser to use  
Type: string  
Default: No default

`covariance`    True if a covariance matrix should be created  
Type: boolean  
Default: true

### 9.2.1. `@minimiser[label].type=adolc`

`tolerance`    Tolerance of the gradient for convergence  
Type: constant  
Default: 0.02

step\_size      Minimum Step-size before minimisation fails  
Type: constant  
Default: 1e-7

evaluations    Maximum number of evaluations  
Type: int  
Default: 4000

iterations      Maximum number of iterations  
Type: int  
Default: 1000

### **9.2.2. @minimiser[label].type=beta\_diff**

tolerance      Tolerance of the gradient for convergence  
Type: constant  
Default: 2e-3

evaluations    Maximum number of evaluations  
Type: int  
Default: 4000

iterations      Maximum number of iterations  
Type: int  
Default: 1000

### **9.2.3. @minimiser[label].type=c\_p\_p\_a\_d**

### **9.2.4. @minimiser[label].type=d\_e\_solver**

population\_size    The number of candidate solutions to have in the population  
Type: non-negative integer  
Default: No default

difference\_scale    The scale to apply to new solutions when comparing candidates  
Type: constant  
Default: 0.02

tolerance      The total variance between the population and best candidate before acceptance  
Type: constant  
Default: 0.01

crossover\_probability    TBA

Type: constant

Default: 0.9

`method`      The type of candidate generation method to use

Type: string

Default: ""

Value: not\_yet\_implemented

Allowed Values: ratio, difference, mean

`max_generations`      The maximum number of iterations to run

Type: non-negative integer

Default: No default

### 9.2.5. `@minimiser[label].type=dlib`

### 9.2.6. `@minimiser[label].type=gamma_diff`

`tolerance`      Tolerance of the gradient for convergence

Type: constant

Default: 0.02

`step_size`      Minimum Step-size before minimisation fails

Type: constant

Default: 1e-7

`evaluations`      Maximum number of evaluations

Type: int

Default: 4000

`iterations`      Maximum number of iterations

Type: int

Default: 1000

## 9.3. Monte Carlo Markov Chain (MCMC)

`@mcmc label`      Define an object type MCMC

`correlation_adjustment_diff`      TBA

Type: constant

Default: 0.0001

`covariance_adjustment_method`      Method for adjusting small variances in the covariance proposal matrix

Type: string

Default: covariance



`length`      The number of chain links to create

Type: non-negative integer

Default: No default

`adapt_stepsize_at`      Iterations in the chain to check and resize the MCMC stepsize

Type: non-negative integer vector

Default: true

`label`      Label

Type: string

Default: No default

`keep`      Spacing between recorded values in the chain

Type: non-negative integer

Default: 1u

`type`      Type

Type: string

Default: ""

`active`      Is this the active MCMC algorithm

Type: boolean

Default: true

`df`      Degrees of freedom of the multivariate t proposal distribution

Type: non-negative integer

Default: 4

`proposal_distribution`      The shape of the proposal distribution (either t or normal)

Type: string

Default: t

`max_correlation`      Maximum absolute correlation in the covariance matrix of the proposal distribution

Type: constant

Default: 0.8

`start`      Covariance multiplier for the starting point of the Markov chain

Type: constant

Default: 0.0

`step_size`      Initial stepsize (as a multiplier of the approximate covariance matrix)

Type: constant

Default: 0.02

### 9.3.1. `@m.c.m.c[label].type=independence.metropolis`

## 9.4. Profiles

**@profile** *label*     Define an object type Profile

*steps*     The number of steps to take between the lower and upper bound

Type: non-negative integer

Default: No default

*lower\_bound*     The lower bounds

Type: constant

Default: No default

*upper\_bound*     The upper bounds

Type: constant

Default: No default

*label*     Label

Type: string

Default: ""

*parameter*     The system parameter to profile

Type: string

Default: No default

## 9.5. Defining catchability constants

**@catchability** *label*     Define an object type Catchability

*label*     Label

Type: string

Default: No default

### 9.5.1. `@catchability[label].type=free`

*q*     The catchability amount

Type: estimable

Default: No default

## 9.6. Defining penalties

**@penalty** *label*     Define an object type Penalty

label      Label  
Type: string  
Default: No default

type      Type  
Type: string  
Default: No default

### 9.6.1. @penalty[label].type=process

log\_scale      Log scale  
Type: boolean  
Default: false

multiplier      Multiplier  
Type: constant  
Default: 1.0

### 9.6.2. @penalty[label].type=vector\_average

parameter      Label of the estimate to generate penalty on  
Type: string  
Default: No default

method      What calculation method to use (k, l, m  
Type: string  
Default: k  
Allowed Values: ratio, difference, mean

k      K Value to use in the calculation  
Type: constant  
Default: No default

multiplier      Multiplier for the penalty amount  
Type: constant  
Default: 1

### 9.6.3. @penalty[label].type=vector\_smoothing

lower\_bound      First element to apply the penalty to in the vector  
Type: non-negative integer  
Default: 0u

`upper_bound`      Last element to apply the penalty to in the vector  
 Type: non-negative integer  
 Default: 0u

`log_scale`      Log scale  
 Type: boolean  
 Default: false

`r`      Penalty applied to rth differences  
 Type: non-negative integer  
 Default: 2u

`parameter`      Label of the estimate to generate penalty on  
 Type: string  
 Default: No default

`multiplier`      Multiplier for the penalty amount  
 Type: constant  
 Default: 1

## 9.7. Defining priors on parameter ratios, differences and means

**@additional\_prior** *label*      Define an object type Additional\_Prior

`method`      Method  
 Type: string  
 Default: No default  
 Allowed Values: ratio, difference, mean

`label`      Label  
 Type: string  
 Default: No default

`type`      Type  
 Type: string  
 Default: No default

### 9.7.1. @additional\_prior[label].type=beta

`mu`      Mu  
 Type: constant  
 Default: No default

`sigma`      Sigma

Type: constant  
Default: No default  
Lower Bound: 0.0 (inclusive)

a     A  
Type: constant  
Default: No default

b     B  
Type: constant  
Default: No default

## 10. Observation command and subcommand syntax

### 10.1. Observation types

The observation types available are,

Observations of proportions of individuals by age class  
Observations of proportions of individuals between categories within each age class  
Relative and absolute abundance observations  
Relative and absolute biomass observations

Each type of observation requires a set of subcommands and arguments specific to that process.

**@observation** *label*     Define an object type Observation

*time\_step*     Time step to execute in  
Type: string  
Default: No default

*likelihood\_multiplier*     Likelihood score multiplier  
Type: constant  
Default: Double(1.0)

*selectivities*     Selectivity labels to use  
Type: string vector  
Default: true

*label*     Label  
Type: string  
Default: No default

*error\_value\_multiplier*     Error value multiplier for likelihood  
Type: constant  
Default: Double(1.0)

`type`      Type of observation

Type: string

Default: No default

`categories`      Category labels to use

Type: string vector

Default: true

`simulation_likelihood`      Simulation likelihood to use

Type: string

Default: ""

`likelihood`      Type of likelihood to use

Type: string

Default: No default

### 10.1.1. `@observation[label].type=process_abundance`

### 10.1.2. `@observation[label].type=process_abundance`

`process_proportion`      Process proportion

Type: constant

Default: Double(0.5)

`process_error`      Process error

Type: constant

Default: Double(0.0)

`obs`      Observation values

Type: string vector

Default: No default

`years`      Years to execute in

Type: non-negative integer vector

Default: No default

`process`      Process label

Type: string

Default: No default

`error_value`      The error values to use against the observation values

Type: constant vector

Default: No default

`delta`      Delta value for error values

Type: constant  
Default: Double(1e-10)

### 10.1.3. **@observation[label].type=process\_biomass**

process\_error      Process error  
Type: constant  
Default: Double(0.0)

obs      Observation values  
Type: string vector  
Default: No default

years      Years to execute in  
Type: non-negative integer vector  
Default: No default

catchability      TBA  
Type: string  
Default: No default

error\_value      The error values to use against the observation values  
Type: constant vector  
Default: No default

delta      Delta value for error values  
Type: constant  
Default: Double(1e-10)

### 10.1.4. **@observation[label].type=process\_biomass**

process\_proportion      Process proportion  
Type: constant  
Default: Double(0.5)

process\_error      Process error  
Type: constant  
Default: Double(0.0)

obs      Observation values  
Type: string vector  
Default: No default

years      Years to execute in

Type: non-negative integer vector  
Default: No default

process      Process label  
Type: string  
Default: No default

catchability      TBA  
Type: string  
Default: No default

error\_value      The error values to use against the observation values  
Type: constant vector  
Default: No default

delta      Delta value for error values  
Type: constant  
Default: Double(1e-10)

#### **10.1.5. @observation[label].type=process\_proportions\_at\_age**

age\_plus      Use age plus group  
Type: boolean  
Default: true

process\_errors      Process error  
Type: constant vector  
Default: true

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

tolerance      Tolerance  
Type: constant  
Default: Double(0.001)

max\_age      Maximum age  
Type: non-negative integer  
Default: No default



ageing\_error      Label of ageing error to use  
Type: string  
Default: ""

delta      Delta  
Type: constant  
Default: DELTA

#### **10.1.6. @observation[label].type=process\_proportions\_at\_age**

process\_proportion      Process proportion  
Type: constant  
Default: Double(0.5)

age\_plus      Use age plus group  
Type: boolean  
Default: true

process\_errors      Process error  
Type: constant vector  
Default: true

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

tolerance      Tolerance  
Type: constant  
Default: Double(0.001)

max\_age      Maximum age  
Type: non-negative integer  
Default: No default

ageing\_error      Label of ageing error to use  
Type: string  
Default: ""

delta      Delta  
Type: constant  
Default: DELTA

**10.1.7. @observation[label].type=process\_proportions\_at\_age\_for\_fishery**

age\_plus      Use age plus group

Type: boolean

Default: true

process\_errors      Process error

Type: constant vector

Default: true

min\_age      Minimum age

Type: non-negative integer

Default: No default

years      Year to execute in

Type: non-negative integer vector

Default: No default

tolerance      Tolerance

Type: constant

Default: Double(0.001)

max\_age      Maximum age

Type: non-negative integer

Default: No default

fishery      Label of fishery the observation is from

Type: string

Default: ""

ageing\_error      Label of ageing error to use

Type: string

Default: ""

delta      Delta

Type: constant

Default: DELTA

**10.1.8. @observation[label].type=process\_proportions\_at\_age\_for\_fishery**

process\_proportion      Process proportion

Type: constant

Default: Double(0.5)

age\_plus      Use age plus group  
Type: boolean  
Default: true

process\_errors      Process error  
Type: constant vector  
Default: true

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

tolerance      Tolerance  
Type: constant  
Default: Double(0.001)

max\_age      Maximum age  
Type: non-negative integer  
Default: No default

fishery      Label of fishery the observation is from  
Type: string  
Default: ""

ageing\_error      Label of ageing error to use  
Type: string  
Default: ""

delta      Delta  
Type: constant  
Default: DELTA

#### **10.1.9. @observation[label].type=process\_proportions\_at\_length**

tolerance      Tolerance for rescaling proportions  
Type: constant  
Default: Double(0.001)

process\_errors      Process error

Type: constant vector

Default: true

years      Year to execute in

Type: non-negative integer vector

Default: No default

length\_bins      Length bins

Type: constant vector

Default: No default

length\_plus\_group      Is the last bin a plus group

Type: boolean

Default: true

delta      Delta

Type: constant

Default: DELTA

#### **10.1.10. @observation[label].type=process\_proportions\_at\_length**

process\_proportion      Process proportion

Type: constant

Default: Double(0.5)

tolerance      Tolerance for rescaling proportions

Type: constant

Default: Double(0.001)

process\_errors      Process error

Type: constant vector

Default: true

years      Year to execute in

Type: non-negative integer vector

Default: No default

length\_bins      Length bins

Type: constant vector

Default: No default

length\_plus\_group      Is the last bin a plus group

Type: boolean

Default: true

delta      Delta  
Type: constant  
Default: DELTA

#### 10.1.11. `@observation[label].type=process_proportions_at_length_for_fishery`

tolerance      Tolerance for rescaling proportions  
Type: constant  
Default: Double(0.001)

process\_errors      Process error  
Type: constant vector  
Default: true

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

length\_bins      Length bins  
Type: constant vector  
Default: No default

fishery      Label of fishery the observation is from  
Type: string  
Default: ""

length\_plus\_group      Is the last bin a plus group  
Type: boolean  
Default: true

delta      Delta  
Type: constant  
Default: DELTA

#### 10.1.12. `@observation[label].type=process_proportions_at_length_for_fishery`

process\_proportion      Process proportion  
Type: constant  
Default: Double(0.5)

tolerance      Tolerance for rescaling proportions  
Type: constant  
Default: Double(0.001)

`process_errors`      Process error  
Type: constant vector  
Default: true

`years`      Year to execute in  
Type: non-negative integer vector  
Default: No default

`length_bins`      Length bins  
Type: constant vector  
Default: No default

`fishery`      Label of fishery the observation is from  
Type: string  
Default: ""

`length_plus_group`      Is the last bin a plus group  
Type: boolean  
Default: true

`delta`      Delta  
Type: constant  
Default: DELTA

### 10.1.13. `@observation[label].type=process_proportions_by_category`

`age_plus`      Use age plus group  
Type: boolean  
Default: true

`selectivities2`      Target Selectivities  
Type: string vector  
Default: No default

`min_age`      Minimum age  
Type: non-negative integer  
Default: No default

`years`      Year to execute in  
Type: non-negative integer vector  
Default: No default

`max_age`      Maximum age  
Type: non-negative integer  
Default: No default

categories2      Target Categories  
Type: string vector  
Default: No default

process\_errors      Process error  
Type: constant vector  
Default: true

delta      Delta  
Type: constant  
Default: DELTA  
Lower Bound: 0.0 (exclusive)

#### 10.1.14. @observation[label].type=process\_proportions\_by\_category

process\_proportion      Process proportion  
Type: constant  
Default: Double(0.5)

age\_plus      Use age plus group  
Type: boolean  
Default: true

selectivities2      Target Selectivities  
Type: string vector  
Default: No default

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

max\_age      Maximum age  
Type: non-negative integer  
Default: No default

categories2      Target Categories  
Type: string vector  
Default: No default

process\_errors      Process error  
Type: constant vector  
Default: true

delta      Delta  
Type: constant  
Default: DELTA  
Lower Bound: 0.0 (exclusive)

#### **10.1.15. @observation[label].type=process\_proportions\_migrating**

age\_plus      Use age plus group  
Type: boolean  
Default: true

process\_errors      Process error  
Type: constant vector  
Default: true

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

max\_age      Maximum age  
Type: non-negative integer  
Default: No default

ageing\_error      Label of ageing error to use  
Type: string  
Default: ""

delta      Delta  
Type: constant  
Default: DELTA

#### **10.1.16. @observation[label].type=process\_proportions\_migrating**

process\_proportion      Process proportion  
Type: constant  
Default: Double(0.5)



age\_plus      Use age plus group  
Type: boolean  
Default: true

process\_errors      Process error  
Type: constant vector  
Default: true

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

max\_age      Maximum age  
Type: non-negative integer  
Default: No default

ageing\_error      Label of ageing error to use  
Type: string  
Default: ""

delta      Delta  
Type: constant  
Default: DELTA

#### **10.1.17. @observation[label].type=time\_step\_abundance**

#### **10.1.18. @observation[label].type=time\_step\_abundance**

time\_step\_proportion      Proportion through the time step to analyse the partition from  
Type: constant  
Default: Double(0.5)

process\_error      Process error  
Type: constant  
Default: Double(0.0)

obs      Observation values  
Type: string vector  
Default: No default

years      Years to execute in  
Type: non-negative integer vector  
Default: No default

error\_value      The error values to use against the observation values  
Type: constant vector  
Default: No default

delta      Delta value for error values  
Type: constant  
Default: Double(1e-10)

#### **10.1.19. @observation[label].type=time\_step\_biomass**

process\_error      Process error  
Type: constant  
Default: Double(0.0)

obs      Observation values  
Type: string vector  
Default: No default

years      Years to execute in  
Type: non-negative integer vector  
Default: No default

catchability      TBA  
Type: string  
Default: No default

error\_value      The error values to use against the observation values  
Type: constant vector  
Default: No default

delta      Delta value for error values  
Type: constant  
Default: Double(1e-10)

#### **10.1.20. @observation[label].type=time\_step\_biomass**

time\_step\_proportion      Proportion through the time step to analyse the partition from  
Type: constant  
Default: Double(0.5)

process\_error      Process error  
Type: constant  
Default: Double(0.0)

obs      Observation values  
Type: string vector  
Default: No default

years      Years to execute in  
Type: non-negative integer vector  
Default: No default

catchability      TBA  
Type: string  
Default: No default

error\_value      The error values to use against the observation values  
Type: constant vector  
Default: No default

delta      Delta value for error values  
Type: constant  
Default: Double(1e-10)

#### **10.1.21. @observation[label].type=time\_step\_proportions\_at\_age**

age\_plus      Use age plus group  
Type: boolean  
Default: true

process\_errors      Process error  
Type: constant vector  
Default: true

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

tolerance      Tolerance  
Type: constant  
Default: Double(0.001)

max\_age      Maximum age  
Type: non-negative integer  
Default: No default

ageing\_error      Label of ageing error to use  
Type: string  
Default: ""

delta      Delta  
Type: constant  
Default: DELTA

### 10.1.22. @observation[label].type=time\_step\_proportions\_at\_age

age\_plus      Use age plus group  
Type: boolean  
Default: true

process\_errors      Process error  
Type: constant vector  
Default: true

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

tolerance      Tolerance  
Type: constant  
Default: Double(0.001)

max\_age      Maximum age  
Type: non-negative integer  
Default: No default

ageing\_error      Label of ageing error to use  
Type: string  
Default: ""

delta      Delta

Type: constant  
Default: DELTA

### 10.1.23. `@observation[label].type=time_step_proportions_at_length`

`tolerance`      Tolerance for rescaling proportions  
Type: constant  
Default: Double(0.001)

`process_errors`      Process error  
Type: constant vector  
Default: true

`years`      Year to execute in  
Type: non-negative integer vector  
Default: No default

`length_bins`      Length bins  
Type: constant vector  
Default: No default

`length_plus_group`      Is the last bin a plus group  
Type: boolean  
Default: true

`delta`      Delta  
Type: constant  
Default: DELTA

### 10.1.24. `@observation[label].type=time_step_proportions_at_length`

`tolerance`      Tolerance for rescaling proportions  
Type: constant  
Default: Double(0.001)

`process_errors`      Process error  
Type: constant vector  
Default: true

`years`      Year to execute in  
Type: non-negative integer vector  
Default: No default

`length_bins`      Length bins

Type: constant vector  
Default: No default

length\_plus\_group      Is the last bin a plus group  
Type: boolean  
Default: true

delta      Delta  
Type: constant  
Default: DELTA

#### 10.1.25. @observation[label].type=time\_step\_proportions\_by\_category

age\_plus      Use age plus group  
Type: boolean  
Default: true

selectivities2      Target Selectivities  
Type: string vector  
Default: No default

min\_age      Minimum age  
Type: non-negative integer  
Default: No default

years      Year to execute in  
Type: non-negative integer vector  
Default: No default

max\_age      Maximum age  
Type: non-negative integer  
Default: No default

categories2      Target Categories  
Type: string vector  
Default: No default

process\_errors      Process error  
Type: constant vector  
Default: true

delta      Delta  
Type: constant  
Default: DELTA  
Lower Bound: 0.0 (exclusive)

**10.1.26. @observation[label].type=time\_step\_proportions\_by\_category**

age\_plus      Use age plus group

Type: boolean

Default: true

selectivities2      Target Selectivities

Type: string vector

Default: No default

min\_age      Minimum age

Type: non-negative integer

Default: No default

years      Year to execute in

Type: non-negative integer vector

Default: No default

max\_age      Maximum age

Type: non-negative integer

Default: No default

categories2      Target Categories

Type: string vector

Default: No default

process\_errors      Process error

Type: constant vector

Default: true

delta      Delta

Type: constant

Default: DELTA

Lower Bound: 0.0 (exclusive)

**10.2. Likelihoods**

**@likelihood** *label*      Define an object type Likelihood

**10.2.1. @likelihood[label].type=binomial**

**10.2.2. @likelihood[label].type=binomial\_approx**

**10.2.3. @likelihood[label].type=dirichlet**

**10.2.4. @likelihood[label].type=log\_normal**

**10.2.5. @likelihood[label].type=log\_normal\_with\_q**

**10.2.6. @likelihood[label].type=multinomial**

**10.2.7. @likelihood[label].type=normal**

**10.2.8. @likelihood[label].type=pseudo**

### 10.3. Defining ageing error

Three methods for including ageing error into estimation with observations are,

- None
- Normal
- Off-by-one

Each type of ageing error requires a set of subcommands and arguments specific to its type.

**@ageing\_error** *label*     Define an object type Ageing\_Error

*label*     Label  
           Type: string  
           Default: No default

*type*     Type  
           Type: string  
           Default: No default

**10.3.1. @ageing\_error[label].type=data**

**10.3.2. @ageing\_error[label].type=normal**

*cv*     CV for Misclassification matrix  
           Type: estimable  
           Default: No default  
           Lower Bound: 0.0 (inclusive)

*k*     TBA  
           Type: non-negative integer  
           Default: 0u



### 10.3.3. @ageing\_error[label].type=off\_by\_one

p2      proprtion of misclassification down by an age

Type: estimable

Default: No default

Lower Bound: 0.0 (inclusive)

Upper Bound: 1.0 (inclusive)

k      The minimum age of fish which can be missclassified

Type: non-negative integer

Default: 0u

Lower Bound: 0.0 (inclusive)

Upper Bound: 1.0 (inclusive)

p1      proprtion of misclassification up by an age, i.e. how proportion of age 3 that are actually age  
4

Type: estimable

Default: No default

## 11. Report command and subcommand syntax

### 11.1. Available reports

The report types available are,

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.

Each type of report requires a set of subcommands and arguments specific to that report.

### 11.2. Report commands and subcommands

**@report** *label*      Define an object type Report

*file\_name*      File Name

Type: string

Default: ""

overwrite      Overwrite file

Type: boolean

Default: true

label      Label

Type: string

Default: No default

type      Type

Type: string

Default: No default

sequentially\_add      Sequentially add a suffix to the file

Type: boolean

Default: false

### **11.2.1. @report[label].type=ageing\_error\_matrix**

ageing\_error      Ageing Error label

Type: string

Default: No default

### **11.2.2. @report[label].type=category\_info**

### **11.2.3. @report[label].type=category\_list**

### **11.2.4. @report[label].type=covariance\_matrix**

### **11.2.5. @report[label].type=derived\_quantity**

### **11.2.6. @report[label].type=estimable**

parameter      Parameter to print

Type: string

Default: No default

time\_step      Time Step label

Type: string

Default: ""

years      Years to print the estimable for

Type: non-negative integer vector

Default: No default

**11.2.7. @report[label].type=estimate\_summary**

**11.2.8. @report[label].type=estimate\_value**

**11.2.9. @report[label].type=initialisation\_partition**

**11.2.10. @report[label].type=mcmc\_chain**

**11.2.11. @report[label].type=mcmc\_covariance**

**11.2.12. @report[label].type=objective\_function**

**11.2.13. @report[label].type=observation**

observation      Observation label  
 Type: string  
 Default: No default

**11.2.14. @report[label].type=partition**

time\_step      Time Step label  
 Type: string  
 Default: ""

years      Years  
 Type: non-negative integer vector  
 Default: true

**11.2.15. @report[label].type=partition\_biomass**

units      Units (Default Kgs  
 Type: string  
 Default: kgs

time\_step      Time Step label  
 Type: string  
 Default: ""

years      Years  
 Type: non-negative integer vector  
 Default: true

---

### 11.2.16. `@report [label] .type=partition_mean_weight`

`time_step`      Time Step label

Type: string

Default: ""

`years`          Years

Type: non-negative integer vector

Default: true

### 11.2.17. `@report [label] .type=random_number_seed`

### 11.2.18. `@report [label] .type=selectivity`

`selectivity`      Selectivity name

Type: string

Default: No default

### 11.2.19. `@report [label] .type=simulated_observation`

`observation`      Observation label

Type: string

Default: No default

### 11.2.20. `@report [label] .type=standard_header`

## 12. Other commands and subcommands

**`@include`** *file*      Include an external file

*file*      The name of the external file to include

Type: string

Default: No default

Value: A valid external file

Condition: The file name must be enclosed in double quotes

Example: `@include "my_file.txt"`

Note: `@include` does not denote the end of the previous command block as is the case for all other commands



---

## 13. Examples



---

## 14. Post processing output using R





---

## 15. Troubleshooting

### 15.1. Introduction

### 15.2. Reporting errors

When reporting a bug or problem ...

### 15.3. Guidelines for reporting a problem with CASAL2

1. Detail the version of CASAL2 are you using? e.g., “CASAL2 2015-10-20 (rev. f610208)Microsoft Windows executable”
2. What operating system or environment are you using? e.g., “IBM-PC Intel CPU running Microsoft Windows 8.1 Enterprise, Service Pack 1”.
3. Give a brief one-line description of the problem, e.g., “a segmentation fault was reported”.
4. If the problem is reproducible, please list the exact steps required to cause it, remembering to include the relevant CASAL2 configuration file, other input files, and any out generated. Specify the *exact* command line arguments that were used, e.g., “Using the command `***. -* -*` reports a segmentation fault. The input configuration files are attached.”
5. If the problem is not reproducible (only happened 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., “SPM 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 is usually not possible to determine the cause of the problem, and we are unlikely to provide any assistance. Note that it is helpful to be as specific as possible when describing the problem.



---

## 16. Acknowledgements



---

## 17. Quick reference

**@ageing\_error** *label*    Define an object type Ageing\_Error

label    Label

type    Type

**@ageing\_error[label].type=data**

**@ageing\_error[label].type=normal**

cv    CV for Misclassification matrix

k    TBA

**@ageing\_error[label].type=off\_by\_one**

p2    proprtion of misclassification down by an age

k    The minimum age of fish which can be missclassified

p1    proprtion of misclassification up by an age, i.e. how proportion of age 3 that are actually age 4

**@age\_length** *label*    Define an object type Age\_Length

distribution    TBA

label    Label

type    Type

time\_step\_proportions

**@age\_length[label].type=data**

cv\_first    CV for the first age class

internal\_gaps

length\_weight    TBA

external\_gaps

cv\_last    CV for maximum age

**@age\_length[label].type=none**

**@age\_length[label].type=schnute**

tau2    TBA

length\_weight    TBA

tau1    TBA

cv\_first    CV for the first age class

y1    TBA

y2    TBA

cv\_last    CV for maximum age

by\_length    TBA

a    TBA

b    TBA

**@age\_length[label].type=von\_bertalanffy**

linf    TBA

length\_weight      TBA  
cv\_first      CV for the first age class  
t0      TBA  
cv\_last      CV for last age class  
k      TBA  
by\_length      Specifies if the linear interpolation of CV's is a linear function of mean length at age.  
Default is just by age  
**@catchability** *label*      Define an object type Catchability  
label      Label

**@catchability[label].type=free**

q      The catchability amount  
**@derived\_quantity** *label*      Define an object type Derived.Quantity  
time\_step      The time step to calculate the derived quantity after  
time\_step\_proportion\_method  
selectivities      The list of selectivities to use when calculating the derived quantity. 1 per  
category  
type      Type  
categories      The list of categories to use when calculating the derived quantity  
time\_step\_proportion

**@derived\_quantity[label].type=abundance**

**@derived\_quantity[label].type=biomass**

**@estimate** *label*      Define an object type Estimate  
lower\_bound      The lowest value the parameter is allowed to have  
transformation  
same      A list of parameters that are bound to the value of this estimate  
prior      The name of the prior to use for the parameter  
upper\_bound      The highest value the parameter is allowed to have  
label      Label  
parameter      The name of the variable to estimate in the model  
type      Type  
estimation\_phase      TBA  
mcmc      TBA

**@estimate[label].type=beta**

mu      Mu  
sigma      Sigma  
a      A  
b      B

**@estimate[label].type=lognormal**

mu      Mu  
cv      Cv

---

**@estimate[label].type=normal**

mu      Mu  
cv      Cv

**@estimate[label].type=normal.by.stdev**

mu      Mu  
sigma    Sigma

**@estimate[label].type=normal.log**

mu      Mu  
sigma    Sigma

**@estimate[label].type=uniform**

**@estimate[label].type=uniform.log**

**@initialisation\_phase label**      Define an object type Initialisation.Phase

label    Label  
type    Type

**@initialisation\_phase[label].type=derived**

exclude\_processes      The processes to exclude from all time steps

**@initialisation\_phase[label].type=iterative**

insert\_processes      The processes to insert in to target time steps  
lambda      Lambda  
exclude\_processes      The processes to exclude from all time steps  
convergence\_years      The years to test for convergence

**@initialisation\_phase[label].type=state.category.by.age**

min\_age      Minimum age to use for this process  
max\_age      Maximum age to use for this process  
categories      List of categories to use

**@likelihood label**      Define an object type Likelihood

**@likelihood[label].type=binomial**

**@likelihood[label].type=binomial.approx**

**@likelihood[label].type=dirichlet**

**@likelihood[label].type=log.normal**

**@likelihood[label].type=log.normal.with.q**

**@likelihood[label].type=multinomial**

**@likelihood[label].type=normal**

**@likelihood[label].type=pseudo**



**@derived\_quantity** *label* Define an object type Derived\_Quantity

time\_step The time step to calculate the derived quantity after

time\_step\_proportion\_method

selectivities The list of selectivities to use when calculating the derived quantity. 1 per category

type Type

categories The list of categories to use when calculating the derived quantity

time\_step\_proportion

**@derived\_quantity[label].type=abundance**

**@derived\_quantity[label].type=biomass**

**@mcmc** *label* Define an object type MCMC

correlation\_adjustment\_diff TBA

covariance\_adjustment\_method Method for adjusting small variances in the covariance proposal matrix

length The number of chain links to create

adapt\_stepsize\_at Iterations in the chain to check and resize the MCMC stepsize

label Label

keep Spacing between recorded values in the chain

type Type

active Is this the active MCMC algorithm

df Degrees of freedom of the multivariate t proposal distribution

proposal\_distribution The shape of the proposal distribution (either t or normal

max\_correlation Maximum absolute correlation in the covariance matrix of the proposal distribution

start Covariance multiplier for the starting point of the Markov chain

step\_size Initial stepsize (as a multiplier of the approximate covariance matrix

**@m\_c\_m\_c[label].type=independence\_metropolis**

**@minimiser** *label* Define an object type Minimiser

active True if this minimiser is active

label Label

type Type of minimiser to use

covariance True if a covariance matrix should be created

**@minimiser[label].type=a\_d\_o\_l\_c**

tolerance Tolerance of the gradient for convergence

step\_size Minimum Step-size before minimisation fails

evaluations Maximum number of evaluations

iterations Maximum number of iterations

**@minimiser[label].type=beta\_diff**

tolerance Tolerance of the gradient for convergence

evaluations Maximum number of evaluations

iterations Maximum number of iterations

---

**@minimiser[label].type=c\_p\_p\_a\_d**

**@minimiser[label].type=d\_e\_solver**

population\_size     The number of candidate solutions to have in the population  
difference\_scale     The scale to apply to new solutions when comparing candidates  
tolerance     The total variance between the population and best candidate before acceptance  
crossover\_probability     TBA  
method     The type of candidate generation method to use  
max\_generations     The maximum number of iterations to run

**@minimiser[label].type=d\_lib**

**@minimiser[label].type=gamma\_diff**

tolerance     Tolerance of the gradient for convergence  
step\_size     Minimum Step-size before minimisation fails  
evaluations     Maximum number of evaluations  
iterations     Maximum number of iterations

**@model label**     Define an object type Model

initialisation\_phases     Define the labels of the phases of the initialisation  
start\_year     Define the first year of the model, immediately following initialisation  
age\_plus     Define the oldest age as a plus group  
min\_age     Minimum age of individuals in the population  
length\_bins  
type     Type of model (the partition structure). Either age, length or hybrid  
max\_age     Maximum age of individuals in the population  
final\_year     Define the final year of the model, excluding years in the projection period  
time\_steps     Define the labels of the time steps, in the order that they are applied, to form the annual cycle  
projection\_final\_year     Define the final year of the model in projection mode

**@observation label**     Define an object type Observation

time\_step     Time step to execute in  
likelihood\_multiplier     Likelihood score multiplier  
selectivities     Selectivity labels to use  
label     Label  
error\_value\_multiplier     Error value multiplier for likelihood  
type     Type of observation  
categories     Category labels to use  
simulation\_likelihood     Simulation likelihood to use  
likelihood     Type of likelihood to use

**@observation[label].type=process\_abundance**

**@observation[label].type=process\_abundance**

process\_proportion     Process proportion

process\_error    Process error  
obs      Observation values  
years      Years to execute in  
process      Process label  
error\_value    The error values to use against the observation values  
delta      Delta value for error values

**@observation[label].type=process\_biomass**

process\_error    Process error  
obs      Observation values  
years      Years to execute in  
catchability    TBA  
error\_value    The error values to use against the observation values  
delta      Delta value for error values

**@observation[label].type=process\_biomass**

process\_proportion    Process proportion  
process\_error    Process error  
obs      Observation values  
years      Years to execute in  
process      Process label  
catchability    TBA  
error\_value    The error values to use against the observation values  
delta      Delta value for error values

**@observation[label].type=process\_proportions\_at\_age**

age\_plus      Use age plus group  
process\_errors    Process error  
min\_age      Minimum age  
years      Year to execute in  
tolerance      Tolerance  
max\_age      Maximum age  
ageing\_error    Label of ageing error to use  
delta      Delta

**@observation[label].type=process\_proportions\_at\_age**

process\_proportion    Process proportion  
age\_plus      Use age plus group  
process\_errors    Process error  
min\_age      Minimum age  
years      Year to execute in  
tolerance      Tolerance  
max\_age      Maximum age  
ageing\_error    Label of ageing error to use  
delta      Delta

---

**@observation[label].type=process\_proportions\_at\_age\_for\_fishery**

age\_plus      Use age plus group  
process\_errors      Process error  
min\_age      Minimum age  
years      Year to execute in  
tolerance      Tolerance  
max\_age      Maximum age  
fishery      Label of fishery the observation is from  
ageing\_error      Label of ageing error to use  
delta      Delta

**@observation[label].type=process\_proportions\_at\_age\_for\_fishery**

process\_proportion      Process proportion  
age\_plus      Use age plus group  
process\_errors      Process error  
min\_age      Minimum age  
years      Year to execute in  
tolerance      Tolerance  
max\_age      Maximum age  
fishery      Label of fishery the observation is from  
ageing\_error      Label of ageing error to use  
delta      Delta

**@observation[label].type=process\_proportions\_at\_length**

tolerance      Tolerance for rescaling proportions  
process\_errors      Process error  
years      Year to execute in  
length\_bins      Length bins  
length\_plus\_group      Is the last bin a plus group  
delta      Delta

**@observation[label].type=process\_proportions\_at\_length**

process\_proportion      Process proportion  
tolerance      Tolerance for rescaling proportions  
process\_errors      Process error  
years      Year to execute in  
length\_bins      Length bins  
length\_plus\_group      Is the last bin a plus group  
delta      Delta

**@observation[label].type=process\_proportions\_at\_length\_for\_fishery**

tolerance      Tolerance for rescaling proportions

process\_errors    Process error  
years            Year to execute in  
length\_bins      Length bins  
fishery          Label of fishery the observation is from  
length\_plus\_group    Is the last bin a plus group  
delta            Delta

**@observation[label].type=process\_proportions\_at\_length\_for\_fishery**

process\_proportion    Process proportion  
tolerance            Tolerance for rescaling proportions  
process\_errors      Process error  
years            Year to execute in  
length\_bins      Length bins  
fishery          Label of fishery the observation is from  
length\_plus\_group    Is the last bin a plus group  
delta            Delta

**@observation[label].type=process\_proportions\_by\_category**

age\_plus          Use age plus group  
selectivities2      Target Selectivities  
min\_age          Minimum age  
years            Year to execute in  
max\_age          Maximum age  
categories2        Target Categories  
process\_errors      Process error  
delta            Delta

**@observation[label].type=process\_proportions\_by\_category**

process\_proportion    Process proportion  
age\_plus          Use age plus group  
selectivities2      Target Selectivities  
min\_age          Minimum age  
years            Year to execute in  
max\_age          Maximum age  
categories2        Target Categories  
process\_errors      Process error  
delta            Delta

**@observation[label].type=process\_proportions\_migrating**

age\_plus          Use age plus group  
process\_errors      Process error  
min\_age          Minimum age  
years            Year to execute in  
max\_age          Maximum age  
ageing\_error        Label of ageing error to use  
delta            Delta

---

**@observation[label].type=process\_proportions\_migrating**

process\_proportion    Process proportion  
age\_plus      Use age plus group  
process\_errors    Process error  
min\_age      Minimum age  
years      Year to execute in  
max\_age      Maximum age  
ageing\_error    Label of ageing error to use  
delta      Delta

**@observation[label].type=time\_step\_abundance****@observation[label].type=time\_step\_abundance**

time\_step\_proportion    Proportion through the time step to analyse the partition from  
process\_error    Process error  
obs      Observation values  
years      Years to execute in  
error\_value    The error values to use against the observation values  
delta      Delta value for error values

**@observation[label].type=time\_step\_biomass**

process\_error    Process error  
obs      Observation values  
years      Years to execute in  
catchability    TBA  
error\_value    The error values to use against the observation values  
delta      Delta value for error values

**@observation[label].type=time\_step\_biomass**

time\_step\_proportion    Proportion through the time step to analyse the partition from  
process\_error    Process error  
obs      Observation values  
years      Years to execute in  
catchability    TBA  
error\_value    The error values to use against the observation values  
delta      Delta value for error values

**@observation[label].type=time\_step\_proportions\_at\_age**

age\_plus      Use age plus group  
process\_errors    Process error  
min\_age      Minimum age  
years      Year to execute in  
tolerance    Tolerance  
max\_age      Maximum age  
ageing\_error    Label of ageing error to use  
delta      Delta

**@observation[label].type=time\_step\_proportions\_at\_age**

age\_plus      Use age plus group  
process\_errors      Process error  
min\_age      Minimum age  
years      Year to execute in  
tolerance      Tolerance  
max\_age      Maximum age  
ageing\_error      Label of ageing error to use  
delta      Delta

**@observation[label].type=time\_step\_proportions\_at\_length**

tolerance      Tolerance for rescaling proportions  
process\_errors      Process error  
years      Year to execute in  
length\_bins      Length bins  
length\_plus\_group      Is the last bin a plus group  
delta      Delta

**@observation[label].type=time\_step\_proportions\_at\_length**

tolerance      Tolerance for rescaling proportions  
process\_errors      Process error  
years      Year to execute in  
length\_bins      Length bins  
length\_plus\_group      Is the last bin a plus group  
delta      Delta

**@observation[label].type=time\_step\_proportions\_by\_category**

age\_plus      Use age plus group  
selectivities2      Target Selectivities  
min\_age      Minimum age  
years      Year to execute in  
max\_age      Maximum age  
categories2      Target Categories  
process\_errors      Process error  
delta      Delta

**@observation[label].type=time\_step\_proportions\_by\_category**

age\_plus      Use age plus group  
selectivities2      Target Selectivities  
min\_age      Minimum age  
years      Year to execute in  
max\_age      Maximum age  
categories2      Target Categories  
process\_errors      Process error  
delta      Delta

**@penalty label**      Define an object type Penalty

---

label     Label  
type     Type

**@penalty[label].type=process**

log\_scale     Log scale  
multiplier     Multiplier

**@penalty[label].type=vector\_average**

parameter     Label of the estimate to generate penalty on  
method     What calculation method to use (k, l, m)  
k     K Value to use in the calculation  
multiplier     Multiplier for the penalty amount

**@penalty[label].type=vector\_smoothing**

lower\_bound     First element to apply the penalty to in the vector  
upper\_bound     Last element to apply the penalty to in the vector  
log\_scale     Log scale  
r     Penalty applied to rth differences  
parameter     Label of the estimate to generate penalty on  
multiplier     Multiplier for the penalty amount  
**@process label**     Define an object type Process

label     Label  
type     Type  
print\_report     Print parameter report

**@process[label].type=ageing**

categories     Categories

**@process[label].type=growth**

**@process[label].type=maturation**

rates     The rates to mature for each year  
selectivities     List of selectivities to use for maturation  
to     List of categories to mature too  
years     The years to be associated with rates

**@process[label].type=mortality\_constant\_rate**

m     Mortality rates  
selectivities     Selectivities  
categories     List of categories  
time\_step\_ratio     Time step ratios for M

**@process[label].type=mortality\_event**

penalty     Penalty label



selectivities List of selectivities  
u\_max U Max  
catches Catches  
years Years

**@process[label].type=mortality\_event\_biomass**

penalty Penalty label  
selectivities Selectivity labels  
u\_max U Max  
catches Catches for each year  
years Years to apply mortality

**@process[label].type=mortality\_instantaneous**

penalty Label of penalty to be applied  
selectivities Selectivities for Natural Mortality  
difference The difference between, two categories natural mortality around the average  
m Mortality rates  
categories Categories for natural mortality  
time\_step\_ratio Time step ratios for M  
average The average mortality rate

**@process[label].type=nop****@process[label].type=recruitment\_beverton\_holt**

r0 R0  
ssb\_offset SSB Offset (year offset)  
standardise\_ycs\_years  
ycs\_values YCS Values  
age Age to recruit at  
b0 B0 Label  
steepness Steepness  
ssb SSB Label (derived quantity)  
prior\_standardised\_ycs Priors for year class strength on ycs values (not standardised ycs values)  
categories Category labels  
proportions Proportions

**@process[label].type=recruitment\_constant**

r0 R0  
age Age  
proportions Proportions  
categories Categories

**@process[label].type=tag\_by\_age**

penalty Penalty label

---

u\_max      U Max  
loss\_rate\_selectivities  
selectivities  
min\_age    Minimum age to transition  
to        Categories to transition to  
years      Years to execute the transition in  
from       Categories to transition from  
initial\_mortality\_selectivity  
loss\_rate  
max\_age    Maximum age to transition  
initial\_mortality  
n

**@process[label].type=tag\_by\_length**

penalty    Penalty label  
u\_max      U Max  
loss\_rate\_selectivities  
selectivities  
min\_age    Minimum age to transition  
to        Categories to transition to  
years      Years to execute the transition in  
from       Categories to transition from  
initial\_mortality\_selectivity  
loss\_rate  
max\_age    Maximum age to transition  
initial\_mortality  
n

**@process[label].type=transition\_category**

from       From  
selectivities    Selectivity names  
to        To  
proportions    Proportions

**@process[label].type=transition\_category\_by\_age**

penalty    Penalty label  
u\_max      U Max  
min\_age    Minimum age to transition  
to        Categories to transition to  
years      Years to execute the transition in  
from       Categories to transition from  
max\_age    Maximum age to transition

**@profile label**    Define an object type Profile

steps      The number of steps to take between the lower and upper bound

lower\_bound     The lower bounds  
upper\_bound     The upper bounds  
label     Label  
parameter     The system parameter to profile  
**@report label**     Define an object type Report  
file\_name     File Name  
overwrite     Overwrite file  
label     Label  
type     Type  
sequentially\_add     Sequentially add a suffix to the file

**@report [label] .type=ageing\_error matrix**

ageing\_error     Ageing Error label

**@report [label] .type=category\_info**

**@report [label] .type=category\_list**

**@report [label] .type=covariance matrix**

**@report [label] .type=derived quantity**

**@report [label] .type=estimable**

parameter     Parameter to print

time\_step     Time Step label

years     Years to print the estimable for

**@report [label] .type=estimate summary**

**@report [label] .type=estimate value**

**@report [label] .type=initialisation partition**

**@report [label] .type=mcmc chain**

**@report [label] .type=mcmc covariance**

**@report [label] .type=objective function**

**@report [label] .type=observation**

observation     Observation label

**@report [label] .type=partition**

time\_step     Time Step label

years     Years

**@report [label] .type=partition biomass**

units     Units (Default Kgs

time\_step     Time Step label

years     Years

---

**@report[label].type=partition\_mean\_weight**

time\_step      Time Step label

years          Years

**@report[label].type=random\_number\_seed**

**@report[label].type=selectivity**

selectivity      Selectivity name

**@report[label].type=simulated\_observation**

observation      Observation label

**@report[label].type=standard\_header**

**@selectivity label**      Define an object type Selectivity

type      Type

**@selectivity[label].type=all\_values**

**@selectivity[label].type=all\_values\_bounded**

v      V

h      H

**@selectivity[label].type=constant**

**@selectivity[label].type=double\_exponential**

y1      Y1

y0      Y0

y2      Y2

x2      X2

alpha      Alpha

x1      X1

**@selectivity[label].type=double\_normal**

sigma\_l      Sigma L

sigma\_r      Sigma R

alpha      Alpha

**@selectivity[label].type=increasing**

v      V

h      High

alpha      Alpha

**@selectivity[label].type=inverse\_logistic**

ato95      aTo95

alpha      Alpha

**@selectivity[label].type=knife\_edge**

alpha     Alpha

**@selectivity[label].type=logistic**

ato95     Ato95

alpha     Alpha

**@selectivity[label].type=logistic\_producing**

ato95     Ato95

alpha     Alpha

h     High

a50     A50

**@length.weight label**     Define an object type Length\_Weight

label     Label

type     Type

**@length.weight[label].type=basic**

units     Units of measure (tonnes, kgs, grams)

a     A

b     B

**@length.weight[label].type=none**

**@time.step label**     Define an object type Time\_Step

processes     Processes

label     Label

---

## 18. References

- R.J.H. Beverton and S.J. Holt. *On the dynamics of exploited fish populations*. Fishery investigations. HMSO, London, 1957.
- B. Bull, R.I.C.C. Francis, A. Dunn, A. McKenzie, D.J. Gilbert, M.H. Smith, R. Bian, and D. Fu. CASAL C++ Algorithmic Stock Assessment Laboratory: CASAL user manual v2.30-2012/03/21. Technical Report 135, NIWA, 2012.
- J. E. Dennis Jr and R.B. Schnabel. *Numerical methods for unconstrained optimisation and nonlinear equations*. Classics in Applied Mathematics. Prentice Hall, 1996.
- A B Gelman, J S Carlin, H S Stern, and D B Rubin. *Bayesian data analysis*. Chapman and Hall, London, 1995.
- W R Gilks, A Thomas, and D J Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 43(1):169–177, 1994.
- Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation: Special Issue on Uniform Random Number Generation*, 8(1):3–30, January 1998.
- Andre E. Punt and Ray Hilborn. *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)., 2001.
- J Schnute. A versatile growth model with statistically stable parameters. *Canadian Journal of Fisheries and Aquatic Sciences*, 38(9):1128–1140, 1981.
- Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, 1995. URL <http://citeseer.ist.psu.edu/182432.html>.



---

## 19.

### CASAL2 software license *check and confirm text*

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

#### 1. DEFINITIONS

"Contribution" means:

- a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
- b) in the case of each subsequent Contributor:
  - i) changes to the Program, and
  - ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents" mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

#### 2. GRANT OF RIGHTS

a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

#### 3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

- a) it complies with the terms and conditions of this Agreement; and
- b) its license agreement:
  - i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
  - ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
  - iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and
  - iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

- a) it must be made available under this Agreement; and
- b) a copy of this Agreement must be included with each copy of the Program.



Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

#### 4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

#### 5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

#### 6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), 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 OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### 7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

## 20. Index

- CASAL2 source code, 2
- About CASAL2, 5
- Abundance or biomass observations, 33
- Age-length relationship, 11, 54
- Ageing, 4, 6
- Ageing error, 36
- Annual cycle, 2, 4, 5, 15
- Available reports, 90
- Basic length-weight relationship, 12
- Bayesian estimation, 20, 21
- Beta prior, 25
- Beverton-Holt recruitment, 4, 5
- Binomial likelihood
  - proportions-by-category, 33
- Binomial likelihood (normal approximation)
  - proportions-by-category, 33
- BOOST C++ library, 2
- Bounds, 18
- Calculation of length-at-age (in an age-based model), 12
- Calculation of mean weight, 13
- CASAL, 6
- Categories, 43
- Category transition, 4
- Citation, 1
- Citing CASAL2, 1
- Command
  - additional\_prior, 69
  - age\_length, 54, 103
  - ageing\_error, 89, 103
  - catchability, 67, 104
  - categories, 43
  - derived\_quantity, 53, 104, 106
  - estimate, 60, 104
  - include, 93
  - Include files, 13
  - initialisation\_phase, 42, 105
  - length\_weight, 57, 118
  - likelihood, 88, 105
  - mcmc, 65, 106
  - minimiser, 63, 106
  - model, 41, 107
  - observation, 70, 107
  - penalty, 67, 112
  - process, 44, 113
  - profile, 67, 115
  - report, 90, 116
  - selectivity, 58, 117
  - time\_step, 43, 118
  - time\_varying, 52
- Command block format, 12
- Command line arguments, 9, 10
- Commands, 11
- Commands
  - Subcommands, 12
- Commenting out lines, 13
- Comments, 13
- Common Public License, 1
- Constant mortality, 7
- Constant Recruitment, 5
- Convergence failure, 19
- Correlation matrix, 19
- Covariance matrix, 17, 19
- Defining ageing error, 89
- Defining catchability constants, 67
- Defining penalties, 67
- Defining priors on parameter ratios, differences and means, 69
- Derived quantities, 5, 11, 53
- Derived quantities by cell, 5
- Determining parameter names, 13
- Differential evolution minimiser, 2, 18
- Estimable parameters, 9
- Estimated parameters, 6, 12
- Estimating parameters, 18
- Estimation methods, 60
- Estimation section, 6
- Event mortality, 7
- Examples, 95
- Exit status value, 14
- Finite differences minimiser, 2, 18
- Getting help, 2
- Hessian, 17, 19
- Include an external file, 93
- Including external files, 10
- Initialisation, 2, 3, 15, 42
  - phases, 3
- Input configuration file, 6, 9

- Input configuration file syntax, 11
- Instantaneous mortality, 8
- Interpolation of length-at-age, 12
- Length-weight, 57
- Likelihoods, 27, 88
- Linux, 1, 2, 6, 9
- Local minimums, 19
- Lognormal likelihood
  - abundance, 35
  - biomass, 35
  - proportions-at-age, 31
- Lognormal prior, 24
- Maturation, 9
- Maturity, in models without maturing in the partition, 13
- Maximum exploitation rate, 8
- Maximum posterior density estimate (MPD), 17
- MCMC, 17, 21
- Microsoft Windows, 1, 2, 6, 9
- Migration, 9
- Mingw, 2
- Model
  - annual cycle, 5
  - derived quantities, 5
  - derived quantities by cell, 5
  - initialisation, 15
  - partition, 5
  - processes, 5
  - state, 5
  - time-steps, 5
- Model overview, 5
- Model structure, 41
- Model years, 4
- Monte Carlo Markov Chain (MCMC), 17, 21, 65
- Mortality, 4, 7
- MPD (Maximum posterior density estimate), 17
- Multi-phase iteration, 3
- Multinomial likelihood
  - proportions-at-age, 30
- Necessary files, 2
- Normal likelihood
  - abundance, 35
  - biomass, 35
- Normal prior, 24
- Notifying errors, 2
- Objective function, 18
- Objective function evaluations, 19
- Observation section, 6, 7
- Observation types, 70
- Observations, 27
- Optional command line arguments, 11
- Output header information, 10
- Parameter names, 13
- Partition, 5
- Point estimation, 18, 63
- Population processes, 4
- Population section, 6, 15
- Population structure, 15
- Posterior profiles, 20
- Priors, 18
- Priors
  - Beta, 25
  - Lognormal, 24
  - Normal, 24
  - Uniform, 24
  - Uniform-log, 24
- Process error, 35
- Processes, 5, 44
- Profiles, 17, 67
- Projection year, 2
- Projections, 4
- Proportions-at-age across aggregated categories, 29
- Proportions-at-age for a single category, 28
- Proportions-at-age for multiple categories, 29
- Proportions-at-age observations, 27
- Proportions-by-category observations, 31
- Pseudo-observations, 37
- Quasi-Newton iterations, 19
- Random number generator, 2
- Recruitment, 4
- Recruitment
  - Beverton-Holt, 4
  - Constant, 4
- Redirecting standard error, 10
- Redirecting standard out, 10
- Redirecting standard output, 10
- Report commands and subcommands, 90
- Report section, 6, 7
- Reports, 39
- Reports section, 39
- Run years, 2
- Running CASAL2, 9
- Schnute growth curve, 12

- Selectivities, 14, 58
  - All-values, 15
  - All-values-bounded, 15
  - Constant, 14
  - Double-exponential, 16
  - Double-normal, 16
  - Increasing, 15
  - Inverse-logistic, 15
  - Knife-edge, 15
  - Logistic, 15
  - Logistic-producing, 16
  - Spline, 16
- Setting the number of threads, 11
- Simulating observations, 36
- Size-weight relationship, 12
- Software license, 1
- Specifying the parameters to be estimated, 18
- standard error, 10
- standard output, 10
- State, 5
- Subcommand argument type, 12
- Successful convergence, 19
- System requirements, 1
- Tag release events, 11
- Tag shedding rate, 11
- Tasks, 9
- Technical specifications, 2
- The Betadiff minimiser, 20
- The differential evolution minimiser, 19
- The estimation section, 6, 17
- The numerical differences minimiser, 18
- The objective function, 17
- The observation section, 6
- The population section, 6, 15
- The report section, 7, 39
- The state object and the partition, 17
- Threads
  - setting, 11
- Time sequences, 0, 2
- Time varying parameters, 52
- Time-steps, 43
- time-steps, 5
- Uniform prior, 24
- Uniform-log prior, 24
- User assistance, 2
- Using CASAL2, 9
- Version number, 1
- von Bertalanffy growth curve, 12
- Weightless model, 13