An introduction to CASAL^2

C.Marsh

October 29, 2015

1 Introduction

This document is a help guide for CASAL^2 (C++ Algorithmic stock assessment laboratory), aimed at users who are new to this program. As the names suggests CASAL^2 is a generalised tool for carrying out stock assessments. There are a range of benefits for using CASAL^2 for catch-at-age/length stock assessments, The first being it is generalised, so has many settings that can be turned on and off for certain situations, it is well tested and a lot of effort has been put into the code base so that it can be easily adapted to new theory and development in integrated stock assessments. This program is covered under "Some licence" see the terms and conditions. There is also supplementary information that may be useful to have access while reading this. This includes a more comprehensive and detailed user manual which breaks down all the processes, a document on how the model is validated and another validation document comparing CASAL^2 predecessor with CASAL^{2}These can all be found in CASAL^2 file that is installed with the installer.

2 How it works

CASAL^2 is program run from a command prompt (windows) or shell (linux), it takes text files as the input. These text files define the model structure and the output wanted, using the command interface the user can then choose what mode to run the program in. For help on the parameters available and there descriptions type `CASAL2 --help`, this will print a help screen. There are multiple modes that CASAL^2 can be run in, these are specified using the following command statement the program name followed by the mode parameter (e.g.`CASAL2 -parameter`). The modes and corresponding parameters include deterministic run `-r`, parameter estimation `-e`, parameter profiling `-p`, mcmc runs `-m`, and projections `-f`. There are two ways of printing output, the default is to print all output to screen, the second is to print output to a file. The second is usually the preferred if you intend on post processing output i.e. create plots. The following example shows how to read in text file that out model is configured in (`My_model.txt`) and run an estimation on some parameters in that model, then print the output to a file named `output.txt`.

```
CASAL2 -e -c My_model.txt > output.txt
```

CASAL2 calls the program, -e tells the program it is going to do an estimation. -c is the parameter that gives the name of the text file with the configured model is, and > is the command to specify the file name where the output is printed.

3 Syntax of a CASAL^2 file

A general structure of CASAL^2 files are that they are split into blocks of subcommands. A block always starts with @ symbol. Blocks describe different aspects of the model, fundamental blocks to have in the model are @model, @initialisation_phase, @categories, @time_step, and @process. Within each block there will be subcommands some will be optional and important subcommands will be mandatory. An example of subcommand is shown for the @model block,

```
@model
type age ## is the model age or length based?
min_age 1 ## minimum age in model
max_age 17 ## maximum age in model
age_plus true ## is the last age group a plus group?
start_year 1972 ## the first year of the model
final_year 2013  ## the first year of the model
initialisation_phases phase1
## The label for the block @intialisation_phase
time_steps step1 step2
## Labels for the block @time_step
```

The subcommands are all the options that follow @model, then there is a space which is where the value for the subcommand goes, i.e. min_age specifies the minimum age in the model and we have set that equal to one but could be any integer. This brings up a useful concept to understand. Different subcommands can take different types of parameters, they can be of type int, double, string and vector. For information about which parameter type a subcommand takes, you should read the syntax section of the manual, there is a field labelled type. If you use the wrong type for a subcommand, for example min_age 1.5, you will get an error. Anything following a # is a comment and is ignored by CASAL^2. It is a useful tool for annotating models.

4 Structure of a model

Will need help with this one as I haven't actually built a model yet.

When setting up a model, I split the model into four sections these are

1. Population
2. estimation

3. observation

4. report

The population section defines the categories that make up the partition, the annual cycle and processes that occur to the partition, along with the parameters that control those processes. The estimation sections defines any parameters that the user wants estimated and any prior information associated with parameters or processes. The observation section defines all the observations and there corresponding likelihoods that make up the objective function. The report section defines all the output the user wishes to have at the end of model runs e.g. objective scores, residuals and the state of the partition at a point in time. To keep models readability I split these sections into there own text file. I have a casal2.txt file that has only the following commands

```
!include "population.txt"
!include "estimation.txt"
!include "observation.txt"
!include "report.txt"
```

The `!include` function looks around in the current directory for the filename that follows it, and reads them all in as a single model.

5 Components of a model

Components of a model that are important to know before setting up a CASAL^2 model are, How many categories are in the partition, what processes occur to which categories in which order, where observations fit in to the model, and what the assumed state of the partition is before the model years run. CASAL^2 runs in yearly cycles each year is split up by time steps, So processes such as fishing and spawning seasons will have an effect on how to specify specify time steps and so will observations such as annual surveys. The next section runs through a very simple example,

6 Simple Example

In the following example describe a situation then go on to configure a CASAL^2 file to run. In this example we have a single area, single stock that has one fishery associated with it. We assume that the partition is made up of a single category (no sex or maturity in the partition). Processes and observations that occur in a typical year in the following order.

1. Recruitment

2. Fishing mortality with natural mortality

3. A survey takes place out of the fishing season and in the spawning season

4. More natural mortaltiy

5. At the end of the year all the fish are aged.

The following model would have the following structure.

```
@model
type age ## is the model age or length based?
min_age 1 ## minimum age in model
max_age 17 ## maximum age in model
age_plus true ## is the last age group a plus group?
start_year 1972 ## the first year of the model
final_year 2013  ## the first year of the model
initialisation_phases phase1
## The label for the block @intialisation_phase
time_steps step1 step2
## Labels for the block @time_step

@categories
format Stock ## format of the category labels
names CHAT4 ## category labels
age_lengths CHAT4_AL ## Lables of age-length relationship for each category
```

The `@categories` command defines the label, number and age-length relationship of categories that make up the partition. A category is a group of individuals that have the same attributes, some examples of such attributes are, life history and growth paths. Characters in a populations that cause differing attributes can be, sex, maturity, multiple area, multiple stock's and tagging information. An example of the `@categories` block for a simple two area model with male and female in the partition.

```
@time_step step1  ## The label from the @model subcommand
processes Recruitment Mortality  ## Labels for @process block

@time_step step2
processes Mortality Ageing
```

The `@time_step` command describes which processes are implemented and in what order. We will continue on from the `@model` block example, where we defined two time steps in the annual cycle (`time_steps step1 step2`). In each year we have two time steps, within each time step we have processes each process must be derined in `@process` block the following processes are described.

```
@process Recruitment ## label of process form @time_step
type recruitment_constant  ## keyword relates to a specific process
## The following are specific subcommands for this type of process
r0 4E7 ## Number of average recruits if no fishing were to occur
age 1 ## age of recruits when entering the partition
```

```
categories CHAT4 ## label of categories that recruits join
proportions 1 ## proportion ofrecruits to each category

@process Mortality
type mortality_instantaneous
categories CHAT4 ## category labels
M 0.19 ## natural mortality rate
selectivities One ## label to a @selectivity block
## this selectivity allows for age varying mortaltiy
time_step_ratio 0.4 0.6 ## If this process is in multiple @time_step blocks
## then this is the proportion of M that occurs in each time step.
table catches
year Fishing
1975 80000
1976 152000
1977 74000
1978 28000
1979 103000
1980 481000
1981 914000
end_table

table fisheries
fishery   category  selectivity  u_max  time_step  penalty
Fishing    CHAT4    FSel  0.7  step1  Catchmustbetaken
end_table

@process Ageing
type ageing
categories CHAT4_AL
```

The above defines all the processes that occur to the partition. In the process `Mortality` we associate a selectivity to natural mortality and in the fisheries table `FSel`, this would be defined as follows.

```
@selectivity One
type constant
c 1

@selectivity FSel
type double_normal
mu 3.82578
sigma_l 1.63038
sigma_r 17
```

If a age-length relationship is specified in the `@categories` block then the `@age_length` block needs to be defined, this block is used to convert age to length which is then used to convert length to weight in an age based model, it is specified as follows,

```
@age_length CHAT4_AL
type von_bertalanffy
length_weight CHAT4_LW ## label for @length_weight block
k 0.164
t0 -2.16
linf 100.8

@length_weight CHAT4_LW ## label from @age_lenght block
type basic
units tonnes
a 4.79e-09
b 2.89
```

The last important block to complete the population text file, is the `@initialisation_phase`. This block of commands specifies how you initialise your partition. This describes the state of the partition before `start_year` of the model, usually this is an equilibrium state. The subcommands available for this block are as follows,

```
@initialisation_phase phase1
type iterative ## Type of initialisation method see manual for more
years 100 ## How many years to run for
```

In the above example we have an iterative initialisation type. This will default to iterating your annual cycle for 100 years, which may or may not cause your partition to hit an equilibrium state. **N.B.** when using this initialisation method you as the user must check if the partition has reached an acceptable equilibrium state.

The next section we are defining is the observation section. We have a survey that occurs in the second time step, which is of relative abundance, this would be defined as follows.

```
@observation Survey ## label of observation
type biomass  ## tyoe of observation
time_step step2 ## which time step the observation occurs
time_step_proportion 0.5 ## the observation occurs half way through the time step
categories CHAT4
selectivities One
catchability q ## The label for @catchability block
years  1992  1993  1994  1995
obs  191000 613000 597000 411000
error_value  0.41 0.52 0.91 0.61
likelihood lognormal ## likelihood to use for the objective function

@catchability q ## label from @observation
q 0.001 ## The value
```

7 Extended example

Add a spawning stock biomass catch at age data multiple categories

8 Analyses of output

CASAL^2 has an extract R library which takes output from `@report` blocks and imports it into R as a list. This library can be found in the directory where you installed the program. There is also another library that helps pull out compress useful information such SSB's and Objective scores for datasets.

```
library(CASAL2)
out = extract(file = "Output_file.txt", path = "Directory_of_file")
```

The first thing to do is install `CASAL2` and set in your computers path or put the executable (.exe) in the directory you are working in. CASAL^2 is run through command prompt for windows and a shell for linux. To view the help in `CASAL2` open a command prompt and type `isam --h`. If you get help output printed to the screen then you have successfully installed `CASAL2` and can continue with the model, if you don't get the help screen you need to install `CASAL2` correctly.

9 Configuring a CASAL^2 model

A general layout of `CASAL2` is that it is split into blocks of commands. A block always starts with `@` symbol. Blocks describe aspects of the model, fundamental blocks to have in the model are `@model`, `@initialisation_phase`, `@categories`, `@time_step`, and `@process`. Within each block there will be subcommands some will be optional and important subcommands will be mandatory. An example of subcommand is shown for the `@model` block,

```
@model
type age ## is the model age or length based?
min_age 1 ## minimum age in model
max_age 17 ## maximum age in model
age_plus true ## is the last age group a plus group?
start_year 1972 ## the first year of the model
final_year 2013  ## the first year of the model
initialisation_phases phase1
## The label for the block @intialisation_phase
time_steps step1 step2
## Labels for the block @time_step
```

The subcommands are all the options that follow `@model`, then there is a space which is where the value for the subcommand goes, i.e. `min_age` specifies the minimum age in the model and we have set that equal to one but could be any integer. This brings up a useful concept to understand. Different subcommands can take different types of parameters, they can be of type int, double, string and vector. For information about which parameter type a subcommand takes, you should read the syntax section of the manual, it has field labelled type. If you use the wrong type for a subcommand, for

example `min_age 1.5`, you will get an error. Anything following a `#` is a comment and is ignored by `CASAL2`. It is a useful tool for annotating models.

The `@categories` command defines the label, number and age-length relationship of categories that make up the partition. A category is a group of individuals that have the same attributes, some examples of such attributes are, life history and growth paths. Characters in a populations that cause differing attributes can be, sex, maturity, multiple area, multiple stock's and tagging information. An example of the `@categories` block for a simple two area model with male and female in the partition.

```
@categories
format sex.area ## format of the category labels
names male.east female.east male.west female.west ## category labels
age_lengths male_AL female_AL male_AL female_AL
## Lables of age-length relationship for each category
```

The @time_step command describes which processes are implemented and in what order. We will continue on from the @model block example, where we defined two time steps in the annual cycle (time_steps step1 step2). In each year we have two time steps, within each time step we can have a number of processes that affect the partition an example of this is,

```
@time_step step1  ## The label from the @model subcommand
processes Recruitment M  ## Labels for @process block

@time_step step2
processes Migration M Fishing Ageing
```

N.B. that order is important, there are many combinations of processes that you can specify. Each having a different effect on the partition. This is a flexible adaptation that must be considered if you are using CASAL2 to compare against other models. An example is that in the predecessor version of CASAL2 there is fixed order in which processes occur in a time step (keep this in mind when setting up the model). Each process label in the @time_step block correspond to a @process block. @process block describe what that process does and which categories it affects, two examples are given,

```
@process Recruitment ## label of process form @time_step
type recruitment_constant  ## keyword relates to a specific process
## The following are specific subcommands for this type of process
r0 4E7 ## Number of recruits
age 1 ## age of recruits when entering the partition
categories male.east female.east ## label of categories that recruits join
proportions 0.5 0.5 ## proportion ofrecruits to each category

@process M
type mortality_constant_rate
categories male.east female.east male.west female.west ## category labels
M 0.19 ## natural mortality rate
selectivities One ## label to a @selectivity block
## this selectivity allows fot age varying mortaltiy
time_step_ratio 0.5 0.5 ## If this process is in multiple @time_step blocks
then this is the proportion of M that occurs in each.
```

There are many other @commands that branch off processes one example in the process M is @selectivity. This creates linking blocking statements, which can make configuration files easy and clear to read.

The last important block is the @initialisation_phase. This block of commands specifies how you initialise your partition. This describes the state of the partition before start_year of the model, usually this is an equilibrium state. The subcommands available for this block are as follows,

```
@intitalisation_phase iphase1
type iterative ## Type of initialisation method see manual for more
years 100 ## How many years to run for
```

In the above example we have an iterative initialisation type. This will default to iterating your annual cycle for 100 years, which may or may not cause your partition to hit an equilibrium state. **N.B.** when using this initialisation method you as the user must check if the partition has reached an acceptable equilibrium state. That was a quick introduction into the syntax and general make up of `CASAL2`. A nice feature of `CASAL2` is that it has sensible error messages, usually telling you the file and line at which the error has occurred.

other useful blocks are `@report`, `@estimation` and `@observation`. Without the `@report` your model will run and exit, which is of little use. Some examples of useful outputs from a model are of the initial partition, selectivity values, and biomass estimates. `@estimation` blocks specify whether a parameter will be estimated in `-e` run, and to estimate parameters you need observations which are specified in the `@obseravtion` block.