# Management Strategy Evaluation for Abalone Fisheries

Malcolm Haddon

Updated on 20 April, 2025

# Table of contents

# Preface

The aim of this document is to introduce and provide examples for using the Management Strategy Evaluation (*MSE*) software encapsulated within the **aMSE** R package (and associated required packages). While the **aMSE** package is aimed at abalone fisheries, it could also be applied to any hard to age, spatially complex invertebrate fishery such as Beche de Mer. Nevertheless, at least in this first edition, all examples will focus upon abalone fisheries, both real and hypothetical.

## What are Harvest Strategies?

Formal Harvest Strategies (HS) have three components, the data used by the HS, the analyses or assessment used to produce the fishery performance measures and combine them if there are more than one, and the harvest control rule used to translate the output or final score from the analyses into a future catch level (Figure 1). They can be either model-based or empirical. Integrated size-based assessment models can be used with abalone fisheries but generally they tend to be managed using empirical harvest strategies, especially where an exploited species does not meet dynamic pool assumptions.

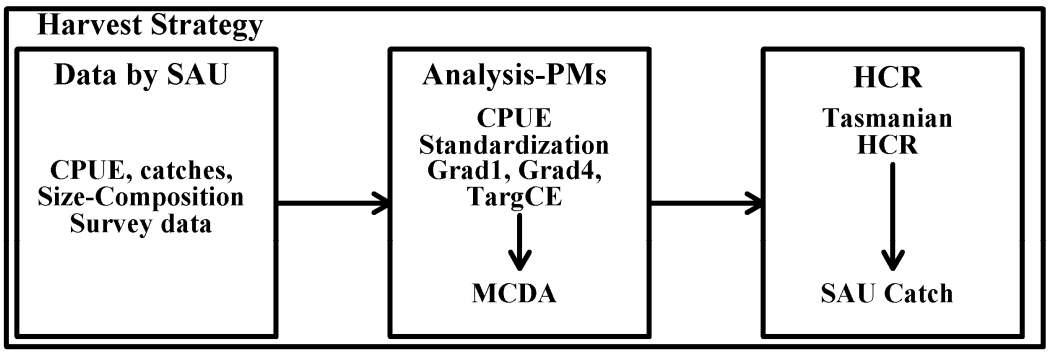


*Figure 1: A diagrammatic representation of the components of a formal Harvest Strategy (HS) as applied to Tasmanian abalone SAU. The three fishery performance measures used are Grad1, Grad4, and TargCE, each based on CPUE (see Bradshaw, 2018).*

## Testing of Fishery Harvest Strategies

At its heart, *MSE* is about testing harvest strategies using simulation. The need for such simulation testing arises because while harvest strategies are invariably devised with the best intentions to provide for sustainable fisheries management of profitable fisheries, even relatively simple systems can develop what are termed *unintended consequences*. While unintended outcomes can be benign, they can also lead to pathological behaviour, such as an inability to increase or decrease total allowable catches when such changes are needed. It is possible to apply new harvest strategies to real fisheries and discover their implications, both good and bad in real time, over a possibly large number of years. However, it is less risky to

simulate the fishery and its management under alternative harvest strategies and, at very least, eliminate the worst options while retaining those that appear beneficial. Enabling such simulations is the intent behind producing **aMSE**.

## The Design of the aMSE R Package

Apparently, when writing software, one should start with a very precise specification of the intent of the software, which should define the inputs and how those are manipulated to generate the outputs. Unfortunately, in our current 'real' world, in terms of abalone (and every other hard-to-age, spatially complex invertebrate) it is clear that not everything that it would be useful to know is already known. In addition, the software is intended to be generally usable by, at least, all Australian abalone jurisdictions and this immediately meant that the software needed to be flexible and adaptable. Thus, rather than starting with a precise specification a more iterative design process was undertaken that could react to innovations such as providing clear descriptions of such things as environmental perturbations.

The underlying population dynamics used to describe changes in the simulated abalone stocks (extremely hard to age both consistently and accurately) was based upon following how the numbers-at-size changed as a result of natural mortality, somatic growth, fishing mortality, recruitment, and small amounts of larval movement. The spatial complexity possible within each simulated fishery also had to be flexible and a scheme of sub-dividing each simulated quota zone into a set of spatial assessment units, each of which could contain a variable number of almost independent populations was adopted. The simplest arrangement possible within **aMSE** is now a Zone with a single spatial assessment unit (*sau*) with only a single population. The upper level of complexity has not been defined but in principle there could be dozens of *sau* with hundreds of populations. In reality, the difficulties in relating the simulation to a real fishery, i.e. conditioning the model on a real fishery is constrained by the availability, and spatial scale, of information and data) will be what places a limit on the spatial complexity adopted in any given case.

The focus of this document is upon the **aMSE** software but the requirement of maintaining flexibility in its use led to the development of two new auxiliary R packages (see **sizemod** and **makehtml**) and the further development of two others (see **codeutils** and **hplot**). When conditioning the simulation model, so that it closely represents a real fishery, it was found that in order to mimic the observed dynamics of the known history of a given fishery it was necessary to estimate how recruitment varied through the time over which observations were available. Initially, this was attempted within the **aMSE** software, and this option is still available. However, it was eventually decided (iterative design) that a separate R package, now called **sizemod**, should be written. This is used to fit a fully articulated size-based integrated stock assessment model (Punt *et al*, 2013) to data available from each *sau* separately. This permits an estimate of the productivity of each *sau* along with providing for the best available description of the historical fishery data. Parameter settings optimized through sizemod are then passed to **aMSE**. Once all *sau* are thus conditioned and adapted within the **aMSE** framework, the *MSE* then projects that conditioned model forward under the control of the alternative harvest strategies so that comparisons of the outcomes can be made. As can be seen in the chapter titled *Using **sizemod** to condition the SAU* package, very different outcomes from the modelling can be obtained simply by selecting different values of natural mortality, steepness, and the hyperstability level of the CPUE. As none of these values are known with any confidence it is not sensible to recommend using **sizemod** in a formal stock assessment except, perhaps, in those SAU with large amounts of biological and fishery information and in a weight-of-evidence context.

An important aspect of any harvest strategy is that it ought to be adaptable and open to improvement. Hence, it was decided that the software code used to implement each jurisdiction's harvest strategy would not be built into **aMSE** but would rather be one of the inputs from each jurisdiction. Initially, this took the form of what is known as a 'source' R file but eventually, for Tasmania, this was converted into yet another a specific R package **TasHS** (for the purposes of the examples in this guide an earlier version of the Tasmanian HS was produced, **EGHS**, which is the one to be used with this guide). This also meant that HSs developed by each jurisdiction could be maintained and managed more easily. It is recommended that each jurisdiction that wants to use **aMSE** should develop such a package, although the somewhat less efficient option of using a 'source' file of suitable functions remains.

In addition, the other R packages that have been produced provide utility routines for general programming (**codeutils**) and plotting routines (**hplot**), while the third (**makehtml**) provides a convenient method for setting up internal websites that show the plots and tables of results across a number of different pages/tabs, which enable examination of results from single runs and comparisons. While this may all sound complex, worked examples in the pages that follow should clarify how these things interoperate.

## The Intended Users

*MSE* can be a complex business (Punt *et al*., 2016). **aMSE** is written entirely in base R (R Core Team, 2024) and is open-source software. This means that anyone with a knowledge of R can discover how it has been implemented. In addition, it can be further developed and maintained by anyone with the correct expertise. But it does not have a nice *shiny* interface (which would need to have had horrendously complex input screens). Because of this a certain facility with the R language is required to run the software, though the threshold requirement is relatively low, and template R code files are provided for running the software (and can even be generated for a user by the software). Much of the difficulty or complexity in use is setting up each simulation to represent a real fishery. It must be emphasized that such representation (known as *conditioning* the operating model) need not always be highly specific. Simulation software can be used to explore the implications of many somewhat more abstract abalone-like fisheries rather than the specific Tasmanian western zone blacklip abalone fishery used in the examples. Describing the conditioning of the simulation model is an important part of this documentation.

One reason **aMSE** does not have a nice 'shiny' graphical interface is to retain a high degree of flexibility during this phase in the evolution of this software. This implies that the user needs to understand much of that flexibility to ensure they do not inadvertently represent the fishery being simulated incorrectly. Ongoing developments are expected to include increasing the number of diagnostics that will highlight assumptions and constraints.

## Living Documentation

**aMSE** is not designed to be a fixed or static piece of software. It is expected that new plots, tables, and other outputs will be produced by users and there is no reason these should not be included in the code base for everyone to use. Having common outputs, plots, and tables would make discussion and comparisons between jurisdictions and different users so much easier. For this reason, the series of Quarto documents that make up this document have also been placed onto a GitHub account, currently at https://www.github.com/haddonm/aMSEGuide so they can be amended and improved whenever required. This documentation should therefore always be considered a draft that

can be improved. It is available as a readable product at https://haddonm.github.io/AMSEGuide/ from where a PDF of the whole can also be downloaded.

## Acknowledgements

Thanks are given for the funding for this project, which was primarily derived from both the Australian Fisheries Research Development Corporation (FRDC Project 2019-118) and the University of Tasmania. Further funding has been provided by the Abalone Industry Reinvestment Fund (projects 2023-63 & 2023-65), which led to expansion and improvements to the aMSE code-base.

# 1. Introduction

## 1.1 Fisheries Assessment and Management

One of the fundamental problems within fisheries management is that it is difficult to measure the status of different harvested stocks directly because surveys are expensive and, despite the cost, remain uncertain. Fortunately, it is generally possible to infer their status from samples, or models fitted to data based upon samples, although these also only provide an uncertain view of a stock. Happily, developing longer time-series of fishery observations (such as catches, catch rates, age- or size-structure data, and many other types of observation) can improve our understanding and modelling of events (assuming the quality and coverage of such data is good enough - a very big assumption). Nevertheless, there always remains a degree of uncertainty in any stock or fishery assessment. Such issues are of even greater importance in the many data-poor or data-limited fisheries and species globally (Vasconcellus and Cochrane, 2005; Pikitch *et al*, 2012). Despite the prevalence of uncertainty, fishery managers are still required to make decisions. This uncertainty, and the consequent difficulties it leads to, have not always been recognized (Smith, 1988) though now the most effective fisheries management jurisdictions attempt to account for uncertainty in explicit ways.

In the 19th century (and into the early 20th) many people believed that the exploitation of natural resources did not require management. This idea, which should now hopefully seem strange, has fortunately evolved into an acceptance that management of wild fisheries is essential, and a wide array of management approaches are now being used around the World (Smith, 1988; Hilborn, 2012). The objectives which systems of fisheries management attempt to achieve have also greatly changed through time. When declines in large fisheries were first identified at the end of the 19th century the focus mainly involved a combination of wanting to maintain catch rates (so as to fish economically) and to maximize the yields from different fisheries (Garstang, 1900). At that time the primary objective was to maximize yield, but it took some years before it was recognized that for many species applying more fishing effort did not necessarily lead to increased catches (the yield-per-recruit problem; Russell, 1931, Beverton & Holt, 1957). It is difficult now to grasp the limited and simplistic view of how fisheries ought to be managed that existed in the 1910s, and extended even up to the 1960s. Larger scale attention only began to be paid to fisheries dynamics and management after the late 1950s, with real progress only commencing in the 1980s onwards (Fournier and Archibald, 1982; Methot, 1989, 1990).

Prior to the late 1950s most thought was given to increasing catches and the efficiency of fishing gear and it still seemed contrary to intuition to recommend limiting catches. For example, at the second FAO conference in 1946, immediately following the second world war, the FAO was strongly urging the development of fisheries as a source of protein and food: “The fishing grounds of the world are teeming with fish of all kinds. Fisheries are an international resource. In underdeveloped areas especially, the harvest awaits the reaper.” (FAO, 1985). The consequences of uninhibited fishing were poorly conceived at that time and attempts to correct the outcomes of such misconceptions from that time are on-going.

Early deterministic stock assessment approaches effectively ignored uncertainty and tended to produce management advice based on the assumption that natural populations are in equilibrium with each other and with any fishing effort imposed on them (Schaefer, 1954, 1957; Gulland, 1965; Megrey, 1989). Assumptions of equilibrium and stability are clearly only an approximation and are invalid in many cases but nevertheless this approach led to

concepts such as the Maximum Sustainable Yield (*MSY*), which related to catch levels, and $F_{max}$, the fishing mortality which related to the effort expected to lead to the maximum yield. Unfortunately for many fisheries, catches relating to $F_{max}$ could be larger than the *MSY*. Both these concepts were early fisheries targets or objectives, with fisheries legislation in many countries still including *MSY* as a primary aim of management. Sadly, the same legislation often neglects to define the concept of *MSY* (although this can be interpreted as an advantage). In the 1970s it became apparent, following the collapse of a number of fish stocks, that *MSY*, as it was then interpreted, was not the safest objective to adopt (Larkin, 1977) and more serious efforts were made to find safer alternatives.

Although the concept of *MSY* is still invoked it has evolved into use as an upper limit to fishing mortality or has been redefined to account for risks of alternative catch levels (Smith and Punt, 2001). In the 1970s and early 1980s, input controls relating to effort, gear, vessel numbers, and closed seasons were the management tools in most fisheries and some of the more successful management objectives focussed on defining an optimum fishing mortality rate. This work led to the concept of $F_{0.1}$, which despite being *ad hoc*, was an advance over $F_{max}$ in terms of sustainability as well as profitability. It usually led to a large reduction in fishing effort (reduction in fishing mortality and costs) but only led to a minor loss in yield (see Hilborn & Walters, 1992, for definitions of such classical fishery objectives). Even though this was an improvement over $F_{max}$ or $F_{msy}$ it was still based on the notion that fish stocks were able to achieve equilibrium with the fishing mortality imposed on them. While this was then assumed to be, at best, an approximation there was still a great deal of development needed to produce the methodologies required for taking uncertainty into account.

The importance of acting to provide management advice in the face of uncertainty was a growing theme in fisheries resource management through the late 1980s and early 1990s. The need to act before scientific consensus could be achieved rather than calling for more research was identified as a key problem for management (Ludwig et al., 1993). The precautionary approach in fisheries is based upon the notion that a lack of scientific certainty about the risk of serious environmental damage must not be used as an excuse for not acting to prevent that damage (FAO, 1995, 1996, 1997).

## 1.2 Formal Recognition of Harvest Strategies

As stock assessments became more sophisticated so were the management options that were developed. In the late 1980s and early 1990s the effects of variability, uncertainty, and associated risks began to be addressed in stock assessments (Francis, 1992) and the notion of presenting a decision table of management options with their associated risks was also developed. Hilborn & Walters (1992, p453) defined a harvest strategy as:

"…a plan stating how the catch taken from a stock will be adjusted from year-to-year depending upon the size of the stock, the economic or social conditions of the fishery, conditions of other stocks, and perhaps the state of uncertainty regarding biological knowledge of the stock."

The harvest strategies discussed at that time revolved mainly around the classical three: 'constant catch' (e.g. TACs; output controls), 'constant fishing mortality' (e.g. $F_{0.1}$; input controls), and 'constant escapement' (e.g. always leaving at least 75% of estimated Mackerel Icefish biomass in the Heard and McDonald Island fishery; mixed input and output controls).

Harvest strategies in the early 1990s focused mainly on setting out fishery objectives (defining biological reference points; Smith et al., 1993) and what constraints should be used. In more recent parlance, this was about determining how to assess each stock's status and what limit and target reference points to put in place. These developments may have been encouraged, at least in part, by new legislation in the USA (the Magnuson & Stevens Act, 1976, 2007) that required definitions of overfishing that would explicitly guard against recruitment overfishing (Mace & Sissenwine, 1993).

A number of very influential documents were published by the FAO in the mid-1990s, including: the *Code of Conduct for Responsible Fisheries* (FAO, 1995), the *Precautionary Approach to Capture Fisheries* (FAO, 1996), and *Fisheries Management* (FAO, 1997); these latter two documents being parts of the Technical Guidelines for Responsible Fisheries series. The authors stated: "Long term management objectives should be translated into management actions, formulated as a fishery management plan or other management framework" (FAO, 1995, p 11). Giving more details, the Guidelines appear to be one of the first documents to describe the components of what are now termed Harvest Strategies.

The 'Guidelines' (FAO, 1996) identified the needs for:

1) targets, described as the desired outcomes for a fishery,

2) operational constraints or limits, described as the undesirable outcomes that are to be avoided, and

3) control rules which specify in advance what action should be taken when specified deviations from the operational targets and limits are observed.

Early work on simulation testing of management arrangements (now known as management strategy or management procedure evaluation) appears to have contributed to this approach to describing harvest or management strategies. Thus, in the FAO Guidelines it defines a management procedure as a description of the data to collect, how to analyze it, and how the analysis translates into actions. This is a standard way to describe a modern harvest strategy: define the data needed, the analysis of status relative to the target and limit reference points, and the control rules used to generate management advice from that status. However, in the FAO guidelines the emphasis given to management procedures was placed on the investigation of how uncertainties influenced the management process (which stemmed from how these management procedures were implemented in South Africa; Butterworth & Bergh, 1993).

The main difference to fisheries management brought about by the adoption of formal harvest strategies was the inclusion of explicit 'decision rules' or 'harvest control rules'. Prior to the introduction of harvest strategies, the data required for stock assessments was certainly collected and the primary thrust of research was the development and articulation of improved stock assessment methodology. Unfortunately, what to do with those assessments to generate management advice sometimes varied from vague to completely unclear. With the addition of formal control rules, management responses become predetermined based on the outcome of the assessment. The use of a formal harvest strategy in a fishery represents a major change to management and constitutes the primary basis for improving the consistency or repeatability, predictability, and transparency of assessment and management.

## 1.3 Properties of Harvest Strategies

An advantage of using formal harvest strategies is that the outcome of their application should not be dependent upon who applies them to data from a fishery. Given the same data collected from a fishery, anyone who applies a given harvest strategy should produce exactly the same management advice. In order for this to be the case requires harvest strategies, and their implementation, to have the following properties:

### 1.3.1 Transparent

The information requirements of each harvest strategy and the objectives being aimed for should be fully documented and publicly available. Assessments invariably use summarized data so confidentiality should not become an issue. Where confidentiality might become an issue (e.g. cpue standardization should use raw data from individual fishers), then formal external, yet confidential, review can avoid this issue. Ideally, if software is used in the application of the harvest strategy that too should be publicly available. Being open to critical review is essential for there to be trust that the management being recommended is free from potentially conflicted outside influence. All reviews take time, effort, and usually costs, so, in potentially contentious cases, care should be taken to ensure that a review is not introduced to delay decisions. One objective of every review should be to assess and maintain transparency.

### 1.3.2 Repeatable

Given the same information and an understanding of the harvest strategy, anyone should be able to generate the required management advice, and that advice should be the same irrespective of who does the work. It should make no difference to the outcome who is in the room when the work is completed and decisions are made. Transparency is critical to achieving this goal, and the consequent repeatability of the outcomes from a formal harvest strategy should instill greater confidence in the management objectives and management actions.

### 1.3.3 Adaptable

A harvest strategy should not be seen as a static invention, it must be open to improvements while retaining the properties of being fully and openly documented, and of being repeatable. Fisheries management must have the capacity of learning and improving as understanding concerning each fishery increases. Being open to review means that if an improvement can be suggested, by anyone, then a change could be made. In addition, directional environmental changes that have occurred in a noticeable manner over the last 50+ years mean that if productivity changes with location, then changes in expectations and hence in the objectives, targets, and limits for each fishery are also likely to need changing.

### 1.3.4 Defensible

The details and effectiveness of each harvest strategy must be defensible. Ideally, a harvest strategy should be simulation tested for effectiveness at meeting their objectives (using management strategy evaluation, *MSE*). Even in the absence of *MSE*, a harvest strategy's performance should be monitored and reviewed regularly. The defensibility of a harvest strategy is also highly dependent on the other three properties. If they are not transparent, repeatable, and adaptable, then they are less defensible.

# 2. Management Strategy Evaluation

## 2.1 Working with MSE

After many years of using what turned out to be overly simplified objectives for commercial fisheries management, there has been a growing trend of explicitly defining the fishery management policy to be used in each jurisdiction. These policies tend to address issues relating to sustainability, managing risk factors, managing bycatch, and actions relating to threatened and endangered species. This is a significant improvement over vague exhortations to achieve the maximum sustainable yield for each species. In addition to the explicit statements defining management objectives, there is a move to introduce explicit and formal harvest strategies aimed at achieving those objectives. There are many possible approaches to developing empirical harvest strategies, with no, as yet, clearly preferred approaches. Unfortunately, unintended consequences are not uncommon when attempting to control or manage relatively complex natural systems so, generally, to avoid potentially disastrous outcomes, it is best to test and compare any new versions of an EHS and the candidate alternative management arrangements before implementing them.

Comparing the relative effectiveness of alternative harvest strategies for particular fisheries generally requires the use of Management Strategy Evaluation (Smith, 1994; Punt *et al*. 2001; Haddon, 2007; Punt *et al*. 2016). As stated by Punt *et al*. (2016, p303):

"Management strategy evaluation (MSE) involves using simulation to compare the relative effectiveness for achieving management objectives of different combinations of data collection schemes, methods of analysis and subsequent processes leading to management actions. *MSE* can be used to identify a 'best' management strategy among a set of candidate strategies, or to determine how well an existing strategy performs. The ability of *MSE* to facilitate fisheries management achieving its aims depends on how well uncertainty is represented, and how effectively the results of simulations are summarized and presented to the decision-makers. Key challenges for effective use of *MSE* therefore include characterizing objectives and uncertainty, assigning plausibility ranks to the trials considered, and working with decision makers to interpret and implement the results of the MSE."

The simulations involve using a mathematical model to mimic both the biological and fishery/fleet dynamics of the fishery being considered. This simulation model, usually termed the *Operating Model*, is either fitted to a real fishery or conditioned (its parameters adjusted) until its dynamic behaviour reflects the observed properties of a real fishery. Once appropriately conditioned (easily stated, not necessarily simple to do), the dynamic behaviour of the operating model is taken to represent reality and is used to simulate the operation of a fishery under the different harvest strategies (*HS*) being compared or tested.

The essential aspect of such a simulation, that makes an *MSE* differ from a standard forward projection of a stock assessment under constant catch or effort (a classical risk assessment; Francis, 1992), is the built-in feedback of the regular management advice into the dynamics described by the operating model. The dynamics of the stock and fishery are simulated each year, the *HS* (data sampling, assessment/analysis, and harvest control rule) is applied however often the *HS* dictates, and the outcome of the specific *HS* is fed back into the operating model as a Total Allowable Catch or Effort, or some other management influence. Such inputs could be expected to alter the path of the expected dynamics (this constitutes the feedback loop). In this way, different *HS* can be expected to have different simulation outcomes over a set number of years (see Figure 2.1).

Not all management strategy evaluations involve simulating a real fishery. There are many general questions that can be answered using hypothetical fishery situations rather than simulating a specific fishery. In some of the examples used in this documentation only a portion of a real fishery is used, which is a mixture of realistic and unrealistic (because in reality the fleet dynamics used relates to a whole quota zone not just one or two *sau*). Nevertheless, such hypothetical situations can be used to explore the influence of different source of uncertainty on differing harvest strategies.

### 2.1.1 Simpler is Not Necessarily Better

*MSE* is often represented in an overly simplistic fashion leading to un-realistic expectations as in: 'if a given harvest strategy has been *MSE* tested, and nothing untoward found, then it must be good'. However, it is important to remember that, as with any model, if an operating model does not include a particular feature in its dynamics (e.g. does not include depensation in its recruitment dynamics, or non-linearity in the relationship between stock biomass and CPUE) then those features can, obviously, never be expressed in the simulations. The structure of an operating model needs to be well documented, and its limitations understood, so that the domain of the *MSE* testing for each *HS* is known. This is especially important to understand because harvest strategies that have been *MSE* tested are generally held in relatively high regard. However, the completeness of any *MSE* testing will always be constrained to the dynamics described by the operating model, so care in its design is required and that design needs to be defensible (see *Operating Model Structure*).

### 2.1.2 Harvest Strategies

In the literature a harvest strategy (Smith, 1997), or *HS*, aimed at achieving a defined policy objective, has a minimum of three components:

1. Specified and representative data collected from the fishery being evaluated,
2. An assessment or analysis of the fishery, based on the data collected, that estimates its status relative to pre-defined target and limit reference points,
3. A formal harvest control rule (decision rule) that defines previously agreed management advice (future catch or effort levels, etc) in response to a stock's status relative to its reference points.

Each of the three components can be varied in multiple ways and, strictly, each such combination would constitute a different harvest strategy (*HS*). Therefore, for any fishery, there will be multiple potential options available for the management. Often, fishery management involves balancing trade-offs between conflicting objectives. For example, with a valuable species such as abalone (e.g. blacklip abalone, *Haliotis rubra*) the two objectives of maintaining a sustainable stock and maximizing the profitable catch are potentially in conflict, or at least can be when considered across different time-frames. Such potential conflicts within fisheries was one reason why management strategy evaluation was developed to enable the comparison of how alternative harvest strategies perform relative to different objectives. *MSE* methods originated in the International Whaling Commission, which had its own contentious issues (Punt and Donavan, 2007). It is used to select or develop an optimum *HS*, or at least reject sub-optimal ones.

### 2.1.3 Management Strategy Evaluation of Abalone Fisheries

There have been previous attempts to conduct management strategy evaluation (*MSE*) of alternative harvest strategies for abalone fisheries in Australia (Haddon *et al*., 2013; Haddon and Helidoniotis, 2013; Haddon and Mundy, 2016). In the current project, an R package **aMSE** (Haddon, 2024) has been developed, and has evolved, from those earlier attempts. This current document constitutes part of that documentation. Other complementary R packages (**codeutils**, **hplot**, **makehtml**, and **sizemod**, (Haddon, 2024b, c, d, e), each with their own documentation) have also been generated or modified from earlier versions to assist with conditioning the operating model within the *MSE*, and the production of a standard set of summary figures and model information.

An example R package **EGHS** has been developed that implements a simpler version of the Tasmanian abalone harvest strategy in a manner useful for the examples described in this guide (Haddon & Mundy, 2024b). Ideally, each jurisdiction that uses **aMSE** would encapsulate their own harvest strategies into such a package, as this simplifies maintenance as well as the use of **aMSE**. The required documentation within each formal R package for an *HS*, and the possibility of including a vignette with more details, means that the workings of the *HS* in each case could be explained and made clear. A formal package avoids the possibility of source files of R functions being modified such that different users inadvertently end up using a different HS structure. Maintenance and possible developments within the code can also be managed more effectively when it is structured as a formal package.

The R code in the **aMSE** package is a greatly developed and translated version of the relatively undocumented Abalone *MSE* code developed and used in Haddon *et al* (2013), which had been developed further, with only slightly better documentation, in Haddon and Mundy (2016). The essential equations describing the operating model dynamics used in those earlier projects were formally published in Haddon and Helidoniotis (2013). Those dynamics have since been found to be overly complex in that they used separate vectors of numbers-at-length to describe the cryptic and emergent components of each of the many abalone populations simulated to make up a simulated quota zone. Other, unpublished, work demonstrated that this separation has no real advantages for representing the stock dynamics but does have serious computational disadvantages. The new **aMSE** package reflects this by changing the previous equations so they now use only a single vector of numbers-at-length to contain the dynamics for each year. In addition, other methods are being incrementally implemented with the intent of speeding the computations. Previously, it could take hours to appropriately explore even a single *HS*, which now, depending on the complexity of the HS, can take only minutes to run a single scenario made up of several hundred replicate runs.

The underlying objective in the current **aMSE** R package is to enable the simulations and *MSE* testing of different *HS* to proceed by users familiar with R, but without months of introductory training with the bare software code-base, as needed previously. This document is a first attempt to provide the background required for others to successfully run the software. While writing this material, sections will be borrowed freely from both Haddon *et al* (2013) and Haddon and Mundy (2016), wherever it is appropriate. These two projects, in which abalone *MSE* code was first developed, did not include sufficient time or resources to generate a more user-friendly or documented code-base for the software. Relatively user-friendly software improves the capacity of other fisheries assessment scientists to apply these methods to more stocks or to new *HS* currently being developed. By making the new R package open-source, freely available, and fully documented, the intent and hope is that more

people will find the methods workable. By being defined within open-source software this also allows for future developments of the package by others should they wish. One primary objective is to allow users to write their own functions or R packages describing any new harvest strategy they might develop.

### 2.1.4 Difficulties Inherent with MSE Testing Abalone HS

The phrase 'abalone stock' sounds meaningful and most would consider this to refer to the standard notion of populations of a species having sufficient genetic homogeneity to represent a connected reproductive whole. However, as demonstrated by Miller *et al* (2009; 2014), genetic studies indicate that abalone populations (at least of blacklip and greenlip abalone) are meta-populations made up of multiple, what might be termed, micro-stocks. Such micro-stocks are functionally independent, while remaining genetically connected such that divergence is not occuring In effect, the management areas for which total allowable catches (TACs) and minimum legal sizes (MLS or LML) are set are spatially recognizable and convenient areas or zones that are composed of numerous mostly isolated micro-stocks. The solution adopted in Haddon *et al* (2013) and Haddon and Mundy (2016) was to generate an operating model made up of numerous discrete populations of abalone, each with their own set of biological properties and fishery productivity that reflected properties observed within real fisheries. One disadvantage of this approach was that there was no chance of fitting this model to a real fishery as no real fishery has sufficient data available concerning differences in growth, size-at-maturity, and other matters relating to productivity. This meant that one could only attempt to condition the operating model to have outputs and properties that approximately matched a real fishery. Of course, as those properties were defined at a large geographical scale there would be no single or unique way of setting up a large number of individual populations to mimic such emergent or sum-of-the-parts properties.

One time-consuming option to counter this problem would be to set up the operating model's underlying populations in a number of ways to determine whether each different arrangement leads, effectively, to the same outcome. While this would be time-consuming it used to be the only way available of characterizing the full uncertainty relating to the spatial dynamics. There is now the possibility, however, that the use and collection of the GPS data-logger data can assist with the conditioning of the operating model, by allowing populations at a smaller scale to be more fully characterized, rather than at the scale of statistical reporting blocks within quota zones. Now there are ten+ years of such data in Tasmania, this database has now been used to define areas of persistent productivity that represent the multiple populations used to define the spatial assessment units (SAU) within the quota zone. Using the longer-term yield obtainable from each such smaller region as a proxy for their relative productivity provides an insight into the required dynamics from each of these 'populations' (or 'areas of persistent production'). This is discussed in the sections dealing with conditioning the operating model.

### 2.1.5 Applications in Victoria

During the research described in Haddon and Mundy (2016), an opportunity arose to apply that abalone *MSE* to some individual reefs within Western Victoria to examine how recovery might proceed under different harvest strategies following the viral destruction brought about by the AVG virus (Haddon & Helidoniotis, 2014). This differed from the applications in Tasmania as it was focussed solely on one reef complex, known as *The Crags*. While the operating model was still only conditioned on reality (rather than fitted) using the biological properties known for the abalone on The Crags, more time was spent improving the

comparison of the length composition and historical performance of the fishery. This could be done by selecting combinations of simulated populations whose combined dynamics more closely followed trends in catches and size-composition through time. However, this process was very time-consuming, so the objectives of each study need to be made clear and explicit before embarking on such detailed conditioning.

### 2.1.6 Why is the Code so Specialized

To date, one reason that running an *MSE* remains relatively specialized and esoteric is that the simulation framework needs to be able to simulate a wide range of processes (see **Figure 2.1**). These processes include a) the dynamics of the selected biological stock, b) the dynamics of the fishery imposed on the stock (the fleet dynamics), c) the generation of simulated fishery data from the fishery, d) the stock assessment or analysis applied to that data, and e) the control rule used to modify the present management options. For abalone this generally means changing the TAC, which is then fed back into the dynamics of the stock in a feedback loop within the modelling framework (Punt *et al*, 2016).

Developing and using such a complex software framework entails specialized expertise. The disadvantage of this is that it constrains its use with real fisheries and limits future developments. This is the primary reason for attempting to translate the *MSE* code into a documented R package. If more people can more easily take on the application of using the management strategy evaluation framework to explore management options for abalone stocks this should increase the chances of the adoption of improvements in management.

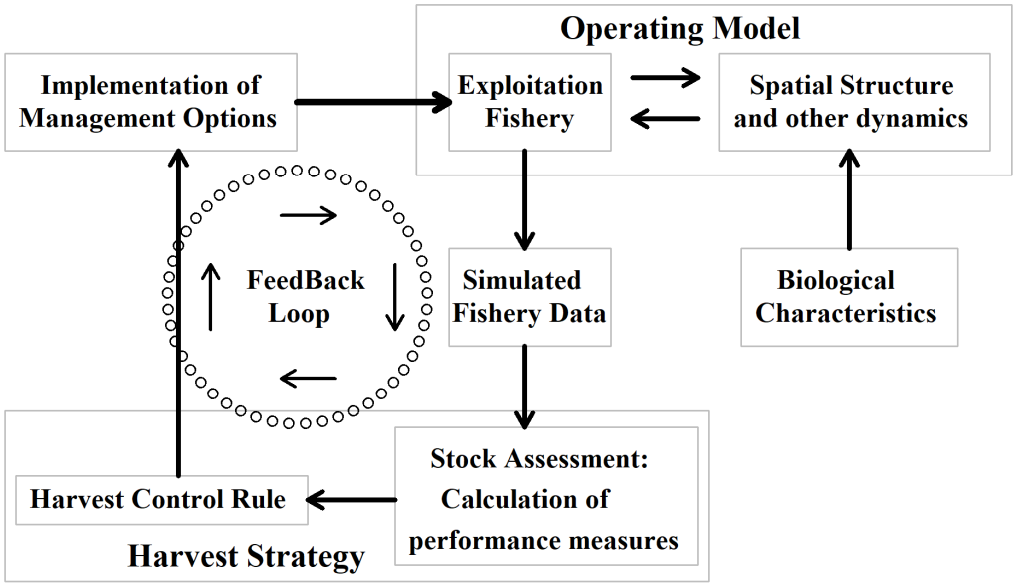


*Figure 2.1: A diagrammatic representation of the main components of an MSE simulation framework, such as used with abalone, and how the feedback cycle operates during the simulations. Redrawn from Haddon et al, 2013*

## 2.2 The Operating Model Structure

The operating model (OM) requires conditioning on a selected abalone fishery because the OM acts as a proxy for reality in the simulations. The task of the OM is to model the stock dynamics across all the micro-stocks embedded in the framework. Currently, a selected abalone fishery zone can have 10s or 100s of sub-populations, each with their own defined properties reflecting somewhat different growth, maturity, and other details relating to productivity. However, such a simulation would be extremely complex to condition to a specific fishery simply because the data requirements would be enormous. To define the operating model within the *MSE*, the number of populations (*numpop* in the R code) need to be specified and how these are conditioned will, obviously, greatly influence the outcomes.

The original abalone *MSE* (Haddon *et al*, 2013) was designed for use with Tasmanian blacklip abalone stocks. Spatially this was structured as statistical blocks within quota zones. In Tasmania, quota zones were introduced from 2000 onward whereas the statistical blocks have been extant since 1965 (Anon, 1966). Biological data was available from many sites around Tasmania although it tended to be summarized at a block level. Each statistical block could be simulated as containing multiple populations (areas of persistent productivity), and each population has its own set of properties with many such properties relating to the productivity. To attempt to capture some of the between block variations in biological properties each block simulated was given its own set of average properties. These included, for many of the properties, a definition of a statistical distribution to describe the expected distribution of values across contained populations, for such things as the growth parameters, the parameters relating to maturity-at-size, and details of the stock recruitment relationship. Thus, each sub-set of populations (in Tasmania representing a statistical block but elsewhere they might represent any spatial assessment unit), was sampled from selected distributions that had been conditioned on what is known about the fishery being simulated. There is also the option of specifying particular values of the productivity parameters (growth, maturity, recruitment) for each 'population' (see the chapter on Conditioning on Populations).

Unfortunately, in fisheries management in general, and abalone fisheries in particular, the terminology used to describe different management details and concepts is not standardized across jurisdictions. Thus, while the term *Zone* is used widely to describe a geographical area over which a particular quota/TAC may be taken, the names given to sub-zones within zones differs between jurisdictions. The statistical *block* from Tasmania is not equivalent to the *reef-code* from Victoria, although it is closer to the *spatial assessment unit* from South Australia, though tend to be larger than the *spatial management units* in Central Victoria. Similarly, in Tasmania, the phrase the *Legal Minimum Length* (LML) is used to define legal size limits, which is equivalent to the *Minimum Legal Length* (MLL), and other such abbreviations from other jurisdictions. In **aMSE** we have used *Zone* for the largest scale of management, *sau* for the current spatial scale of assessment (*sau* can either be capitalized or not), and *LML* for the legal minimum length. When using the *MSE*, people in different jurisdictions should hopefully be able to manage any required translation without excessive indignation; it is the concept that matters not the word or phrase.

In an operational simulated zone different SAUs are likely to have different numbers of contained populations, and the populations are likely to be of very different sizes, each population with its own particular properties. Within the Tasmanian application of **aMSE**, the populations have been defined from the GPS data-logger information and are more akin to 'areas of persistent productivity'.

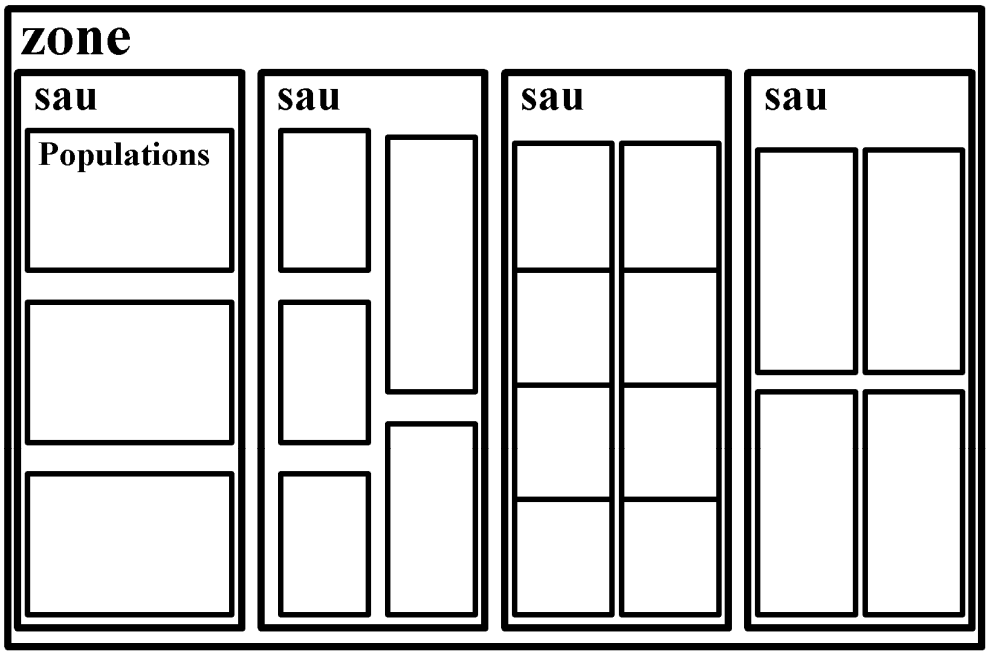


*Figure 2.2: A diagrammatic representation of a simulated zone made up of four spatial assessment units (SAUs) each made up of a number of separate populations, each with its own properties. Redrawn from Haddon et al, 2013.*

# 3. Using aMSE

## 3.1 A Worked Example

Rather than continue with theoretical considerations we will begin working through an example *MSE* run in some detail. In this way users will see what actions are needed in practice rather than deal with more abstract notions. During the development of the **aMSE** R package for FRDC 2019-118, the Tasmanian western zone was conditioned on both biological and fisheries data to provide for further development and testing of the software. The summary data required are now included as default settings in the *ctrlfiletemplate()* and *datafiletemplate()* functions. In addition, we can also use the *data(lfs)* internal data-set and the function *rewritecompdata()* to generate the size-composition data required.

In this chapter, we will use these functions to produce a working example scenario that will provide an overview of what is needed to use the software and what to expect to come out of it when run. It is recommended that the help for each function and data-set mentioned in this documentation be read for a fuller understanding of what is going on. For example, after running *library(aMSE)*, typing *?rewritecompdata*, or whatever function name is of interest, into the RStudio console - see the chapter *A Non-Introduction to R* in Haddon (2021) for details about examining R code within a package. To read a list of all visible functions within a loaded package, in the console type *?codeutils* (or whichever package is of interest). Its help screen should open. Scroll to the bottom and click on the 'Index' within [Package **codeutils** version 0.0.16 Index] and a listing of each function and its title will appear. An alternative approach is to use *ls("package:aMSE")* or *ls.str("package:aMSE")*. The latter also provides the syntax of each function.

### 3.1.1 Requirements to Run aMSE

Before running the example, it is, of course, necessary to install the required R packages containing the software used. The following description will continue under the assumption that the user will be using R with RStudio (see https://posit.co/). If some other development environment is being used, then it will be assumed that the user will be able to adapt the following to their own system.

It is necessary to install the following R packages (and their dependencies).

The first two can be downloaded from CRAN in the usual way. In RStudio, use the Packages tab and click on Install, then type their names in the input box with a space between each one:

- **rmarkdown** required for the vignettes within **aMSE**.
- **knitr** required for the vignettes and for the *kable* function that is used in the output *.html* files to generate readable tables.

Four packages specific to **aMSE** are required for all jurisdictions (**aMSE**, **codeutils**, **hplot**, and **makehtml**). The first R package, **aMSE** is one of the outputs from FRDC project 2019-118, **codeutils** and **hplot** are earlier packages that now contain extra functions added during the execution of 2019-118. **makehtml** was developed for other uses and is simply used by aMSE to generate the internal websites that display the results. In addition, a fifth user-supplied package (or R-source file) is needed, which contains functions implementing the Harvest Strategy for the jurisdiction of interest (see the *JurisdictionHS_Requirements* chapter), here we will use the **EGHS** package. The four **aMSE** related R packages are available through either installation or cloning from the GitHub account at

*https://www.github.com/haddonm*, or they can be installed from the build source files (ending in tar.gz) located in the directory: *dropbox/National_abalone_MSE/aMSE_files*. Obviously, the **EGHS** package is only required if the user is intending to use or explore the Tasmanian harvest strategy.

- **aMSE** is the primary package for running the *MSE*, it contains the functions and some data-sets that drive the *MSE* code-base.
- **codeutils** is a package containing (surprise) an array of utility functions used especially when reading files, parsing text, and so on.
- **hplot** is a package containing functions written in base R to assist with plotting various types of specialized graphics that **aMSE** uses.
- **makehtml** is a package used to automatically generate the HTML and CSS files that make up the summary internal web-page for displaying the many results from single scenarios using the MSE and, similarly, when comparing scenarios.

If installing from the R source files one would need to have copies of the latest version of **aMSE** and dependent packages sourced using the GitHub link above (where the version number is at least that shown or larger, which would imply more recent):

- *aMSE_0.3.5.tar.gz*,
- *codeutils_0.0.15.tar.gz*,
- *hplot_0.0.19.tar.gz*,
- *makehtml_0.1.2.tar.gz*, and, if using the Tasmanian HS,
- *EGHS_0.1.15.tar.gz*.

In RStudio, under the Packages tab one first presses the *Install* button and then uses the install from option box that pops up (each users library location will likely differ):

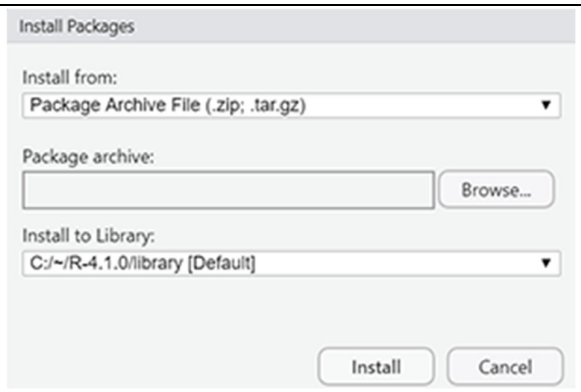


*Figure 3.1: The Install Packages interface from RStudio. The dropdown arrow is used to select 'Package Archive File (.zip; or .tar.gz)', and then you browse to point the box at the required source file (R is now version 4.4.1).*

## 3.2 Running aMSE

### 3.2.1 Organize Scenario Results

The number of possible scenarios it is possible to generate in a relatively short time is very large so to facilitate making comparisons as simple as possible, it is best to be organized about how to save the results for each scenario. The sub-directory name and path for all the input files and the results that follow for a given scenario is named in the code as the *rundir* (for hopefully obvious reasons). For example, in the text below we will introduce what will

be termed the 'EG' sub-directory (EG as in example). So, the *rundir* would be the generic path leading to all the scenario sub-directories combined with *EG/*, the actual scenario in this case. The scenario name *EG/* is known as the *postfixdir*, while the generic path to all scenario sub-directories is the *prefixdir*. On the computer in which this example is being developed the *prefixdir* is, in fact, *pathtopath(getDBdir(),"A_CodeUse/aMSEUse/scenarios/")* (note that the R / sub-divider can, as usual, be replaced by the double backslash, \\, if that is preferred; the *getDBdir()* function finds the path to the DropBox directory, to use a different *prefixdir* then define it however you wish). The *pathtopath()* function combines the character strings irrespective of what sub-divider is used or what each component starts with or ends with; In each case the sub-dividers should not be mixed as in / with \\.

Graphically, this can all be represented, as an example, so that the *prefixdir* would be:

..DropBox/A_CodeR/aMSEGuide/runs

and sub-directories below the 'runs' sub-directory might be:

- runs/EG
- runs/EGM12
- runs/EGM3
- runs/EGMall
- ..

which indicates four sub-directories where EG is the example basecase where none of the meta-rules are used. The meta-rules are used to modify the outcome of the analysis of the three fishery performance measures to more closely match the desired HS objectives. In the EGHS there are currently three meta-rules, each described later when comaprisons between HS will be made. The EGM12 is where only meta-rules 1 and 2 are used, similarly EGM3 and EGMall are where meta-rule 3 only is used and all meta-rules are used, respectively. In all cases natural mortality = 0.15, steepness = 0.7, and the hyperstability lambda = 0.75 (see the chapters on conditioning the operating model to make sense of those parameters. These four scenarios can then be compared to determine which had the most effects that best matched the desired objectives of the harvest strategy. When dealing with Windows machines a useful shortcut is to realize the classical "~" used by Unix and Linux systems, refers to the user's "Documents" directory. Thus, on my own computer, in R, *path.expand("~")* gives rise to "C:/Users/Malco/Documents" (partly because Windows 11 is pathetic and its default user directory cannot spell Malcolm). Of course, this being Microsoft, one cannot rename either the default 'user' or the 'Documents' directories but it can be used if required.

### 3.2.2 A Possible Workflow

Once the R packages are installed then a potential workflow might begin with these nine steps (see the R code chunks below in Section 3.3 for code details):

1. For each scenario to be considered, select or create a directory somewhere in your own system that will become the *rundir* within which the control file and the data files are placed (and all the results will be written as separate files). The directory path pointing to the different scenarios, each in their own *rundir*, is termed the *prefixdir*, which in the example to follow will be *C:/Users/Malco/Dropbox/A_CodeR/aMSEGuide/runs/*, but obviously a user will need to set up their own directory structure. One then identifies a *postfixdir*, which is

appended to the *prefixdir* to form each *rundir*, so, initially, here we will have a *postfixdir = EG/*. If one then uses the function *confirmdir()*, it either confirms that the *rundir* exists or it asks whether you would want to create it, after which it creates that sub-directory ready for your work. Again, read the function's help for more options, and/or use *?confirmdir*. If the 'ask' argument is set FALSE then it will just create a missing directory without asking, so check the spelling of your *postfixdir* first if you use that *confirmdir()* option.

In the text below (section 3.3) there are example R-code blocks which can also be found in the associated file. You will find such a directory, called *EG*, in the *Dropbox/National_abalone_MSE/aMSE_files/scenarios* directory. You could copy that to the location of your choice. In it you will also see a file called *run_aMSE.R*, which contains the code as described across the code blocks below. Once the user has edited the directory names, they could use that to run the *MSE* when the time comes. This file can be kept anywhere convenient (I tend to keep it in the 'runs' or *prefixdir* sub-directory and modify which subdirectory it points to for different scenarios.

2. Generate a draft control file for a scenario using the **aMSE** function *ctrlfiletemplate()* (the default control file name is *controlEG.csv*). As it stands this sets up a system of 8 *SAU* with *56* populations distributed among them (as in the Tasmanian western fishery *MSE*). In the *EG* subdirectory (*rundir*) this is called *controlEG.csv* (= Base Case of natural mortality = 0.15, steepness = 0.7, lambda = 0.75, with 8 sau with a total of 56 populations).

3. If necessary, edit the .csv file created by *ctrlfileTemplate()* to match the conditioning data available for the fishery being simulation tested (here we will leave it as-is but further details of each component in the control file are given in the *The Input Files and Conditioning the MSE* chapters).

4. Generate a draft data file for the number of stock assessment areas (termed *sau* in the R code) to be simulated using the **aMSE** function *datafileTemplate()*, being sure that the data file name matches that pointed to in the control file (see the code blocks below, but the default data file name is *saudataEG.csv*, as already described in the default control file).

5. If necessary, edit the .csv file created by *datafileTemplate()* to match the conditioning data available for the fishery being simulation tested (here, again, we will leave it as-is but, once again, further details are given in the *The Input Files and Conditioning_the_MSE* chapters). Descriptions of each entry in the control and data files are given in the *The_Input_Files* documentation.

6. A second data file of size-composition data is required if the user wants to compare the predicted size-composition of catches against those observed during the operating model conditioning. Here, such a file is generated using one of the internal data-sets to produce a file called *lf_WZ90-20.csv*, implying length frequency data from the western zone for years 1990 to 2020, with some missing years (again this file can be found in *EG* in the *Dropox/National_abalone_MSE/aMSE_files/scenarios* directory), but the code used to generate it is given below.

7. Each jurisdiction in Australia currently has very different harvest strategies with very different requirements. To allow for these differences a list of HS related properties is input to the software so that each HS can operate appropriately. This list was named

*hsargs* (obviously short for harvest strategy arguments). Each harvest strategy needs to have detailed documentation regarding what arguments are required. An example for Tasmania is given in the code blocks below. See also the Appendix in this *aMSEGuide* (Haddon, 2024a).

8. From this point there is a choice of running either the *do_condition()* or the *do_MSE()* functions. The first reads in the control and two data files, generates the equilibrium, unfished simulated zone and then conditions the operating model on any available fishery data (catches, indices of abundance, size-composition data). It then tabulates and plots up the conditioned result to enable the success of conditioning on a real fishery to be determined This is used when adjusting the conditioning of the model to a fishery. The *do_MSE()* function does all that the *do_condition()* function does, but then also conducts the projections under control of the harvest strategy whose definition is contained in the **EGHS** R package (of whichever harvest strategy is being used). Before running *do_MSE()* for the first time for a new fishery it is best to run *do_condition()* repeatedly so that the conditioning can be adjusted prior to making more formal MSE scenario runs for later comparisons. Here we could forego that step as the template files reflect a pre-conditioned base-case scenario for the western zone blacklip fishery in Tasmania.

9. Finally, one uses the *makeoutput()* function, which plots and tabulates the results, and produces the HTML and CSS files, into the defined *rundir* and has the option of opening the internal web-page ready for inspection. The R objects output by *do_MSE()* can be very large (~1.8Gb for 250 replicates if the *includeNAS* (include numbers-at-size) argument is set = TRUE) so it is best to save that to a fast hard drive which is not synced to a cloud somewhere (ie NOT DropBox, and absolutely not to a shared DropBox folder, or at least not one shared with me). However, a run of 250 replicates using the Tasmanian HS currently takes about 2.9 minutes (on an Asus Zenbook S with an Ultra 7 processor) so repeating scenarios is not overly onerous. Keep in mind that comparisons are done with saved results.

As will be seen in the code chunks below, this whole process sounds more complex than it is in practice.

## 3.3 The Workflow in Practice

### 3.3.1 The Setup

As usual, if a user is unfamiliar with a function then use *?function-name* to get help on what it does and what its arguments are. Alternatively, just type *function-name* in the console (with no following brackets) and the function's code will be printed to the console for inspection or try *args(function_name)* for just a listing of the arguments.

Diagrammatically, the process of running the aMSE software can be summarized as in Figure 3.2, although a verbal description and example R code are also given below.

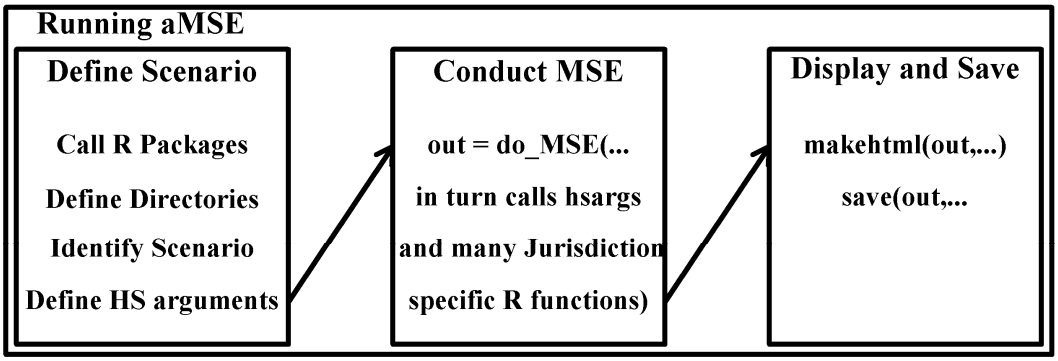


*Figure 3.2: Diagram of the sequence of actions or steps required to run the aMSE software for a given scenario.*

First one sets up R options (if desired), calls the required libraries, and sets up the directory information:

```
options("show.signif.stars"=FALSE, # some R options I find helpful
        "stringsAsFactors"=FALSE,  # now a default in R4
        "max.print"=50000,
        "width"=240)
suppressPackageStartupMessages({  # declare libraries ------------
  # this is the minimum, of course others can be added if desired
  library(aMSE)
  library(EGHS)     # obviously only if using the Tasmanian HS
  library(codeutils)
  library(hplot)
  library(makehtml)
  library(knitr)
})
# OBVIOUSLY, modify the rundir definition to suit your own setup!!!
prefixdir <- pathtopath(getDBdir(),"A_codeR/aMSEGuide/runs/")
postfixdir <- "EG"     # a descriptive name for the rundir
rundir <- pathtopath(prefixdir,postfixdir) # define rundir
startime <- Sys.time() # to document the time taken
verbose <- TRUE        # send messages to the console about the run progress
controlfile <- paste0("control",postfixdir,".csv") # match control file name
outdir <- "C:/aMSE_scenarios/EG/"   # storage on a non-cloud hard-drive
confirmdir(rundir,ask=FALSE)   # make rundir if it does not exist

c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EG  already exists

confirmdir(outdir,ask=FALSE)   # to be interactive needs ask = TRUE

C:/aMSE_scenarios/EG/  already exists
```

### 3.3.2 Making the control and data files

Now we can generate the control and the two data file using **aMSE** template functions. If these files already exist then, obviously, we do not need to run this code again.

```
controlfile <- "controlEG.csv" # default example file name, change it as needed
# Now make the controlfile. Read the help file using ?ctrlfiletemplate. This
# explains what the devrec argument is all about.
ctrlfiletemplate(indir=rundir,filename=controlfile,devrec=0)
# Within controlfile, the default data file name is 'saudatapostfixdir.csv'. If
# you want to call it something else (maybe aloysius.csv or machynlleth.csv?)
# then edit the seventh line of the controlfile that holds the definition
# of the name of the SAU data file, and change it in this code.
datafiletemplate(indir=rundir,filename="saudataEG.csv")
# The program also needs some length composition data. For this we are going
# to use one of the inbuilt data-sets called 'lfs' (see its help). The default
# filen is already written into the controlfile. Change that as necessary.
data(lfs)
writecompdata(indir=rundir,lfs,filen="lf_WZ90-20.csv")
# dir(rundir)  # listing the contents of rundir can be a useful check
```

I recommend that you take a look at the contents of these newly generated files (either in RStudio or Excel, though Excel is best avoided), which will be found in *EG*, but, for now, do not alter anything, although if you do (perhaps the data file name) be sure to re-save it as a .CSV file and not an .XLSX file. Full details of these files are given in the *The Input Files* section of the documentation. For now, the user should know that the scenario includes a natural mortality = 0.15, a recruitment steepness = 0.7, and a cpue hyper-stability lambda = 0.75. The latter implies that the dynamics are affected by hyper-stable CPUE rather than the more classical assumption of a linear relationship between CPUE and exploitable biomass (the assumption of linearity may be common and classical, but it is incorrect for abalone and lots of other species).

### 3.3.3 The HS Package or JurisdictionHS.R File

Here we are using the **EGHS** R package, and the default *hsargs* (see code chunk below). If an R package has been developed that implements a harvest strategy for a specific jurisdiction, it will need its own set of hsargs. The functions that are used to define the harvest strategy being tested can be read in via a source R file, or if an R package exists, the relevant library(s) loaded.

```
# In Tasmania, the HS is included in a package called EGHS, and this
# contains all the required functions.
# Alternatively use source("pathtoafilecontainingtheHSfunctions.R")
# But, if the EGHS package is used instead of a source file, we still need
# the global object hsargs, which contains the settings used by the the
# Tasmanian HS. For details of hsargs, see the documentation for ?EGHS or
# for whichever HS you are exploring.

hsargs <- list(mult=0.1, #expansion factor for cpue range when calc the targqnt
               wid = 4, # number of years in the grad4 PM
               targqnt = 0.55, # quantile defining the cpue target
               maxtarg = c(150,150,150,150,150,150,150,150), # max cpue Target
               pmwts = c(0.65,0.25,0.1),  # relative weights of PMs
               hcr = c(0.25,0.75,0.8,0.85,0.9,1,1.05,1.1,1.15,1.2), # hcr mults
               hcrm3 = c(0.25,0.75,0.8,0.85,0.9,1,1.1,1.2,1.25,1.3),# mr 3
               startCE = 2000, # used in constant reference period HS
               endCE = 2019,   # used in constant reference period HS
               metRunder = 0,  # should the metarules be used. 0 = No
               metRover = 0,   # use metarules 0 = No
               decrement=1, # use fishery data up to the end of the time series
```

```
                pmwtSwitch = 0, # n years after reaching the targCE to replace
                stablewts = c(0.8, 0.15, 0.05), # pmwts with stablewts mr3
                hcrname="constantrefhcr",     # the name of the HCR used
                printmat=NULL) # something needed for some HS, not TAS
```

If using a *jurisdictionHS.R* source file, then it needs to contain a number of specific functions and constants, even where some may, in fact, do nothing:

```
tasFIS <- function(x) { # currently no FIS data is used in TAS
  return(NULL)          # though this may change
}
```

For full details see the *JurisdictionHS_Requirements* section.

## 3.4 Running do_Conditioning

Once setup with all the required files and packages, one would normally attempt to condition the model by changing input parameters so that it generates simulated dynamics that more closely follow those observed in the real fishery. For example, it is possible to modify recruitment deviates in particular years, which would modify the available exploitable biomass in later years, which, in turn, will change the predicted CPUE and associated size-composition of catches, with the intention of improving the match between the observed and that predicted by the operating model. However, if, when running the *ctrlfiletemplate* function, you used the *devrec=0* argument (as we did), then the recruitment deviates for the western zone, under a scenario of natural mortality M = 0.15, recruitment steepness = 0.75, and a hyperstability parameter lambda = 0.75, have already been optimized using the **sizemod** size-based modelling package to provide the best statistical fit between the observed and predicted CPUE and the observed size-composition of the catch and that predicted by the model (see *Using_sizemod_to_condition_the_SAU*).

Despite having the operating model already conditioned, here we will run the code needed to condition the model on the historical data to illustrate what the outputs will look like. The code in the next block automatically generates a set of web-pages using HTML and CSS to layout and display the results. If you run this code while having Windows Explorer (or whatever equivalent you are using on your machine) open on your *rundir* you should see the generation of an array of files prior to the web-pages being opened automatically in a browser. If you were to set the argument *openfile = FALSE* then the browser would not be activated, but you could open it all by double clicking on the EG.html file within the EG directory. Alternatively, one can enter, into the RStudio console, *browseURL(pathtopath(rundir,"EG.html"))*.

```
prodout <- FALSE  # no estimation of production properties saves time but
# this runs the MSE on a control rule that uses no meta-rules. It will be
# compared with scenarios that include meta-rules in a later example.
out <- do_condition(rundir,controlfile,  # controlfile defined above
                    calcpopC=calcexpectpopC,  # from EGHS
                    verbose = TRUE,
                    doproduct = prodout, # prodout=FALSE, no details re MSY
                    dohistoric=TRUE,
                    matureL = c(70, 200), # length range for maturity plots
                    wtatL = c(80, 200), # lengths for weight-at-length plots
                    mincount=120,  # minimum obs for including length comps
                    uplimH=0.35,   # not used because doproduct=FALSE
                    incH=0.005,
```

```
                  deleteyrs=0,  # all length comp years used
                  prodpops=NULL) # no individual pop producivity plotted
```

```
All required files appear to be present
Files read, now making zone
Time difference of 0.7189279 secs

Conditioning on the Fishery data
```

```
makeoutput(out,rundir,postfixdir,controlfile,hsfile="EGHS Package",
           doproject=prodout,openfile=TRUE,verbose=FALSE)
```

The local web page displaying the output has a number of tabs. The *condition* tab illustrates the match between the observed cpue and that predicted by the operating model, while the *predictedcatchN* tab illustrates the match between the observed size-composition of the catches and the predicted values. It is clear that those *sau* with small sample sizes of size-composition data only have a relatively poor model fits in some years.

Each of these tabs are repeated when running the *do_MSE()* function so more detailed descriptions of each tab will be given in the next section where the *MSE* will be run using the default Tasmanian harvest strategy. Note that the main page is named 'EG', which reflects the fact that the scenario webpage top-lvel tab is named after the *postfixdir*.

## 3.5 Running do_MSE

Once conditioning the model has been completed to the degree desired, one can run projections for a specific management scenario with the simulated abalone *zone's* management being determined by the harvest strategy as implemented in **EGHS** (see the discussion concerning the spectrum of options relating to conditioning operating models and how that related to the specific objectives of the simulation testing being undertaken).

If one keeps an Explorer window open looking at *rundir*, the addition of different files can be watched in real time. The *MSE* can be run using the following code (I suggest you examine the help files for any functions new to you). *do_MSE()* provides notifications to the console about the progress of calculations if 'verbose=TRUE' is set. The *checkhsargs()* function is defined to examine *hsargs* but is only valid for the TAS EHS. Any other jurisdiction would need to write an equivalent function to interact with their own HS and *hsargs* requirements.

```
checkhsargs(hsargs)  # lists chosen options before run, only for TAS
```

```
The hcr being used is:  constantrefhcr
hsargs$startCE and hsargs$endCE are used to define the reference periods
for each sau
```

```
# ?do_MSE provides more detailed descriptions of each function argument
# see EGHS package for help with the HS functions
# do_MSE function can take a few minutes to run so be patient
out <- do_MSE(rundir,controlfile, # already known
              hsargs=hsargs,    # defined as global object
              hcrfun=constantrefhcr,   # the main HS function
              sampleCE=tasCPUE, # processes cpue data
              sampleFIS=tasFIS, # processes FIS data (if any)
              sampleNaS=tasNaS, # processes Numbers-at-Size data
              getdata=tasdata,  # extracts the data from the zoneDP object
```

```
              calcpopC=calcexpectpopC, #distributes catches to populations
              makeouthcr=makeouthcr, # generates updateable HS output object
              fleetdyn=NULL,   # only used in SA and VIC so far
              scoreplot=plotfinalscores, # a function to plot HS scores
              plotmultflags=plotmultandflags, # function to plot multipliers
              interimout="", # save the results after projecting,
              varyrs=7, # years prior to projections for random recdevs
              startyr=48, # in plots of projections what year to start
              verbose=TRUE, # send progress reports to the console
              ndiagprojs=4, # individual trajectories in DiagProj plots
              cutcatchN = 56,  # cut size-at-catch array < class = 112mm
              matureL = c(70, 200), # length range for maturity plots
              wtatL = c(50, 200),   # length range for weight-at-length plots
              mincount = 120,   # minimum sample size in size-composition data
              includeNAS = FALSE, #include the numbers-at-size in saved output?
              depensate=0, #will depensation occur? see do_MSE help for details
              kobeRP=c(0.4,0.2,0.15), # ref points for a kobe-like phase plot
              nasInterval=5,  # year interval to use when plotting pred NaS
              minsizecomp=c(100,135), #min size for pred sizes, min for catches
              uplimH=0.35,incH=0.005, #H range when estimating productivity
              deleteyrs=0)    # all length comp years used

All required files appear to be present
Files read, now making zone
Now estimating population productivity between H  0.005  and  0.35
makeequilzone  25.24291 secs

Conditioning on the Fishery data
dohistoricC  1.205479 secs

Conditioning plots completed  1.575177 secs

Preparing for the projections
Projection preparation completed  10.94988 secs
Doing projections
2021  2022  2023  2024  2025  2026  2027  2028  2029  2030  2031  2032
2033  2034  2035  2036  2037  2038  2039  2040  2041  2042  2043  2044
2045  2046  2047  2048  2049  2050
All projections finished  1.51891 mins
Now generating final plots and tables
Starting the sau related plots

Finished all sau plots  14.65523 secs
Starting size-composition plots

Finished size-composition plots 6.194708 secs

Plotting fishery information

hsargs.txt saved to rundir
plotting HS performance statistics

plotting Population level dynamics

All plots and tables completed  16.06431 secs
```

You could check out the list of files in *rundir* using *dir(rundir)*. If the numbers-at-size are saved (includeNAS=TRUE) the *out* object can be gigabytes rather than megabytes. Best to only use this option when making a final scenario run.

The *do_MSE()* function currently requires 29 input parmeters.The *rundir* and *controlfile* were defined earlier, the remaining 27 arguments are new and are described in the help page for *do_MSE()* (?do_MSE).

A flag can be set (*verbose*) to determine whether to have updates on progress sent to the console as they were above. This can be helpful, especially during development. While you should examine the help for *do_MSE()* some of the more important arguments will be considered here. The function makes great use of R's ability to use function names as arguments to other functions. In that way, while we have standard names for functions within *do_MSE()* one can point each of those to custom functions devised for each jurisdiction's approach to managing its abalone stocks. Just as each jurisdiction will require *hsargs* to have a different set of components, the arguments listed here all require functions to be written for each jurisdiction to match the requirements of each HS.

• hcrfun=constantrefhcr, is the main function driving the HS. It gathers all the inputs and is expected to output everything wanted from the HS. In the case of Tasmania, the minimum would be the *acatch* for each SAU. In fact, many more outputs are output and stored (in outhcr) so that the HS performance can be plotted and monitored.

• sampleCE=tasCPUE, controls how the CPUE data from the simulations are smapled and processed prior to use within the HS (see the R code by typing tasCPUE into the console with no following brackets).

• sampleFIS=tasFIS, similarly this function would process any FIS data is any were available.

• sampleNaS=tasNaS, processes the Numbers-at-Size data from the populations to generate the predicted number-at-size in the catch.

• getdata=tasdata, extracts the data from the zoneDP object, which is used to hold the dynamics of the dynamic zone during projections.

• calcpopC=calcexpectpopC, once the *acatch* (or TACC) is known then this function is used to distribute that catch across the populations within each SAU (needed even where management does not use SAU scale catch allocation.

• makeouthcr=makeouthcr, this is used in the function *doprojections* where it is input into hcrfun (here that is constantrefhcr), which outputs an updated version. A the end of the projections it will therefore contain all the outputs from hcrfun ready for characterization.

• fleetdyn=NULL, where the default fleet dynamics is not used (which predicts the catch from each SAU based on available exploitable biomass plus noise) this is where a custom function can be input (this is needed in South Australia)

• scoreplot=plotfinalscores, a jurisdiction specific function to plot HS scores

• plotmultflags=plotmultandflags, a function to plot eh catch multipliers as implemented in each jurisdiction.

When you run this *do_MSE()* code, in fact it will be doing rather a lot of work, even though in this instance it is only running 100 replicates (more normally do 250 or 500). Apart from the results contained in the multiple .png files (plots), .csv files (tables), and some other .csv files, all the results can be found in the *out* object associated with the *do_MSE()* statement.

```
str1(out)  # function from codeutils = str(out,max.levels=1)

List of 29
 $ tottime     : 'difftime' num 2.572
 $ runtime     : POSIXct[1:1], format: "2025-04-17 16:26:03"
 $ starttime   : POSIXct[1:1], format: "2025-04-17 16:23:29"
 $ glb         :List of 20
 $ ctrl        :List of 13
 $ zoneCP      :List of 56
 $ zoneD       :List of 14
 $ zoneDD      :List of 14
 $ zoneDP      :List of 14
 $ NAS         : NULL
 $ projC       :List of 5
 $ condC       :List of 15
 $ sauout      :List of 10
 $ outzone     :List of 13
 $ production  : num [1:71, 1:6, 1:56] 256 245 235 225 216 ...
 $ condout     :List of 2
 $ HSstats     :List of 2
 $ saudat      : num [1:32, 1:8] 21.5 0.3 130 1 54.3 ...
 $ constants   : num [1:33, 1:56] 6 21.5 0.3 130 1 ...
 $ hsargs      :List of 16
 $ sauprod     :List of 3
 $ zonesummary :List of 2
 $ kobedata    : num [1:8, 1:4] 0.33 0.449 0.259 0.242 0.313 ...
 $ outhcr      :List of 8
 $ scoremed    : num [1:30, 1:7] 42.8 35.4 31.5 31.3 33.4 ...
 $ popmedcatch :List of 8
 $ popmedcpue  :List of 8
 $ popmeddepleB:List of 8
 $ pops        : num [1:56, 1:26] 1 2 3 4 5 6 7 8 9 10 ...
```

Many of these objects are lists and so one might use `str()` or `str2()` to examine their structure and contents, but this has been done for you in the *R_Object_Structure_of_Output* section of the documentation (though you could/should still try it yourself). The many plots and tables produced remain only a start. Any number of further plots and tables can be produced from what is already there but especially when alternative scenarios are compared. So, knowing the object structure of what comes out is extremely helpful.

Most of the analyses we can see are conducted at the primary grouping level (i.e. the *sau* level for **EGHS**). Thus, the main objects for initial study would be the *sauout*, *NAS*, *outzone*, and even the *zoneDP* objects (population level dynamics). See the *R_Object_Structure_of_Output* section for more details.

You should run the *dir(rundir)* to see how many files have been created by the MSE ready for use.

We can now construct the internal web-page to display results. If you set the argument 'openfile=TRUE' then the internal website will open automatically. If set to FALSE then you would need to open the *rundir* and double-click the scenario *html* file. For example, if your scenario is called **EG** then you would click on **EG.html**, alternatively, one can enter, into the RStudio console, *browseURL(paste0(rundir,".html"))*, which will automatically use the latest analysis conducted into *rundir*.

```
makeoutput(out,rundir,postfixdir,controlfile,
           hsfile="EGHS Package",openfile=TRUE,verbose=FALSE)
```

Obviously, this is a multi-faceted summary of what has been done. It includes the pages from *do_condition* plus many more.

### 3.5.1 Save Outputs for later Comparison

The primary intent of the *aMSE* software is to make comparisons between alternative harvest strategy scenarios. For this it is necessary to save the *out* objects (which, without the NAS object are hundreds of mega-bytes and with the NAS object can be giga-bytes). Because of their size it is best to store these on a local and fast hard drive so that when making comparisons the separate *out* object for each scenario can be examined and the outcomes compared. Here we illustrate the required R code and we will use the results later in the documentation when comparing the basecase with alternatives.

```
save(out,file=pathtopath(outdir,paste0(postfixdir,".RData")))
```

### 3.5.2 Home Page

The *Home* tab contains lines of information concerning the particular scenario run of the *MSE*. This information includes the directories used, the files used, some details of the run (number of replicates, years projected, number of *populations* and *sau*) but especially the randomseed used when generating the different *populations* within the *sau*. If that randomseed, which is set in the controlfile, is altered one would expect the complete conditioning to be different as many of the properties of each population would change slightly. What this number does is ensure that the same simulated zone is generated each time a scenario is run.

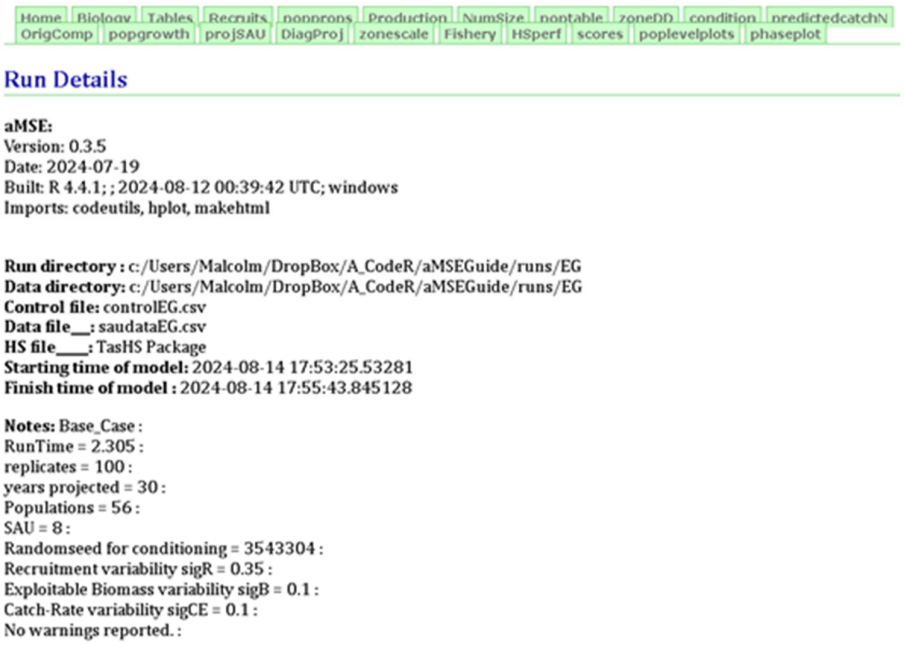


*Figure 3.3: The home page of the internal web-site generated from the results of the do_MSE R function from aMSE. Of course, your own will differ from this in the details but all the tabs should be there.*

In the following pages are a wide variety of results and outputs but, of course, any number of extra, additional, or alternative plots and tables could be included on each page (although that would entail updating the *do_MSE()* function, or functions it refers to). Each figure and table caption contains the name of the .png or csv file from which it is produced should the figure or table be needed for any other purpose. Clicking on any figure will enlarge it for ease of examination (use the return arrow at top left in the browser to return to the original page). In the sections below describing each tab a figure is given illustrating at least one plot from each tab. See the respective tab for the related caption.

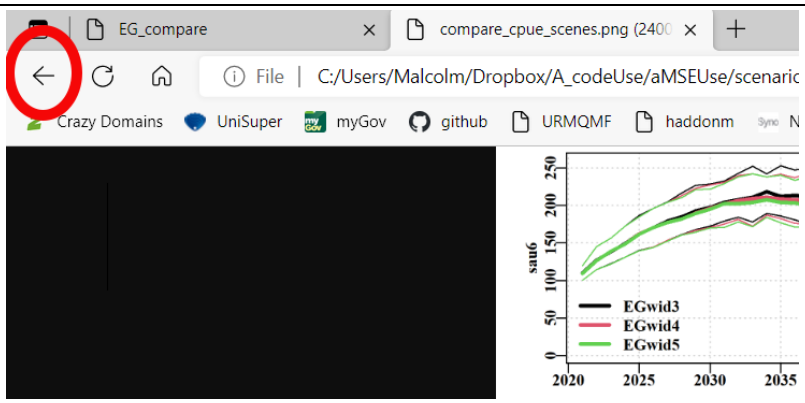


*Figure 3.4: A view of a browser page where a particular figure has been enlarged by clicking on it (here most of it is obscured to the right). Use the highlighted arrow to return to the ordinary scale view.*

### 3.5.3 Biology Page

This contains plots of the maturity-at-length curves, weight-at-size curves, and emergence-at-size curves for each population in each SAU. For each metric, there is a pane of plots for each SAU, each of which contains indicators for every population in that SAU

The final eight histograms of biological properties illustrate the variation among population's for their values of natural mortality, *M*, the *MaxDL*, *L95* and *L50* growth parameters across all SAU and populations. There are also histograms of each population's MSY, steepness, and the biological LML (size-at-maturity plus two years' growth. Finally, *AvRec*, the unfished recruitment, are also variable, but this reflects the fact that in each *sau* only a proportion of the total *sau AvRec* is allocated to each population. There is almost no random variation in the *AvRec* values between R runs as the standard deviation is set to 1e-05 in each case (because the *AvRec* value is fitted within **sizemod**.

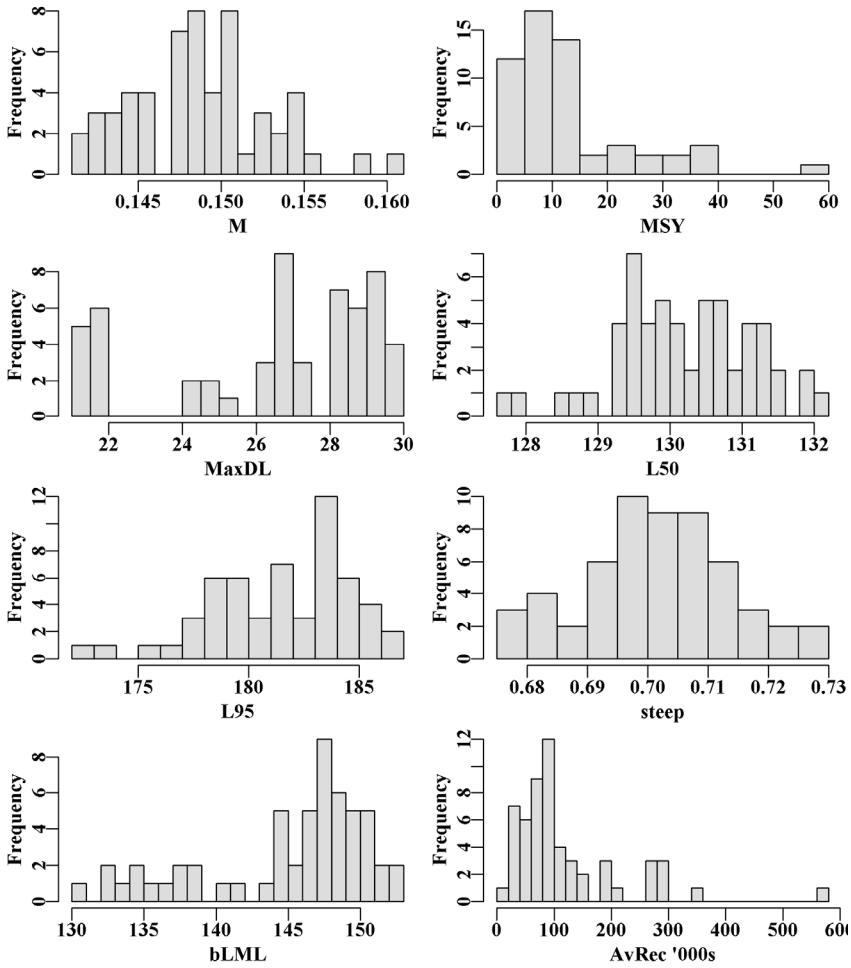


*Figure 3.5: The distribution of various biological properties across the 56 populations.*

### 3.5.4 Tables Page

Currently, this holds only two tables, the first being the productivity properties of each *sau* (*B0*, *MSY*, etc), and the second being the tabulated contents of *zonebiology.csv*. This

represents the various actual values for an array of biologically important variables (including those plotted as histograms on the Biology page), all by *population*. The input constants for population biological properties are derived from adding random variation to the SAU scale property values.

*Table 3.1: SAU properties relating to recruitment and productivity.*

| | sau6 | sau7 | sau8 | sau9 | sau10 | sau11 | sau12 | sau13 |
|---|---|---|---|---|---|---|---|---|
| R0 | 258881 | 461928 | 239640 | 1058383 | 836448 | 1720648 | 1546067 | 583240 |
| B0 | 438.882 | 872.171 | 524.479 | 2438.878 | 2071.964 | 4152.789 | 3126.261 | 915.911 |
| ExB0 | 414.822 | 845.227 | 522.023 | 2450.284 | 2117.053 | 4164.472 | 3203.565 | 869.511 |
| MSY | 20.283 | 43.738 | 24.955 | 122.512 | 102.811 | 207.707 | 156.662 | 41.758 |
| steep | 0.691 | 0.705 | 0.7 | 0.698 | 0.707 | 0.702 | 0.705 | 0.695 |

### 3.5.5 Recruits Page

This holds two plots. The first is a representation of the stock recruitment relationship for each *sau*, all on one plot. This illustrates how different each *sau* is in terms of productivity. The second plot includes a plot of the stock-recruitment relationships of each population within each *sau*, which illustrates the variation apparent in the defined populations. Each plot has its own y-axis scale, so care is needed with interpretation.

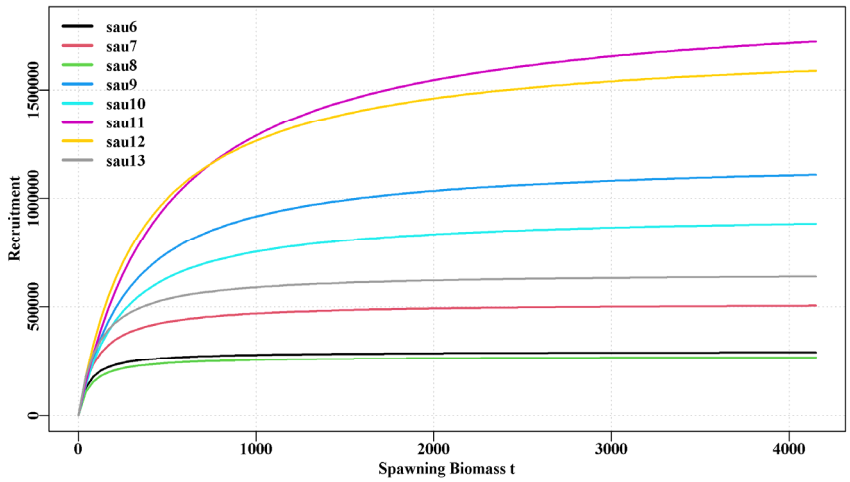


*Figure 3.6: Total stock-recruitment curves for the eight SAU in the example.*

### 3.5.6 popprops Page

Tabulates some of the properties of the individual populations within each sau. Currently this reflects only the proportion of the average recruitment by sau allocated to each population, which repeats what is found in the saudataEG.csv file.

### 3.5.7 Production Page

As the name suggests, this tab provides figures and tables realting to productivity. It has a plot of the production curve relative to expected CPUE for each *sau*, including estimates of the *MSY* and the predicted CPUE at *MSY*. The second plot is of the productivity curves for the whole zone. This more complex plot could be generated for each *sau* as desired using the *production* array (try *str1(out$production)*).

Finally, there is a small 5 x number-of-sau matrix of $B0$, $B_{MSY}$, $MSY$, $D_{MSY}$ (depletion at *MSY*), and $CE_{MSY}$ (predicted cpue at $B_{MSY}$) for each sau. This has extra information to the first table in the *Tables* page.

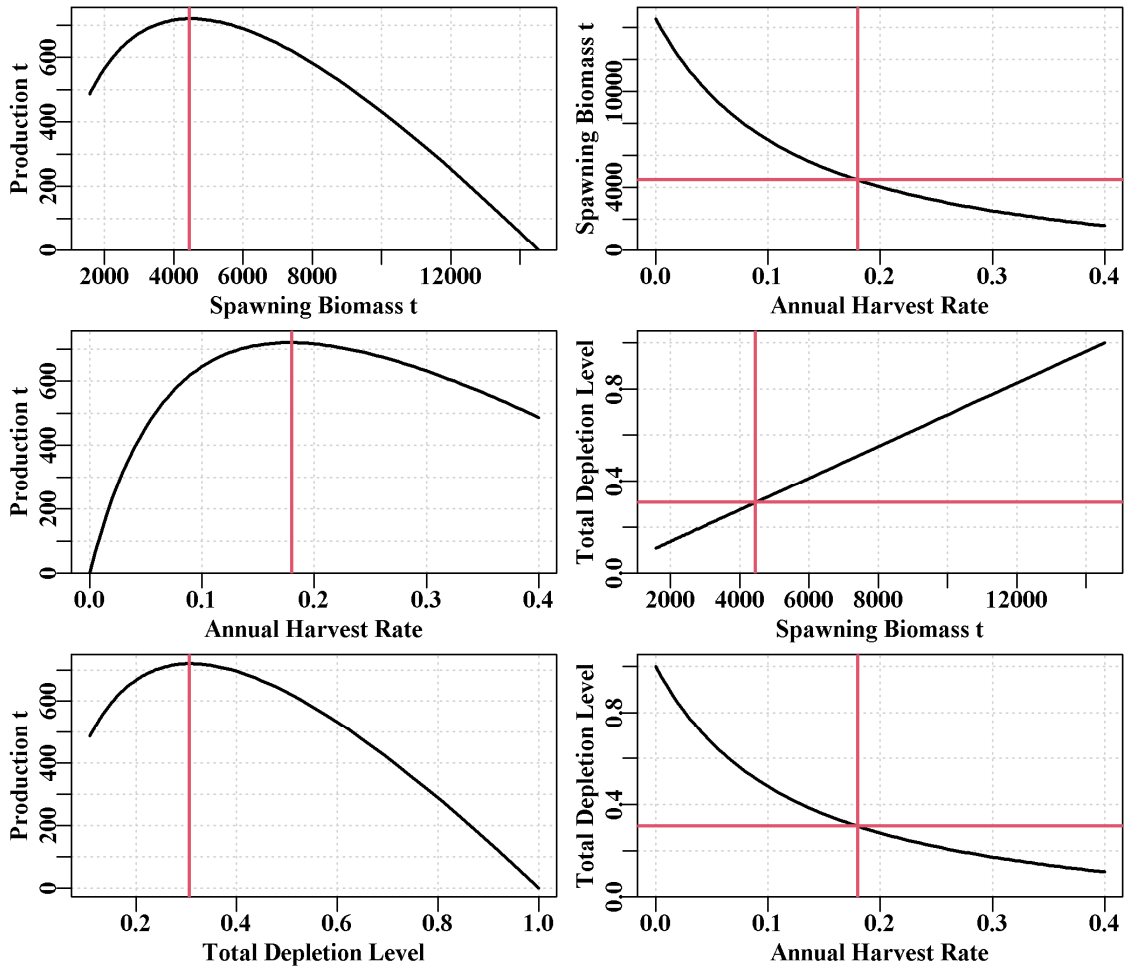


*Figure 3.7: Total stock-recruitment curves for the Zone in the example.*

### 3.5.8 NumSize Page

This remains in need of further development. It includes a single plot containing the expected equilibrium, unfished numbers-at-size for the whole *zone*, and a second plot illustrating the numbers-at-size for each *population.* With 56 populations individual population details cannot be discerned, but it still illustrates the variation in growth, modal progression and

productivity among populations). Following the messy all population plot is a separate plot for each SAU, with snapshots of the size distribution of the stock in 5-year steps across the projection period. Following these is a similar set of 5-year snapshots of the size-distribution of each replicate's catch for each SAU.

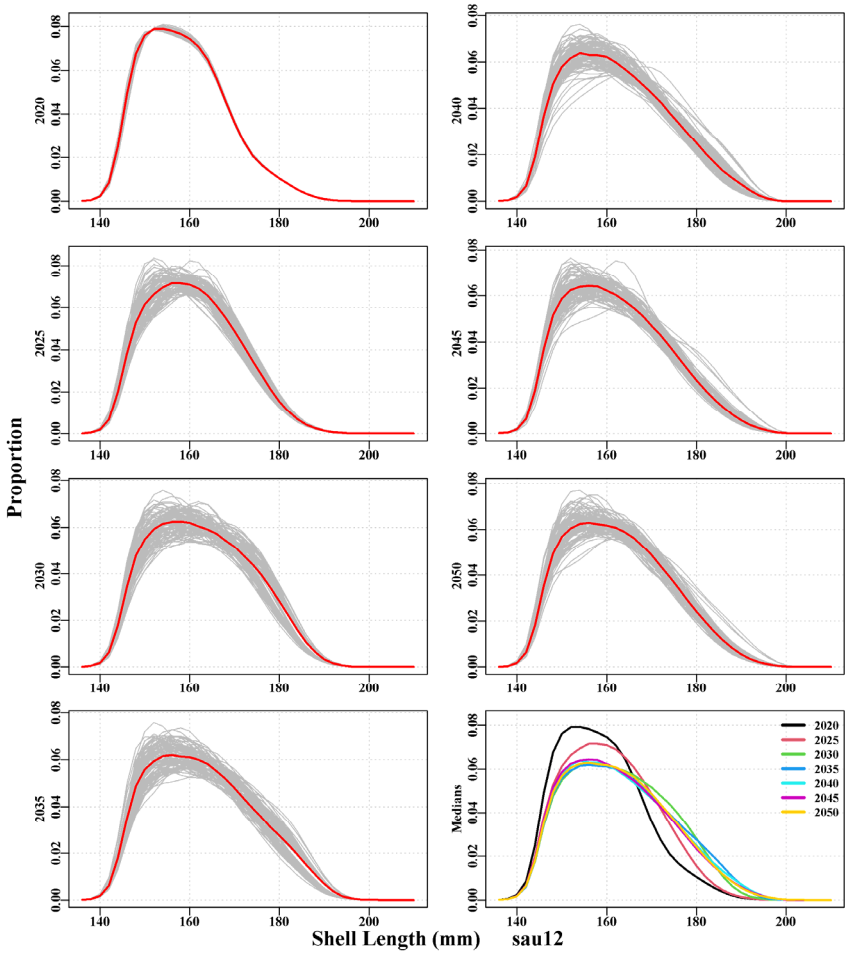


*Figure 3.8: Predicted size-composition of the catch in sau12 from 2020 - 2050 in 5 yearly steps.*

## 3.5.9 poptable Page

This contains the basic biological properties for each of the 56 populations in the example. This is obviously a large table, which is also saved as popprops.csv inside the *out* object. These values are based on the values provided as input in saudata.csv, with added variation within the range specified.

## 3.5.10 zoneDD Page

As with any web page on Windows, pressing 'ctrl +' increases the size of the contents and 'ctrl -' decreases the size. With the figures, each can clicked and they enlarge automatically, at top left is the back arrow that will return one to the broader view (Figure 3.4). Selecting the

*zoneDD* tab leads to a page with a figure made up of histograms of some of the major derived properties of the 56 populations (as in the Biology tab) and then four large tables.

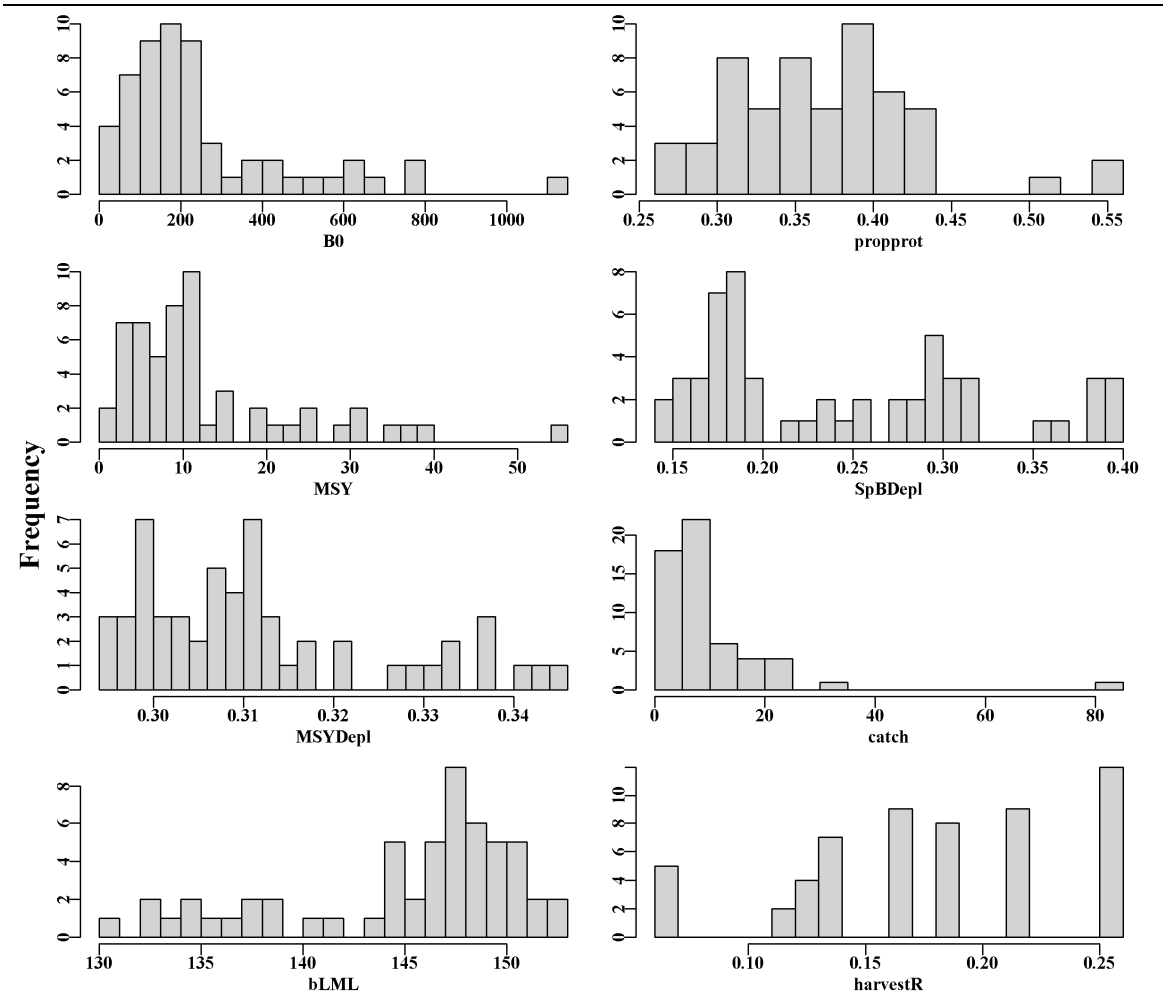


*Figure 3.9: Histograms of eight of the major biological parameters and properties across all 56 populations within the zone.*

The top table on this tab shows the contents of the *propertyDD.csv* file now found in *rundir*, some of which is plotted in Figure 3.9. These are the conditioned emergent properties of each population in the zone. The second is a table of the last ten years of harvest rates during the conditioning for each population from the *final_harvestR.csv* file. Very high values in any *SAU* or population indicate a potentially poor fit. In this case there are some values in the first three populations (sau6) above 0.4 in years V2, V4 and V5, but *sau6* to *sau8* are difficult *sau* to fit a model to. The third is the table of *saudat.csv*, which contains the data read in from the *saudataEG.csv* file for reference. Finally, the fourth table, *popdefs.csv*, contains the specific parameter values produced for various properties of each *population*.

### 3.5.11 condition Page

With the 8 *SAU* in the Tasmanian conditioning, this tab contains 12 plots and two tables. The first plot compares the observed CPUE for each *SAU*, from the historical time series input in the control file, with those predicted by the operating model after biological conditioning and fitting the data using the R package **sizemod**. If you click on the plot in the web-page with your mouse, it will expand to become easier to read the details. Return to the main page using the left arrow (back) at the top of your browser **(Figure 3.4)**. The match between the observed and predicted values, **(Figure 3.10)** appear very good, although sau8 is clearly a complex situation for any model to fit as since about 2012 the dynamics have been influenced by other factors not in the model.

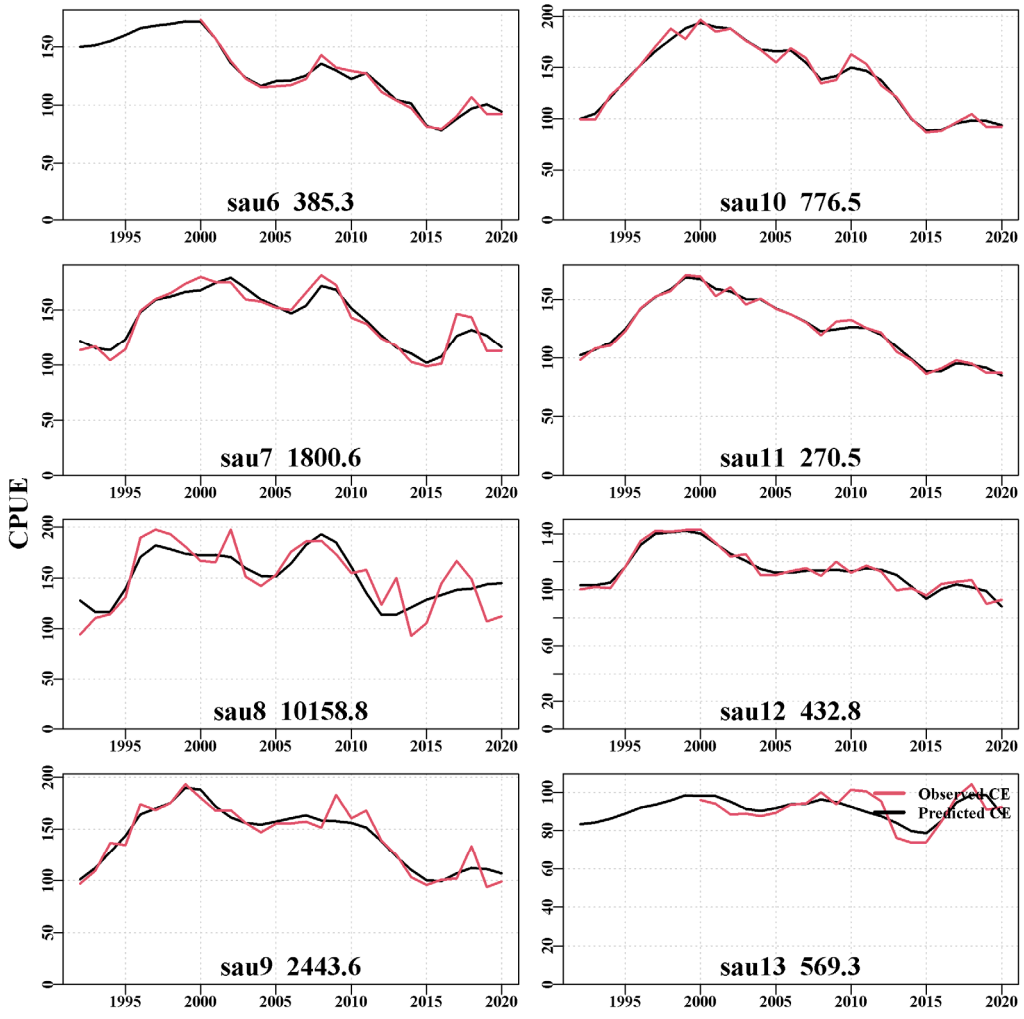


*Figure 3.10: The top plot from the condition page of the internal web-page generated from the results of the do_condition and do_MSE R functions from aMSE. The values beside the SAU names are the simple sum-of-squared differences between the observed and predicted values.*

An example of the individual *SAU* plots is given for *SAU* 10. Below the comparison of CPUE plot there are 8 plots, one for each sau, illustrating the trajectory of the mature biomass

depletion level, the catch through time, the cpue (with the observed values in green), the implied annual harvest rate, the mature biomass (in tonnes), and the implied recruitment level (prior to 1982 and post 2014 are deterministically taken from the stock recruitment curve).

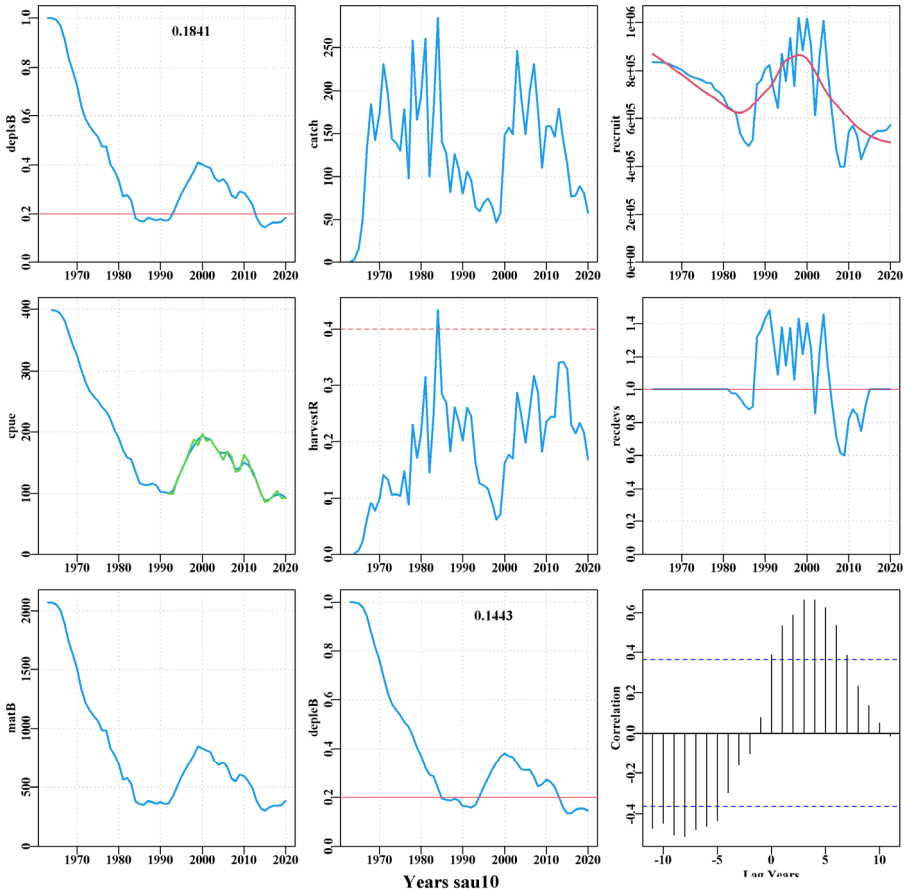


*Figure 3.11: The SAU plot illustrating the dynamics of the combined populations within SAU 10. This, and plots for all other SAU are to be found in the 'condition' tab.*

Below the plots for each *sau* is a summary plot of each recruitment trace used to simplify looking for correlated recruitment events predicted by the model. It is important to remember that the recruitment deviates produced by **sizemod** and reproduced in **aMSE** are estimates and not observed data. Nevertheless, it is possible to see repeating patterns of positive and negative recruitment levels between *sau*. For example, *sau10* to *sau13* all show a decline prior to 2010 followed by a rise, although the exact timing appears to differ by a year or so between some *sau*. *sau6* and *sau13* only have recruitment deviates from 1990 to 2014 because they were only formed in 2000 when zonation was introduced, which limits the data available for conditioning.

The final plot is a histogram of the depletion in the final year of conditioning. This illustrates the effectiveness of including recruitment variation when preparing for the projections (which was set in *do_MSE()* using the *varyrs=7* argument). This figure is then complemented by a table of the quantiles of that final year's depletion level.

### 3.5.12 predictedcatchN

Illustrates the fit to the observed numbers-at-size in the catch obtained during the conditioning. The quality of fit is very much affected by the number of observations involved. These plots are for data at the *sau* scale, so they integrate across however many populations are deemed to make up the *sau*'s dynamics. Given this is the case, the fit in some cases is remarkably good, for example, in *sau11* to *sau13*, the last plots on the page, the fits to the later samples where sample sizes are large are excellent. *sau6* has the least data and generates the worst fit to size-composition data.

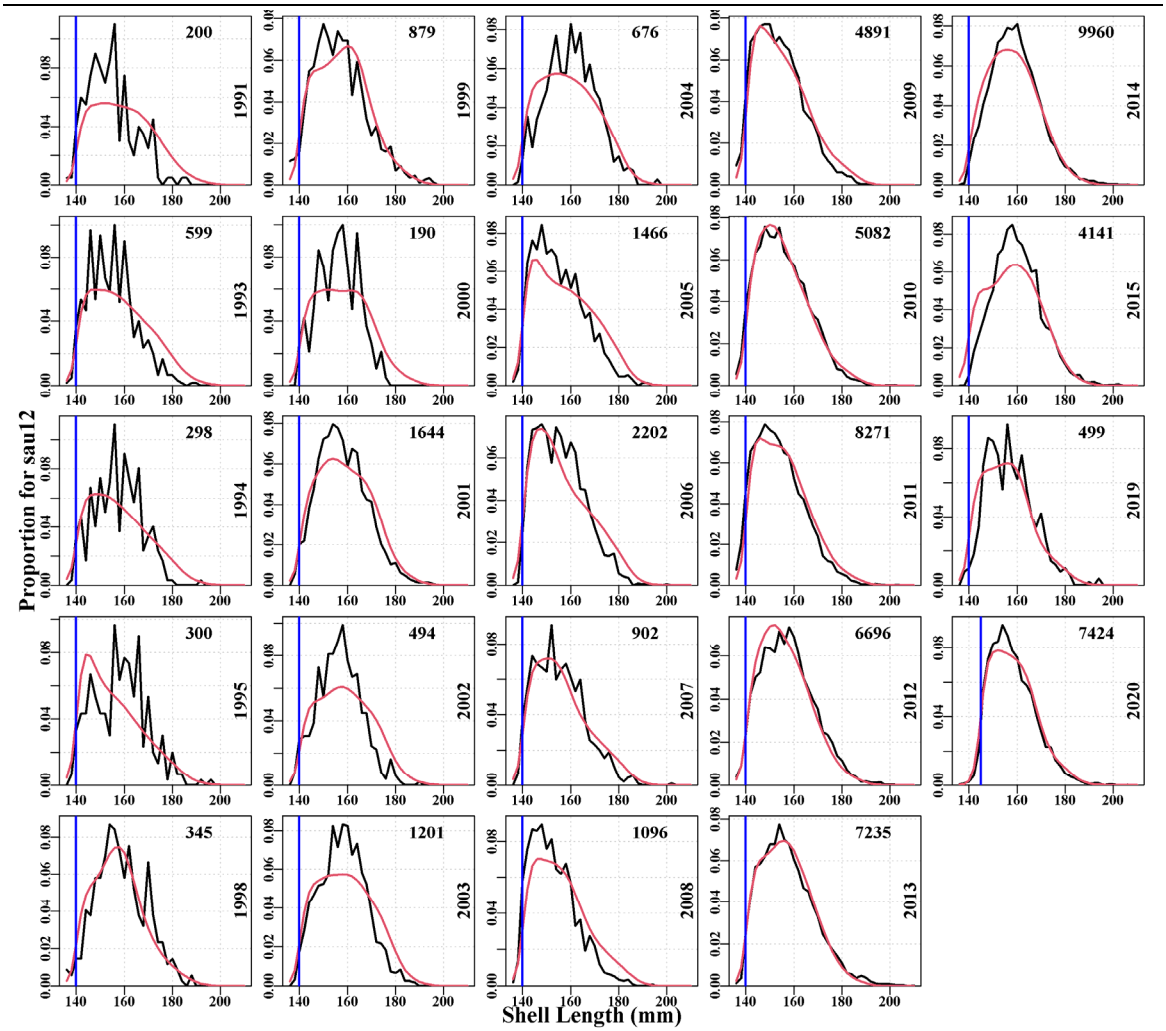


*Figure 3.12: Size-composition of the catch in sau12 with the model fitted size-distribution. black = observed, red = predicted.*

### 3.5.13 OrigComp Page

Contains plots of the original size-composition of catch data. This is primarily for reference but also provides an image that allows for an appreciation of changes through time and of any gaps in the time-series. Understanding one's data is vital when attempting to understand how the dynamics in a stock may be changing.

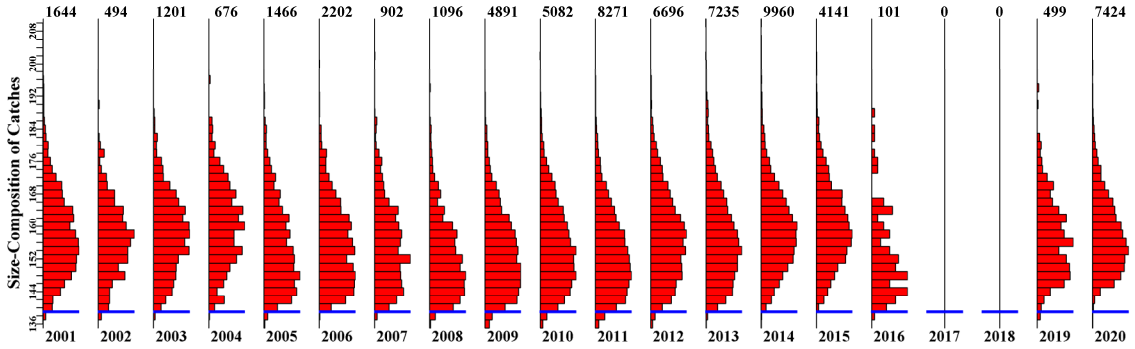


*Figure 3.13: Size-composition of the catch in sau12 for samples taken 2001-2020. Number of observations at the top of each plot with year at the bottom.*

## 3.5.14 popgrowth Page

Has plots of the individual growth curves expressed in each population within each sau. This enables a better appreciation of the variation in growth included within and between sau.

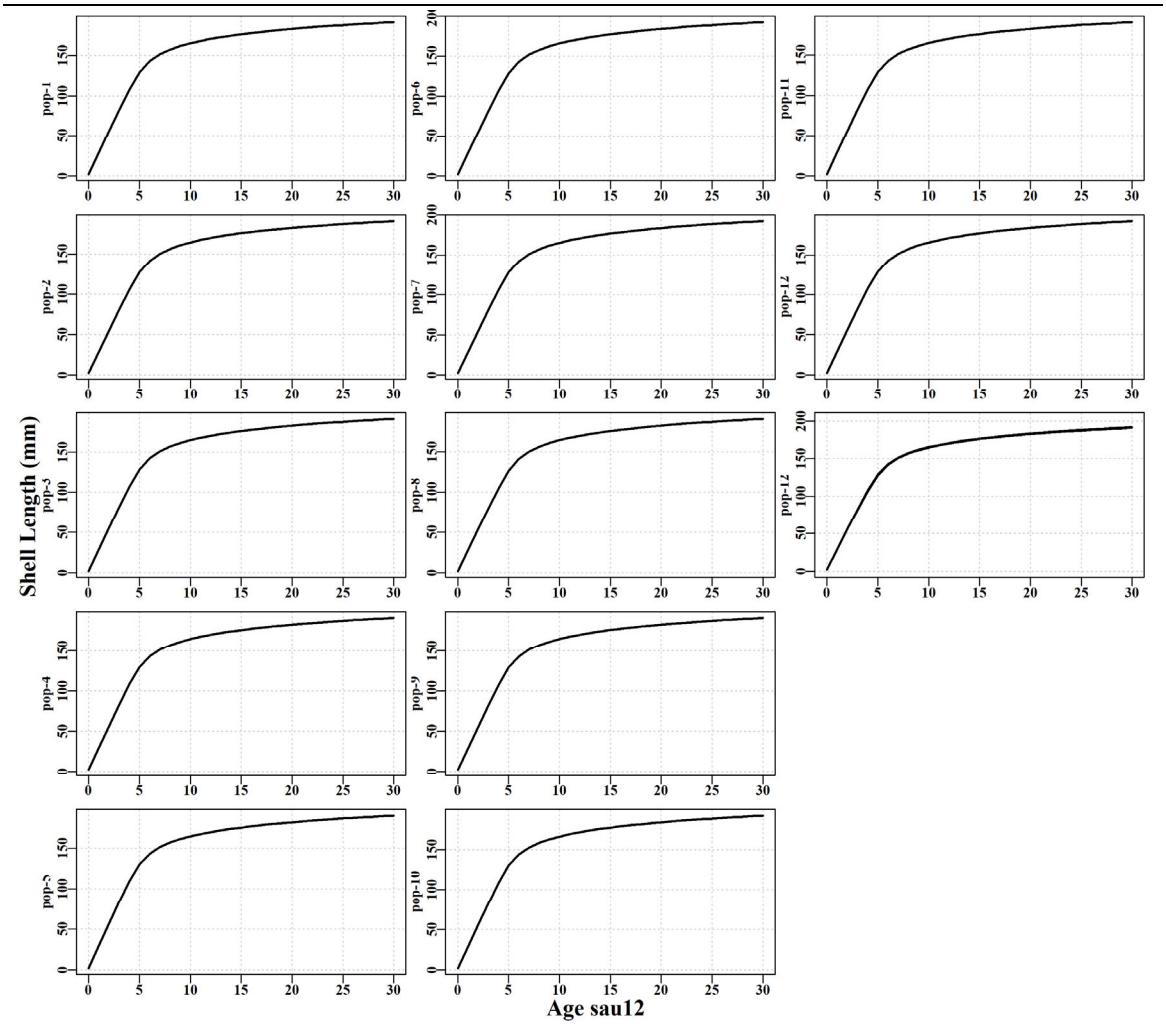


*Figure 3.14: Implied size at age for the 12 'populations' within SAU 12. The curves all appear similar in form but differ in the details. These growth curves will be identical across all scenarios, assuming the same random number seed is used.*

## 3.5.15 projSAU Page

This page illustrates the effect of the harvest strategy on the dynamics of each *sau*. There are seven plots, the first being for the predicted CPUE by *sau*, then the predicted actual catch and aspirational catch (the first, not surprisingly, being more variable than the second because that is one source of variation added in the *MSE*), then comes the mature biomass followed by the exploitable biomass (very similar and currently reflective of the CPUE trends). Finally, comes the predicted recruitment trend and the last plot is of the predicted annual harvest rate.

The grey lines represent the trajectory of individual runs, and illustrate the variation among replicates within each SAU. The each plot also has the median value of replicates (blue line) and the 90th quantile bounds (fine red lines). It is important to appreciate that these are derived values across all replicates, and do not represent an individual run.

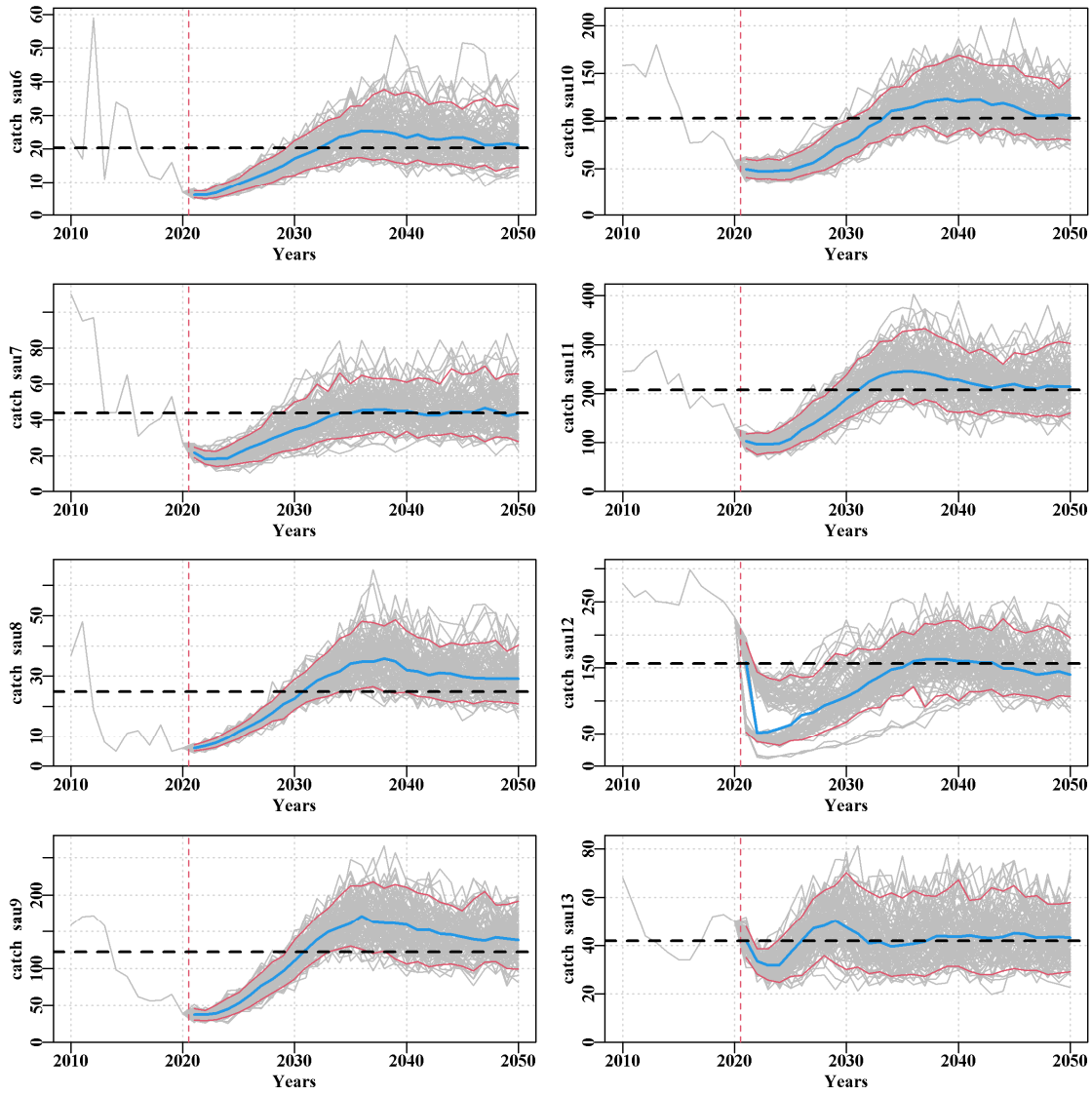


*Figure 3.15: Projected catches for the EG example for each SAU. The dashed line in each case is the estimated MSY.*

## 3.5.16 DiagProj Page

Currently contains three plots designed to characterize some of the variability of the projections and to ensure that the expected dynamics are behaving in a manner akin to the real fishery.

The first plot is of the differences between the actual *SAU* catches and the aspiration *SAU* catches, while useful this plot needs improvement to denote the proportional error in each *SAU*. Within the current Tasmanian HS when actual catches begin to approach 120 percent of the aspirational catch then that *sau* is closed to further commercial fishing. The 120% value has been reduced in recent years, meaning the *withsigB* value needs to be reduced. This acts to limit the variation now possible. This can be used to 'tune' the *withsigB* value that is used to control how precisely the fleet dynamics adheres to the aspirational catches derived from the harvest strategy. If *withsigB* was set extremely small, then the variation included in how

much catch was taken from each *sau* and each *population* would also be small. The *calcpopC()* function, which estimated the aspirational catches (and TAC) is defined in **EGHS** (or whichever harvest strategy is being used). The algorithm underlying the dynamics is described and explained in the documentation to **EGHS** (and hopefully in any other HS as well). The fleet dynamics essentially describes how the divers interact with the harvest strategy and hence needs to be part of the input functions as part of the HS package or R source file.

The second plot is a comparison of a limited number of randomly selected trajectories from the scenario being explored showing the actual catches by *SAU*, as solid lines, and the related aspirational catches as dotted lines. This plot functions to show that the fleet dynamics model is behaving in a realistic manner but also to illustrate whether the expected variation in catches by *sau* are plausibly realistic. In fact, they are far less variable than during the history of the fishery (see the *fishery* tab) but this is a reflection of the meta-rules that control the deviation from the aspirational catches, that were only introduced in 2019. The number of lines plotted is determined by the *ndiagprojs=4* argument within the *do_MSE()* functions argument list. With four lines, this is readable but if this is set too large the plot becomes too busy and loses any value for diagnostics, even when magnified.

The third and final plot is of the same number of individual CPUE trajectories for each *sau*. Again, its purpose is to determine whether the variation in the predicted CPUE trends appear realistic for the fishery concerned.

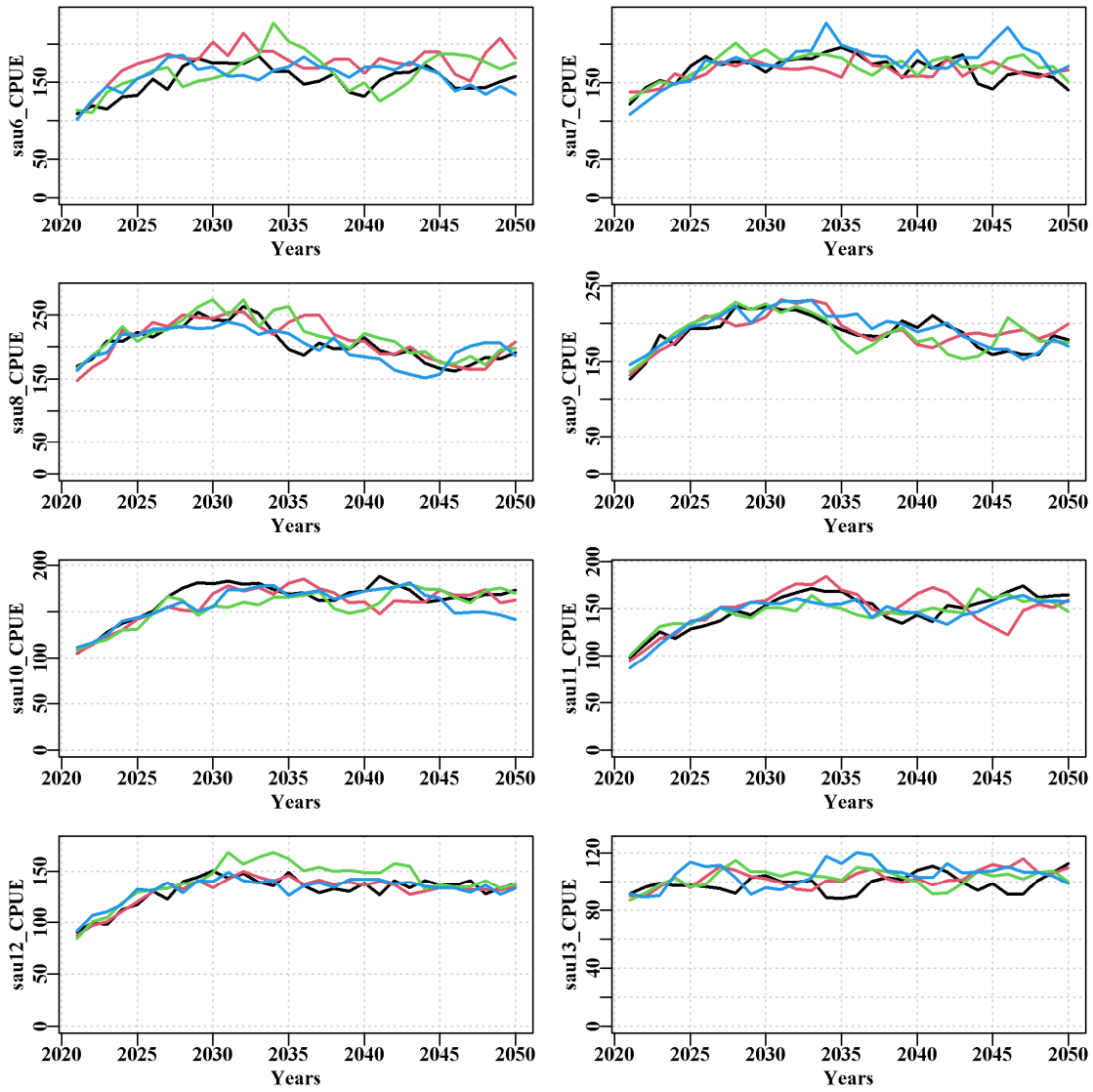


*Figure 3.16: Projected cpue for four randomly selected trajectories in each SAU to determine visually whether they appear realistic.*

## 3.5.17 zonescale Page

Like the projSAU page, this contains seven plots of the same set of plots as seen in the projSAU page, but of *zone* scale dynamics. The projected values for each *population* within each *SAU* have all been combined at the *zone* scale (catch weighted where required, see the help for the function using *?poptozone*). The plots include total catch, the TAC (essentially the same plots in Tasmania), CPUE, and mature biomass depletion level, mature biomass (mirrors the depletion plot), annual harvest rate, and annual recruitment.

Of course, the *zone* scale dynamics obscure the variation observed at the *SAU* level and even more so at the *population* level. The zone scale ultimately, is of principal interest in terms of the consequences of a particular HS scenario for the TACC, while the SAU and population

scale details provide an understanding of the spatial complexity within the zone and the relative reliance on the different SAUs.

The page ends with a table containing the median values of each of the plotted variables.

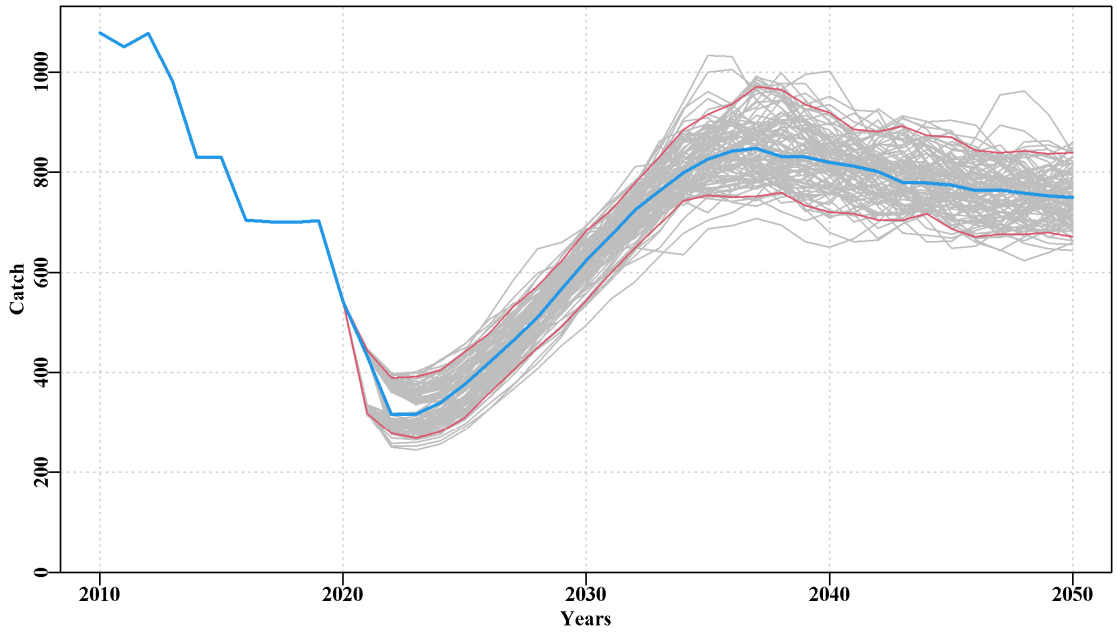


*Figure 3.17: Projected zonal catch for the EG example with the median and inner 90th quantiles.*

### 3.5.18 Fishery Page

This page was intended to illustrate the historical fishery data. It currently contains a plot of the selectivity curves relating to all LML that occurred during the historical conditioning period and through the projection period. Then a plot of the individual catches by *sau*, which illustrates the remarkable inter-annual variation in historical catches in teh Tasmanian western zone blacklip fishery. Some *sau* exhibit halving and doubling of catches in the space of two years. Even the final zone-wide sub-plot exhibits some large changes between years, although this calmed down a good deal at least at the zone scale once quota zones were introduced in 2000.

The final plot is of the cpue observed in each *sau*. Note the truncation of the time-series in sau6 and sau13, which occurs because the introduction of zonation in the year 2000 split these *sau* across different zones.

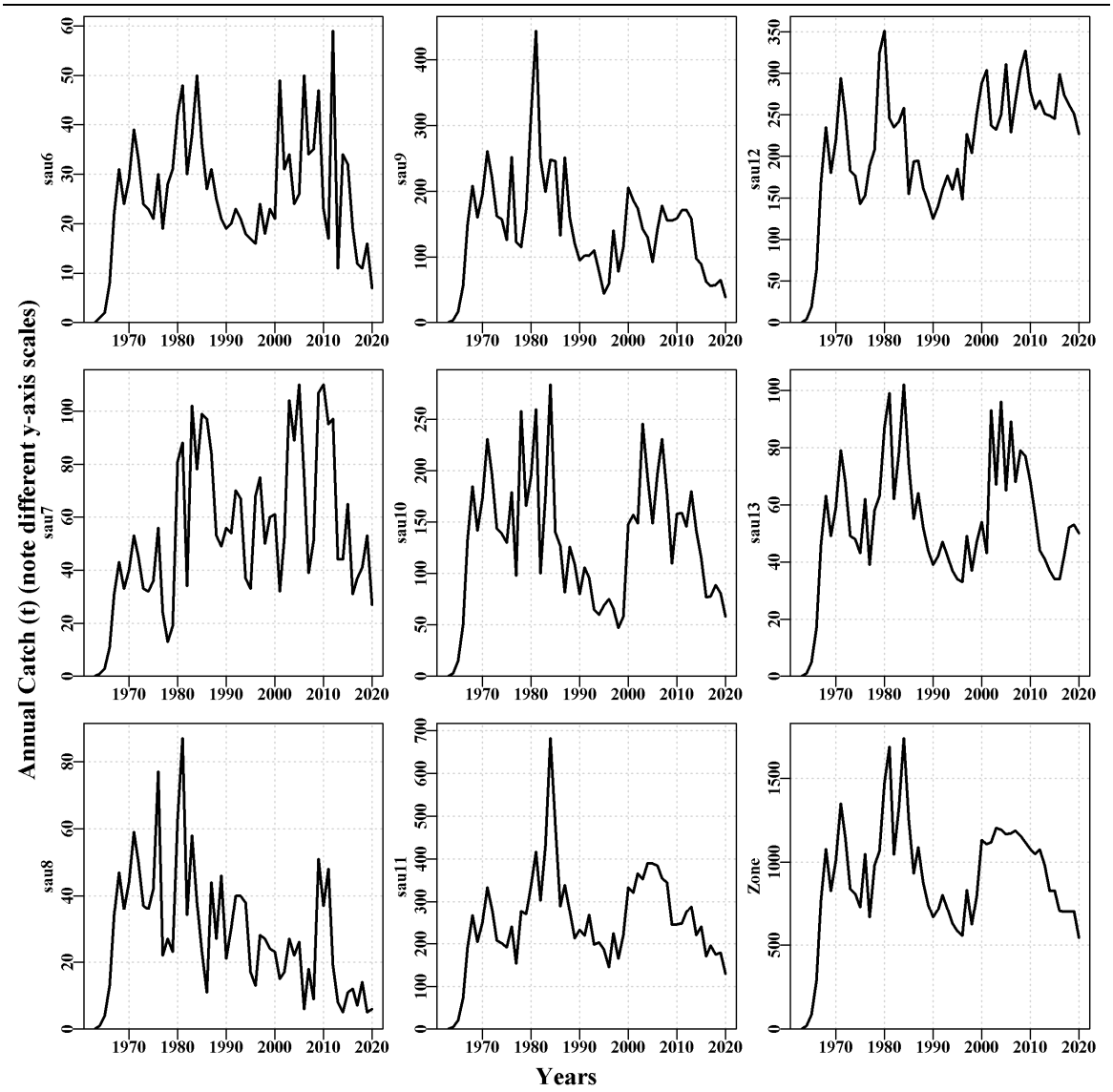


*Figure 3.18: Historical catches for each western sau. Note the sometimes extreme variability between years, which can be highly destructive.*

## 3.5.19 HSperf Page

This page currently holds a listing of the arguments used by the harvest strategy (*hsargs*) and plots of the cumulative sum of total catches by *sau* at 5-years and at 10-years as well as plots of the mean catches at 5 and 10 years, again by *sau*. The *hsargs* listing is simply a double check that the values used are what was wanted.

The bi-modal outcome for sau12 is obvious and appears to relate to some random recruitment events leading to lower biomass and CPUE, and in some cases an *aCatch* reduction of 75% or even two such reductions. The final table is an example from a single replicate of what target cpue is achieved through the projections for each sau. It is the case that sau6 to sau10 invariably breach the 150kg/hr imposed maximum, while sau11 - sau13 all achieve a somewhat lower target, with the more easterly sau having lower targets.

It is expected that these tables and plots will differ for each harvest strategy examined.

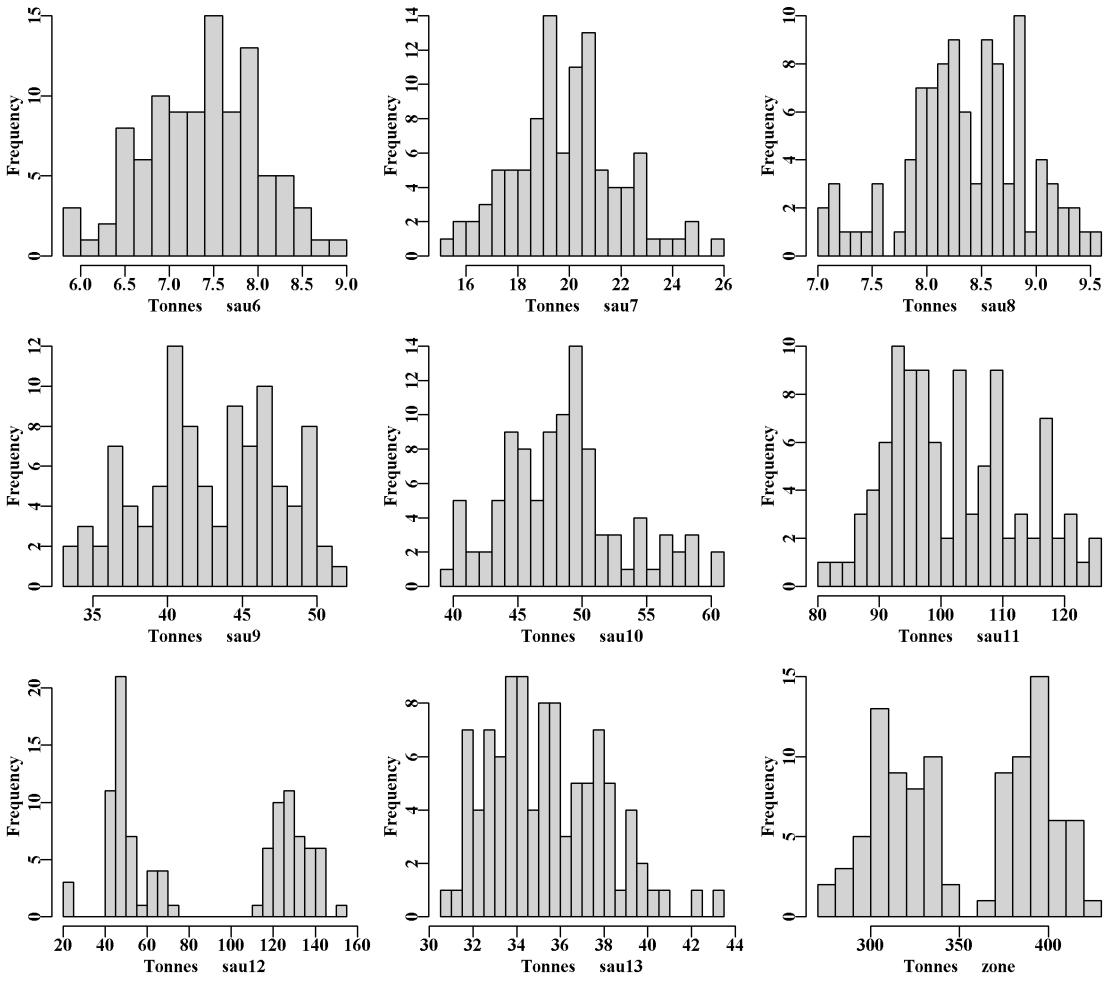


*Figure 3.19: Mean annual projected catch after 5 years in the EG example. SAU 12 has high yields and a bimodal outcome, leading to the zone outcome being bimodal.*

## 3.5.20 scores

The *scores* page illustrates the HS statistics for each *sau*. Each plot depicts the projected catch and cpue, and the scores for the grad1, grad4, and targetCPUE performance measures (PM). Finally, it provides the targetCPUE and the final total score from the harvest control rule.

These plots enable the user to determine which fishery PM contributed most influence on the catches and when.

The red lines, in each case are the median values.

Once again, the illustrated plots relate to the Tasmanian harvest strategy and its variants. Differing plots will be required to be relevant for different jurisdictions. Equally, the interpretation of any resulting patterns will be dependent on the jurisdiction's HS. For

example, in Tasmania, the median final TasHS score is expected to stabilize at 6, which is the index of the multiplier = 1.0 in the *hcr* argument of Tasmania's *hsargs*.

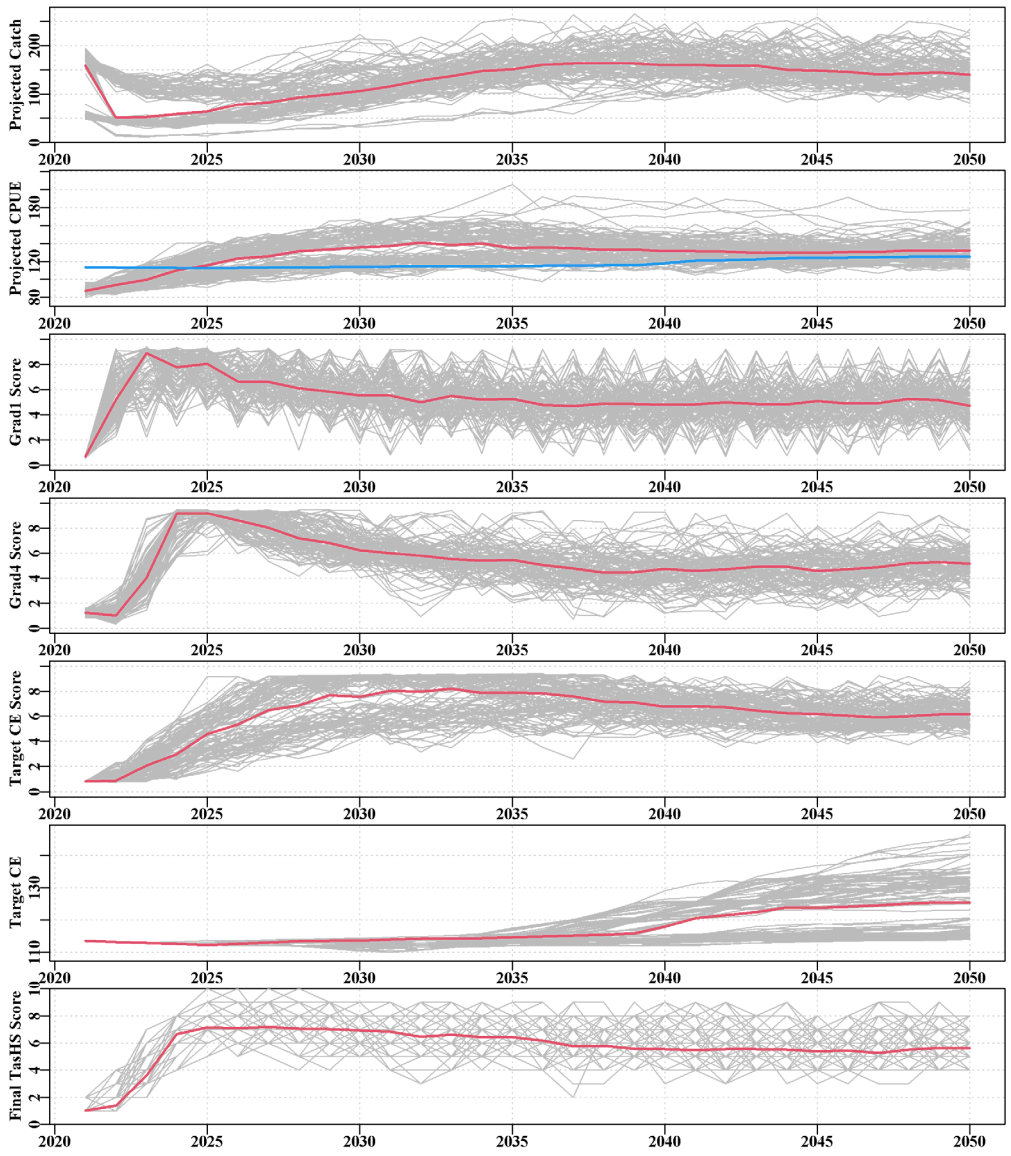


*Figure 3.20: Projected Tasmanian HS scores for the three CPUE performance measures, total score, and reference plots of catch and cpue for example EG for SAU 12.*

## 3.5.21 poplevelplots

Each *sau* has a given number of 'areas of persistent production' or *populations*. This tab on the website contains plots of the replicate trajectories within each such *population* within each *sau*. The median value for each *population* is also plotted as a unique colour. Such *population* level plots illustrate the degree of heterogeneity in productivity between the populations within each *sau*.

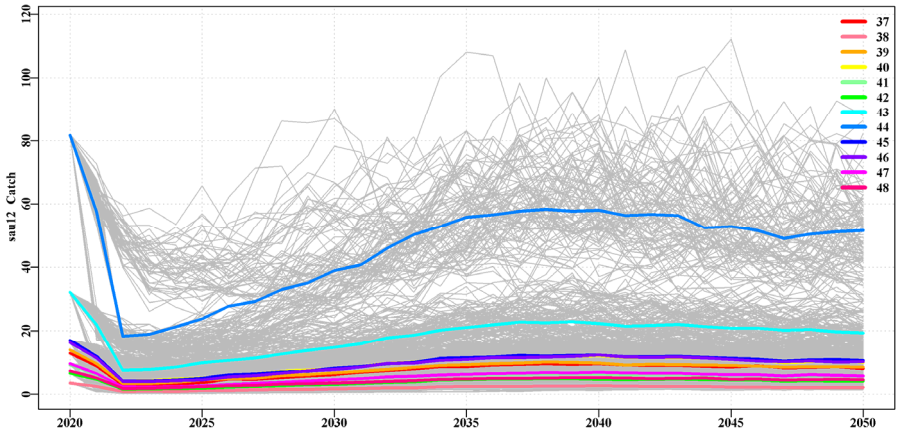


*Figure 3.21: Projected catches in each population within sau12 for example EG.*

## 3.5.22 phaseplot

The phaseplot tab contains eponymous plots for each *sau*. For each *sau* there are two phase plots (or Kobe plots if you will). The first is a more classical plot of the predicted Harvest Rate vs the predicted Mature Biomass Depletion level and contains all the data from the start of the CPUE time-series to the end of the projection (the projected parts use the median, although this may evolve to show some possible bounds). The second plot is of the CPUE gradient-4 score from the **EGHS** vs the CPUE Target score from **EGHS**. The first is to act as a proxy for the fishing mortality or harvest rate, and the CPUE target score acts as a proxy for the mature biomass. The two plots for each *sau* are placed side by side to make comparisons between the theoretical optimum variables and their proxies (which do surprisingly well).

A table is provided containing the final values (2050) for each SAU, of the metrics used in the phaseplots.

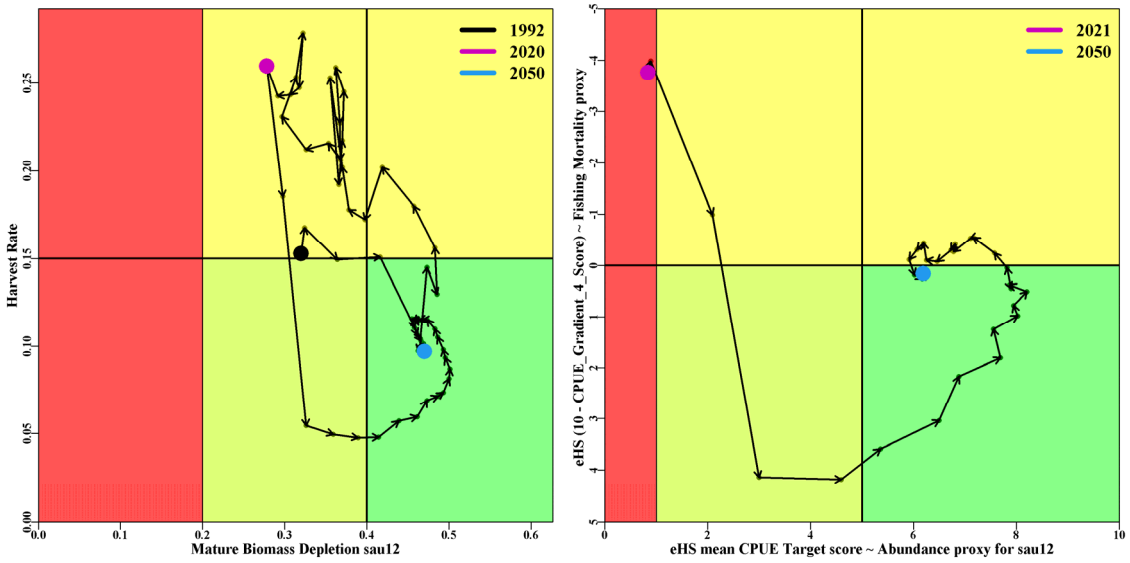


*Figure 3.22: Modelled phase plot and empirical proxy phase plot for sau12 in example EG.*

# 4. Comparing MSE Scenarios

## 4.1 Introduction

In the previous chapter a description was given of how to run the **aMSE** software and produce an array of results. That has many interesting aspects but the underlying intent of the *MSE* software is to test and compare alternative harvest strategies. In general terms formal harvest strategies have three main components: 1) specified data sets, 2) a specified data analysis or assessment relative to limits and targets, and 3) a harvest control rule that produces a specified catch or fishing mortality. In addition to these three components, optionally, it is common, especially with empirical harvest strategies, to include meta-rules that are used to modify the harvest strategy's behaviour under different circumstances. These metarules are designed to change the normal behaviour of the HS, should a particular threshold be reached, or set of conditions apply, such as number of consecutive years the CPUE above the target.

Strictly, if a change is introduced into any of the details of the three main requirements of a harvest strategy, or the meta-rules if present, that would constitute a different harvest strategy. It is therefore very simple to devise multiple alternative harvest control rule or meta-rule variants for comparison. The objective of making such comparisons is to determine which of the alternatives performs the best (best means it achieves the objectives of the HS most effectively). Sometimes, as often in politics, the choice is limited to the least-worst.

Once again, before going into the details, it is useful to have a real-world example in mind to aid making sense of the theoretical descriptions. To that end, in this chapter, we will develop an example that makes a comparison between four versions of a Tasmanian HS (Haddon & Mundy, 2024b) in which a constant set of reference years are used. This will introduce the processes required to set up and then make comparisons between the results of different scenarios, which will, hopefully, give the user/reader a more intuitive grasp of how to use the suite of R packages that revolve around **aMSE**. The Tasmanian HS is under constant exploration for ways to make it more effective, however, an unvarying version has been produced for the examples in this guide: **EGHS**. This contains the *constantrefhcr*, which uses a fixed set of reference years (defined in *hsargs*) and the *consthcr*, a very simple HCR that uses a set of constant aspirational catches for each sau and that ignores any responses of the stock. This is effectively projection with a constant catch and strictly is more a risk assessment than a MSE.

### 4.1.1 The Example

The example will involve changing which meta-rules are included in the example Tasmanian HS found in **EGHS**, that is, we need to change components of *hsargs*. The user should read the documentation for the Tasmanian harvest strategy (at very least read the help page *?EGHS::constantrefhcr* and *?EGHS::EGHS*, but also the latest Tasmanian abalone assessment and Bradshaw, 2018). The three input arguments, within *hsargs*, that are to be changed are the *metRunder*, *metRover*, and *pmwtSwitch* values. A description of each of the components within *hsargs* is provided in the help for the **EGHS** package. After calling *library(EGHS)* then one can type *?EGHS* in the console and the packages help will be listed, including the components making up the *hsargs*.

The Tasmanian HS currently uses three performance measures for each *sau*:

1)the *targCPUE*, which is the current standardized cpue relative to a target CPUE,

2)the *grad1*, which is the gradient of change in cpue over the previous 12 months, relative to a target gradient of zero,

3)the *grad4*, which is the gradient of change in cpue over the past four years, including year-to-date, relative to a target gradient of zero.

The Tasmanian HS is based on Multi-Criteria Decision Analysis principles. A utility function is developed to assign a score from 0 to 10 for each PM, with a score of 5 for the Target value.

There are currently 16 or 17 components in the input *hsargs* (depending on whether two are used to define a reference period or one vector; and this may change as other meta-rules are considered). The three arguments relating to the current three meta-rules can be turned off by setting them all = 0, or turned on by setting the first two to any number > 0, and the third is set to the number of years after the target CPUE has been achieved that CPUE needs to keep increasing before the alternative, more generous, HCR multipliers are used and the weights attributed to the different performance measures are altered to whatever vector is out into the *stablewts* argument. There will be four scenarios compared:

- No meta-rules - named EG from the previous chapter (already saved)
- only meta-rule 1 & 2 activated - named EGMR12
- only meta-rule 3 activated - named EGMR3
- all meta-rules 1 - 3 activated - named EGMRall

## 4.2 Running the Three Extra Scenarios

As in the previous chapter we will first start with a base-case that uses the current Tasmanian HS but it will have the first two meta-rules activated (if required, see the previous chapter for a description of these steps).

```
options("show.signif.stars"=FALSE, # some R options that I find can help
        "max.print"=50000,
        "width"=240)
suppressPackageStartupMessages({  # declare libraries ------------
  # this is the minimum, others can be added if desired
  library(aMSE)
  library(EGHS)     # obviously only if using the Tasmanian HS
  library(codeutils)
  library(hplot)
  library(makehtml)
  library(knitr)
})
dropdir <- getDBdir()
prefixdir <- pathtopath(dropdir,"A_codeR/aMSEGuide/runs/")
# we then use the _copyto()_ function to copy the required files from the
# original example EG from the previous chapter. This is merely to facilitate
# ensuring that the necessary files are put in the correct places.
# _copyto()_ fixes the runlabel and datafile names but if changes are made to
# internal fixed settings, such as to _lambda_ or _M_ then the copied files
# will still require editing. In these examples we will only be altering the
# hsargs metRunder, metRover, and pmwtSwitch values so no other changes are
```

```
# required.
filelist=c("controlEG.csv","lf_WZ90-20.csv","saudataEG.csv")
aMSE::copyto(prefixdir=prefixdir,fromdir="EG",todir="EGMR12",
             filelist=filelist)



controlEG.csv  has been copied to  EGMR12  as  controlEGMR12.csv
lf_WZ90-20.csv  has been copied to  EGMR12  as  lf_WZ90-20.csv
saudataEG.csv  has been copied to  EGMR12  as  saudataEGMR12.csv
Be sure to change the control, data, and run files where necessary

[1] "c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EGMR12"
```

Now we can define the *rundir* for the first scenario where no meta-rules are implemented. In all cases you would, obviously, define your own *rundir* by defining the *prefixdir* and *postfixdir* to suit your own computing environment.

```
startime <- Sys.time()   # just to document the time taken
#  OBVIOUSLY, modify the rundir definition to suit your own setup!!!
postfixdir <- "EGMR12"  # a name for the rundir, in fact, same as base-case
rundir <- pathtopath(prefixdir,postfixdir)  # define the rundir
verbose <- TRUE
controlfile <- paste0("control",postfixdir,".csv")
outdir <- "C:/aMSE_scenarios/EG/"   # for storing results
# check all is well with the directories
confirmdir(rundir,ask=FALSE)   # automatically make it if it does not exist

c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EGMR12  already exists

confirmdir(outdir,ask=FALSE)   # interactively ask default = TRUE

C:/aMSE_scenarios/EG/  already exists
```

One can see that the new *rundir* (*EGMR12*) now contains the three files required to run the *MSE*. It is a good idea to check the *controlEGMR12.csv* file where you should find the the runlabel and the datafile names have been changed appropriately by the *copyto()* function. As the change to be made is in the *hsargs* input then everything is ready to run the MSE, as before. Normally we conduct a run with at least 250 replicates, or perhaps 500, for this example, we will leave the number of replicates, as set in *controlEG.csv*, at 250, which is generaly sufficient to understand the implications of a given HS. Note also that we have set *includeNAS=FALSE* to keep the saved outputs relatively compact.

```
#  set metUnder and metOver = 2, and pmwtSwitch = 0
hsargs <- list(mult=0.1, wid = 4,
               targqnt = 0.55, maxtarg = c(150,150,150,150,150,150,150,150),
               pmwts = c(0.65,0.25,0.1),  # relative weights of PMs
               hcr = c(0.25,0.75,0.8,0.85,0.9,1,1.05,1.1,1.15,1.2),
               hcrm3 = c(0.25,0.75,0.8,0.85,0.9,1,1.1,1.2,1.25,1.3),
               startCE = 2000, endCE = 2019,
```

```
              metRunder = 2, metRover = 2, # set = 2 means metarule 1 and 2
              decrement=1, pmwtSwitch = 0, # set = 0 means no metarule 3
              stablewts = c(0.8, 0.15, 0.05),
              hcrname="constantrefhcr", printmat=NULL)
checkhsargs(hsargs)
```

```
The hcr being used is:  constantrefhcr
meta-rule 2 is being used
meta-rule 1 is being used
hsargs$startCE and hsargs$endCE are used to define the reference periods
for each sau
```

```
out <- do_MSE(rundir,controlfile,hsargs=hsargs,hcrfun=constantrefhcr,
              sampleCE=tasCPUE,sampleFIS=tasFIS,sampleNaS=tasNaS,
              getdata=tasdata,calcpopC=calcexpectpopC,makeouthcr=makeouthcr,
              fleetdyn=NULL,scoreplot=plotfinalscores,
              plotmultflags=plotmultandflags,interimout="",
              varyrs=7,startyr=38,verbose=TRUE,ndiagprojs=4,cutcatchN=56,
              matureL=c(40,170),wtatL=c(50,200),mincount=120,
              includeNAS = FALSE,depensate=0,kobeRP=c(0.4,0.2,0.15),
              nasInterval=5,minsizecomp=c(100,135),uplimH=0.35,incH=0.005)
```

```
All required files appear to be present
Files read, now making zone
Now estimating population productivity between H  0.005  and  0.35
makeequilzone  25.2109 secs

Conditioning on the Fishery data
dohistoricC  1.144564 secs

Conditioning plots completed  1.497504 secs

Preparing for the projections
Projection preparation completed  10.57106 secs
Doing projections
2021  2022  2023  2024  2025  2026  2027  2028  2029  2030  2031  2032
2033  2034  2035  2036  2037  2038  2039  2040  2041  2042  2043  2044
2045  2046  2047  2048  2049  2050
All projections finished  1.468512 mins
Now generating final plots and tables
Starting the sau related plots

Finished all sau plots  14.37874 secs
Starting size-composition plots

Finished size-composition plots 5.972448 secs

Plotting fishery information

hsargs.txt saved to rundir
plotting HS performance statistics

plotting Population level dynamics

All plots and tables completed  15.46443 secs
```

```
makeoutput(out,rundir,postfixdir,controlfile,hsfile="EGHS Package",
           doproject=TRUE,openfile=TRUE,verbose=FALSE)

save(out,file=pathtopath(outdir,paste0(postfixdir,".RData")))
```

Now we need to repeat this a further two times using *EGMR3* and *EGMRall*, which involves redefining the *postfixdir*, which in turn, redefines the *rundir*. Do not forget to change the meta-rule values, else you only repeat the previous analysis! Remember also, if you want the web-sites to open automatically after they have finished you need to set the *openfile* argument in *makeoutput()* = TRUE.

```
# set metUnder and metOver = 0, and pmwtSwitch = 4, include hcr3 vector
postfixdir <- "EGMR3"     # a new name for the rundir
rundir <- filenametopath(prefixdir,postfixdir)  # define the rundir
controlfile <- paste0("control",postfixdir,".csv")
# check all is well with the directories
confirmdir(rundir,ask=FALSE)   # automatically make it if it does not exist

c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EGMR3  already exists

filelist=c("controlEG.csv","lf_WZ90-20.csv","saudataEG.csv")
copyto(prefixdir=prefixdir,fromdir="EG",todir=postfixdir,filelist=filelist)



controlEG.csv  has been copied to  EGMR3  as  controlEGMR3.csv
lf_WZ90-20.csv  has been copied to  EGMR3  as  lf_WZ90-20.csv
saudataEG.csv  has been copied to  EGMR3  as  saudataEGMR3.csv
Be sure to change the control, data, and run files where necessary

[1] "c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EGMR3"

hsargs <- list(mult=0.1, wid = 4, # mult changed to = 0.05
               targqnt = 0.55, maxtarg = c(150,150,150,150,150,150,150,150),
               pmwts = c(0.65,0.25,0.1),  # relative weights of PMs
               hcr = c(0.25,0.75,0.8,0.85,0.9,1,1.05,1.1,1.15,1.2),
               hcrm3 = c(0.25,0.75,0.8,0.85,0.9,1,1.1,1.2,1.25,1.3),
               startCE = 2000, endCE = 2019, metRunder = 0, metRover = 0,
               decrement=1,
               pmwtSwitch = 4, stablewts = c(0.8, 0.15, 0.05),
               hcrname="constantrefhcr", printmat=NULL)
checkhsargs(hsargs)

The hcr being used is:  constantrefhcr
meta-rule 3 is being used
hsargs$startCE and hsargs$endCE are used to define the reference periods
for each sau


out <- do_MSE(rundir,controlfile,hsargs=hsargs,hcrfun=constantrefhcr,
              sampleCE=tasCPUE,sampleFIS=tasFIS,sampleNaS=tasNaS,
              getdata=tasdata,calcpopC=calcexpectpopC,makeouthcr=makeouthcr,
              fleetdyn=NULL,scoreplot=plotfinalscores,
              plotmultflags=plotmultandflags,interimout="",
```

```
            varyrs=7,startyr=38,verbose=TRUE,ndiagprojs=4,cutcatchN=56,
            matureL=c(40,170),wtatL=c(50,200),mincount=120,
            includeNAS = FALSE,depensate=0,kobeRP=c(0.4,0.2,0.15),
            nasInterval=5,minsizecomp=c(100,135),uplimH=0.35,incH=0.005)

All required files appear to be present
Files read, now making zone
Now estimating population productivity between H  0.005  and  0.35
makeequilzone  24.41401 secs

Conditioning on the Fishery data
dohistoricC  1.235547 secs

Conditioning plots completed  1.58161 secs

Preparing for the projections
Projection preparation completed  10.40617 secs
Doing projections
2021  2022  2023  2024  2025  2026  2027  2028  2029  2030  2031  2032
2033  2034  2035  2036  2037  2038  2039  2040  2041  2042  2043  2044
2045  2046  2047  2048  2049  2050
All projections finished  1.629163 mins
Now generating final plots and tables
Starting the sau related plots

Finished all sau plots  15.87958 secs
Starting size-composition plots

Finished size-composition plots 6.616973 secs

Plotting fishery information

hsargs.txt saved to rundir
plotting HS performance statistics

plotting Population level dynamics

All plots and tables completed  17.78126 secs

makeoutput(out,rundir,postfixdir,controlfile,hsfile="EGHS Package",
          doproject=TRUE,openfile=TRUE,verbose=FALSE)

save(out,file=pathtopath(outdir,paste0(postfixdir,".RData")))
```

And now the final scenario EGMRall, including all three meta-rules:

```
# set metUnder and metOver = 2, and pmwtSwitch = 4, include hcr3 vector
postfixdir <- "EGMRall"     # a new name for the rundir
rundir <- filenametopath(prefixdir,postfixdir)  # define the rundir
controlfile <- paste0("control",postfixdir,".csv")
# check all is well with the directories
confirmdir(rundir,ask=FALSE)   # automatically make it if it does not exist

c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EGMRall  already exists
```

```
filelist=c("controlEG.csv","lf_WZ90-20.csv","saudataEG.csv")
copyto(prefixdir=prefixdir,fromdir="EG",todir=postfixdir,filelist=filelist)
```

```
controlEG.csv  has been copied to  EGMRall  as  controlEGMRall.csv
lf_WZ90-20.csv  has been copied to  EGMRall  as  lf_WZ90-20.csv
saudataEG.csv  has been copied to  EGMRall  as  saudataEGMRall.csv
Be sure to change the control, data, and run files where necessary

[1] "c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EGMRall"
```

```
hsargs <- list(mult=0.1, wid = 4, # mult changed to = 0.15
               targqnt = 0.55, maxtarg = c(150,150,150,150,150,150,150,150),
               pmwts = c(0.65,0.25,0.1),  # relative weights of PMs
               hcr = c(0.25,0.75,0.8,0.85,0.9,1,1.05,1.1,1.15,1.2),
               hcrm3 = c(0.25,0.75,0.8,0.85,0.9,1,1.1,1.2,1.25,1.3),
               startCE = 2000, endCE = 2019, metRunder = 2, metRover = 2,
               decrement=1, pmwtSwitch = 4, stablewts = c(0.4, 0.5, 0.1),
               hcrname="constantrefhcr", printmat=NULL)
checkhsargs(hsargs)
```

```
The hcr being used is:  constantrefhcr
meta-rule 2 is being used
meta-rule 1 is being used
meta-rule 3 is being used
hsargs$startCE and hsargs$endCE are used to define the reference periods
for each sau
```

```
out <- do_MSE(rundir,controlfile,hsargs=hsargs,hcrfun=constantrefhcr,
              sampleCE=tasCPUE,sampleFIS=tasFIS,sampleNaS=tasNaS,
              getdata=tasdata,calcpopC=calcexpectpopC,makeouthcr=makeouthcr,
              fleetdyn=NULL,scoreplot=plotfinalscores,
              plotmultflags=plotmultandflags,interimout="",
              varyrs=7,startyr=38,verbose=TRUE,ndiagprojs=4,cutcatchN=56,
              matureL=c(40,170),wtatL=c(50,200),mincount=120,
              includeNAS = FALSE,depensate=0,kobeRP=c(0.4,0.2,0.15),
              nasInterval=5,minsizecomp=c(100,135),uplimH=0.35,incH=0.005)
```

```
All required files appear to be present
Files read, now making zone
Now estimating population productivity between H  0.005  and  0.35
makeequilzone  27.15966 secs

Conditioning on the Fishery data
dohistoricC  1.298785 secs

Conditioning plots completed  1.652807 secs

Preparing for the projections
Projection preparation completed  11.93203 secs
Doing projections
2021  2022  2023  2024  2025  2026  2027  2028  2029  2030  2031  2032
2033  2034  2035  2036  2037  2038  2039  2040  2041  2042  2043  2044
```

```
2045  2046  2047  2048  2049  2050
All projections finished  1.682187 mins
Now generating final plots and tables
Starting the sau related plots

Finished all sau plots  14.53013 secs
Starting size-composition plots

Finished size-composition plots 6.138116 secs

Plotting fishery information

hsargs.txt saved to rundir
plotting HS performance statistics

plotting Population level dynamics

All plots and tables completed  16.17092 secs

makeoutput(out,rundir,postfixdir,controlfile,hsfile="EGHS Package",
          doproject=TRUE,openfile=TRUE,verbose=FALSE)

save(out,file=pathtopath(outdir,paste0(postfixdir,".RData")))
```

Now, in your browser, there should be at least four webpages, each with the standard array of tabs.

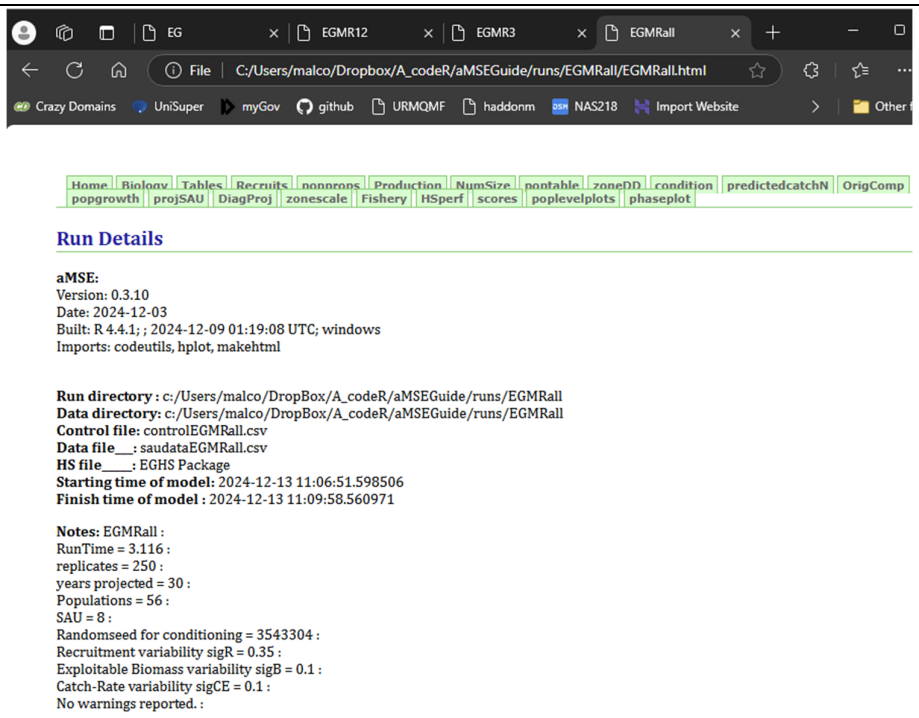


*Figure 4.1: A view of a browser with four scenarios run and completed.*

## 4.3 Making Comparisons

While it would be possible to open the same tab in each of the three pages and step between them for a visual comparison that is not conducive to making useful comparisons. Instead, R functions have been generated that make a series of comparisons between the scenarios by first reading in the stored *out* objects from each *MSE* run. This involves knowing the *outdir* that was used and selecting those stored objects that contain the information about each scenario that the user wants to compare. So, assuming that a number of scenarios have already been run (as above), first we set up the required libraries and directories and then list the contents of *outdir* so an explicit selection can be made. If the user has setup a different way of storing their results then, obviously, a different way of selecting and reading the required scenarios will be required.

### 4.3.1 Prepare the Analysis

```
options("show.signif.stars"=FALSE,"stringsAsFactors"=FALSE,
        "max.print"=50000,"width"=240)
suppressPackageStartupMessages({
  library(aMSE)
  library(EGHS)
  library(codeutils)
  library(hplot)
  library(makehtml)
  library(knitr)
})
dropdir <- getDBdir()
prefixdir <- pathtopath(dropdir,"A_codeR/aMSEGuide/runs/")

postfixdir <- "EG_compare"
verbose <- TRUE
rundir <- pathtopath(prefixdir,postfixdir)
outdir <- "C:/aMSE_scenarios/EG/" # obviously define one that suits you
confirmdir(rundir,ask=FALSE)

c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EG_compare  already exists

confirmdir(outdir,ask=FALSE)

C:/aMSE_scenarios/EG/  already exists

files <- dir(outdir)
printV(files)

           value index
1   BCtest.RData     1
2       EG.RData     2
3   EGMR12.RData     3
4    EGMR3.RData     4
5  EGMRall.RData     5
6 MR4noLML.RData     6
7  testMR4.RData     7
```

We will compare all four 'EG' scenarios, though that will make some of the plots relatively busy.

### 4.3.2 Make the Comparisons

Currently, in the vector called *files* there are six scenarios, there could be very many more so a method of selecting which files to use is needed. This is handled by the wrapper function *do_comparison()* using an argument *pickfiles*, which is merely a vector of the index of each file wanted in a comparison. In this case, *pickfiles=c(1,2,3,4)*, if there had been two files listed before the three we are interested in it would have been *pickfiles=c(3,4,5,6)*. To understand all the arguments to *do_comparisons()* see the function's help. Running the following code will generate a new webpage with multiple tabs (see the help for *?do_comparison*). The output from the *do_comparison()* function is directed into an object, called here, *result*. The structure of the contents of *result* are listed in the appendix to this chapter. This is made available so that individuals can develop new plots and comparisons of their own.

```
pfiles <- grep("EG",files) # or some other way of selecting, even manually.
result <- do_comparison(rundir=rundir,postfixdir=postfixdir,outdir=outdir,
                        files=files,pickfiles=pfiles,verbose=TRUE,
                        intensity=100,zero=FALSE,Q90=TRUE,
                        altscenes=NULL,juris="",
                        ribbonleg="topleft",scencol=c(1,2,3,4))

Loading  EG.RData  which may take time, be patient
Loading  EGMR12.RData  which may take time, be patient
Loading  EGMR3.RData  which may take time, be patient
Loading  EGMRall.RData  which may take time, be patient

          Base_Case     EGMR12      EGMR3    EGMRall same
reps         250.00     250.00     250.00     250.00    1
randseed 3543304.00 3543304.00 3543304.00 3543304.00    1
sigR           0.35       0.35       0.35       0.35    1
sigB           0.10       0.10       0.10       0.10    1
sigCE          0.10       0.10       0.10       0.10    1
projyrs       30.00      30.00      30.00      30.00    1
numpop        56.00      56.00      56.00      56.00    1
nSAU           8.00       8.00       8.00       8.00    1
Nclass       105.00     105.00     105.00     105.00    1
hyrs          58.00      58.00      58.00      58.00    1
pyrs          30.00      30.00      30.00      30.00    1
larvdisp       0.01       0.01       0.01       0.01    1
initdepl       1.00       1.00       1.00       1.00    1
Now doing the comparisons

Now doing the HSPM tab

Now doing the C_vs_MSY tab

Now doing the Scenario PMs

Now doing the Zone tab

Now doing the Catch tab
```

```
Now doing the cpue tab

Now doing the depletion spawnB tab

Now doing the depletion exploitB tab

Now doing the harvestR tab

Now doing the phaseplots tab
```

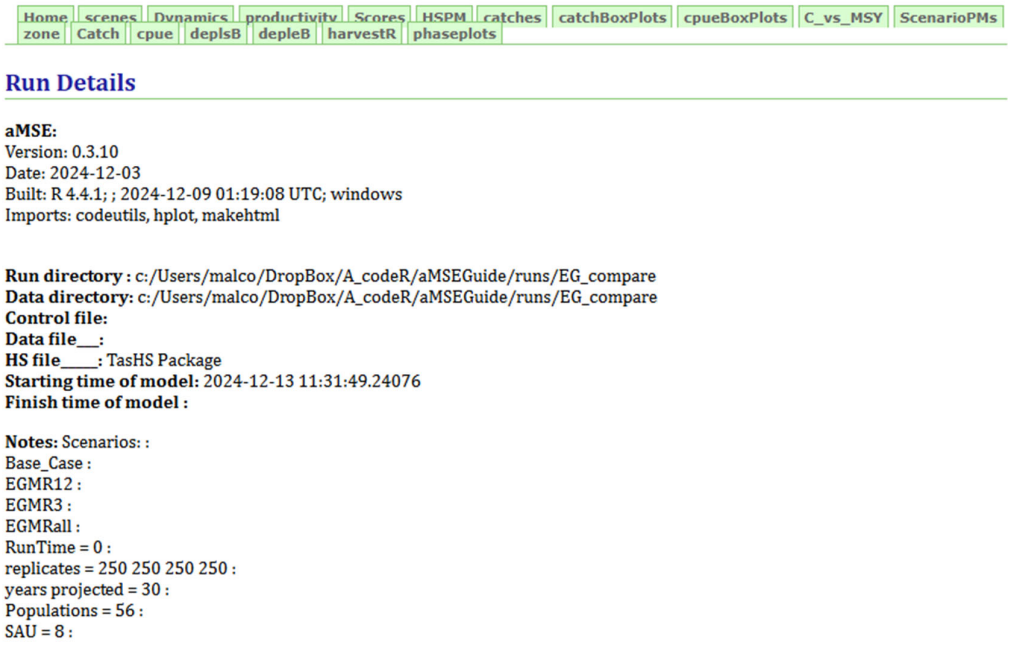


*Figure 4.2: A view of a browser page with the tabs currently produced by the do_comparison() function.*

### 4.3.3 The Different Tabs

Hopefully, by now, it is not unexpected that the *Home* tab exhibits details relating to the run.

The comparisons begin in the *scenes* tab, which currently only contains a single table detailing the underlying characteristics of each scenario. This table is a repeat of the table of scenario characteristics printed to the console.

Given the challenge is to recover the stock in each SAU and that should allow the recovery of the fishery, a balance or trade-off is required between the rapidity with which catches are returned to the fishery and the rate at which stock biomass is predicted to recover. As the plots in the Dynamics tab demonstrate, returning catches too rapidly does not allow the biomass to increase as quickly, as evidenced by the predicted CPUE rising then falling rapidly. To approach a more stable and resilient fishery such oscillations are best avoided.

*Dynamics* tab, where the median and 90th quantiles of the predicted projections for cpue, actual catch, mature biomass, and harvest rate for each scenario are plotted against each

other. Few differences occur between scenarios in their CPUE until between 2028 – 2030, depending on which Sau is considered. SAU 13 exhibits the relatively similar trends through time. In all other SAU the no meta-rule (noMR) and meta-rule 3 (MR3) scenarios exhibit a similar oscillatory trend except the noMR has a time-lag to reach the minimum CPUE of about 10-11 years. In all cases in all scenarios (except in SAU13) CPUE remains above 100kg following the period of stock recovery. The MRall appears to achieve a relatively stable CPUE more rapidly than MR12 and does not reach such high levels.

In terms of predicted catches MR3 returns catch to the fishery the fastest but then rapidly reduces it again to become more stable eventually. As with CPUE, noMR follows a similar pattern just 10 years more slowly. MRall leads to somewhat higher stable catches than MR12 and once again appears to find an improved balance between speed of recovery of the fishery and a conservative return of catches. These outcomes are reflected in the predicted annual harvest rates as well.

The mature biomass follows approximately the same trajectories in all scenarios until 2028-2030 where they begin to diverge. noMR and MR3 aer both oscillatory with noMR lagging behind the MR3. MR12 and MRall are both more stable in their trajectories with the MR12 scenario ending up with a somewhat higher level of mature biomass in all SAU except SAU13. While that is a positive thing for the stock it implies a somewhat lower final catch.

The *Productivity* tab only exhibits a single table, which implies that all scenarios share the same productivity characteristics. If the selected scenarios differed in some factor that influenced productivity (perhaps different natural mortality values) then each different scenario would have its own table of productivity properties by *sau*.

The *Scores* tab illustrates the final median harvest strategy scores for each *sau* and compare them in the plot. But each scenario has a table with the final scores tabulated. These data derive from functions within EGHS and will quite likely require changes when other Harvest Strategies are implemented. Given the harvest control rule implemented in the HS the median target would be a value of six (where values between 5 and 6 imply no change in catch).

The *HSPM* tab contains tables that, for each *sau* (and sometimes the whole zone), is gives the median annual average variation in catch (aavc), the sum of the first 5 projection years of catch, and the sum of the first 10 projection years of catch. The variability of catches during the recovery stage is relatively low and it only begins to increase after about 10 years. Even so, this table demonstrates that MR3 tends to be more variable in its catches than the other three scenarios in all SAU. The sum to 5 and sum to 10 years illustrate differences between the scenario with the sum10 reflecting the rapid return of catches made by MR3 with MRall providing a more rapid return than either noMR or MR12.

The *Catches* tab contains three plots and a table. The key plots are where the average annual variation of catches is plotted for each scenario x each *sau*, this is repeated for the sum of catches over the first five years, and then for the first 10 years. The effect on *sau12* become apparent once again in each of these plots. The final table contains the boxplot statistics for each comparison and each *sau*.

The *catchBoxPlots* is similar to the *catches* tab but contains boxplots for each *sau* for each scenario, for the three harvest strategy performance measures: aavc, sum5, and sum10.

The *cpueBoxPlots* tab contains boxplots for each *sau* within each scenario of the years taken to achieve the maximum cpue, and for each *sau* x scenario, the years taken to achieve the median maximum cpue.

The *C_vs_MSY* tab contains plots for each scenario and *sau* of the ratio of the predicted catches divided by the predicted *MSY* for each *sau*, which is one of the preferred MSE performance measures. In the current example, the trajectory of return to near the MSY might be better illustrated by a new ribbon plot as seen in the zone tab, and this will be implemented.

The *ScenarioPMs* tab contains a set of histograms of the deviations from a loess fitted, in each scenario, to all replicates across years. The mean and standard deviation of those distributions are tabulated below. This is a different measure of catch variability.

The *zone* tab contains ribbon plot outlines of each scenario's 90th quantile bounds for the total zone statistics, overlaid so that areas of overlap and differences become clear. The use of rgb colours allows for transparency so that the degree of overlap can become clear. When too many scenarios are included in a plot this can become confusing.

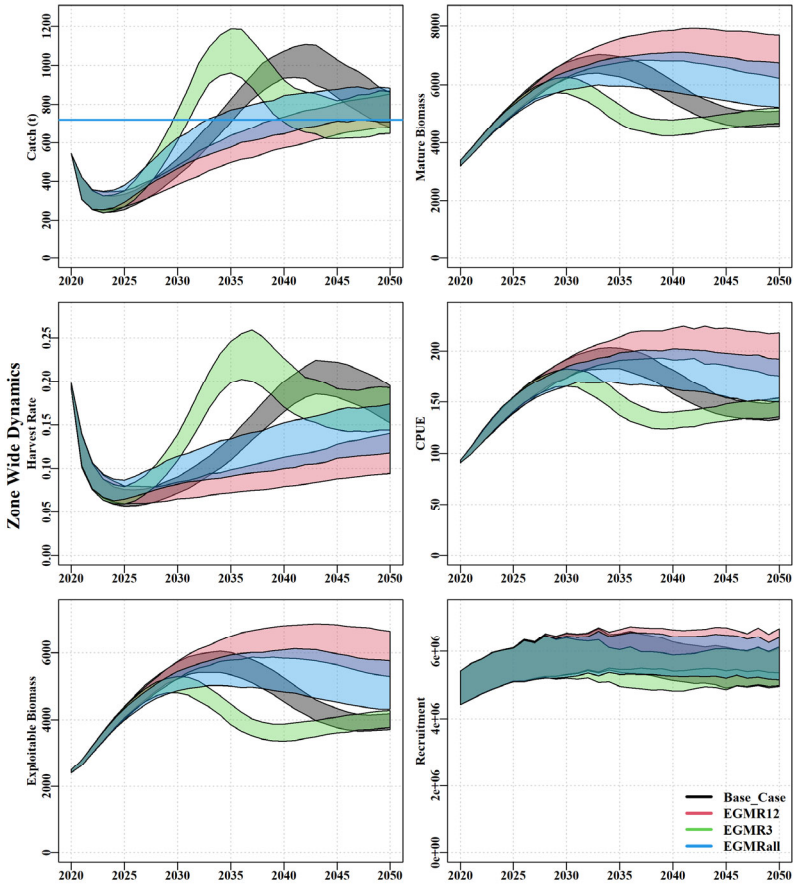


*Figure 4.3: The zone tab generated by the do_comparison() function. The base-case = EG, no meta-rules.*

The Catch, cpue, deplsB, depleB, and the harvestR tabs all use the same ribbon plots as in the zone tab except they repeat the information in the Dynamics tab in a manner that makes comparisons simpler for some people.

The *Catch* tab uses ribbon plots to illustrate the overlap and differences between the scenarios in terms of the combined catch for each *sau*. The use of semi-transparent colours helps to discern the level of overlap.

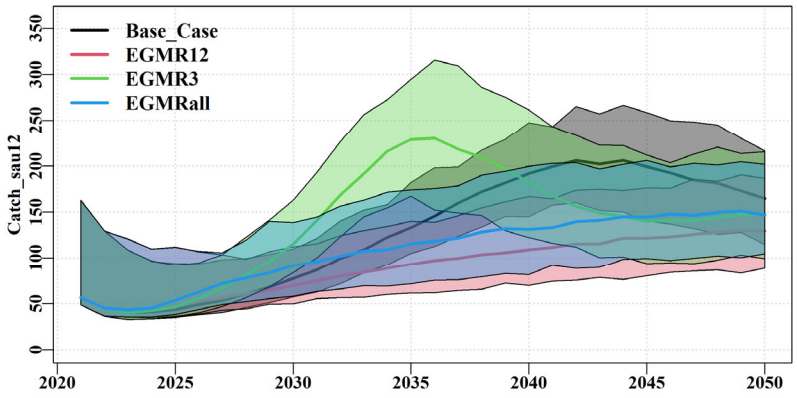


*Figure 4.4: The Catch tab generated by the do_comparison() function for sau12. The base-case = EG, no meta-rules.*

The *cpue* tab uses ribbon plots to illustrate the overlap and differences between the scenarios in terms of the cpue in each *sau*. Once again, the use of semi-transparent colours helps to discern the level of overlap.

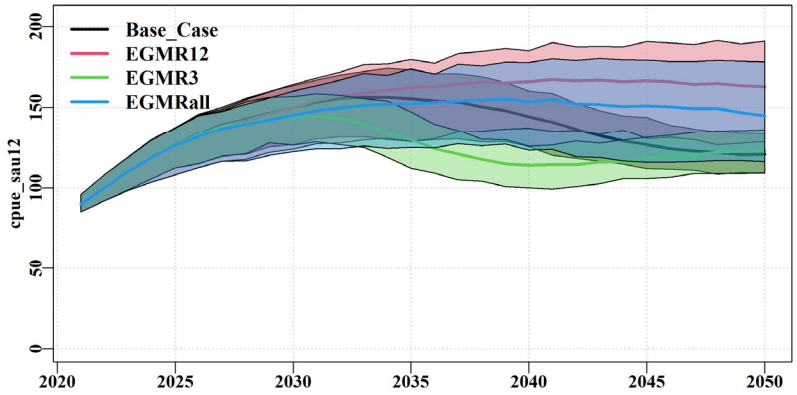


*Figure 4.5: The cpue tab generated by the do_comparison() function for sau12. The base-case = EG, no meta-rules.*

The *deplsB* tab uses ribbon plots to illustrate the overlap and differences between the scenarios in terms of the mature biomass depletion levels in each *sau*.

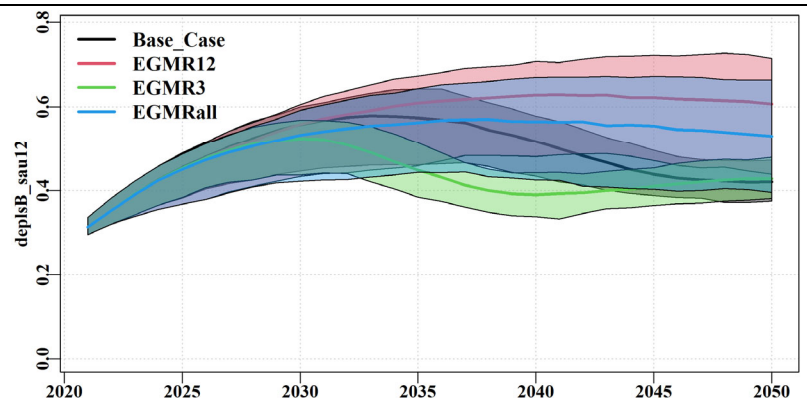


*Figure 4.6: The spawning biomass depletion tab generated by the do_comparison() function for sau12. The base-case = EG, no meta-rules.*

The *depleB* tab uses ribbon plots to illustrate the overlap and differences between the scenarios in terms of the exploitable biomass depletion levels in each *sau*. In the example, the differences found in *sau12* become very clear. The results here are similar to those exhibited with the spawning biomass depletion, but remain of interest because the exploitable biomass is more closely associated with cpue than the spawning biomass.

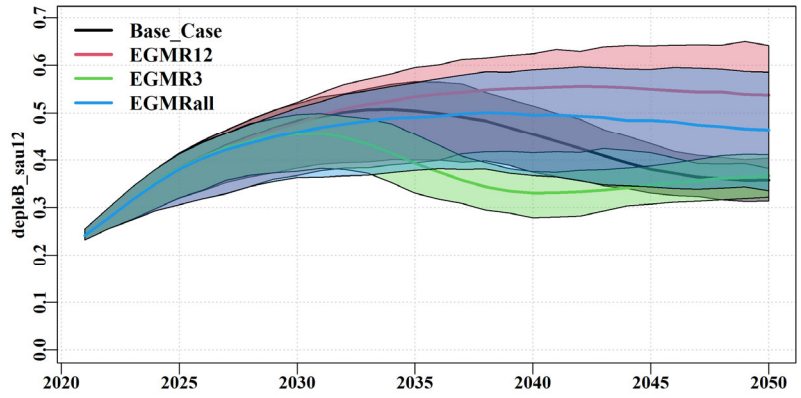


*Figure 4.7: The exploitable biomass depletion tab generated by the do_comparison() function for sau12. The base-case = EG, no meta-rules.*

The *harvestR* tab makes it very clear that MR3, sustains oscillatory behaviour in terms of harvest rates (and other dynamics) much longer than the other scenarios.

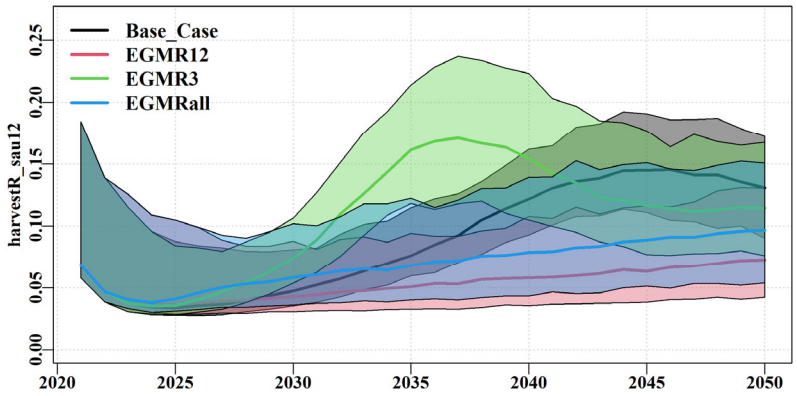


*Figure 4.8: The harvest rate tab generated by the do_comparison() function for sau12. The base-case = EG, no meta-rules.*

Finally, the *phaseplots* tab contains ten plots depicting, for each *sau* the effect of the different scenarios on the relationships, during the projections, of the following combinations of variable (using the medians in all cases): 1) actual catch/MSY vs exploitable biomass depletion, 2) actual catch/MSY vs mature biomass depletion, 3) actual catch/MSY vs Mature Biomass/Bmsy, 4) actual catch vs exploitable biomass depletion, 5) Actual catches vs mature biomass depletion, 6) Aspirational catch vs mature biomass, 7) Aspirational catch vs Mature Biomass depletion, 8) CPUE vs Actual catches, 9) Annual harvest rate vs actual catches, and 10) Actual catches vs Exploitable biomass.

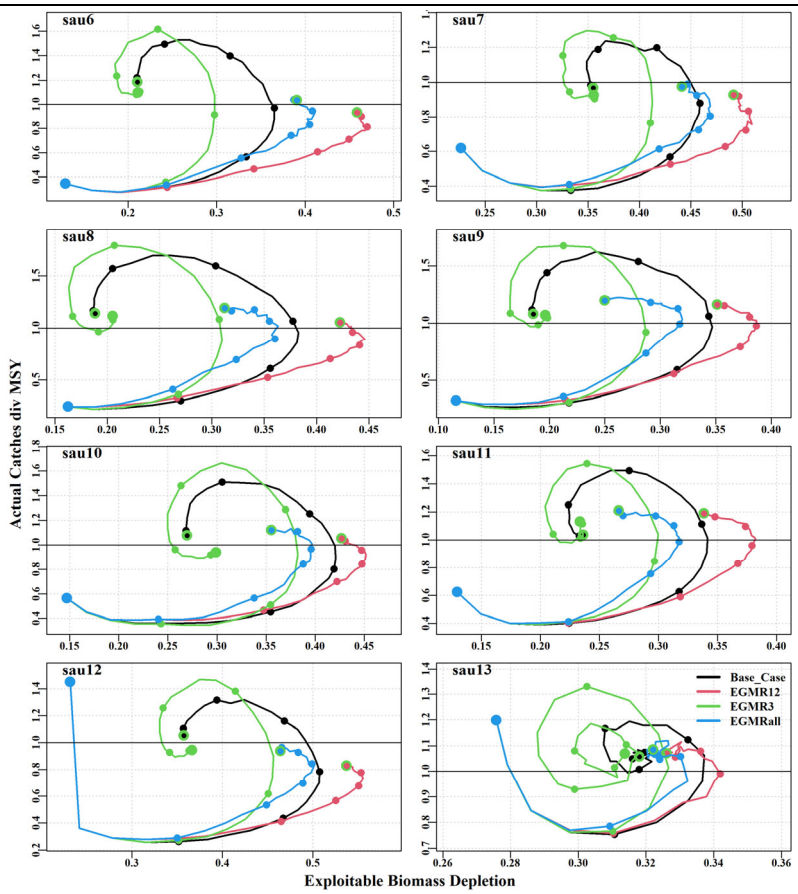


*Figure 4.9: The harvest rate tab generated by the do_comparison() function for sau12. The base-case = EG, no meta-rules.*

## 4.4 Further Developments

While above the text ended on "Finally, …" the comparison of scenarios is as areas still under active development and further additions continue to be made with different requirements for different jurisdictions. This, along with each jurisdiction's harvest strategy, is another area where the needs of the different jurisdictions may differ and require either extra R source files or packages to be developed in each jurisdiction to provide the analyses and plots wanted in each region.

## 4.5 Appendix Structure of the output from do_comparison()

Here we explore the structure of the outputs from *do_comparion()* by using the four scenarios comared in this chapter.

The top level components of *result* are:

str1(result) : List of 7

- scenes : chr [1:4] “Base_Case” “EGMR12” “EGMR3” “EGMRall”
- ans :List of 4
- quantscen:List of 4
- dyn :List of 4
- prods :List of 4
- scenprops: num [1:13, 1:5] 2.50e+02 3.54e+06 3.50e-01 1.00e-01 1.00e-01 …
- devout :List of 2

The *scenes* object is, as shown above, a character vector with the names of each scenario

The *ans* object contains the outputs from each *do_MSE()* scenario.

str1(result$ans) : List of 4

- EG :List of 29
- EGMR12 :List of 29
- EGMR3 :List of 29
- EGMRall:List of 29

where each of he scenario lists have the following structure:

str1(result$ans$EG) : List of 29

- tottime : ‘difftime’ num 2.94
- runtime : POSIXct[1:1], format: “2025-01-07 07:54:54”
- starttime : POSIXct[1:1], format: “2025-01-07 07:51:58”
- glb :List of 19
- ctrl :List of 13
- zoneCP :List of 56
- zoneD :List of 14
- zoneDD :List of 14
- zoneDP :List of 14
- NAS : NULL
- projC :List of 5
- condC :List of 15
- sauout :List of 10
- outzone :List of 13
- production : num [1:71, 1:6, 1:56] 256 244 233 223 213 …
- condout :List of 2

- HSstats :List of 2
- saudat : num [1:32, 1:8] 21.5 0.3 130 1 54.3 …
- constants : num [1:33, 1:56] 6 21.5 0.3 130 1 …
- hsargs :List of 16
- sauprod : num [1:7, 1:8] 438.882 149.874 20.272 0.341 140.632 …
- zonesummary :List of 2
- kobedata : num [1:8, 1:4] 0.33 0.449 0.258 0.242 0.313 …
- outhcr :List of 8
- scoremed : num [1:30, 1:7] 42.8 35.3 31.8 31.5 33.4 …
- popmedcatch :List of 8
- popmedcpue :List of 8
- popmeddepleB:List of 8
- pops : num [1:56, 1:26] 1 2 3 4 5 6 7 8 9 10 …

and would include the numbers-at-size objects is these were saved with NAS=TRUE.

The result$quantscen object and internal structures contain the quantiles for the dynamics of each SAU for the four variables *cpue*, *catch*. *matureB*, and *harvestR*, with the 0.025, 0.05, 0.5, 0.95, and 0.975 quantiles across the *reps* replicates, giving the central 95%, and 90% ranges and the median.

str1(result$quantscen) : List of 4

- cpue :List of 4
- catch :List of 4
- matureB :List of 4
- harvestR:List of 4

str1(result$quantscen$cpue) : List of 4

- Base_Case:List of 8
- EGMR12 :List of 8
- EGMR3 :List of 8
- EGMRall :List of 8

str1(result$quantscen$cpue$Base_Case) : List of 8

- sau6 : num [1:5, 1:30] 97.6 99.9 110.3 121.3 123.1 …
- sau7 : num [1:5, 1:30] 111 112 124 136 140 …
- sau8 : num [1:5, 1:30] 149 150 161 174 177 …
- sau9 : num [1:5, 1:30] 117 119 131 143 148 …
- sau10: num [1:5, 1:30] 96 97.1 104.5 112.8 114.7 …
- sau11: num [1:5, 1:30] 89 90.4 97.2 106.6 107.9 …
- sau12: num [1:5, 1:30] 83.4 85 89.9 95.9 96.9 …
- sau13: num [1:5, 1:30] 84.7 85.5 89.5 93.7 95.2 …

The structure of the *dyn* object within *result* provides all replicates within the dynamics for each SAU. Each of the scenarios is made up of arrays of 88 years including the historic years and the projections, the 8 SAU, and the 250 replicates.

str1(result$dyn) : List of 4

- EG :List of 10
- EGMR12 :List of 10
- EGMR3 :List of 10
- EGMRall:List of 10

str1(result$dyn$EG) : List of 10

- matureB : num [1:88, 1:8, 1:250] 439 438 436 429 409 …
- exploitB: num [1:88, 1:8, 1:250] 415 414 413 409 395 …
- midyexpB: num [1:88, 1:8, 1:250] 0 431 430 428 421 …
- catch : num [1:88, 1:8, 1:250] 0 1 2 8 22 …
- acatch : num [1:88, 1:8, 1:250] 0 1 2 8 22 …
- harvestR: num [1:88, 1:8, 1:250] NaN 0.00232 0.00465 0.0187 0.05228 …
- cpue : num [1:88, 1:8, 1:250] 0 437 436 432 422 …
- recruit : num [1:88, 1:8, 1:250] 259139 259078 258961 258478 257088 …
- deplsB : num [1:88, 1:8, 1:250] 1 0.998 0.994 0.977 0.933 …
- depleB : num [1:88, 1:8, 1:250] 1 0.999 0.996 0.985 0.953 …

The structure of *result$prods* object provides the productivity characteristics of each SAU.

str1(result$prods) ; List of 4

- EG : num [1:8, 1:7] 439 872 524 2439 2072 …
- EGMR12 : num [1:8, 1:7] 439 872 524 2439 2072 …
- EGMR3 : num [1:8, 1:7] 439 872 524 2439 2072 …
- EGMRall: num [1:8, 1:7] 439 872 524 2439 2072 …

result$prods$EG

- `B0 Bmsy MSY Dmsy CEmsy Hmsy Bexmsy`
- sau6 438.8820 149.8744 20.27160 0.3414914 140.63196 0.195 91.32144
- sau7 872.1711 272.7968 43.73620 0.3127790 115.88669 0.200 191.66429
- sau8 524.4789 166.0719 24.93913 0.3166417 194.81373 0.175 126.23452
- sau9 2438.8778 748.3589 122.48502 0.3068456 198.17074 0.180 600.52685
- sau10 2071.9641 616.6733 102.77587 0.2976274 142.30645 0.170 536.12436
- sau11 4152.7893 1257.9882 207.63196 0.3029261 140.28520 0.180 1018.44438
- sau12 3126.2611 951.3633 156.59858 0.3043135 91.10815 0.175 792.96791
- sau13 915.9109 303.9922 41.73923 0.3319015 75.48181 0.190 193.28620

The *result$scenprops* object contain the properties of each scenario used to ensure that like is being compared with like. The *same* column identifies any rows which have difference between scenarios.

result$scenprops

- Base_Case EGMR12 EGMR3 EGMRall same

- reps 250.00 250.00 250.00 250.00 1
- randseed 3543304.00 3543304.00 3543304.00 3543304.00 1
- sigR 0.35 0.35 0.35 0.35 1
- sigB 0.10 0.10 0.10 0.10 1
- sigCE 0.10 0.10 0.10 0.10 1
- projyrs 30.00 30.00 30.00 30.00 1
- numpop 56.00 56.00 56.00 56.00 1
- nSAU 8.00 8.00 8.00 8.00 1
- Nclass 105.00 105.00 105.00 105.00 1
- hyrs 58.00 58.00 58.00 58.00 1
- pyrs 30.00 30.00 30.00 30.00 1
- larvdisp 0.01 0.01 0.01 0.01 1
- initdepl 1.00 1.00 1.00 1.00 1

The *result$devout* object contains, for each SAU, the mean deviates from a loess fitted to each *catch* replicate as well as their standard deviation. This repeats the tables and complements the plots seen in the ScenarioPMs tab in the *do_comparions()* HTML output. It compares the variability of catches within each SAU across scenarios across the full set of projection years. In the meandevs table below for example, notice in SAU6 how the mean for EGMR3 is more than twice that of the other scenarios.

str1(result$devout) : List of 2

- meandevs: num [1:4, 1:8] 22.4 16.9 55.2 21.4 51 …
- sddevs : num [1:4, 1:8] 5.2 4.27 15.61 6.65 13.6 …

result$devout$meandevs

- sau6 sau7 sau8 sau9 sau10 sau11 sau12 sau13

- Base_Case 22.36759 51.00718 27.86222 123.9374 89.31076 211.9045 139.0051 100.81643
- EGMR12 16.88400 42.09966 23.32073 116.9489 88.71891 196.1574 127.8207 75.98318
- EGMR3 55.17959 98.20418 75.87913 344.6366 185.62404 519.1445 314.5054 147.18156
- EGMRall 21.43495 51.51618 32.01075 160.5546 119.10000 265.3326 172.1482 87.70842

# 5. The Input Files

## 5.1 aMSE Requirements

There are a number of requirements when conducting a Management Strategy Evaluation of alternative harvest strategies on an abalone fishery using the **aMSE** R package. It is currently best to use the *RStudio* integrated development environment, as this facilitates the installation of different R packages, and the editing and running of the R files used to conduct the actual scenario runs (see the *Using the aMSE Software* chapter).

The R packages required to run a scenario include at least:

- **aMSE** the main R package containing the operating model and auxiliary functions used.
- **codeutils** an R package containing numerous utility functions that are common to many simulation tasks.
- **hplot** an R package containing functions that facilitate the generation of various plots.
- **makehtml** an R package used to generate the multi-tabbed web-pages that can display the results generated. Not strictly required for running the scenarios but is helpful for displaying the results in an accessible manner.
- *jurisdictionHS.R* a source R file containing functions unique to a particular jurisdiction's harvest strategy, or a package such as:
- **TasHS** is a new R package that encapsulates the harvest strategy used in Tasmania, which is used as an alternative to having a *jurisdictionHS.R*. The input constants used by some of those Tasmanian HS R functions still need to be declared in *hsargs* before running the MSE. Using an R package rather than an R source file helps guarantee that the same software is used in all scenarios and makes it easier to maintain or modify the harvest strategy as required. In the examples herein, a stable but cut-down version of TasHS called EGHS is used.

These five packages are available from https://www.github.com/haddonm.

- **knitr** an R package used in vignette preparation and the production of formal tables in **aMSE**'s output (obtainable from CRAN).

Most importantly, for this chapter, there are also a number of other data files required for a scenario run and these are all *.csv* files. In the sections below we will go through their structure and contents in detail to act as a guide to their format and functioning. Each can be generated using the template functions within **aMSE** (see the *Using_aMSE* chapter for details):

- a *control_scenario.csv* file (whose name can be anything the user wishes), that holds control options, many other constants, and identifies other input files.
- a *saudata.csv* file, that holds parameterizations of the biological properties of the *sau* and the *populations* that make up the operating model.

There may also be other data files if such data are available, these include:

- a size-composition of commercial catch data file (default from the *rewritecompdata()* function is *lf_WZ90-20.csv*, which is also used in the control file from *ctrlfiletemplate()*)

- a size-composition of fishery-independent survey file, with the same internal format as the fishery size-composition file.

When using the R package **aMSE** a typical scenario run would entail a number of fundamental steps, although most of these are not immediately obvious tpothe user:

- Initiate each *population* within each *sau* in the operating model with its biological properties.
- Generate the equilibrium unfished zone across all populations and estimate the productivity of each *population* (and hence each *sau* and the total *zone*).
- Condition the operating model using either a hypothetical depletion or apply historical fishery data until the simulated *zone* is ready to be projected under control of the particular harvest strategy being considered (Figure 5.1).
- Conduct the projections and summarize the results.

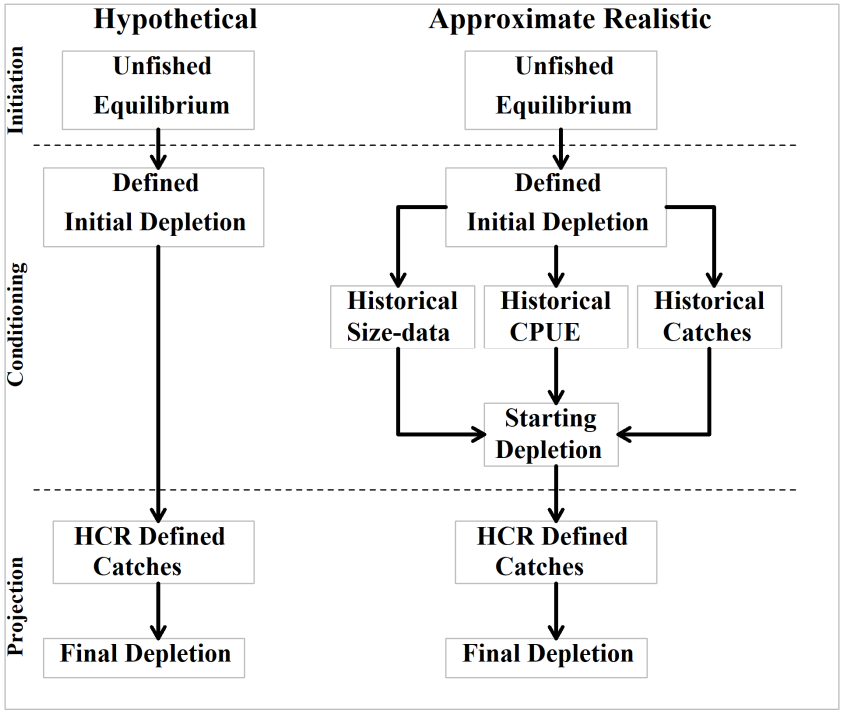


*Figure 5.1: A spectrum of alternative approaches for initiating and using the MSE framework. The initial depletion can be no initial depletion.*

## 5.2 File Locations for each Scenario

Within **aMSE**, for each scenario considered, it is designed so that all related input files (control, data, and size composition data, (and jurisdictionHS.R if that is being used rather than an HS package), and all result files will be contained in the same directory. Within the **aMSE** code, this directory is referred to as the *rundir* (as in run-directory). It is suggested that

each scenario run be given a unique name that is then used as the *rundir* name (the *postfixdir* as described in the *Using aMSE* chapter). For example, if examining the influence of the legal minimum length (LML) on the outcome of using a particular harvest strategy (perhaps using the **TasHS** R package) one might have the following scenarios: rundir = "EGlml140", "EGlml145", "EGlml150", and so on, all contained in a sub-directory named 'scenarios' (or whatever you wish; for example, *Malcolm*, is a good name, though you may be able to think of more appropriate ones). This would facilitate future comparisons by simplifying identification and selection of those scenarios to be compared. The names also suggest they are all variants of the "EG" basecase scenario. But of course, a user may name their *rundir* whatever they wish. The aim of **aMSE** is to facilitate running an *MSE* on abalone, not to force people to work effectively or efficiently!

The intent is that the *control.csv*, *saudata.csv* and the *lf_data.csv* files (and *jurisdictionHS.R* file if you have one) are stored in the *rundir* for each given scenario explored. This may entail duplicating the *saudata.csv*, *lf_data.csv*, and *jurisdictionHS.R* files between scenarios (one reason an HS R package is a sensible option) so care needs to be taken that if one changes these underlying files then those changed files need to be propagated across all sub-directories in which they are used. **This is the responsibility of the operator**. A simple way of making such copies has been illustrated when discussing the comparison of scenarios in the *Comparing Scenarios* chapter.

Here we will focus solely on the control and data files. There are functions to generate data file templates for each of the data-file types *ctrlfiletemplate()*, *datafiletemplate()*, and *rewritecompdata()*, which uses the internal data set data(lf). These can be run and the resulting templates edited to form the required files, this also automatically illustrates the required format. Alternatively, another option is to use the example data-files, made available with the **aMSE** package, which can be saved, copied and edited appropriately.

Similarly, within **aMSE**, there are functions to read in each type of data-file. If there is a formatting error within the data-file it should throw an error which identifies where it has gone wrong. Further development is intended to these error checks. Using the template functions and editing the results can be simpler than starting from scratch.

The input data-files, currently, are all required to be *.csv* files where components with multiple values use a comma as a separator. This makes their preparation using software such as *Microsoft Excel* relatively simple. They can also be edited directly in the editor pane of RStudio (or any other text editor).

The first of the two files is the *control_scenario.csv* (I name mine by including reference to the scenario, as in *controlM15h7L75.csv*, (natural mortality M=0.15, steepness h=0.7, and lambda L=0.75). This file contains a wide range of settings concerning the structure of the *Operating Model* as well as where to find any historical conditioning data from a given fishery (catches, cpue, recruitment deviates), if such conditioning is required. The second file would then be named *saudataM15h7L75.csv* and contains the data that is used to define the biological properties of each *sau* that make up the simulated *zone*. The rest of this document provides a detailed description of the contents and format of these files. The third file, another data file, would contain the size-composition data used to condition the scenario, if available.

If the user has used longer filenames to describe their scenarios this can lead to many plots becoming confused as the scenario names are used in the figure legends when making

comparisons. This is why *do_comparisons()* has an *altscenes* argument, that allows alternative scenario names to be allocated for use in the figures.

## 5.3 The Conditioning Requirements

There is a spectrum to the degree to which the operating model needs to be conditioned on a real fishery. At the least specific, one still has to include a spatial structure that would reflect a typical abalone stock. In **aMSE** the top level is the *zone*, which can contain even a single *SAU*, which must contain one or more *populations*. Hence the smallest simulated zone would constitute a single *SAU* with a single population (this might be used to compare the effect of included an array of populations within an sau). The example data-sets within **aMSE** exhibit a spatial structure of 8 SAU with a total of 56 populations. This hierarchy can be customized to suit a particular jurisdiction and species, and one could then include what might be deemed typical characteristics of growth, maturity, emergence, and other details, including variation among the different populations found within each SAU. Even with such details it would remain a generic abalone MSE (Figure 5.1).

It is possible to take such conditioning a step further and customize the details of the biology of each *SAU*, or even each *population* making up the spatial structure, to reflect the biology known for a real fishery *zone* (assuming there are field data available). One could also scale predicted productivity and cpue to match a real fishery. This would be less generic but still not exactly fitted to a real fishery. The detailed spatial structure internal to the operating model is what makes it impossible to fit the operating model to fishery data as a whole. However, it might still be possible to fit the productivity and fishing history of individual SAU to real fishery data and thereby gain estimates of actual productivity and possibly even recruitment deviates (see *Using_sizemod_to_Condition the SAU*). Of course, while this latter approach would appear, at first glance, to be the best strategy for examining a harvest strategy, as it might operate in a real situation, it is also the most demanding in terms of data and difficulty. One can spend large amounts of time iteratively improving an operating model fit to observed catches, length-composition, and CPUE data for each SAU (Figure 5.1). While this is a fine way to spend time (a lot of time, and I really mean a lot of time) it is also a good strategy to remember that there is a spectrum of how closely one needs to reflect nature and that some questions about harvest strategies do not require the best possible fit to a real fishery to remain useful for determining the implications of alternative management options (see for example, Pourtois *et al*, 2022).

## 5.4 The *control_scenario.csv* File

The control file is a *.CSV* file made up of a series of sections and each will be described in turn. The values expressed in this document may differ from those generated by the *ctrlfiletemplate()* function, but the descriptions are valid. Within the *control.csv* file (which can have any name you wish, it does not have to be *control.csv*, though it does have to be a *.csv* file), the main section headings are usually written using capital letters for increased clarity. The correct spelling of such section title is important, as is the spelling of the variable names in the first column. The order of the sections does not need to be strictly maintained but the order of any components within each section must be maintained as described. This is required because in a number of cases they are being parsed in sequence, though many are also being obtained explicitly by name (which is why typing/spelling the names correctly is also important; the template files are all correct). Where the labels or names are not clear as to the intent of the variable then a description can be given to the right of the actual values. Comments can be inserted anywhere outside of any of the sections. This is all exemplified in

the .csv file generated by *ctrlfiletemplate()*, which we will assume was given the default name *controlEG.csv*.

To aid understanding this document it would help the reader to have a copy of the *.csv* file generated by *ctrlfiletemple()* open in a window.

### 5.4.1 DESCRIPTION

At the top of the *controlEG.csv* file is an optional description section that can be used to describe the objective of the scenario implied by the control and population files. The author can add as many lines of text to this section as desired as, currently, it is not used in the model run. This may change later so it may act as meta-data for each run. It would be useful to treat this section as such meta-data anyway.

*DESCRIPTION*

*Control file containing details of a particular run*
*Modify the contents to suit your own situation*
*I have added these next lines, which do not influence reading the file*
*If you wish you should modify the contents of this description to suit*
*the scenario being executed, use as many lines as required for*
*your own description.*

### 5.4.2 START

The start section currently has four components, a name for the particular run (which gets used on the home tab/page of the output web-page but is most useful when comparing scenarios so make it informative!), the full name of the *saudataEG.csv* file (which is expected to be found in the same *rundir* as the controlfile). In each case, note the use of commas to separate a variable's label from each variable's value (cannot be seen if opened in Excel, but as long as the file is saved as a csv file all will be well). The hopefully descriptive text after each variable's value(s) is ignored by the program and can be expanded if desired (as long as it remains as a single line, the comments below are expanded to include more explanation but you should not have multiple lines). Such descriptive text should not include commas as this will break it up into separate Excel cells

*START*

- runlabel, Base_Case, label for particular scenario. Very important when comparing scenarios.
- datafile, saudataBC.csv, name of saudata file = saudataBC.csv
- *bysau*, 1, 1=TRUE, 0 = FALSE are we expecting data by SAU or population. Generally, one would use by sau, but an option exists to operate with populations explicitly. (see *Conditioning by Population*.).
- parfile, NA, name of file containing optimum parameters from **sizemod**, this is used to simplify transferring information from **sizemod** to **aMSE** during conditioning. Generally, this is NA, which means it gets ignored (see the chapter on *Using sizemod to condition the SAU*).

### 5.4.3 zoneCOAST

The zoneCOAST section (an odd/old name, which may change to become something more descriptive, but once something is named then coding inertia can set in) currently contains four components.

*zoneCOAST*

- *replicates, 100, number of replicates normally at least 250 or 500*, in the code this is found in *glb* named *reps*.
- *withsigR, 0.35, recruitment variability eg 0.35*, this literally varies recruitment in a Log-Normal fashion during projections and the last few years of the conditioning.
- *withsigB, 0.1, process error on exploitable biomass*. This affects the fleet dynamics so that, in Tasmania, the actual SAU catches end up rather different from the aspirational catches, though they all sum to the TAC. Elsewhere it could be used in the equivalent fleet dynamics function within the *JurisdictionHS.R* source file or R package.

- *withsigCE, 0.1, process error on cpue calculations*. This will vary the relationship between cpue and exploitable biomass by including Log-Normal errors. For example, a value of 0.025 will varying the apparent exploitable biomass in individual model events between 0.9 - 1.1 times the actual value.

In the code base, the three sources of variation (*withXXXX*) reside in the *ctrl* object.

*withsigR* relates to the year-to-year recruitment variability. This does not include any special recruitment events or recruitment deviates that may be included. During the generation of the equilibrium, unfished model, recruitment variation is set to a very small number, 1e-08, so that recruitment processes are effectively deterministic off the Beverton-Holt stock recruitment relationship. While there appears to be no evidence of auto-correlated recruitment residuals the intent is that these will eventually be included as an option (see *Operating Model Structure* for further details of the model equations). It is possible to include exceptional events in particular years within the projections that influence recruitment and/or survivorship. This is intended to allow for the exploration of the implications of such things as marine heat waves, exceptional storms, and other potentially damaging events. Details are given in the *Perturbations within Projections* chapter.

$$R_{p,t} = \frac{4hR_0B^S_{p,t}}{(1-h)B_0 + (5h-1)B^S_{p,t}} e^{\varepsilon_{p,t} - \sigma^2_R/2}$$

where $\varepsilon_{p,t}$ is defined as:

$$\varepsilon_{p,t} = N(0, \sigma^2_R)$$

*withsigB* the fleet dynamics of where divers elect to take their catches within an SAU mean that the actual catches from each SAU differ from the aspirational catches allocated to each SAU according to whatever harvest control rule is used. This will be determined within each jurisdictions HS R package, and may be different to the description given here, which only reflects the case in Tasmania. This is implemented by first allocating catches to the *SAU* in a deterministic manner and then modifying them by imputing variability to the perceived distribution of biomass among *SAU*, denoted $u$, which is then propagated across component *populations*:

$$B_{u,t}^{E,*} = B_{u,t}^{E} e^{\varepsilon_{u,t} - \sigma_B^2/2}$$

where:

$$\varepsilon_{u,t} = N(0, \sigma_B^2)$$

$\sigma_B$, *withsigB*, is the standard deviation of the variation that occurs between the real exploitable biomass and the perceived distribution of exploitable biomass across *SAU's*, and also includes other sources of uncertainty.

This variation is introduced into the potential *SAU* catches using:

$$C_{u,t}^{*} = TAC_t \times \frac{B_{u,t}^{E,*}}{\sum B_{u,t}^{E}}$$

where $TAC_t$ is the sum of the HCR's aspirational catches in each year $t$. One diagnostic used during conditioning (see the *DiagProj* tab in the webpage) is to examine the variation between the actual catches and the aspirational catches previously determined by the HCR, if one has such data. Otherwise, it should be characterized from the outputs, which will allow its variation to be examined for plausibility.

*withsigCE* designates the imprecision in the relationship between cpue and exploitable biomass. Essentially, it is used to generate a Log-Normally distributed random number about the value 1.0, which is then used to multiply the actual exploitable biomass. This is all conducted at the population scale (see *oneyearcat* code in *dynamics.R*). Given:

$$\varepsilon_{u,t} = N(0, \sigma_{ce}^2)$$

then bias-corrected Log-Normal errors depicted as:

$$err = e^{\varepsilon_{u,t} - \sigma_{ce}^2/2}$$

If $\sigma_{ce}$ is set extremely small (say, 1e-08), then $err$ is essentially equal to one, otherwise some variation around 1 will occur and be introduced into the estimate of cpue or each population. This would be whether hyperstability is also implemented in the cpue.

$$CE_p = q_p \times \hat{B}_p^E \times err \times 1000$$

where $\hat{B}_p^E$ is the average of the mid-year and end of year exploitable biomass. A $\sigma_{ce} = 0.025$ value of provides a range of multipliers on the exploitable biomass of between 0.9 and 1.0:

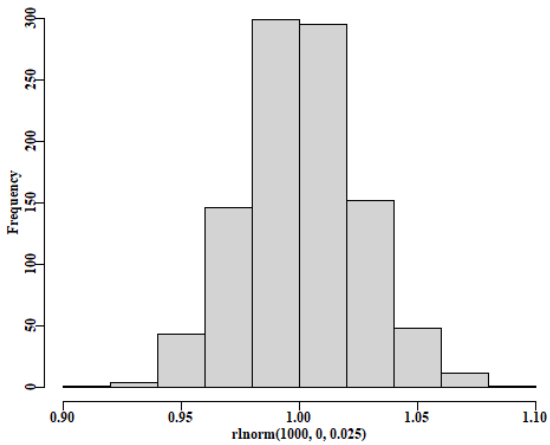


*Figure 5.2: Typical range of process error multipliers for a sigce value = 0.025.*

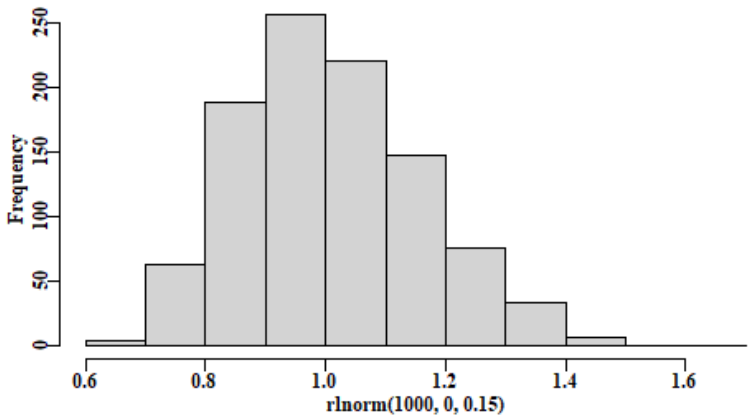


*Figure 5.3: Typical range of process error multipliers for a sigce value = 0.15.*

## 5.4.4 ZONE

Again, the use of commas to separate variable values is essential. The zone is defined in terms of the number of *SAU* (spatial assessment units) it is made up of.

*ZONE*

- *nSAU, 8, number of spatial assessment units eg 2*
- *SAUpop, 3, 3, 5, 7, 9, 9, 12, 8, number of populations per SAU in sequence*. The phrase, 'in sequence' relates to the idea of distributing the SAU, and their contained populations, along the coastline in an approximately linear manner. This simplifies the larval dispersal matrix, which currently assumes the populations are aligned along the coastline approximately linearly.
- *SAUnames, sau6, sau7, sau8, sau9, sau10, sau11, sau12, sau13, labels for each SAU*
- *initdepl, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, initial depletion levels for each SAU.*

In the code, *nSAU*, *SAUpop*, and *SAUnames* all reside in the *glb* object, inexplicably, *SAUnames* in the *glb* object is called *saunames*. The *initdepl* values are held in the *condC* object used when conditioning the zone.

The example file from *ctrlfiletemplate()* inserts the number of populations among SAU that reflect the current selection in the Tasmanian Western Zone, but each simulated zone would be expected to be different. The SAUnames should only be mixtures of letters and numbers but must not contain spaces. The *initdepl* is the intended initial depletion before either the model run or further conditioning. If any of the *initdepl* values for any SAU is < 1.0 then, once the equilibrium zone is achieved it will undergo a depletion event (see the help on the function *depleteSAU()*)

## 5.4.5 SIZE

The size structure range used in the example data files was originally selected to suit the Tasmanian blacklip abalone fishery. Hence it extends the midpoints of each 2mm size class from 2mm - 210mm, with the 210mm class being a plus group. These values will not necessarily make sense for other jurisdictions or species. *Haliotis roie*, for example, will require a much smaller upper size, and might possibly use 1mm size classes, depending on how data are collected.

*SIZE*

- *minc, 2, centre of minimum size class, classes are 1-3,3-5,5-7…, centered at 2,4,6,…*
- *cw, 2, class width mm*
- *Nclass, 105, number of size classes, leading to a maximum midpoint of 210mm*

These values lead to the dynamics being described within vectors and matrices ranging from 2 – 210 in 2mm size classes. Of course, each of these values can/should be altered to suit the biology of the species concerned. The derived *midpts* defining the size-classes, and the *Nclass* reside in the *glb* object.

## 5.4.6 RECRUIT

Larval movement between Tasmanian blacklip populations has been demonstrated to be remarkably low. Nevertheless, there will undoubtedly be some small amount of larval drift between populations, which operate on a smaller scale than the SAU within which they are found. With no generic estimates of this movement rate it is assumed that the populations are generally aligned linearly along the coastline and such larval movement is a small constant rate.

A linear arrangement along the coast is obviously an approximation but one should attempt to arrange the SAU and component populations in this manner or else the movement matrix will need to become rather more complex (which remains a possibility, but in the absence of data one can only wish you luck).

*RECRUIT*

*larvdisp, 0.01, rate of larval dispersal eg 0.01 = 0.5 percent of recruits in each direction*

The value of *larvdisp* is used to construct a simple larval movement matrix that assumes movement only occurs between adjacent populations at a rate of half the *larvdisp* value. While a value of 0.01 may appear almost trivial this does affect the equilibrium dynamics, but the effects are generally only minor. However, by including this it is now possible to determine the explicit effect of different levels of larval movement, and this would enable the importance of the relative isolation of the different species to be explored. The movement matrix is contained with the ‘globals’ object *glb*, where it can be tabulated and visualized.

### 5.4.7 RANDOM

There will often be a need to control the generation of random numbers used during both the definition of the populations and the projections. The use of a random seed is especially important when conditioning the operating model as many random numbers are used during that process and if the conditioning is intended to simulate a specific fishery, then the distribution of productivity needs to remain the same during each model run. The truth of this will become more apparent during the description of the *saudataEG.csv* file contents and uses in sections below.

The use of specific random seeds might be wanted, for example, if one wanted to be sure to repeat a particular analysis exactly, of perhaps to see only the effect of changing an argument, perhaps the LML, while making no other changes. So, to ensure the production of the same sequence of pseudo-random numbers, one uses *set.seed*. Under *RANDOM* we have two values, the first, *randomseed* is used for repeatability of population generation when conditioning the model. By default one would expect this to have a value. The second value, *randomseedP* is used for repeatability during the projections. Each scenario could have a different *randomseed* and, optionally, *randomseedP*, although one should think carefully about what one is comparing between scenarios when making this choice. Using the same seed for all scenarios compared ensures any differences seen are due the changes to the harvest strategy and are not due to the biological properties used. Ideally, one would conduct comparisons with the same initial *randomseed*. But would use both the same or different *randomseedP* for the projections so as to capture the fill range of variation. One can (should) use *getseed()* from **codeutils** to generate such seeds. It is recommended that one not try to generate one by oneself. The literature suggests that 12345 tends to be used more often than would be expected at random(!) so it, and others akin to it are just not good enough.

If one does not want to set the random seeds and just use any old random sequence that comes along then set *randomseed* and *randomseedP* to 0. By default, *randomseedP* is set to zero.

*RANDOM, Set these to zero for non-repeatable starting points.*

- *randomseed, 3543304, for repeatability of population definitions if >0*
- *randomseedP, 0, for repeatability of projections by simply continuing with the sequence started by the randomseed value, if a different randomseedP is used, then a new set sequence of pseudo-random numbers will occur, or if set to NA a random new sequence will occur each scenario run.*,

### 5.4.8 initLML

However the model is to be run (Figure 5.1), all scenarios begin by generating an unfished, equilibrium model. This includes a characterization of the equilibrium productivity of each population and this will require a description of selectivity to be used, hence we need an initial LML. The fishery productivity is altered when the LML is changed because it alters the balance between natural mortality, growth, and fishing mortality. Smaller LML do not always generate the highest levels of productivity (this becomes a yield-per-recruit problem, although that also, of course, depends upon the biological properties of growth).

*initLML, 140, used to generate unfished zone if no historical catches and LML present*

If the conditioning is to include the application of historical catches the different SAU may still require an initial depletion (which can be set at 1.0) and an initial LML. If conditioning

on historical catches, then it makes sense to initiate the model at the LML in use at the start of the historical catches. If a generic MSE is being conducted, for example, a generic west coast Tasmania starting projections in 2020 or 2021, then perhaps initiate the model at an LML of 140, although, as in the file generated by *ctrlfiletemplate()*, project the dynamics forward from 2020 with an LML of 145 (see PROJLML later).

### 5.4.9 PROJECT

Basic information about the projection period of the simulations. Elsewhere we add together the projection years (here *pyrs*=30) and the historical years, *hyrs*, value (see CATCHES) to obtain the total number of years of the final projections. These values are found in the *glb* object, along with the year names for each sequence of years (as in *hyrnames* and *pyrnames*).

*PROJECT, 30, number of projection years for each simulation*

### 5.4.10 ENVIRON

The entries under the ENVIRON heading are designed to allow the introduction of exceptional events into specific years during the projections. For example, the incidence of marine heat wave events is increasing and are of such a degree that their impacts can influence subsequent recruitment and the survivorship of settled animals. If one wants to introduce such an event in a given year, or years, then define how many years after the ENVIRON header. Here we will introduce values that relate to two years of events, happening in the fifth and eighth projection years. In each case, the proportion of recruits in that year that survive is listed for each sau. Because there are two years the software expects two rows (beginning with 'proprec'). If only one year, then there would be only one 'proprec' line (and so on). Following those lines are the survivorships of the settled animals, in this case, 99 percent are given as surviving, so the effect is primarily through the recruitment. Note that setting *ENVIRON* to zero means any environmentally induced effects are set to *NULL* and will have no effect. For more details and examples, see the *Perturbation within Projections* chapter.

*ENVIRON, 0, in how many years will an event occur*, 0 means have no events
*eyr, 5, 8, which projection years will have an event*
*proprec1, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, one for each sau*
*proprec2, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, one for each sau*
*propNt1, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99,*
*propNt2, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99,*

The passage of any ENVIRON objects through aMSE follows a simple path:

In *inputfiles.R* readctrlfile -> envimpact -> glb ->

In *projection.R* doprojections ->

In *dynamics.R* envimpact is used by both [oneyearcat & oneyearsauC]

The eyr determines which years have an impact.

If the year is in the eyr vector, then:

- in sauyearsauC, sigR is reduced by a factor of 10 and the value of the stock recruitment curve is multiplied by proprec, thus reducing the basline recruitment.
- in oneyearcat, the input numbers-at-size for that year is multiplied by the proportion surviving the environmental effect: inNt=(inN[,popn] * survP).

### 5.4.11 PROJLML

Under the *PROJECT* heading we read in a line for each of the projection years. Assuming that PROJECT > 0, then *readctrlfile()* looks for the heading *PROJLML*. The LML used in the projections are listed here explicitly so that changes through time can be easily implemented. There needs to be at least as many defined as the value of *PROJECT* (which becomes *pyrs*, see *str1(glb)*), here that is 30. The projLML are used to define the selectivity for each of the *pyrs* during the projections. Tasmania introduced a west coast LML = 150 in 2024 so that change should be included in future projections, though obviously it has been omitted from simulations made before 2024. The LML for the projections are included in the *projC* object.

*PROJLML, need at least the same number as there are projection years*

*2020,145, the year and the Legal Minimum Length (LML, MLL, MLS) e.g. 140*
*2021,145,*
*2022,145,*
*2023,145,*
*...*
*...*
*2050,145,*

### 5.4.12 CATCHES

If *CATCHES* is > 0 then one would need to include that number of years of historical catches by SAU along with the year and LML that was used when the catches were taken. An example is given below of the required format. If *CATCHES* = 0, then any data here will be ignored. If *CATCHES* > 0 then *readctrlfile()* looks for a heading *CondYears* and then reads in *CATCHES* rows with the following format. Here the catches are in tonnes.

*CATCHES,58, if > 1 then how many years in the histLML*

*CondYears,LML,6,7,8,9,10,11,12,13, column names for convenience only*
*1963,127,0,0,0,0,0,0,0,0,, a line of zeros for initial equilibrium state*
*1964,127,1,1,1,4,3,5,4,1, the year, the LML, the catches by SAU*
*1965,127,2,3,4,17,15,21,19,5,*
*...*
*...*
*1987,132,31,84,44,251,82,339,195,64,*
*...*
*...*
*2019,140,16,53,5,65,81,179,251,53,*
*2020,145,7,27,6,39,58,129,227,50,*

## 5.4.13 CEYRS

*CEYRS* is similar to *CATCHES* in that it defines how many years of cpue data will be read in. The required format is given below for *CEYRS* = 29.

IMPORTANT: If *CEYRS* = 0, then any cpue data will be ignored. Note that in the early years there appears to be cpue data missing from SAU6 and SAU13, which are present in the latest years. This is because, in Tasmania, zonation was only introduced in 2000, so cross-zone blocks such as 6 and 13 cannot be included prior to 2000.

*,6,7,8,9,10,11,12,13,* again, not used here just as labels

*CEYRS,29, if >0 then number reflects number of historical CPUE records by SAU*
*1992, ,113.4, 94.2, 97.0, 99.4, 98.1, 100.19, ,*
*1993, ,116.8, 110.38, 109.37, 99.1, 107.88, 102.20, ,*
*…*
*…*
*2018, 106.68, 143.57, 148.43, 133.70, 104.38, 95.01, 106.88, 104.08,*
*2019, 92.22, 112.70, 107.11, 93.65, 91.89, 86.94, 90.03, 90.78,*
*2020, 92.01, 113.01, 112.10, 99.10, 92.10, 87.10, 93.10, 92.10,*

## 5.4.14 SIZECOMP

Size-composition data is akin to the biological properties data in being potentially a large amount of data which might confuse the structure and editing of of the *control* file. Hence we use this to refer to a filename containing the required data. If *SIZECOMP* = 0 then no file will be read in, otherwise a list of filenames will be expected and these should be stored in the *rundir*. The filenames should begin in the row immediately below *SIZECOMP*.

*SIZECOMP,1*
*lf_90-20.csv*

The format of the size-composition .CSV data file should be the following, of course the years and size classes used should reflect your own data (see the output of *writecompdata()* for the full format):

length, sau, 1984, 1985, 1986, … , 2019, 2020
120, 7, 0,0,0, … , 0,0
122, 7, 0,0,0, … , 0,0
124, 7, 1,1,2, … , 0,0
126, 7, 1,3,4, … , 0,0
128, 7, 3,19,26, … 0,0
…, 7,
…, 7,
208, 7, 0,0,0, … , 0,0
210, 7, 0,0,0, … , 0,0

By using this format it is possible to use the function *getLFdata()* to read the data into the program. Try *?getLFdata*. These data are read in and returned to the program within the *condC* object.

## 5.4.15 RECDEV

Conditioning the data on historical catches and cpue will entail searching for an *AvRec* value that approximates the long-term productivity of the unfished stock. This should place the predicted cpue approximately through the observed cpue. However, one can expect to see deviations, some relatively large. After fitting numerous example models it becomes clear that average recruitment off the stock recruitment curve provides an inadequate description of the dynamics of any abalone fishery. It is possible to adjust the rise and fall of the predicted cpue by imposing recruitment deviates in particular years. Recruitment deviates are expected to take the form of Log-Normal variation, which vary around the value 1.0. If the model predicted recruitment matched the Beverton-Holt predicted recruitment exactly this would imply a recruitment deviate value = 1.0 as each average recruitment value is multiplied by the deviate. Thus, if recruitment is taken to be lower than the stock-recruitment curve would predict, then the deviate would be less than 1.0 and if recruitment was greater than expected the deviate would be greater than 1.0. These are best fitted iniitally using the **sizemod** R package but if this fails through, for example, inadequate contrast in available data, then cruder alternatives exist within **aMSE** (see *Conditioning the Operating Model*).

IMPORTANT: By relying primarily on recruitment deviates to match the observed against the predicted cpue, and the observed vs the predicted size-composition data, we would be ignoring any extra non-fishing related mortality, such as marine heat-wave events or other environmentally driven mortality events. But as a first approximation it should be able to move each SAU closer to the observed state.

The format of the recruitment deviates is as follows:

RECDEV, 58
CondYears, 6,7,8,9,10,11,12,13 This line is here for convenience
1963, -1, -1, -1, -1, -1, -1 -1, -1, The -1 values mean ignore recruitment deviates
1964, -1, -1, -1, -1, -1, -1 -1, -1, The -1 values mean ignore recruitment deviates
1965, -1, -1, -1, -1, -1, -1 -1, -1, The -1 values mean ignore recruitment deviates
1966, -1, -1, -1, -1, -1, -1 -1, -1, The -1 values mean ignore recruitment deviates
…
…
1980,1.207,1.097,0.939,1.014,0.643,0.911,0.504,1.197, some SAU +ve some -ve
1981,1.083,0.140,0.500,1.547,0.591,0.503,0.066,0.645,
1982,0.922,0.915,1.130,0.994,0.790,0.995,0.807,1.067,
1983,1.019,0.933,1.129,1.114,0.887,0.923,0.784,1.360,
…
…

If a line, after the year value is >0 then values for the whole line MUST be given, even if they are merely set at 1, which implies the recruitment level should be taken off the mean of the stock recruitment relationship.

The values illustrated were derived from simple code that first searches for an optimum *AvRec* value, and then searches, sequentially through each SAU, for an optimum set of recruitment deviates between a given range of years. One needs to account for the expected time-lag in years between when a cohort might settle and when it then will begin to enter the fishery. The values included in the *ctrlfiletemplate()* function were obtained using **sizemod** to fit each sau's data to an array of parameters, including recruitment deviates.

## 5.5 The *saudataEG.csv* File

This file is identified in the *START* section of the *control* file. Like the *control* file the *saudataEG.csv* file can also start off with some optional descriptive text lines.

*SAU definitions listing Probability density function parameters for each variable*
*These are randomly allocated to each population except for the proportion of recruitment,*
*which is literally allocated down in the popREC section*

As many of these descriptive lines as required can be included. The start of input that is used by the MSE is signaled by the *SPATIAL* heading (remember use capital letters for the section headings where indicated and getting the spelling correct are important). The *saudataEG.csv* file contents are read in by the function *readsaudatafile()*, and each variable is identified and selected by its name (so type carefully or use the *datafiletemplate()* function to make a start). Despite this approach, reading the values is still split into groups that relate to their function. First there are lines describing the implied geographical spatial structure being simulated, then comes a matrix of properties for each *SAU*, following on from the *PDFs* heading.

Not all the input variables are used to define probability density functions. Thus, each of the *nsau*, *saupop*, *saunames* are defined explicitly after deciding on the geographical structure to impose. They need to be listed in the order in which they are expected to occur along the coast. Each population is a member of a given *SAU*, and here we are using the Tasmanian block number as the SAU index (and the *saunames*). Each SAU can have a different number of populations. These three *SPATIAL* inputs appear to be duplicates of those in the *controlEG.csv*, and they are, but are used for different purposes.

*SPATIAL*

*nsau, 8, number of spatial management units eg 6 or in TAS's case 8*
*saupop, 3, 3, 5, 7, 9, 9, 12, 8, number of populations per SAU in sequence*
*saunames, 6, 7, 8, 9, 10, 11, 12, 13, labels for each SAU; not used*

### 5.5.1 PDFs

*PDFs, 32,*

This line is read in to determine the number of variable values to be read (the rows to the matrix). The number of columns of the matrix are defined by the *nsau* defined above.

The precision with which this matrix is filled will constitute a large part of any conditioning. Approximate and plausible values will produce a generic MSE whereas, tuning the matrix to the biological properties of a particular zone, and especially particular SAU, as best one can, will generate a much more specific MSE, especially if that is followed up by conditioning the resulting modelled stock on historical fishery data. Thus, if one then included historical catches and other fishery data the result would be as realistic a modelled representation as one could get when using such a complex spatial structure and without an enormous amount of data (which no-one has).

Here we only imply a plausible and generic zone, as can be inferred by the repeated values (which will still lead to different values between populations because variability is included as each population is generated - variables prefixed s, are all large enough to influence a random number). If specific values for a given population are wanted then set the respective

variation value to 1e-08. All variables are sampled from a Normal distribution except for the average unfished recruitment, which strongly defines the upper productivity of each population, and is sampled from a log-normal distribution (see the function *definepops()*).

### 5.5.2 Growth

The parameters of the Inverse Logistic model describing growth (Haddon *et al*, 2008; Haddon & Mundy, 2016). Alternative growth functions will be implemented in the future, if desired, and will be selected in this section. In each case there is the main parameter (MaxDL, L50, L50inc, and SigMAx) followed immediately by the variation to be attributed to a given population's value (obtained through sampling from a normal distribution). If exact or specific values are wanted for the populations in a given *SAU*, then set the respective variation value to 0. Estimates of *MaxDL* and *L95* (*L50inc = L95 - L50*) can be obtained for a given *sau* by using **sizemod**.

*DLMax ,26, 30, 30, 34, 34, 34, 34, 29, maximum growth increment*
*sDLMax ,2, 2, 2, 2, 2, 2, 2, 2, variation of MaxDL within each SAU*
*L50 ,135.276, 135.276, 135.276, ..., ..., ..., ..., ..., Length at 50% MaxDL*
*sL50 ,2, 2, 2, 2, 2, 2, 2, 2, variation of L50*
*L50inc ,39, 39, 39, 39, 39, 39, 39, 39, L95 - L50 = delta =L50inc*
*sL50inc ,1.5, 1.5, 1.25, 1.25, 1.25, 1.25, 1.25, 1.25, variation of L50inc*
*SigMax ,3.3784, 3.3784, 3.3784, ..., ..., ..., ..., ..., max var around growth*
*sSigMax ,0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, var of SigMax*

### 5.5.3 LML

It is not impossible that different SAU operate under different LML, this allows for the generation of the equilibrium zone to reflect such variation if present. If, however, historical catches are available and are used, they are defined along with the LML used in each year and those LML are used in the *zone* definition instead. The entry here is only for situations where no historic data are used, but it still needs to be present under all circumstances.

*LML ,127, 127, 127, 127, 127, 127, 127, 127, Initial legal minimum length*

### 5.5.4 Weight-at-Size

The weight-at-size relationships for each population are defined using the standard equation:

$$W_{p,L} = a_p L_p^{b_p}$$

which defines the weight of animals in population $p$ at length $L$. For this we need both the $a_p$ and $b_p$ parameters.

It was found in Haddon *et al* (2013; see pages 208-209) that there was a power relationship between the $b_p$ parameter and its corresponding $a_p$ parameter. This was useful but lacked any intuitive sense. An examination of weight-at-size relationships within *sau* (see the appendix on _weight-at-size *vs_location*) found that within *sau* these relationships mainly had very similar values for the `$a$` parameter with minor variation in the `$b$` parameter. While that variation was low, it nevertheless led to important differences between populations. So

low variation is included in the exponent, `*b*` but only extremely low variation included for the `*a*` parameter.

*Wtb ,3.161963, 3.161963, 3.161963, ..., ..., ..., ..., ..., weight-at-length exponent*
*sWtb ,0.000148, 0.000148, 0.000148, ..., ..., ..., ..., var of Wtb*
*Wta ,6.4224e-05,5.62e-05, ..., ..., ..., ..., ..., ..., power curve intercept*
*sWta ,1e-08, 1e-08, ..., ..., ..., ..., ..., ..., variation of Wta*

### 5.5.5 Natural Mortality

This is a troublesome variable. Previous work has generally assumed a natural mortality rate of about 0.2, but this suggests that most animals would be close to senile by the age of 23 (only 1 percent would be expected to survive that long). However, very many very large animals have been found, despite their growth characteristics implying that to reach such sizes would take much longer than 23 years (given the very slow growth rates of large animals). This casts doubt on the notion of M = 0.2, which, if traced, stems from some rather limited, and potentially flawed (blocking respiratory pores with tags) tagging data from decades ago. As with the growth characteristics, tagging may negatively affect survivorship, this potentially affecting any early tagging study of natural mortality (biasing it high). Here we have selected 0.15 as the mean or base-case against which to compare the rest, but the uncertainty means that it would be worthwhile comparing the outcomes of any HS when M = c(0.1, 0.125, 0.15, 0.175, and possibly even 0.2).

*Me ,0.15, 0.15, 0.15, 0.15, 0.15, 0.15, 0.15, 0.15, Nat Mort*
*sMe ,0.005, 0.005, 0.005, 0.005, ..., ..., ..., ..., var of Me*

### 5.5.6 Recruitment

The average unfished recruitment is a major influence on the productivity of each SAU and population. During conditioning it is recommended that this be one of the first variables to adjust to set-up each population's unfished production. At very least it should be possible to determine the relative long-term production of each SAU, and this can be used as a guide to determine the relative productivity of each SAU, which should then be distributed among its constituent populations. If a time-series of GPS data-logger information is available (or potentially a series of surveys) this can be used to scale the relative productivity of each constituent population within each SAU (this is done in the *propREC* section of the *saudataEG.csv* file, as described below). The *AvRec* unfished recruitment ($R0$) value, in each case, is in linear-space for ease of use, but is commonly expressed as log(AvRec) in the model. This is defined using steepness as shown in the equations within the *zoneCOAST* section above. Estimates of *AvRec* for each *sau* can be obtained by using **sizemod**.

*AvRec,217500,410000,185000,905000,760000,1440000,1350000,355000,*
*sAvRec,0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, R0 variation*
*defsteep ,0.7, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7, 0.7, Beverton-Holt steepness*
*sdefsteep ,0.0125, 0.0125, 0.0125, 0.0125, ..., ..., ..., ...,*

### 5.5.7 Emergence

Emergence has been studied in Tasmania through an examination of the growth of encrusting animals and algae on the animal's shells. The cover by algae (even encrusting algae) being taken as evidence of emergence (exposure to light). This only becomes influential if the emergence curve overlaps with the selectivity curve, which may happen when the LML is low. If no emergence data is present, then just set values so that the emergence curve always runs at smaller sizes than the selectivity curves (see later). These values below are mostly invented (for now; from very limited data) but would influence availability when the LML was down at 127mm.

*L50C ,126.4222, 126.4222, 126.4222, ..., ... , length at 50% emergent*
*sL50C ,0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,*
*deltaC ,10, 10, 10, 10, 10, 10, 10, 10, length at 95% - 50% emergent*
*sdeltaC ,0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,*

### 5.5.8 CPUE

Earlier versions of **aMSE**, assumed linearity between cpue and exploitable biomass and a maximum cpue was used to designate or scale the predicted cpue to match nominal observed cpue levels. Now, however, the option (the very much preferred option) is to assume hyper-stability of cpue (a non-linear relationship between cpue and exploitable biomass) and this has made the next two parameters redundant. They are retained however, as the maximum cpue that would be obtained in an unfished population can be stored in this variable once it is calculated. The values contained in this variable are no longer used but nsau values are required in each row in all cases. The variation can be set to zero (again, no longer used).

*MaxCEpars, 0.4,0.425,0.45,0.475,0.45,0.45,0.375,0.3, max cpue t-hr*
*sMaxCEpars, 0, 0, 0, 0, 0, 0, 0, 0,*

### 5.5.9 Selectivity

Even though there is an LML, the precision with which abalone are taken with respect to the LML is not always perfect with some legal sized animals being left behind and occasionally some just sub-legal being taken (though these observations may be due to measuring error). Here we use a simple addition to the selectivity ogive used for each population. The *selL95p* parameter can be estimated within **sizemod**, and often has values between 4 - 6. The selectivity curves used in the modelling (which are a combination of selectivity and availability) are illustrated in the *Fishery* tab of the MSE output.

*selL50p ,0 ,0, 0, 0, 0, 0, 0, 0, L50 of selectivity, 0 = no bias*
*selL95p ,5, 5, 5, 5, 5, 5, 5, 5, L95 of selectivity*

### 5.5.10 Size-at-Maturity

It was found in Haddon *et al* (2013; see pages 210-211) that there were relationships between the two parameters used to define the logistic maturity-at-size relationships for different populations. A fixed parameter value of -16 for the $a_p$ parameter led to a range of plausible

values for the $b_p$ parameter. A better alternative is to use size-at-maturity data from each *sau* if it is available. If available it can be analysed using the **biology** package (the values included in the file created by the *datafiletemplate()* function derive from maturity samples analysed in that way.

*SaMa, -22.3,-22.3,-22.2,-22.1,-24.0,-15.2,-21.2,-22.1, maturity logistic a par*
*L50Mat, 98.8,98.8,116.3,116.7,122.9, 112.2,121.1,105.8, L50 for maturity b = -1/L50*
*sL50Mat, 3, 3, 3, 3, 3, 3, 3, 3,*

### 5.5.11 cpue Hyper-stability parameter

The *lambda* parameter in the equation

$$I_y = qest \times B_E^{\lambda}$$

can define a linear relationship between cpue and exploitable biomass only when *lambda* or $\lambda$ has a value of 1.0. If $\lambda < 1.0$, then that relationship curves down to generate hyper-stable cpue values. Hyper-stability of, at least Tasmanian blacklip abalone, has now been demonstrated using the GPS logger data, and so now, in Tasmania, the base-case being used assumes a $\lambda = 0.75$. No variation is currently included in that parameter. The *qest* parameter (estimated within **sizemod**, although it could be approximated by trial and error) is designed to scale the exploitable biomass (raised to the exponent $\lambda$) so that the resulting cpue matches the observed nominal scale. If $\lambda = 0.5$ this would be equivalent to using the square-root of the exploitable biomass, which is obviously a smaller number than the original, hence the scaling factor *qest* needs to adjust for that appropriately for each SAU.

lambda, 0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75 ,
qest, 4.75, 2.25, 5.17, 1.63, 1.27, 0.77, 0.61, 1.45,

## 5.6 propREC

The *propREC* section provides details of the distribution of the recruitment levels across the populations within each *SAU*. The three columns of values are the *SAU*, the *population* index, and its expected proportion of its respective *SAU* recruitment each year. These values need to be set manually during the conditioning and can be varied until the dynamics approximate the desired dynamics. In Tasmania, once again the GPS logger data were able to be used to identify within each *sau*, areas of persistent productivity, which were equated to the populations. These varied in area and in average yield, but provided a more objective way to define the population structure within each *sau* than simply ascribing different numbers of populations and proportions of recruitment.

*SAU,pop,propR*
*6, 1, 0.742, in SAU 6 about 1/6 of recruitment goes to pop 1*
*6, 2, 0.094, in SAU 6 most recruitment goes to pop2*
*6, 3, 0.163, in SAU 6 almost no recruitment goes to pop3*
*7, 4, 0.203, in SAU 7 a more even distribution of recruitment occurs*
*7, 5, 0.59*
*7, 6, 0.207*

*8, 7, 0.041*
*…*
*…*
*12, 48, 0.03*
*13, 49, 0.121*
*13, 50, 0.084*
*13, 51, 0.124, in SAU 13 there are 8 populations with variable levels*
*13, 52, 0.164*
*13, 53, 0.215*
*13, 54, 0.126*
*13, 55, 0.062*
*13, 56, 0.104*

That completes the current structures and formats of the *control.csv*, *saudata.csv*, and size-composition data files.

# 6. Conditioning the MSE with the sizemod Package

## 6.1 Introduction

A Management Strategy Evaluation can attempt to simulate a real-world fishery. Conditioning the operating model involves modelling the properties of the fished stock so that the dynamic behaviour of the operating model mimics, or is at least has strong similarities to, the observed dynamics of the real-world fishery. Where operating models assume no or only simple spatial structure, it is sometimes possible to use standard stock assessment modelling methods to fit the operating model to available data from the fishery. This would ensure that estimates of productivity and other aspects of how the stock responds to fishing pressure are as good as the available data permits. Fitting the operating model to available data would also provide estimates of uncertainty around key parameters, which could then be included in the MSE simulations to provide a relatively realistic reflection of how the stock could respond when using alternative harvest strategies to provide management advice through time.

In Australia (and elsewhere) abalone stocks tend to be made up of meta-populations with notoriously complex spatially structures, with each component population being mostly self-sustaining through localized recruitment processes meaning limited larval movement and with very limited to no effective movement of settled animals. One outcome of such spatial structure is that most abalone populations can be considered data-poor (Haddon et al., 2005; Orensanz et al., 2005; Parma et al., 2003). Parma et al. (2003) and Orensanz et al. (2005) discuss the notion of S-fisheries, which are generally spatially structured (meaning patchy with heterogeneous biological properties), often targeting species of lower economic importance, and potentially subject to serial depletion. Wilson et al, (2013) added to this list of properties, by referring to the common mismatch between scale of fishing, scale of reporting, and scale of management (lots of S's, hence S-fisheries). Most importantly, for this discussion, such fisheries do not conform to the assumptions of the classical dynamic-pool notion of fish stocks. For a fishery to be a dynamic pool then some population event (be it a fishing event, a recruitment event, etc) will affect the whole stock within a relatively short period (often, at most, within a year). This requires there to be relatively high levels of mobility and mixing of individuals, or at least high levels of larval movement. Such an assumption may well be valid in scale-fisheries but is far less often true in fisheries for sessile, sub-tidal, hand-gathered species. Abalone cannot be considered to be of lower economic value but closely meet every other criterion for S-fisheries.

Given the complex spatial structured exhibited by Australian abalone fisheries it would be invalid to attempt a zone-wide assessment of a fishery's dynamics and expect such a thing to reflect the observed dynamics at smaller geographical scales. At best, it might be possible to conduct an assessment at the smallest spatial scale at which data from the fishery has been reliably collected. In the case of the Tasmanian west coast, that scale is that the level of statistical block (= *sau* or *SAU*).

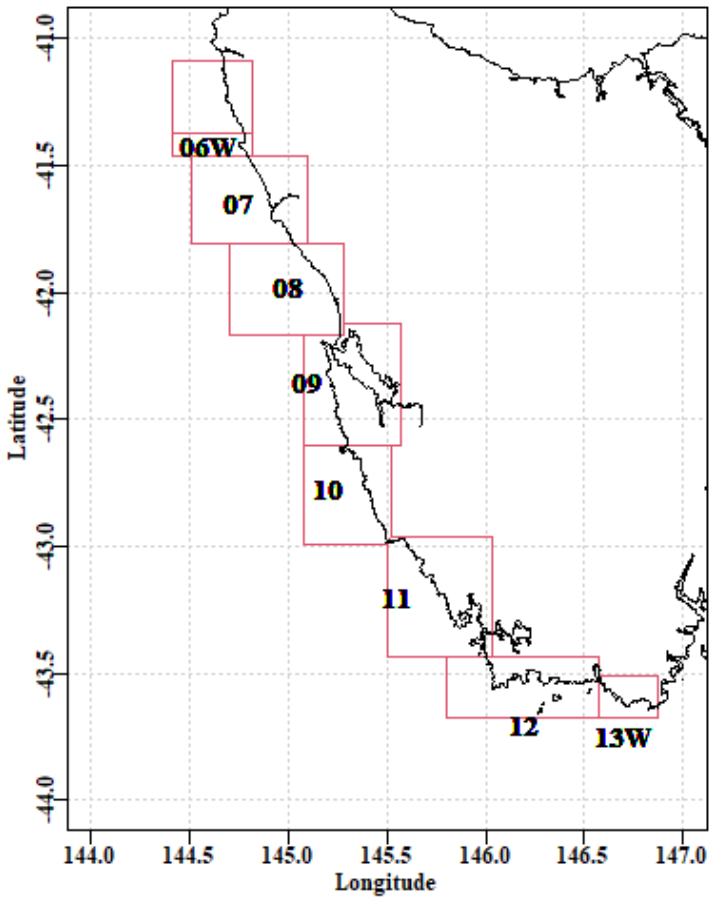


*Figure 6.1: A sketch map of the statistical blocks or sau in the current Tasmanian Western Quota Zone, which are included in the Tasmanian MSE. Block or SAU numbers are as labelled. sau6 and sau13 have a 'W' postfix because, surprisingly, they also have subblocks in other zones.*

Generally, high value species around the world are assessed using relatively complex stock assessment models with statistical integrated assessment models becoming more common (Maunder and Punt, 2013; Punt et al., 2013). High value species such as abalone are not often considered to be data-poor species, but there are many difficulties in collecting representative data from such a patchily distributed, highly variable, sub-tidal species. In the case of Tasmanian blacklip abalone (*Haliotis rubra*) data collection with respect to the biological properties of size-at-maturity, growth, and size-structure, has continued for decades. However, the extent and complexity of the Tasmanian coastline means that many areas and depths have no such biological samples at all. Spatial complexity and heterogeneity becomes a problem when attempting to condition the **aMSE** operating model to mimic the dynamics of an Australian abalone stock (see the appendix on *Maturity vs Location* for an example of small spatial scale biological heterogeneity).

### 6.1.1 What Models are Possible?

Surplus production models (Prager, 1994; Haddon, 2011, 2021) have been used with some success with catches and standardized CPUE data from some Tasmanian abalone statistical blocks. The SAU on the west coast with the most data (SAU 9 - 12) provided relatively

convincing model fits to the available data. The sparser data from SAU 6 - 8 generated less stable model fits, mainly because catches, and therefore number of records, were relatively low and the limited data were therefore more variable. The original idea for conditioning the Tasmanian MSE was to use such models to estimate the productivity of each block/SAU and continue improving the fit to observations from there. One problem with this strategy is that simple surplus production models assume that recruitment is also a simple and deterministic process. Unfortunately for this strategy, in Tasmania, abalone recruitment is sometimes extensive and far from average across whole quota zones and at other times appears patchy and variable in intensity. In the operating model, annual recruitment is modelled using a Beverton-Holt stock recruitment curve. Exploratory attempts to impose trial and error recruitment deviations from the modelled averages into the surplus production models, demonstrated that recruitment deviates were necessary for the predicted CPUE and predicted size-composition of the catch to approximate the observed data. However, attempting to introduce such artificial recruitment deviates was clumsy at best and often led to implausible model inputs at worst. Another disadvantage of using surplus production models is that they ignore what is known about the biological properties of the different areas (differing growth, etc.), and they cannot use any of the size-composition of catch data, which are now regularly collected, when fitting the model. These disadvantages are a problem because together they mean such models cannot provide strongly defensible estimates of any recruitment deviates that may have occurred.

Despite the serious issue regarding how representative of the multiple populations expected in any one SAU standard sampling can be, the use of a size-based integrated assessment model was explored to determine whether it could provide more plausible estimates of productivity and, more especially, of the expected recruitment dynamics during the known history of the fishery. An R package, called **sizemod** has been and continues to be developed to simplify the application of such a model; it is documented elsewhere and within the **sizemod** package. Here we will focus on different ways of using it and what the modelling results imply for conditioning *aMSE's* operating model.

## 6.2 Using sizemod as a Size-Structured Production Model

The R package **sizemod** contains a number of data files that allow for the illustration of how to use the software. One data set, *fish*, contains the fishery data, another, *sizecomp*, contains observed size-composition data from catches across a number of years, and another *setup* contains other details required to get a model run to work (see **sizemod** documentation, which also contains a formal description of model structure.).

```
require(sizemod)
data(fish)
data(sizecomp)
fiscomp <- NULL
omega=c(1,1,0,0) # include data streams for cpue and size-composition, no FIS
data(setup)
ctrl <- setup$ctrl
glb <- setup$glb
constants <- setup$constants
glb$maxage=50
glb$phase=1
glb$lambda <- 0.75 # CPUE and Exploitable biomass relationship now non-linear
# when first fitting SA model set glb$sigce = 0, so between year CPUE
```

```
if (glb$sigce == 0) {   # variability is first approximated
  glb$sigce <- getrmse(fish,invar="cpue",inyr="year",natlog=TRUE)$rmse
}
biol <- makebiology(glb$midpts,constants) # define biological properties
kable(biol[50:59,],digits=c(3,3,3,3))
```

*Table 6.1: Proportion mature, weight-at-length, proportion emerged, and maturity x weight at length for length classes 100 - 118 mm. The full range of sizes is 2 - 210 mm in Tasmania.*

| | mature | WtL | emergent | MatWt |
|---|---|---|---|---|
| 100 | 0.024 | 131.371 | 0.000 | 3.102 |
| 102 | 0.033 | 139.920 | 0.001 | 4.648 |
| 104 | 0.047 | 148.843 | 0.001 | 6.928 |
| 106 | 0.065 | 158.149 | 0.002 | 10.257 |
| 108 | 0.090 | 167.845 | 0.004 | 15.056 |
| 110 | 0.123 | 177.942 | 0.008 | 21.853 |
| 112 | 0.166 | 188.448 | 0.014 | 31.265 |
| 114 | 0.220 | 199.372 | 0.025 | 43.929 |
| 116 | 0.286 | 210.722 | 0.044 | 60.369 |
| 118 | 0.363 | 222.508 | 0.077 | 80.825 |

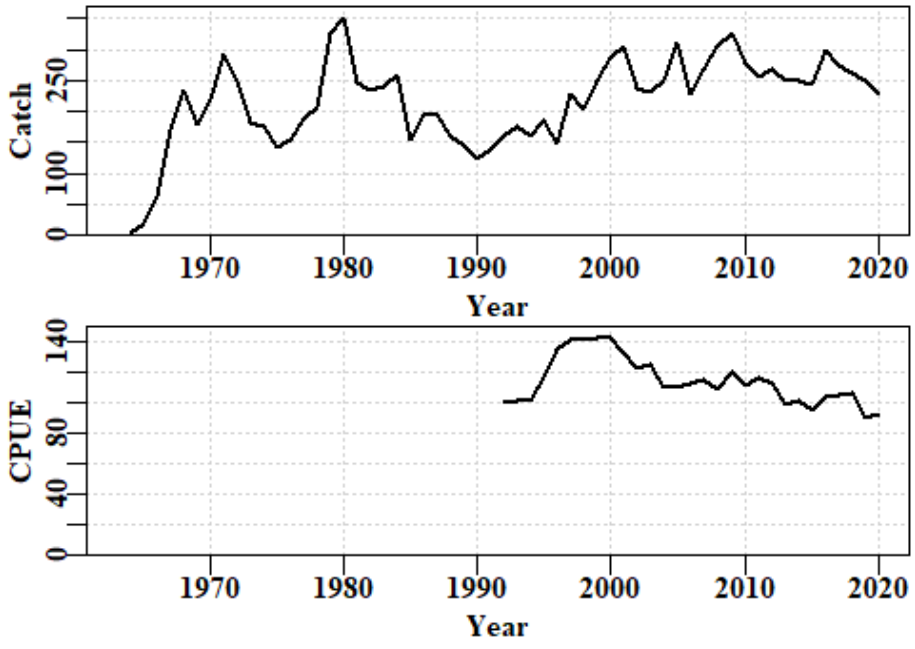


*Figure 6.2: Fishery history in SAU 12 in the western zone in Tasmania. Catches in tonnes, CPUE in kg/hr.*

The catch time-series exhibits some remarkable variation from year to year (see Figure 6.2). The 100t increase between 1999 and 2001 in SAU12 was a response to the introduction of quota zones, which allocated a TAC to the west of Tasmania with the aim of distributing dive effort more widely around the State. The CPUE time-series only starts in 1992 because prior to that the data are an unknown mixture of records by month, day, individual diver, and

collections of divers. A change in the reporting requirements in 1992 (daily by individual diver), and associated changes to the database helped solve those issues.

The size-composition data used was generally collected as samples of at least 100 individual measurements from individual landings by divers. Hence the early observations were often the outcome of sampling only very few landings. When combined with the 2mm size-classes in the assessment and operating models, this explains why the early data exhibits such spikiness. One great value of the size-composition data is that when it is compared with the theoretical unfished size-distribution of catches it provides a measure of the level of depletion imposed upon the populations within SAU 12. The size-composition data remains surprisingly noisy. For example, in 2019, the catches appear to be biased constistently high off the LML by about 5mm. Whether this was because divers had been instructed to try to ignore smaller animals for marketing reasons, or some other reason, is unknown. It suggests that time-blocking of any selectivity parameters might lead to improvements in model fit, although, once again, the question of whether the data are representative of an area as large as an SAU remains problematic (see Figure 6.1); the coastline of sau12 extends across about 0.4 of a degree of latitude and 0.5 degree of longitude.

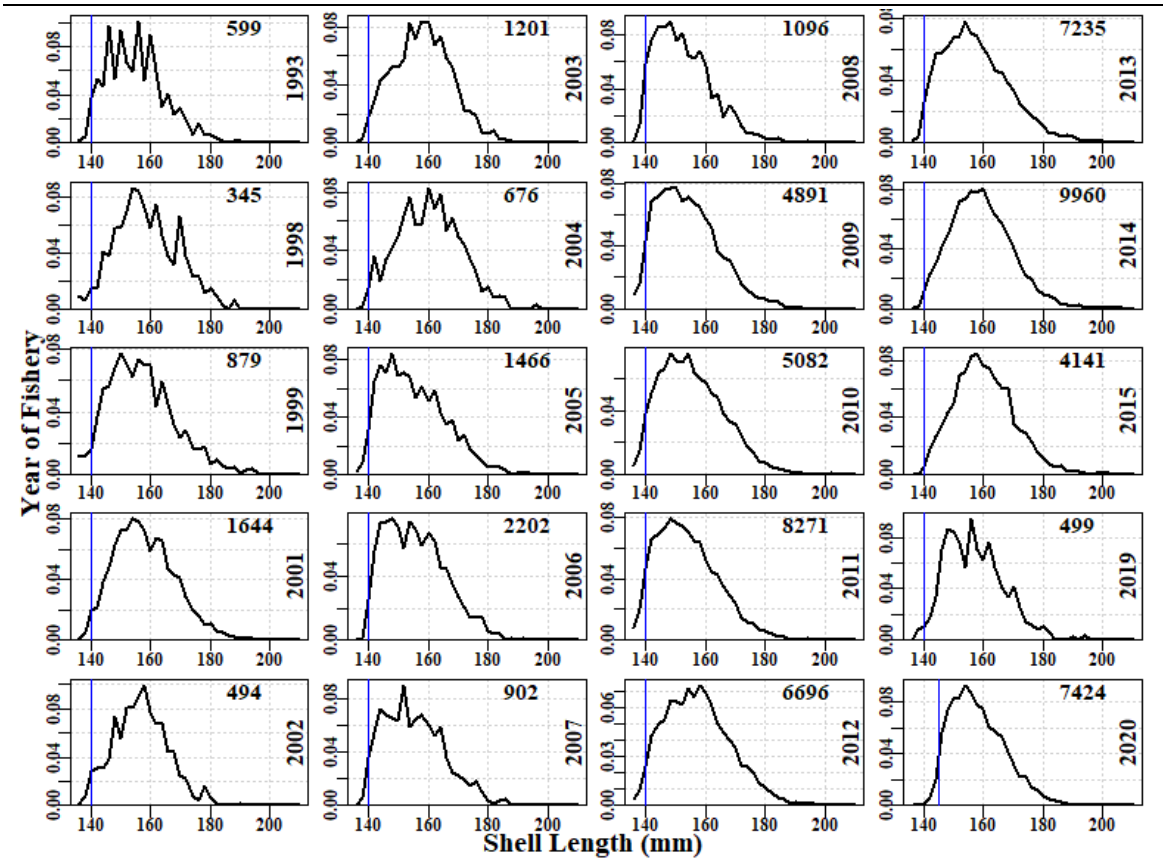


*Figure 6.3: The observed size-composition of catches sampled from 1990 - 2020, with 11 missing years. The sample sizes increased from 2007 onwards. The vertical blue lines are the LML in each year; note the increase in 2020.*

### 6.2.1 The Initial Parameters

The model is currently set up to estimate about 35 parameters (depending on how many recruitment deviates are included), all of which are log-transformed to help stabilize the estimation process. These include the unfished recruitment (*LnR0*), the *MaxDL* of the inverse logistic growth curve used (Haddon et al, 2008), the *L95* of the growth curve, the catchability (implemented to allow for a non-linear relationship between CPUE and exploitable biomass = hyperstability), and the difference between the 50% and 95% selectivity curve. In addition to these five primary parameters there are 30 recruitment deviates from 1985 - 2014. The number of deviates estimated depends upon the number of years of informative size-composition data there are available. Preliminary values for these are put into the model using a 35 x 2 array, with the second column containing a zero if the parameter is to be held constant and any number greater than zero (we use 1) if it is to be estimated by the model. By setting all the recruitment deviates to zero (back-transformed = 1.0), and setting the model not to estimate their value, it is possible to run **sizemod** as a size-structured surplus production model.

```
Difference in equilibrium population Numbers-at-size > 10

Difference in equilibrium population Numbers-at-size > 10
```

The initial parameters making up the *pindat* were obtained partly by examining the outcome of tagging estimates of growth in the area, and partly (mainly) by trial and error until the predicted CPUE at least approximated the shape of the observed CPUE (Figure 6.4).

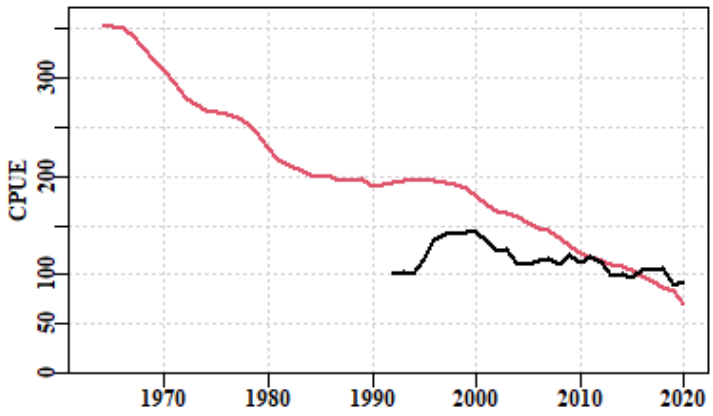


*Figure 6.4: Predicted CPUE (red, 1964 - 2020) relative to standardized observed CPUE (black, 1992 - 2020) when modelled using the initial parameter values before fitting the model using maximum likelihood.*

### 6.2.2 The 5-Parameter Model Fit

By searching manually for initial parameters that provide an approximate solution, as suggested by the two curves approaching each other (Figure 6.4), then formally fitting the model to the available data becomes more efficient (see appendix for all the code used).

```
starttime <- Sys.time()
  outmod <- fitlbm(pindat,negLLP,funk=dynamics,biol=biol,glb=glb,fish=fish,
                   constants=constants,omega=omega,sizecomp=sizecomp,
                   fiscomp=fiscomp,both=TRUE,tol=1e-06,initH=0)

initial  value 2565.279326

iter  10 value 1591.668939
iter  20 value 1527.720011
final  value 1527.056570
converged
  0:     1527.0566:  14.9410  3.16484  5.13042 -1.23772 -5.06285
Difference in equilibrium population Numbers-at-size > 10

  fitpar <- outmod$ans2$par
  optpar <- allpin
  optpar[modin$notfix] <- fitpar
  neglogL <-
negLLP(pars=fitpar,initpar=optpar,funk=dynamics,biol=biol,glb=glb,
                   fish=fish,constants=constants,notfixed=modin$notfix,
                   omega=omega,finalcomp=sizecomp,fiscomp=fiscomp,full=TRUE)

Difference in equilibrium population Numbers-at-size > 10

  likelihoods <- cbind(oldlogL,neglogL,(abs(oldlogL - neglogL)))
  print(round(likelihoods,4))

               oldlogL     neglogL
LLce         1163.1697     67.6058 1095.5638
compL        1401.0268   1457.8080   56.7812
penaltyR        0.0000      0.0000    0.0000
sigmaCE         0.0337      0.0337    0.0000
wtsc            0.0070      0.0070    0.0000
penLnR0         0.3574      0.9994    0.6420
penDL           0.0000      0.0370    0.0370
pencatch        0.0000      0.0000    0.0000
penH            0.7255      0.6053    0.1202
penaltyCE       0.0000      0.0000    0.0000
totalL       2565.2793   1527.0556 1038.2238
sad          1020.8780    288.1041  732.7739
sadce        1014.4832    283.4456  731.0376
sadcomp         6.3948      4.6585    1.7363
sigmaR          0.5000      0.5000    0.0000
lambda          0.7500      0.7500    0.0000
steep           0.7000      0.7000    0.0000
M               0.1500      0.1500    0.0000
LLsc       200720.4761 208855.3306 8134.8546
LLfis           0.0000      0.0000    0.0000
LLfissc         0.0000      0.0000    0.0000
wtfis           0.0070      0.0070    0.0000
fiscompL        0.0000      0.0000    0.0000
limLnR0        16.0000     16.0000    0.0000
maxDL          32.0000     32.0000    0.0000
minDL          23.0000     23.0000    0.0000
```

```
  pindat[modin$notfix,"param"] <- fitpar
endtime <- Sys.time()
print(endtime - starttime)

Time difference of 2.026953 secs
```

With only five parameters the model fit is generally very quick. Later, when we fit recruitment residuals the solution can take more time to be found. The improvement in model fit between the start and the 5-parameter fit is apparent both in the reduced total negative log-likelihood from 6462 to 2701, but in the reductions to both the log-likelihoods for CPUE (LLce) and the composition data (compL). When this model fit is plotted in full the improvement in fit to CPUE is also visible, although overall it remains poor, appearing to be a one-way decline.

```
Difference in equilibrium population Numbers-at-size > 10
```

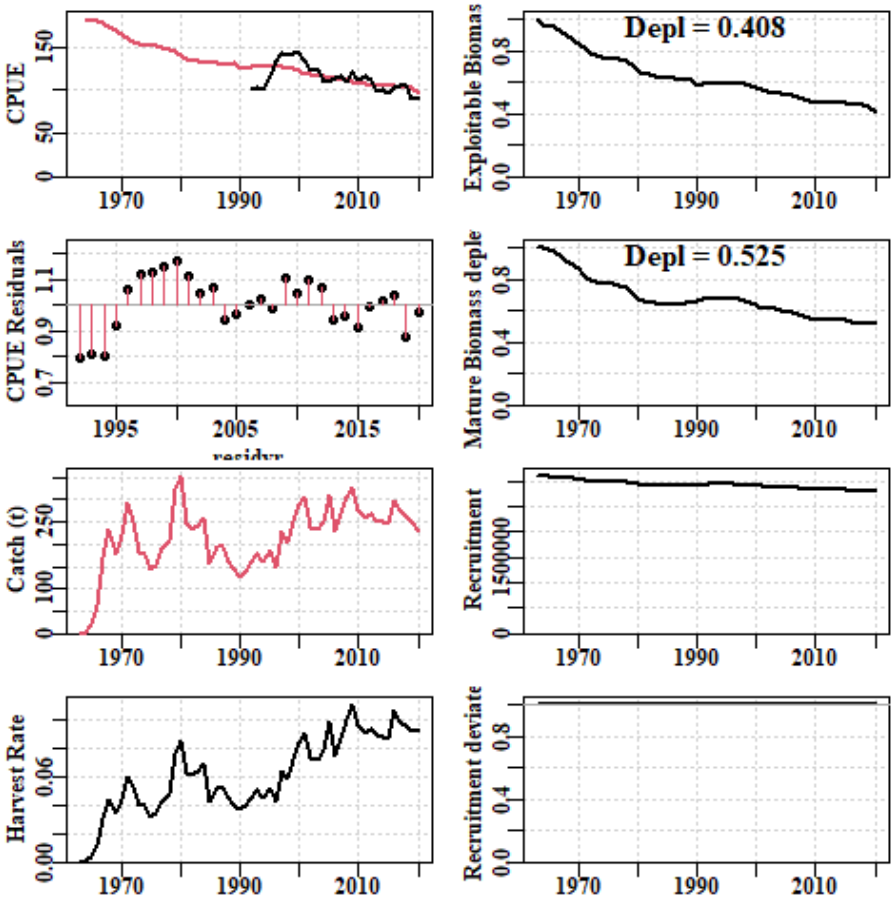


*Figure 6.5: Summary plots for the fit to the model when only fitting the first five parameters and treating sizemod as a size-structured surplus-production model.*

While the model fit will automatically provide an estimate of current depletion for both the mature biomass (important for recruitment dynamics) and exploitable biomass (important for the estimate of CPUE), it should be noted that the final predicted CPUE is well above the observed CPUE (Figure 6.5). The trends in the residuals also indicate that despite the improved model fit there remain some significant biases that demonstrate a serious model misspecification. The model has done its best to fit to the data, but the assumption of constant recruitment (bottom right panel) does not provide for sufficient flexibility for the modelled dynamics to be able to fit closely to the data.

The fit to the size-composition data also has issues, although it is remarkably good in some years, in particular 2012, 2013, 2020 (Figure 6.6). Surprisingly, even the earlier years from 1994 to 2006, where sample sizes were relatively small, the fits that were produced are clearly approximating the distributions despite the noisiness of the data. Notice that some years, when the data were first considered (2014, 2015, and 2019), appear unusual in that the rising edge of the commercial data is right shifted away from the LML.

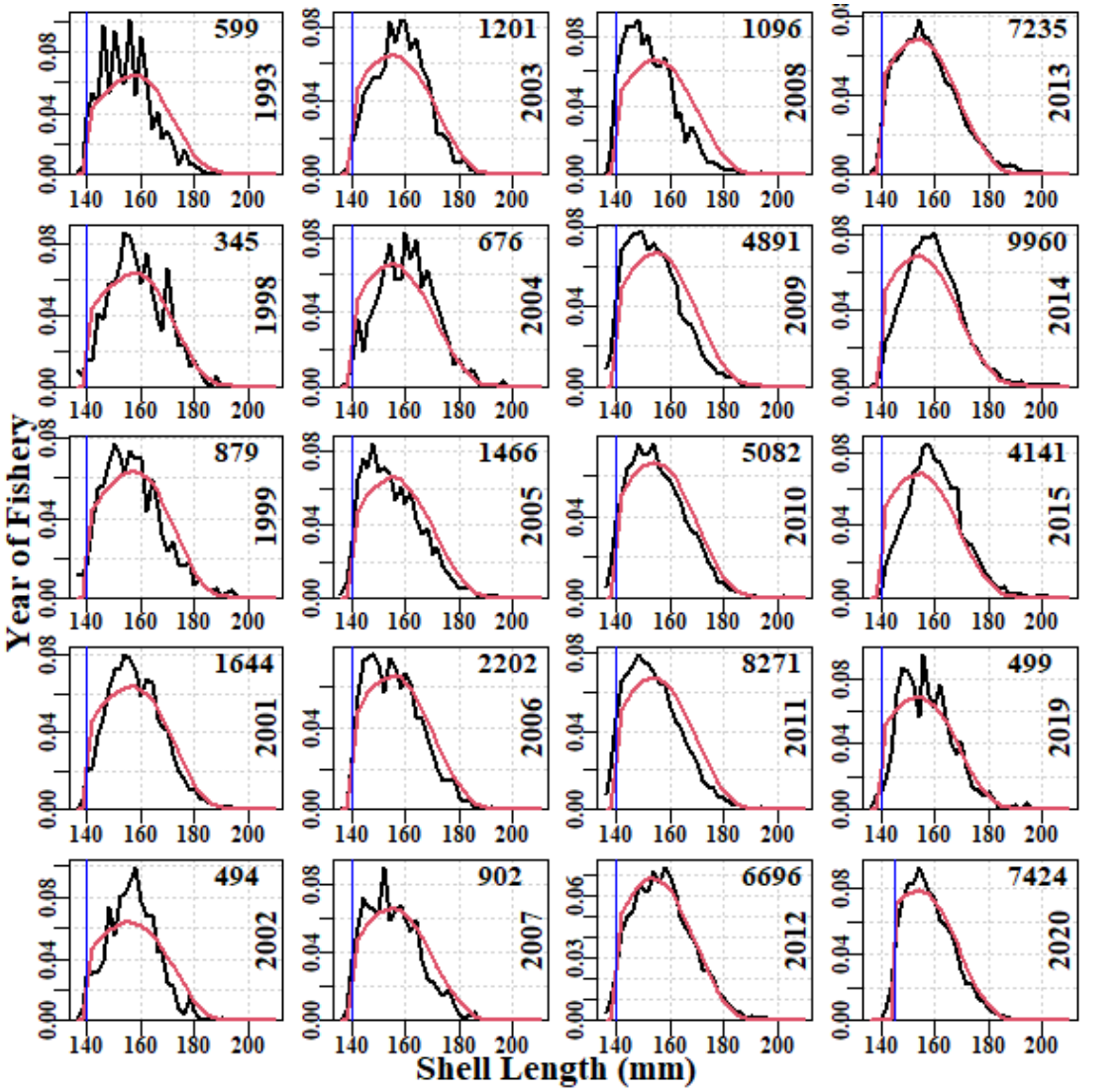


*Figure 6.6: Using sizemod as a size-structured surplus-production model generates plausible model fits to some of the year*

*of size-composition data.*

Importantly, the descending edge of the observed and fitted size-composition distribution, when compared to the theoretical unfished distribution contributes evidence towards estimating the depletion in each year (assuming selectivity is not dome-shaped; the model implemented logistic selectivity).

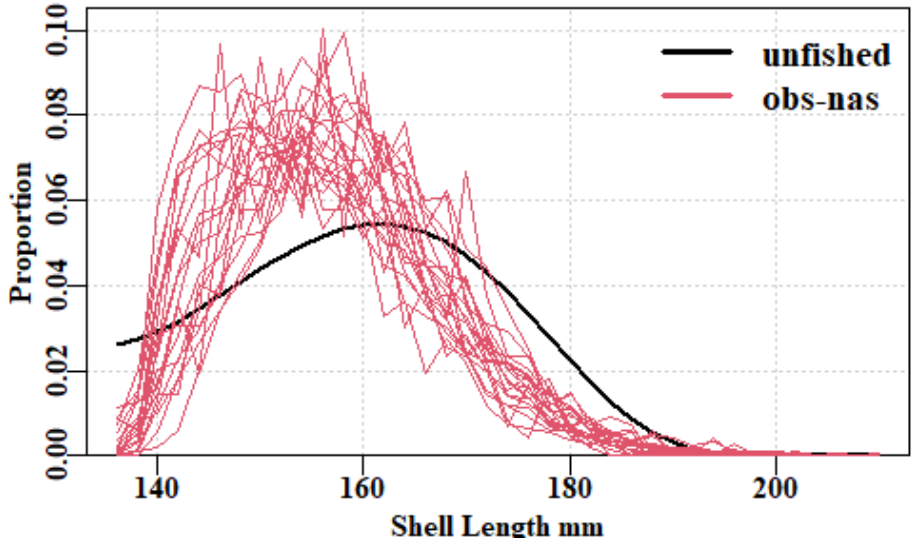


*Figure 6.7: A comparison of all years of observed size-composition of catch data with the predicted unfished equilibrium size-distribution of the population. The left-ward shift of the mode of the observed data away from the mode of the unfished population is indicative of depletion. The lines have different heights because the proportions across each curve all sum to equal 1.0, 'obs-nas' is the observed numbers-at-size.*

## 6.3 The 35-Parameter Model Including Recruitment Deviates

To allow the estimation of recruitment deviates it is only necessary to alter the 'phase' column in *pindat* from zero to 1. This could be done in sub-groups gradually, if problems arose during the fitting process. Alternatively, if the model has problems converging on a biologically plausible solutions then one can manually adjust some of the recruitment deviates time-lagged prior to increases in CPUE to improve the initial fit of predicted CPUE to that observed. This approach is sometimes needed to aid in obtaining a final plausible model fit. However, for data from statistical block 12, model fitting was able to proceed directly to estimating 35 parameters. With this many parameters, each iteration in the minimization takes longer, though now that **Rcpp** routines have been included to speed the calculations the whole process is much faster than the original R only code.

```
pindat[6:35,"phase"] <- 1
pindat[6:35,"param"] <- 0.1
pindat[1:5,"param"] <- c(14.25,3.29,5.2,-0.46,1.36)
```

Now, when the model is refit to the data it will alter all those zeros to adjust the recruitment in each of those years (1985 - 2014) and thereby improve the fit to the CPUE and size-composition data.

```
modin <- getpin(pindat)
pin <- modin$pin
notfix <- modin$notfix  # in this case all parameters are not fixed
allpin <- modin$allpin
allpin[notfix] <- pin
oldlogL <- negLLP(pars=pin,funk=dynamics,initpar=modin$allpin,
                  biol=biol,glb=glb,fish=fish,constants=constants,
                  notfixed=modin$notfix,omega=omega,finalcomp=sizecomp,
                  fiscomp=fiscomp,full=TRUE)

Difference in equilibrium population Numbers-at-size > 10

starttime <- Sys.time()
  outmod2 <- fitlbm(pindat,negLLP,funk=dynamics,biol=biol,glb=glb,fish=fish,
                    constants=constants,omega=omega,sizecomp=sizecomp,
                    fiscomp=fiscomp,both=TRUE)

initial  value 2845.113909

iter  10 value 1407.251819
iter  20 value 1354.454309
iter  30 value 1333.216319
iter  40 value 1328.034588
iter  50 value 1327.222530
iter  60 value 1326.524064
iter  70 value 1325.738847
iter  80 value 1325.237478
iter  90 value 1324.613119
iter 100 value 1323.847725
iter 110 value 1323.340747
iter 120 value 1323.224617
iter 130 value 1323.127476
iter 140 value 1323.085858
```

```
iter 150 value 1323.082623
iter 150 value 1323.082623
iter 150 value 1323.082622
final  value 1323.082622
converged
  0:     1323.0826:  14.0987  3.33405  5.18416 -0.433344  1.40928
0.0578526 0.129311 0.227778 -0.254128 0.773467 0.805066 0.445180 0.200832
0.423153 0.533578 0.0481762 -0.140197 0.435316 -1.27799 0.513452 0.395607
0.363445 0.116582 0.819583 0.0648155 0.595222 0.117573 0.0616644 0.0915990
-0.267316 0.796522 0.328776 0.501531 -1.10073 0.890477
 25:     1323.0799:  14.0982  3.33385  5.18448 -0.433050  1.41080
0.0593352 0.132020 0.233038 -0.250369 0.770941 0.808362 0.444856 0.202573
0.422683 0.533153 0.0479880 -0.140003 0.436237 -1.27846 0.515117 0.395481
0.363972 0.118817 0.820425 0.0653228 0.595210 0.116981 0.0615609 0.0929899
-0.265075 0.795719 0.332357 0.500808 -1.09839 0.890727
 50:     1323.0613:  14.0769  3.33441  5.18561 -0.425872  1.41083 0.252045
0.267807 0.233927 -0.190501 0.793396 0.841358 0.445018 0.234640 0.426804
0.551504 0.0671789 -0.124130 0.455780 -1.28758 0.533297 0.410752 0.376703
0.139459 0.833419 0.0852341 0.608918 0.135467 0.0779464 0.107331 -0.236060
0.802577 0.364037 0.500774 -0.989233 0.894203
Difference in equilibrium population Numbers-at-size > 10

  fitpar <- outmod2$ans2$par
  optpar2 <- fitpar
  neglogL <- negLLP(pars=fitpar,initpar=optpar2,funk=dynamics,biol=biol,
                    glb=glb,fish=fish,constants=constants,
                    notfixed=modin$notfix,omega=omega,finalcomp=sizecomp,
                    fiscomp=fiscomp,full=TRUE)

Difference in equilibrium population Numbers-at-size > 10

  likelihoods <- cbind(oldlogL,neglogL,(abs(oldlogL - neglogL)))
  print(round(likelihoods[1:12,],4))

            oldlogL   neglogL
LLce      1427.5735  -68.0738 1495.6473
compL     1411.6218 1390.3194   21.3024
penaltyR     0.0000    0.0000    0.0000
sigmaCE      0.0337    0.0337    0.0000
wtsc         0.0070    0.0070    0.0000
penLnR0      0.2821    0.0937    0.1884
penDL        0.0000    0.0000    0.0000
pencatch     0.0000    0.0000    0.0000
penH         5.6366    0.7218    4.9148
penaltyCE    0.0000    0.0000    0.0000
totalL    2845.1139 1323.0610 1522.0529
sad        659.8509   34.2540  625.5969

  pindat[modin$notfix,"param"] <- fitpar
endtime <- Sys.time()
print(endtime - starttime)

Time difference of 50.60049 secs
```

The negative log-likelihoods have improved in that the *LLce* for CPUE has decreased from 67.606 in the 5-parameter fit to -68.074 and the *compL* for the size-composition data has decreased from 1458 down to 1390. When the data streams are completely consistent with each other and if the recruitment penalty has been omitted, the minimizer would have kept working until the fit to the CPUE and size-composition can be as perfect as possible. Not surprisingly, the resulting dynamics are rather different from that produced by the size-structured surplus-production model version. Other details of the model fit, such as the various penalties and limits are explained in the documentation to **sizemod**. **sizemod** includes a bias-ramp on the recruitment deviates, uses the Francis (2013) weighting on the cpue index of relative abundance, and iterative re-weighting of the relative weight given to the size-composition data. Once again, these details are described in detail and explained in **sizemod**'s documentation; using the built-in data sets these details have already been attended to in the example.

```
Difference in equilibrium population Numbers-at-size > 10
```

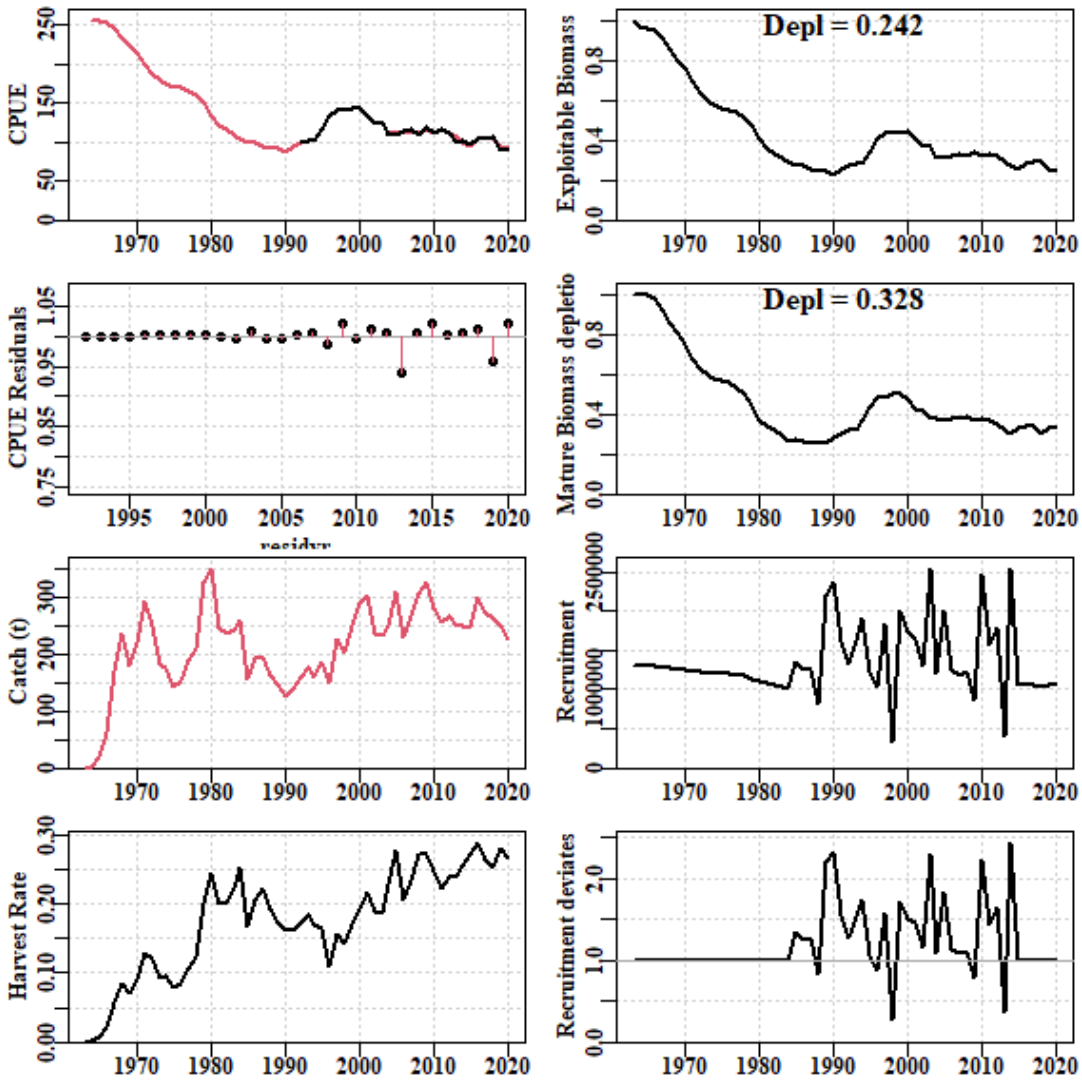


*Figure 6.8: Summary plots for the fit to the full model fitting all 35 parameters. The fit to the cpue appears perfect though the residual plot demonstrates that differences are still present.*

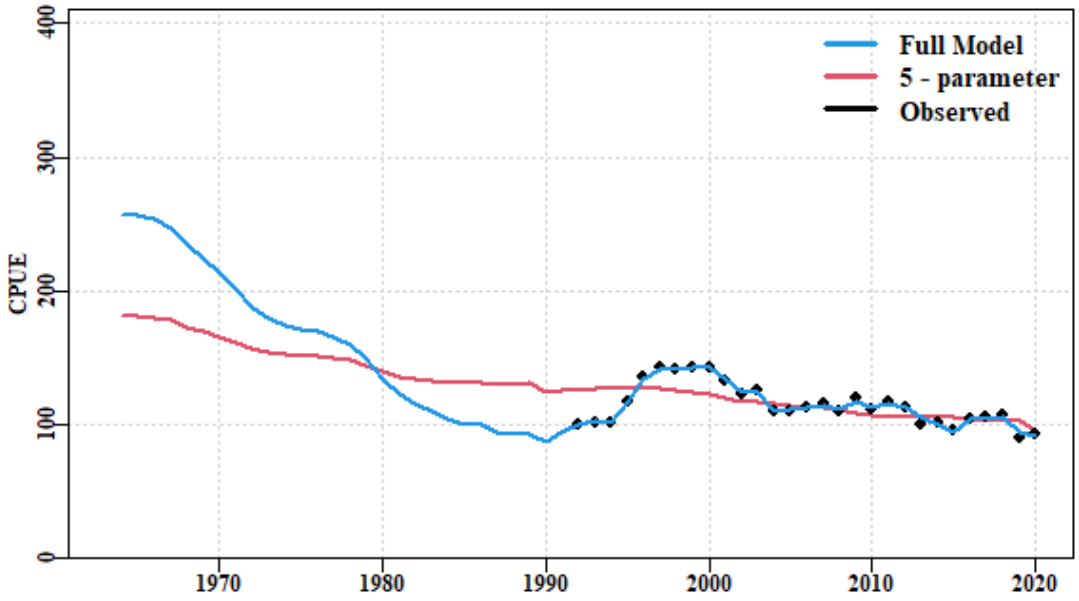


*Figure 6.9: Fitted CPUE from the 5-parameter size-based surplus-production version, and the full model compared. Note the greater depletion level implied by the full model.*

Comparing the model fits to the size-composition data (Figure 6.10) illustrates that the differences are more subtle than with the CPUE. Nevertheless, clear improvements are exhibited by the modified expected numbers-at-size in the catch. There are numerous improvements in the fits to the earlier noisy years of data, whereas in the years with larger samples the improvements are seemingly negligible or more subtle. However, for example, in the 2011 data the fit to the descending limb of the distribution is a clear improvement, as is the increased height of the rising limb.

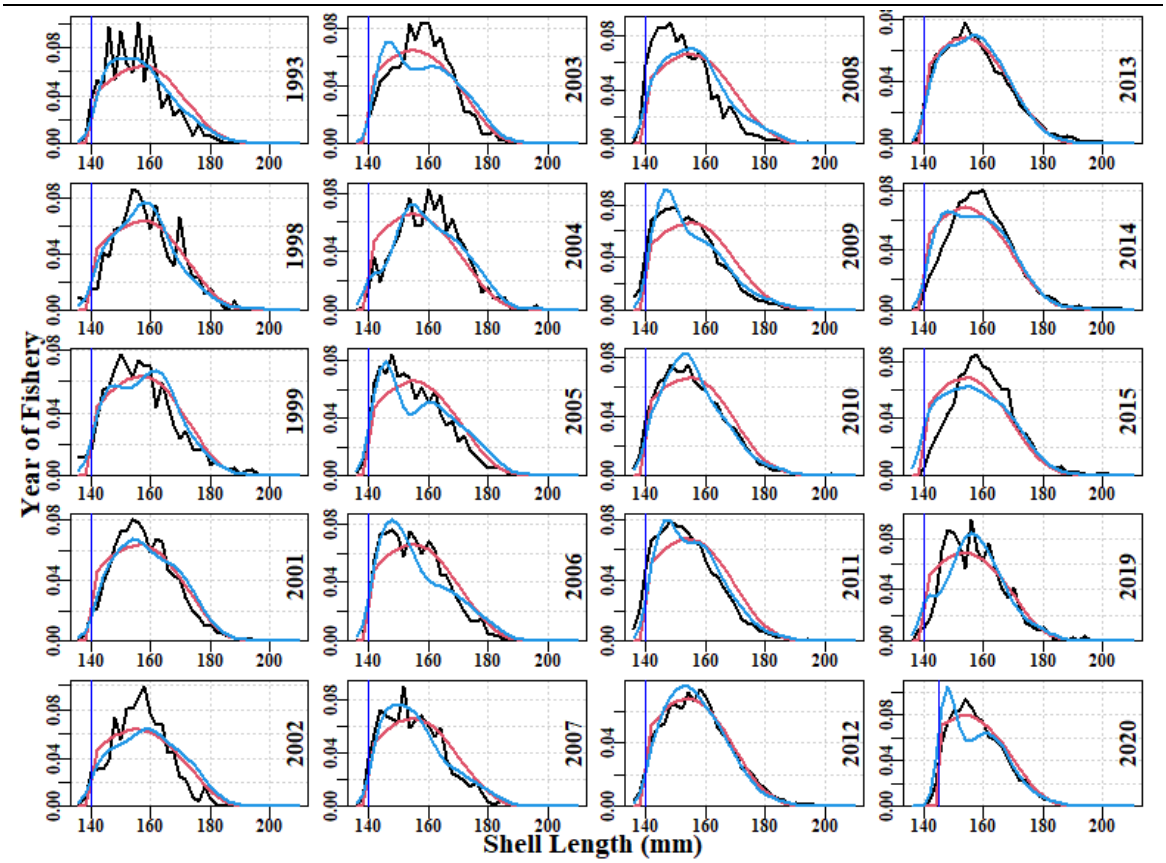


*Figure 6.10: Comparison of the size-composition of catch data fitted to the 5-parameter surplus-production version (red line), and the 35-parameter model (blue line).*

### 6.3.1 Non-Linearity of CPUE

The situation modelled in this scenario used a $\lambda = 0.75$. This may be understood by examining the following equation:

$$\widehat{I_t} = qB_t^{\lambda} \qquad (6.1)$$

where $\widehat{I_t}$ is the predicted CPUE in year $t$, $q$ is the catchability, $B_t$ is the exploitable biomass in year $t$, and $\lambda$ is a parameter that can alter the relationship between CPUE and exploitable biomass. In most stock assessments, $\lambda = 1$ is assumed, which asserts a linear relationship. By using $\lambda = 0.75$ the relationship between curvi-linear, which when illustrated clarifies the implications of the equation (Equation 6.1).

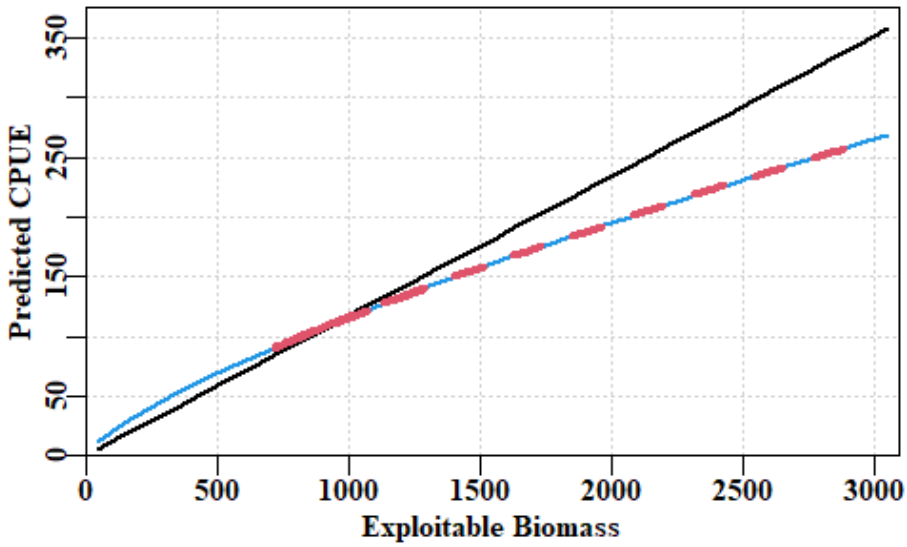


*Figure 6.11: The predicted relationship between CPUE and exploitable biomass when lambda = 0.75. The black line reflects how it would look if lambda = 1.0. Where the red line appears solid it reflects the range over which observed CPUE varies from 1992 - 2020, where it is dashed reflected where there is no usable CPUE data.*

Note that the maximum CPUE predicted when $\lambda = 0.75$ is 100kg/hr lower than if a linear relationship was used. This value appeared to divers to be more realistic when the fishery first started, as, at that time, handling time would have been much greater, and the risk of disturbing other abalone when removing one would mean many would lock down leading to damage if removal were attempted.

## 6.4 Initial Discussion

A classical, purely biomass based surplus production model can fit the abalone data remarkably well (Haddon, 2011, 2021), however, it only provides estimates of productivity and no information on the size-composition of the catch or the growth parameters. It also turns out to be relatively unstable for some of the statistical blocks and is extremely sensitive to variation in its parameters, especially the *r* parameter. The 5-parameter size-structured surplus production model has at least three advantages over the classical biomass-dynamic model:

1)by including the available size-composition data it can provide information concerning selectivity and growth as well as fishing mortality and final depletion (it can also inform about recruitment deviations but not in the 5-parameter model),

2)it can more easily model the dynamics that include the full history of catches, and

3)it is more stable when implementing non-linear relationships between the CPUE and exploitable biomass. Despite these advantages the observed fit to the CPUE data does not appear as good as that in the biomass-based surplus production model.

On the other hand, the 5-parameter model is also being fitted to the size-composition data so the model fit ends as a compromise between fitting the CPUE data and the size-composition data.

In its turn, the full 35-parameter model has an important advantage over the size-structured surplus-production model in that it can directly estimate the recruitment residuals required to adjust the model fits to both the CPUE (Figure 6.9) and the size-composition data (Figure 6.10).

Ignoring the recruitment deviates, the initial five parameter estimates also differ between the 5-parameter and the 35-parameter models:

*Table 6.2: Comparison of the five main parameters when fitting the size-based model.*

| | 5-param | 35-param |
|---|---|---|
| LnR0 | 3081586 | 1327381 |
| MaxDL | 23.685 | 28.052 |
| L95 | 169.088 | 178.423 |
| qest | 0.2900 | 0.6483 |
| seldelta | 0.006 | 4.093 |

Much of the variation in CPUE is accounted for by the implementation of recruitment deviates in the 35-parameter model. Hence the differences between parameter values appear relatively large as they reflect differences in how productivity is expressed (more due to growth and less due to the average unfished recruitment in the 35-parameter model). Such changes were required to account for the changed dynamics when recruitment deviates were included. In fact, given these are numerical solutions, it is possible, if the initial parameter values are varied, to obtain essentially identical final model fits where the total likelihood differs at the second or third decimal place (of the order of a 10,000th of a percent difference). It is the case that, given the highly variable size-composition data and the highly variable catch data from year to year, a good deal of uncertainty can be expressed in the model outputs.

There are also some strong assumptions included in the assessment that are associated with the constant values given to some important parameters. These strong assumptions include that the natural mortality = 0.15, that the steepness of the Beverton-Holt stock recruitment relationship was 0.7, and that the lambda parameter, describing the non-linearity between CPUE and exploitable biomass, was 0.75. These values were fixed because for abalone they remain unknown or, at best, estimated for very small samples from few populations. Varying these parameters does not affect the model fit very much but does alter the implied unfished CPUE. In summary, the mathematical optimum model fit is different from the optimum biologically plausible model fit (initial catch rates staying within plausible values). The three assumed values appear to provide for the best compromise between optimizing both the statistical fit and the most biologically plausible set of implied population dynamics.

It has been demonstrated that it is possible, for a statistical block in Tasmania, to use the size-structured integrated assessment model to estimate the block's productivity, the selectivity characteristics, the growth characteristics, and a series of recruitment deviates used to help describe the history of dynamics for which data are available. The uncertainty over what values to attribute to natural morality ($M$), recruitment steepness ($h$), and the non-linearity parameter ($\lambda$) remains a problem. Nevertheless, the use of the size-based integrated

assessment to characterize the dynamics within each SAU is a great improvement over the simpler surplus production models.

## 6.5 Final Adjustments once in aMSE

After transferring the parameters from the sizemod estimates into each SAU within the aMSE operating model further adjustments are required to the AvRec and recruitment deviates to optimize the fits between the predicted CPUE at the SAU level and the observed standardized CPUE as well as optimizing the fits between the predicted size-composition of catch and those observed. This is done within **aMSE** using the two functions *adjustavrec()* and *optimizerecdevs()* (see their individual help pages within **aMSE** for their arguments and the syntax for how to use them.

The differences between the predicted MSY and AvRec for **sizemod** and **aMSE** is such that for each SAU in the Tasmanian western zone the MSY in aMSE is about 9.5% less than the sizemod estimate for each SAU, and the AvRec is about 3.8% larger in aMSE, although that relationship is much more variable between SAU. These differences between the **sizemod** and the **aMSE** values are a result of the dynamics in aMSE being split between each SAU's populations rather than a single dynamic-pool within each SAU and the scale of the differences alters with the number of populations used within an SAU.

# 7. MSE Operating Model Structure

## 7.1 Model Dynamics

The operating model generates the population dynamics for a simulated abalone zone by following the numbers-at-size through time of each of the component populations within each of the sau. It does this by including how each is affected by natural mortality, somatic growth, fishing mortality, and recruitment. The model developed in Haddon *et al*. (2013) and Haddon and Helidoniotis (2013), and further developed in Haddon & Mundy (2016), used separate vectors of numbers-at-size to describe the cryptic $N_t^C$ and emergent $N_t^E$ components of each population. While this can be considered as a more realistic representation of nature further testing demonstrated it was relatively inefficient. Here the model is somewhat simplified through the cryptic and emergent components of the population being contained in the single vector $N_t$, where $N_t$ is assumed to be the numbers-at-size (shell length in mm) at the start of each year $t$. This simplifies the equations and helps speed the calculations although with this approach the effect of emergence needs to be included explicitly in some of the equations describing the dynamics (see the *Selectivity* section below).

Being based upon difference equations, the model structure adopted to describe the assumed annual dynamics, begins at the start of each year and involves a number of steps: 1) half of the survivorship from natural mortality being applied first, 20 this is followed by individual growth, 3) then survivorship from fishing mortality (if fishing occurs), 4) followed by the remaining survivorship from natural mortality, 5) finally, each population $p$, will give rise to $R_p$ recruits in year $t$, and if any larval dispersal occurs (described by the movement matrix *Phi*, $\mathbf{\Phi}$), would lead to the re-distribution of a small proportion of those recruits among the populations (Miller *et al*, 2009), and they are then added to the first size class of each population vector, $p$, at the end of year $t$, $\mathbf{N_{p,t}}$, which is equivalent to be ing the start of hte following year. If natural mortality is implemented as half natural mortality, that is $S_h = e^{-M_p/2}$, twice a year, with other dynamics between the natural mortality events then the dynamics for the numbers-at-size can be represented in matrix notation (which is read right to left) as:

$$\mathbf{N_p^{t+1}} = \mathbf{\Phi R_p} + S_p^h \mathbf{A_p G_p} S_p^h \mathbf{N_p^t} \qquad (7.1)$$

where $S_p^h$ is the survivorship of population $p$ following half of the instantaneous natural mortality in each population, $M_p$ (some small variation between populations is assumed), $\mathbf{A_p}$ is the survivorship following the imposition of any fishing mortality occurring in population $p$ (which is implemented as vector multiplication with the vector result of $\mathbf{G_p} S_p^h \mathbf{N}_{p,t}$), $\mathbf{G_p}$ is the growth transition matrix for population $p$, and $\mathbf{\Phi R_p}$ is the vector of recruits, $\mathbf{R_p}$, from each population multiplied by the movement matrix $\mathbf{\Phi}$ among populations, and then added into each size-class within $\mathbf{N_{p,t+1}}$ (the same as if it had been added at the end of the year, $\mathbf{N_{p,t}}$; because the end of each year is the same as the start of the next.

The survivorship following the fishing mortality rate over a year is defined as the complement of an annual harvest rate **A**:

$$A_{p,L} = \left(1 - s_{p,L,t} H_{p,t}\right) \qquad (7.2)$$

where $A_{p,L}$ is the survivorship of length class $L$ for population $p$, $s_{p,L,t}$ is the selectivity of length class $L$ in year $t$, and $H_{p,t}$ is the fully selected harvest rate in year $t$ for population $p$ (the harvest rate being the proportion of exploitable biomass taken as catch). We explicitly use exploitable biomass because the standard use of a legal minimum length (LML) in abalone fisheries can lead to the exploitable biomass being very different from the mature biomass.

an alternative view of the survivorship would be:

$$\mathbf{A_{p,t}} = e^{-\mathbf{s_{p,t}} F_{p,t}} \qquad (7.3)$$

where $\mathbf{s_{p,t}}$ is the vector of selectivity-at-length (or size). Strictly, selectivity is assumed to be equal for all populations within a zone (although as selectivity is combined with emergence, which varies by population, the $p$ subscript is also required; see below), and $F_{p,t}$ is the fully selected, instantaneous fishing mortality rate for population $p$ in year $t$. This simplification means that now the transition from cryptic and emergent no longer needs to be included in the annual dynamics. However, it does require that selectivity is now a combination of selectivity-by-diver and Emergence, which is only influential on the final selectivity if the logistic used to describe emergence overlaps the legal minimum length (which is quite possible in some areas of the western zone where the size at maturity can be relatively large and in the early years of the fishery in Tasmania and the LML was only 127mm).

### 7.1.1 Model Initiation

Model initiation will always begin with each population being assumed to be at equilibrium in the absence of fishing. At equilibrium, $\mathbf{N}^*$, the absence of fishing mortality implies that survivorship from the annual harvest rate equals one, $\mathbf{A = 1.0}$, which can therefore be omitted from the initiation):

$$\mathbf{N_p^*} = \mathbf{\Phi R_p} + \mathbf{S_p^h G S_p^h N_p^*}$$

If it is assumed that there is no larval movement, $\mathbf{\Phi = I}$, the Unit matrix, then that matrix can also be ignored (for the time-being) in the dynamics, which can be re-arranged to obtain an analytic expression for the equilibrium numbers-at-length $\mathbf{N_p^*}$ (Sullivan *et al*, 1990):

$$\mathbf{N}_p^* - S_p^h \mathbf{G_p} S_p^h \mathbf{N_p^*} = \mathbf{N}^* \left(\mathbf{I} - S_p^h \mathbf{G} S_p^h\right) = \mathbf{R_p}$$

which, finally, implies (Sullivan *et al*, 1990):

$$\mathbf{N_p^*} = \left(\mathbf{I} - \mathbf{S_p^h G S_p^h}\right)^{-1} \mathbf{R_p} \qquad (7.4)$$

If, however, there is even a minor degree of larval dispersal, and for blacklip abalone in Tasmania a value of 0.5 percent (0.005) between populations is plausible, then the analytical solution is no longer valid. In practice, within **aMSE**, the R-package from this project (R Core Team, 2024; Haddon, 2024), the analytic solution is used to obtain the starting point for an iterative application of the unfished dynamics until an equilibrium is obtained (see the help for the function *testequil*).

### 7.1.2 Initial Depletion

By definition the model is initiated at an unfished equilibrium. However, having the complete fishery history is not a luxury afforded to every fishery so there will be instances where prior to conditioning the mdoel prior to applying the known historical catches, all or at least some *SAU* may already be depleted to different degrees. The level of such depletion may be suggested by the application of the **sizemod** size-based integrated assessment program to each SAU (see chapter on *Conditioning the MSE with the sizemod Package*). Alternatively, alternative levels may be applied in a hypothetical manner to determine their influence.

If an initial depletion is required this is input in the control file under the ZONE section in the *initdepl* vector, with a value for each SAU (even if it equals 1.0, meaning no initial depletion). This preliminary depletion is conducted within the *depleteSAU()* **aMSE** function. If the *initdepl* value for an SAU is < 1.0, then this uses a simple trial and error search for the harvest rate that depletes each SAU to a level closest to the value in *initdepl* and uses that value to deplete the populations within the SAU so the final depletion is as close as can be to the required value. Only then does the conditioning (fitting the operating model to the available observations) occur. The dynamics for applying any initial depletion level is set to use the selectivity that is reported for the first year of observational data.

## 7.2 Biology and Stock Related Statistics

### 7.2.1 Emergence

A logistic curve (Haddon, 2011, 2021) can be used to describe the transition from the cryptic to the emergent component of the population, but this could only become influential on the dynamics if natural mortality differs between the two components or if the emergence logistic overlaps with the selectivity curve, and or the Legal Minimum Length (LML).

$$E_{p,L} = \frac{1}{1 + exp\left(-log(19)\left(L - L_{E_p50}\right) / \left(L_{E_p95} - L_{E_p50}\right)\right)}$$
$$E_{p,L} = \frac{1}{1 + exp\left(-log(19)\left(L - L_{E_p50}\right) / \delta_p\right)} \qquad (7.5)$$

where $E_{p,L}$ is the proportion of size-class $L$ that are emergent in population $p$, and $L_{E_p50}$ and $L_{E_p95}$ are the usual logistic parameters defining the lengths at which 50% and 95% are emergent in population $p$. The term $\delta_p$ is the constant $\left(L_{E_p95} - L_{E_p50}\right)$. Emergence from crypsis only becomes an issue for the dynamics of the model if they are considered to have different natural mortality rates within crypsis and/or when the emergence curve overlaps with the selectivity curve (which it can do when the LML is low, e.g. 127mm early on in Tasmania, especially on the west coast). Where the selectivity and emergence curves overlap then the proportion remaining in crypsis would act as a refuge, effectively reducing the fishing mortality on those size classes.

### 7.2.2 Selectivity

Selectivity-by-diver is assumed to be equal across populations within a zone, but a $p$ subscript could be added if different LML were expressed in different parts of a zone (this

might be the case if different LML were imposed in different parts of the same zone, which can occur in Tasmania!). In the operating model Selectivity, $s_{L,t}$ for length $L$ in year $t$ needs to be defined by year to permit changes in the LML to be reflected in the selectivity by divers. This implies that rather than a single vector of values, a matrix of selectivity values will be required, one column per year. Each year's selectivity is defined using a lower-case $s$ to distinguish it from Survivorship:

$$s_{L,t} = \frac{1}{1 + exp[-log(19)(L - L_s50)/\delta_s]} \qquad (7.6)$$

where $\delta_s = L_s95 - L_s50$.

Strictly, in case the emergence curve overlaps the selectivity curve, to define selectivity we should multiply the selectivity-at-length by the emergence-at-length (element x element, or Hadamard, multiplication). That way, if there is overlap it will alter the selectivity appropriately, and if there is no overlap then selectivity will not be affected. Importantly, because emergence varies by population, this means that the selectivity expressed would not necessarily be the same across all populations.

$$\begin{gathered} s_{p,L,t} = s_{L,t} \times E_{p,L} \\ \mathbf{s_{p,t}} = \mathbf{s_t} \otimes \mathbf{E_p} \end{gathered} \qquad (7.7)$$

### 7.2.3 Growth

The growth from size-class $j$ to size-class $i$ is described by the elements of a growth transition matrix defined by:

$$\begin{aligned} G_{p,i,j} &= \int_{-\infty}^{L_i+\frac{LW}{2}} \frac{1}{\sqrt{2\pi}\sigma_{p,j}} \exp\left(-\left[\frac{L_i - \bar{L}_j}{2\left(\sigma_{p,j}\right)^2}\right]\right) dL \quad L_i = L_{Min} \\ G_{p,i,j} &= \int_{L_i-\frac{LW}{2}}^{L_i+\frac{LW}{2}} \frac{1}{\sqrt{2\pi}\sigma_{p,j}} \exp\left(-\left[\frac{L_i - \bar{L}_j}{2\left(\sigma_{p,j}\right)^2}\right]\right) dL \quad L_{Min} < L_i \leq L_{Max} \end{aligned} \qquad (7.8)$$

where $G_{p,i,j}$ is the probability of growing from size class $j$ into size class $i$ in population $p$, $LW$ is the size-class width, $\sigma_{p,j}$ is the standard deviation of the normal curve describing the growth increments of animals starting in size class $j$ in population $p$, $L_i$ is the length of size class $i$, and $\bar{L}_j$ is the mean growth increment of animals starting from the mean of size-class $j$. $L_{Min}$ and $L_{Max}$ are the minimum and maximum size-classes, with the maximum being treated as a plus group. To ensure that all columns sum to 1.0 (to prevent growth implying losses or gains of its own), and to make $L_{Max}$ a plus group, the final row of the matrix is modified for each column $j$ as:

$$G_{p,L_{Max,j}} = G_{p,L_{Max,j}} + \left(1 - \sum_{i=L_1}^{L_{Max}} G_{p,i,j}\right) \qquad (7.9)$$

The expected mean growth increment from size-class $j$ (so it grows into size class $i$) is defined using an inverse logistic growth curve that has been found to describe blacklip abalone growth well (Haddon et al. 2008; Helidoniotis et al., 2011):

$$\bar{L}_{p,i,j} = L_{p,j} + \frac{Max\Delta L_p}{1 + exp\left[log(19)\left(L_{p,j} - L_{p,j}50\right)/\delta_{p,j}\right]} + \varepsilon_{L_{p,j}}$$

$Max\Delta L_p$ is the maximum growth increment for the population $p$, $L_{p,j}50$ and $L_{p,j}95$ are the usual logistic parameters defining the initial lengths at which 50% and 5% of the maximum growth increment are expressed ($\delta_{p,j}$ is simply the 95% minus the 50% parameters). Note that the $log(19)$ is positive, which inverts the logistic curve (compare with the equation for emergence; Equation 7.5). The $\varepsilon_{L_{p,j}}$ is the variation around the mean expected growth increment. It is assumed to be normally distributed with a standard deviation that varies with the growth increment (Haddon et al. 2008):

$$\sigma_{L_{p,j}} = \frac{Max\sigma_{L_p}}{1 + exp\left[log(19)\left(L_{p,j} - L_{p,g}95\right)/\delta_{p,j,\sigma}\right]} \qquad (7.10)$$

The $\left(L_{Max} - L_{p,g}95\right)$ remains a constant and can be parameterized as such ($\delta_{p,g,\sigma}$). Alternative descriptions of this variation, such as the use of a power law (Haddon, 2021), may be explored for their impact. Additional growth curve description options could be included in the R package **aMSE** (Haddon, 2025) if requested by users.

An important influence on the growth increment estimates is the negative bias that appears to be introduced by the tagging methods used in their estimation (See Haddon *et al*, 2013, page 191, Figure 70). This means that when conditioning the abalone operating model, it will be necessary to compare the predicted unfished numbers-at-size with those obtained from the fisheries catch-at-size. In that earlier work, after using the best estimates of growth increment from tagging, the predicted size-distribution of the unfished population had a smaller average size than the observed frequencies in the catches from a non-pristine fishery. Clearly modifications to the growth parameters were required and options are discussed in the growth section of the *Conditioning_the_MSE* chapter.

### 7.2.4 Weight-at-Length

The weight-at-length, $W_L$, relationship involves two constants:

$$W_{p,L} = a_p L_p^{b_p} \qquad (7.11)$$

During the conditioning of the model it is possible that an observed power relationship that exists between the two parameters can be used instead of estimating both (see Haddon *et al*, 2013, page 209, Figure 75). Using this has the advantage that any correlation between the two parameters is maintained even when random pairs are used.

### 7.2.5 Maturity-at-Length

Maturity at size, $m_{p,L}$, uses an alternative logistic curve, again with two parameters, $\alpha$ and $\beta$, only this time $L_{m_p 50} = -\alpha_p/\beta_p$ and the inter-quartile distance is $2log3\beta_p = 2.197225\beta_p$.

$$m_{p,L} = \frac{exp(\alpha_p + \beta_p L)}{1 + exp(\alpha_p + \beta_p L)} = \frac{1}{1 + \left(exp(\alpha_p + \beta_p L)\right)^{-1}} \qquad (7.12)$$

such curves are best fitted to observed data using a Generalized Linear Model that uses binomial residual errors (see the associated R package **biology**).

### 7.2.6 Spawning and Exploitable Biomass

Mature or spawning biomass needs to include numbers-at-size by maturity-at-size and weight-at-size. The operating model (OM) operates at the population scale but the 'predicted' data from the OM needs to be at the SAU level. Hence, the outputs from each population need to be combined in a valid manner (see later under *Sampling*). Whatever the case for sampling, we will still need population-based estimates of such things as exploitable biomass, numbers-at-size, and so on, so methods for combining the appropriate populations into a single SAU are required. Some, such as numbers-at-size, will need simple summation while others, such as cpue, might require catch-weighted values. A population-based estimate of spawning biomass at time $t$, $B^S_{p,t}$, can be obtained through:

$$B^S_{p,t} = \sum_{L=L_{Min}}^{L_{Max}} \left(m_{p,L} W_{p,L} N_{p,L,t}\right) \qquad (7.13)$$

Spawning biomass is, like exploitable biomass, calculated in the same units as the $W_{p,L}$ equation (dependent upon what parameter values are used). If that is in grams then it requires division by 1e6 to estimate tonnes, if in kg then division by 1000 is required. Exploitable biomass, when used to calculate CPUE, is estimated after half of natural mortality and growth have occurred and before any fishing mortality occurs in any single year. Remember that the selectivity for each population, $s_{p,L,t}$ includes any effects of Emergence:

$$B^E_{p,t} = \sum_{L=L_{Min}}^{L_{Max}} s_{p,L,t} W_{p,L} G_{p,L,j} e^{-M_p/2} N_{p,L,t} \qquad (7.14)$$

where the exploitable numbers-at-size $L$ in year $t$, $N^E_{p,L,t}$, is obtained from:

$$N^E_{p,L,t} = G_{p,L,j} e^{-M_p/2} s_{p,L,t} N_{p,L,t} \qquad (7.15)$$

where $N_{p,L,t}$ is the numbers-at-size $L$ at the start of year $t$ for population $p$.

For internal consistency, however, exploitable biomass is also reported as start-of-year values as well as mid-year values, and so is simply the sum across size-classes of the final numbers-at-size at the end of the previous year by the selectivity times the weight-at-length:

$$B^E_{p,t+1} = \sum_{L=L_{Min}}^{L_{Max}} s_{p,L,t} W_{p,L} N_{p,L,t} \qquad (7.16)$$

where $B^E_{p,t+1}$ is the start-of year exploitable biomass (equals the end of year exploitable biomass).

### 7.2.7 Catchability and CPUE

At least in the Tasmanian blacklip abalone fishery, the relationship between catch-rates and catches is usually observed to be linear, hence the relation between catches and effort is also linear. Because of this catch-rates are assumed to have at least some influence over the distribution of catches among areas. Observed catch-rates (cpue) would naturally be expected to be variable through time and across areas and so are modelled as:

$$I_{t,p} = q_p \left(B^E_{p,t}\right)^{\lambda} e^{N(0,\sigma_q)}$$

where $I_{t,p}$ is the cpue in year $t$ for area $p$, $q_p$ is the catchability coefficient within population $p$. $B^E_{p,t}$ is the mid-year exploitable biomass in year $t$ and area $p$, with a non-linearity coefficient of $\lambda$, and $e^{N(0,\sigma_q)}$ is a Log-Normal random deviate, with $\sigma_p$ being the standard deviation of the catchability coefficient $q_p$. If $\lambda = 1.0$ then the relationship between cpue and exploitable biomass is linear, values other than $\lambda = 1.0$ lead to non-linear relationships, which can make rather a large difference to the $q_p$ value. Given how important the use of cpue is in all Australian abalone harvest strategies, this is one assumption whose influence is in dire need of testing. We are using $p$ as a sub-script for population (or area of persistent production), because in the conditioning each population within each Spatial Assessment Unit corresponds to a particular area within that SAU. Effectively, area and population are the same thing in the model.

### 7.2.8 Annual Model Dynamics

Once each population is initiated its dynamics can be projected forwards a year at a time depending on how much catch is expected to be taken or how much effort is expected to be focussed into each population. The population initiation sets up the equilibrium numbers for the properties defined for each population. Then, given a specific harvest rate for each population they can be projected forward in yearly steps. This projection is based around how the numbers-at-size change through fishing, growth, natural mortality, and recruitment. As before, the fishing mortality rate over a year is defined as the complement of an annual harvest rate and is distributed down the diagonal of an otherwise zero matrix **A**:

$$A_{p,L} = \left(1 - s_{p,L,t} H_{p,t}\right)$$

where $A_{p,L}$ is the survivorship of length class $L$ for population $p$, $s_{p,L,t}$ is the selectivity of length class $L$ in year $t$, and $H_{p,t}$ is the fully selected harvest rate in year $t$ for population $p$ (the harvest rate being the proportion of exploitable biomass taken as catch). We can define the survivorship from applying half of natural mortality as follows:

$$S^h_p = e^{-M_p/2}$$

$S_p^h$ does not need to be a vector as multiplying a matrix or vector by a constant is simpler and quicker. We apply this survivorship twice in a year with the other dynamics occurring between:

$$\mathbf{N_p^{t+1}} = S_p^h\left[\mathbf{G_p A_p^t} S_p^h \mathbf{N_p^t}\right] + \mathbf{\Phi R_p}$$

### 7.2.9 Recruitment Processes

Recruitment is described using a vector with all new recruits allocated to the first size class and all other size classes being set to 0. This may need modification if small size classes are used (perhaps 1mm) and post-larval forms are variable in size. A Beverton-Holt stock recruitment relationship was assumed, as re-parameterized by Francis (1992), with a and b parameters that were restructured in terms of steepness, $h$, unfished mature or spawning biomass, $B_0^S$, and the average unfished recruitment level, $R_0$:

$$a_p = \frac{4h_p R_{p,0}}{5h_p - 1} \text{ and } b = \frac{B_{p,0}\left(1 - h_p\right)}{5h_p - 1}$$

Using this re-parameterization the Beverton-Holt relationship becomes:

$$R_{p,t} = \frac{4hR_{p,0}B_{p,t}^S}{\left(1 - h_p\right)B_{p,0} + \left(5h_p - 1\right)B_{p,t}^S} e^{\varepsilon_{p,t} - \sigma_{p,R}^2/2}$$

where $\varepsilon_{p,t}$ is defined as:

$$\varepsilon_{p,t} = N\left(0, \sigma_{p,R}^2\right)$$

The expected residual error distribution around the recruitment is log-normal; $\sigma_{p,R}$ is the standard deviation of the natural logarithm of the recruitment residuals, and $-\sigma_{p,R}^2/2$ is a bias correction term that ensures that the time series of estimated recruitment values relates to the mean rather than the median recruitment level (Hastings & Peacock 1975). This requires, for each population, that $h_p$, $R_{p,0}$, $B_{p,0}$, and $\sigma_{p,R}$ be defined for each population, and for each year that $B_{p,t}^S$ be estimated. Should deterministic recruitment be required (as when calculating the unfished, equilibrium zone structure), then set $\sigma_{p,R}$ to a very small number (when 1e-08 is squared this is the smallest number possible on most computers with 15 significant digits).

$A_{p,0}$ is the virgin biomass per recruit and is defined as the mature stock biomass that would develop given a constant recruitment level of 1. Thus, at a biomass of $A_{p,0}$, distributed across a stable size distribution, the resulting recruitment level would be $R_{p,0} = 1$. $A_{p,0}$ acts as a scaling factor in the recruitment equations by providing the link between $R_{p,0}$ and $B_{p,0}^S$. $A_{p,0}$ can thus be estimated by setting the annual recruitment level to 1, obtaining the equilibrium size distribution using unfished dynamics. At the virgin biomass per recruit, $A_{p,0}$, the average unfished recruitment level, $R_{p,0}$, is related directly to the unfished mature, or spawning, biomass, $B_{p,0}^S$:

$$B^S_{p,0} = R_{p,0}A_{p,0}$$

In Tasmania, during the conditioning, the average unfished recruitment level for each SAU (AvRec = $R_0$) is adjusted so that the fit of the predicted CPUE is close to the observed CPUE for each SAU. The populations within each SAU are defined using the Tasmanian GPS logger data to identify areas of persistent productivity (see the *Conditioning* section). Then the $R_0$ for each SAU is distributed among the populations in proportion to the relative yield reported for each of the GPS defined areas of persistent productivity (APPs, which we refer to as *populations*). Without a time-series of such detailed GPS data then the relative size of each population could be defined by randomly selecting (from a log-normal distribution) an initial unfished average recruitment, $R_{p,0}$, which, given the equilibrium size distribution, provides an estimate of each population's unfished mature biomass $B^S_{p,0}$.

## 7.2.10 Larval Dispersal

The annual dynamics include recruitment but a small proportion of each population's larval production can disperse away from that population. To be general we use a matrix of the potential movement from population to population but, given the linear structure to most reefs around coastlines very few cells in the matrix will be filled. The diagonal of the movement matrix defines the proportion of larvae that are expected to settle within the population that generates them. The immediate sub-diagonal and super-diagonal relate to the movement between adjacent populations. This simplification is used in the operating model because while a small amount of movement is known to occur (Miller *et al*, 2009), little or nothing else is known about that dispersal.

Alternative matrix structures could be generated to account for more complex spatial arrangements of populations but such things would need to acknowledge they were based on speculation rather then even approximate estimates.

$$\Phi = \begin{matrix} p_{1,1} & p_{2,1} & 0 & 0 & 0 & 0 & 0 \\ p_{1,2} & p_{2,2} & p_{3,2} & 0 & 0 & 0 & 0 \\ 0 & p_{2,3} & p_{3,3} & p_{4,3} & 0 & 0 & 0 \\ 0 & 0 & p_{3,4} & p_{4,4} & p_{5,4} & 0 & 0 \\ 0 & 0 & 0 & p_{4,5} & p_{5,5} & p_{p6,5} & 0 \\ 0 & 0 & 0 & 0 & p_{5,6} & . & . \\ 0 & 0 & 0 & 0 & 0 & . & p_{n,n} \end{matrix}$$

## 7.2.11 Fleet Dynamics

The application of any harvest control rule in this complex spatial model will entail the translation of SAU level aspirational catches (which should sum to the zone's TAC), into catches applied to each individual population within each SAU. Of course, the aspirational catches per SAU derived from the harvest control rules (HCR) within each harvest strategy will not be taken exactly, The overall TAC will eventually constrain total catches, but some SAU will experience larger catches than expected, and others will experience smaller catches. As a result, the application of the TAC as catches to particular populations requires two steps.

The first step is to include variation into the aspirational catches for each SAU from the HCR and then scale them until their sum equals the original TAC. The second step requires that each SAU's actual catch is distributed across their component populations in some manner that reflects plausible fleet dynamics.

The assumption is made in the Tasmanian HS that the sum of the catches across the SAU equates to the TAC. If ever the TAC were not completely caught this assumption would obviously have to change. Each year the aspirational catches for either the zone TAC or for each SAU will be determined by application of the harvest control rule for whatever harvest strategy is being used. In Tasmania, the harvest control rule uses a multi-criterion decision analysis (mcda) based around different aspects of cpue. In the operating model this is implemented initially without error.

$$C_{u,t,hcr} = C_{u,t-1} \times multhcr$$

where $C_{u,t-1}$ is the aspirational catch from SAU $u$ in year $t-1$, $C_{u,t,hcr}$ is the aspirational catch from SAU $u$ in year $t$, and $multhcr$ is the previous year's *acatch* multiplier derived from whatever HCR is used. It is important to understand that it is the aspirational catches that are modified each year, not the actual catches.

The new $TAC_t$ will be:

$$TAC_t = \sum_{u=1}^{nSAU} C_{u,t,hcr}$$

The first step on the way to setting the potential actual SAU catch is to include Log-Normal variation to the SAU exploitable biomass on the assumption that the divers gain an appreciation of how much is present in each SAU, but they make mistakes and availability varies across years so there is a misallocation of effort:

$$B_{u,t}^{E,*} = B_{u,t}^{E} e^{\varepsilon_{u,t} - \sigma_B^2/2}$$

where:

$$\varepsilon_{u,t} = N(0, \sigma_B^2)$$

$\sigma_B$ is the standard deviation of the variation that occurs between the real exploitable biomass and the perceived distribution of exploitable biomass across SAU's, which also includes other sources of uncertainty.

This variation is introduced into the potential SAU catches using:

$$C_{u,t}^{*} = TAC_t \times \frac{B_{u,t}^{E,*}}{\sum B_{u,t}^{E}}$$

where $C_u^*$ is the actual SAU catch scaled to the new TAC.

A useful diagnostic for use during conditioning, and during model runs, would be to examine the predicted variation between the actual catches and aspirational catches determined by the

HCR, if one has such data from a real fishery. Otherwise, it should be characterized from the outputs, which will allow its variation to be examined for plausibility.

### 7.2.12 Calculation of MSY

The maximum sustainable yield for each population is estimated for each one using a numerical approach that characterizes the expected production curve but starting with the equilibrium unfished population (or app) and sequentially applying a sequence of harvest rates for two or three times the number of historical years of data until it approximates an equilibrium yield for each applied harvest rate. The maximum yield achieved across the range of harvest rates is the MSY. The precision of such numerical methods is constrained by the steps selected between the individual harvest rates. The smaller the increments on harvest rate the finer the resolution. The dynamics of this numerical process, by default, uses the selectivity that will be applied at the end of the projection years.

If, for some, currently un-imagined, reason the selectivity from an alternative year is wanted then the index for that year (between 1 and hyrs + pyrs) can be put into the *selectyr* argument for the *do_MSE()* function.

## 7.3 Model Output

## 7.4 Sampling from the Operating Model

The MSE operates by simulating the yearly dynamics and then, using data from the operating model (OM) it applies a harvest control rule, at a period determined by the respective harvest strategy being tested, generates the required management advice and conducts subsequent years using that management advice. This is repeated for as many years as are included in the simulation.

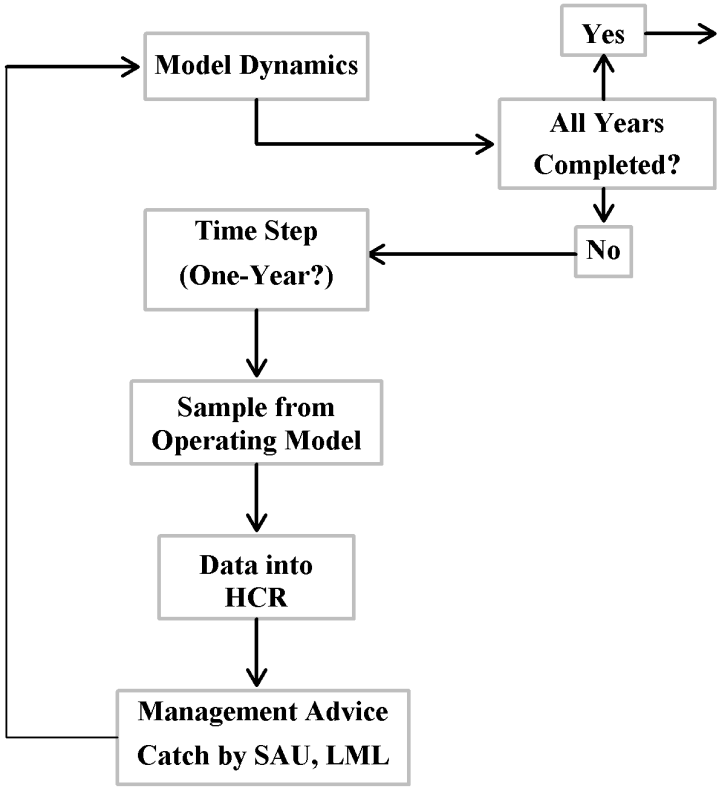


*Figure 7.1: The feedback loop of an MSE simulation framework as used with abalone. The data can include anything from the operating model although always at the scale of SAU, including catches, cpue (and/or survey indices), length-composition of catches, each with error, as well as error-free statistics from the simulated population.*

# 8. Conditioning by Population

## 8.1 Summary

The AIRF project 2023_63 has enabled the **aMSE** code base developed during the FRDC project 2019-118 to be expanded to allow for population-based conditioning of individual populations within different spatial assessment units (SAU). The option of population conditioning is now available although if is not used it has no effect on the simulation outputs. Population based conditioning is required when attempting to answer tactical issues such as the effect of making a change to the legal minimum length (LML) in areas where the heterogeneity of productivity is very great (such variation is common in blacklip abalone stocks). In addition to adding these new options, an array of new outputs relating to population-based dynamics and properties have also been implemented, which, again, are designed to facilitate answering SAU specific questions relating to optimizing management decisions other than catch levels. All of these changes now operate seamlessly within the codebase.

## 8.2 Introduction

In previous sections, methods for conditioning the operating model within the MSE have been described. Assuming it has been possible to use the **sizemod** package to fit to data from each SAU, ideally, the dynamics of each SAU, as they are predicted by the **aMSE** software, should be very similar to that expressed by the individual SAU in the **sizemod** software. Differences can be expected because in the **aMSE** package each SAU, instead of being treated as a single dynamic pool population (as it is in **sizemod**), is represented as a set of mostly separate populations.

Prior to the changes described in this chapter the biological properties of each population within each SAU are based upon the properties expressed at the SAU scale, albeit with random variation included on almost all biological properties. The degree of variation added in the examples is set to a level in each case to mimic the spatial heterogeneity in productivity typically expressed by the blacklip abalone around Tasmania). When conditioning by SAU the current property that is an exception to this strategy is the average unfished recruitment level (*R0* or *AvRec*), estimated as a parameter for each SAU by **sizemod**. Now, in **aMSE**, the SAU level *AvRec* is sub-divided across each SAU's populations on a fixed proportional basis, which will be unique to each SAU (and has been derived from the relative yield from the population areas as defined using the GPS logger data). The method of implementation can be seen in the Chapter 5 under the *propREC* section. As a result of this variation, there can be many differences between and within SAU and how they are represented in the MSE.

The development of both the **aMSE** and **sizemod** R packages was partly driven by needing to solve the various problems associated with providing a generalized framework for conducting MSE analyses on spatially structured benthic fisheries on species that required size-structured dynamics (because they were hard to age). The **sizemod** R package was developed as a means of conditioning the MSE operating model within **aMSE** such that its predicted dynamics attempted to match both the observed CPUE as well as the observed size-composition of the catch of abalone. The advent of **sizemod** led, in turn, to improvements to **aMSE** because of the new options presented by the outputs of **sizemod**. It is still the case, however, that some of the outputs from **sizemod** need to be modified slightly when they are represented by multiple populations within each SAU.

In some parts of the fishery more information is available concerning some aspects of the biology such as the size-at-maturity and growth. There are sometimes large differences in yield between some areas of persistent production (app or population), which are not able to be accounted for merely by including random variation and modifying the average unfished recruitment (*AvRec*) for a particular app/population. In addition, it can sometimes be desirable to set up scenarios where groups of *apps* within SAU have either low, middling, or relatively high productivity. In such cases some way of manipulating the variables important to productivity within individual apps (=populations) is required. The proportional strategy used to allocate unfished recruitment across the collection of apps within each SAU is not suitable for fixing values of size-at-maturity or growth in particular populations, so an alternative approach was developed and implemented.

### 8.2.1 Variables Important for Productivity and Yield

A number of model parameters relating to biological properties important for productivity exist. These include:

- average unfished recruitment, *AvRec*
- the growth parameters, here we use the maximum growth increment, *DLMax*, and the length at 5 percent of the maximum increment, *L95* (= *L50* + *L50inc*)
- the size-at-maturity, here we use the *L50mat* parameter (= length at 50 percent maturity) and the *SaMa*, or size-at-maturity 'a' parameter, the intercept of the maturity ogive with length
- the instantaneous natural mortality rate, *M*
- the steepness of the stock recruitment relationship, *h* (from the Beverton-Holt stock recruitment curve)
- the weight-at-length relationship, here we use the exponential *b* parameter of the $W_L = aL^b$ that represents the weight at a given length $W_L$.

How these can be allocated individually to particular populations/apps is the subject of this chapter.

### 8.2.2 The Difference between Productivity and Yield

The notions of productivity and yield sometimes appear confounded in the fisheries literature but their differences are clear.

The yield from any area is simply the amount of catch that has been removed over whichever time-frame is being considered. Whether the 'yield' should include any other fishing related mortality is a complication best avoided by treating that separately. With abalone, non-fishing related mortality should be minimal, especially as when divers accidentally remove sub-legal sized animals from the rock, they are required to place the animals back on a rock surface and ensure they adhere. Illegal, unreported, and unregulated catches (IUU catches), such as that due to poaching or even recreational fishing, should be treated separately where there is available information.

Productivity is a more complex concept than simple yield. It relates to the rate of production of potential yield, which is one reason the two concepts can get confused. For example, the somatic growth of individuals is a strong contributor to a population's productivity. Thus, a low productivity reef of the same area as a higher productivity reef will, on average, produce a lower yield because the individuals in the more productive population are expected to grow

past the legal minimum length (LML) faster and to grow to a larger, heavier size. Two equal sized reefs might produce the same yield in a given year, but the more productive reef will be able to sustain such catches for longer. Generally, a lower productivity area would be expected to have slower growth, a smaller maximum size, and possibly a lower size-at-maturity. The possible yield is determined both by the biological productivity and the legal minimum size, as well as the area of a reef. A key factor for abalone sustainability is setting the legal minimum size appropriately.

The caveat above about 'same size reef' is important, because, in principle, it would not be impossible to obtain a seemingly high yield from a relatively low productivity area if that area was large. Even so, it would not be expected to be able to sustain large catches for as long as an equivalent high productivity area. Similarly, if the available reef area of a highly productive population was low then the potential yield from it will also be low. Despite this, a difference between the two is that it is easier to deplete a higher productivity area if the legal minimum size is set at a size that enables the fishery to deplete the spawning biomass. If the LML is set at a size that allows good access to slower growing, less productive populations in an SAU (spatial assessment unit), this will increase the risk of over-fishing for faster growing, more productive areas, which, very sensibly, are generally preferred by divers.

As natural mortality is generally strongly related to somatic growth rates and maturity, natural mortality is also expected to be strongly influential on productivity (Beverton, 1992; Jensen, 1996). However, as is typical in most fisheries, estimates of natural mortality for any species are generally poorly defined, and this is especially the case with species that are difficult to age. Abalone around Tasmania appear able to live to at least 30 years of age (based on growth rates and maximum sizes), which using simple life-history relationships implies a natural mortality rate of about 0.15. This is generally used along with slight variation among populations. When attempting to adjust the productivity of individual populations (=apps) it is recommended that changes be made to growth, maturity, and recruitment, rather than to natural mortality.

The average long term yield from an area is correlated with the area of reef fished (determined using the combined areas of the KUD derived from the GPS logger data). So, when attempting to set the productivity *AvRec*, or the proportion of recruitment allocated to a population, should perhaps be the first variable to modify. The total *AvRec* influences the SAU's total population, so changing that will alter each population according to its allocated share.

After having set up the SAU approximately to the level reflecting that observed in the fishery, it is recommended that when proceeding to condition on individual populations then the parameters on which to concentrate relate to somatic growth and maturity. Of course, if required, any of the model's parameters can be added to the list of those fixed by population. For example, the steepness (*defsteep*) in the data file) can be influential because it also affects the stock recruitment curve, although once the other parameters are fixed the effects of steepness become relatively minor.

## 8.3 Methods

A description of conditioning each SAU might summarize earlier sections by describing how a base case set of constants for natural mortality, steepness, and cpue hyperstability, *lambda*, were chosen for implementation within **aMSE**. Once selected, the available fishery and biological data for each SAU could be put through **sizemod** and the model fitted in a manner that estimated some growth parameters, the unfished average recruitment, and the average

diver selectivity across sizes of abalone. In addition, **sizemod** estimates recruitment deviates across those years where sufficient size-composition data exist to inform the model. Other constants, such as those describing the weight-at-length are estimated outside of the model.

### 8.3.1 Relative Weighting of Data Sets

Within **sizemod** the relative weight attributed to the cpue index of relative abundance is described by Francis (2011; the so-called Francis weighting). At the same time, an iterative re-weighting routine is used to discover the optimum weighting to apply to the size-composition data. Finally, relative bias ramps are applied to the recruitment deviates to account for the varying amount of information available at the limits of the observed size-composition data (Method & Taylor, 2011).

Routines have now been developed that collect together the final optimum parameters into a matrix and these can now be automatically transferred to the control and data files used by **aMSE** so that the latest estimates of the important productivity parameters are easily transferred accurately from **sizemod**. To enhance comparability between the outputs of both **sizemod** and **aMSE** it is important that the assumed constants for such things as the weight-at-length, maturity-at-length, growth and all the rest are the same for each SAU in both software systems.

After transferring the parameters from the **sizemod** estimates into each SAU within the **aMSE** operating model further adjustments are required to the *AvRec* and recruitment deviates to optimize the fits between the predicted CPUE at the SAU level and the observed standardized CPUE as well as optimizing the fits between the predicted size-composition of catch and those observed. This is done within **aMSE** using the two functions *adjustavrec()* and *optimizerecdevs()* (see their individual help pages within **aMSE** for their arguments and the syntax for how to use them.

The differences between the predicted MSY and *AvRec* for **sizemod** and **aMSE** is such that for each SAU in the Tasmanian western zone the MSY in **aMSE** is about 9.5% less than the **sizemod** estimate for each SAU, and the *AvRec* is about 3.8% larger in **aMSE**, although that relationship is much more variable between SAU. These differences between the **sizemod** and the **aMSE** values are a result of the dynamics in **aMSE** being split between each SAU's populations rather than a single dynamic-pool within each SAU SAU and the scale of the differences alters with the number of populations used within an SAU.

### 8.3.2 Conditioning the Operating Model at a Population Level

Much of the data used to condition the operating model is contained within the 'saudataXXX.csv' file (see Chapter 5 for a detailed description of the contents of each input file). Each SAU has a particular average value of the important variables to be set, such as *DLMax* and *L50mat*. In addition, there is an associated variability *sDLMax* and *sL50mat* used to define the variation used when randomly varying each parameter across the populations within each SAU. For what follows it is important that the name of the variation parameter is identical to the variable it relates to except for the prefix 's'. This is because the names are used explicitly when identifying which parameters are to be set by population and which are to remain randomly allocated across populations. Those that are to be set by population need to have their associated variation set to a very small number (e.g. 1e-06 or smaller) so that when variation is added it does not affect the set value.

### 8.3.3 An Hypothetical Example

An hypothetical sauX can be used to illustrate what changes are required to implement population-based parameter setting. The example will be unrealistic as only four populations will be implemented within a single SAU so that the example is simple to follow (a matrix of four rows is easier to follow than one of 30 rows). In real simulations there would generally be many more separate populations. For example, the western zone simulation used in FRDC project 2019-118 had 56 populations across the eight SAU. When conditioning by population the potential to increase the spatial detail is greater.

We are using the catch, cpue, and size-composition data from sau5 in Tasmania's Northern zone to form the basis of our hypothetical SAU. In reality, the number of identifiable populations within sau5 is somewhere between 13 and 24, with a final number to be decided through consultation with experienced industry members regarding on-the-ground fishing behaviour.

The structure and contents of the control, data, and size-composition data files are described in the chapter on *The Input Files* (Chapter 5). No changes are required to the control file. The *bysau* flag, under the START section in the control file should remain set = 1:

- START,,,,
- runlabel, sauX , the scenario label,,
- datafile, saudatasauX_by_sau.csv , name of saudata file,,
- bysau,1, 1=TRUE and 0=FALSE,,
- …

Prior to AIRF project 2023_63 the *bysau* flag was inserted into the code base in readiness in case an opportunity came to condition the population properties directly rather than as random variation from SAU properties. Once starting the 2023_63 project, after some false starts, it soon became clear that there was a more effective way of implementing population-based conditioning that did not need the *bysau* flag. Rather than produce a large and difficult to use matrix of all variables (minus the variation terms) by all populations it is much more efficient to only modify those variables that influence productivity (or other aspects of the fishery that are to be explored) and leave the rest as random variation from SAU averages. The *bysau* flag is now deprecated and eventually it will be removed from the code. It is retained in the meantime to avoid the potential for disruption of other users of the **aMSE** software. If the flag is set to its default value of 1 (as in: bysau, 1,) then the flag will continue to have no effect on any of the scenarios.

Instead of using the *bysau* flag a simpler solution for the user was to allow the software to determine which parameters were to be fixed for each population when reading in the data file. This is the source of the requirement that care is needed when typing the names of the selected parameters. When no population conditioning is used, which was the only option developed prior to AIRF project 2023_63, the *propREC* section only referred to the proportion of each SAU's *AvRec* (average unfished recruitment level) allocated to each population. Obviously, the proportions across all populations sums to 1.0. In the data file this is represented using the following lines at the bottom of the file (see Chapter 5; the symbols at the start of each line in the text are not included in the data files):

- propREC,,
- sau, pop,AvRec,

- 5, 1, 0.1,
- 5, 2 ,0.4,
- 5, 3, 0.1,
- 5, 4, 0.4,

In the single-SAU-four-population example of sauX we will illustrate fixing the *L50Mat*, the size at 50% maturity, *SaMa*, the intercept of the maturity ogive , *DLMax*, the maximum growth increment, *L50*, the size at half the maximum growth increment, and *L50inc*, which is used to produce the *L95* growth parameter ($L_{95} = L_{50} + L_{50inc}$).

When implementing population-based conditioning, where selected variable/parameter values are defined or fixed in each population, it is necessary to provide values for each of the selected parameters for each population implemented in the operating model. The implementation involves expanding the original *propREC* section at the bottom of the data file. Perhaps the section should no longer be termed *propREC*, but again, to prevent disruption to other users this name will be retained (for the time being; naming things when writing software is harder than it might look).

The names used in the first line below the *propREC* label must have exactly the same spelling and capitalization as is used in the table of SAU parameters that make up the top of the data file (see Chapter 5). The first three columns remain identical to the default conditioning setup (as above). We have added the five parameter names and the respective values that we wish to allocate to each of the four populations.

In the north-west for example, we might use an analysis of the extensive data set available on the size-at-maturity across sau 5 and 6 to define a more specific set of values for the different populations identifiable using the GPS logger data. Similarly, in consultation with Industry divers, the relative productivity expected from each smaller area within each SAU can be used to select appropriate values for the growth parameters. Such conditioning at a finer geographical scale than the SAU is very dependent upon the GPS logger data. But this finer scale is required to make sense of the heterogeneity in productivity expressed by the various areas of persistent productivity identified within the north-west.

*Table 8.1: An example expansion of the propREC data fields at the bottom of the MSE data file required to set the values of five extra parameters in each population.*

| sau | pop | AvRec | L50Mat | SaMa | DLMax | L50 | L50inc |
|---|---|---|---|---|---|---|---|
| 5 | 1 | 0.1 | 99.5 | -15.969 | 21.8 | 112 | 31.5 |
| 5 | 2 | 0.4 | 98.9 | -15.969 | 24.0 | 112 | 38.5 |
| 5 | 3 | 0.1 | 113.0 | -25.853 | 26.6 | 115 | 34.9 |
| 5 | 4 | 0.4 | 115.0 | -25.853 | 22.6 | 115 | 40.1 |

The data file, with most unchanged lines omitted, should now have the following form:

- Biological properties by population for hypothetical sauX
- SPATIAL,,
- nsau,1, number of spatial management units
- saupop,4, number of populations per SAU in sequence
- saunames,X, labels for each SAU

- PDFs,32,
- DLMax ,23.50603217, maximum growth increment
- sDLMax ,1.00E-06, variation of MaxDL NOTE tiny value
- L50 ,112, Length at 50% MaxDL
- sL50 ,1.00E-06, variation of L50 NOTE tiny value
- L50inc ,30.34957282, L95 - L50 = delta =L50inc
- sL50inc ,1.00E-06, variation of L50inc NOTE tiny value
- …
- AvRec,878733.028324352,
- sAvRec ,1.00E-07, NOTE tiny value
- …
- SaMa,-22.371, maturity logistic a par
- L50Mat,98.8992042, L50 for maturity b = -1/L50
- sL50Mat,1.00E-07, NOTE tiny value
- …
- propREC,,
- sau, pop,AvRec,L50Mat,SaMa,DLMax,L50,L50inc,
- 5,1,0.1, 99.5,-15.969,21.8,112,31.5,
- 5,2,0.4, 98.9,-15.969,24.0,112,38.5,
- 5,3,0.1,113.0,-25.853,26.6,115,34.9,
- 5,4,0.4,115.0,-25.853,22.6,115,40.1,

Note the single-sau-four-population spatial structure under SPATIAL. Each of the five parameters fixed to population specific values has its related variation value denoted with the same name prefixed with *s*, as, for example, *sDLMax* and *sL50inc*. The *SaMa* parameter does not have additional variation as the weight-at-length relationship it affects is extremely sensitive to this value and adding even small amounts of variation led to unpredictable outcomes. In each case these *sParName* values have been set to very low value such as 1e-06 or 0.000001. When each population is generated within the **aMSE** software, including this much variation leads to no noticeable change to the fixed values.

If the user forgets to set the variation variable to a very low value and yet sets up the rest of the data file to fix specific parameters, then a warning message is generated. Similarly, if a parameter is not set up to be fixed for each population but the variation term is still set very low, a different warning message is generated.

### 8.3.4 Other Details

Earlier data files were somewhat odd in that the variation term for the *DLMax* was named *sMaxDL*.

- PDFs,32,
- DLMax, 23.50603217, maximum growth increment
- sMaxDL, 1.00E-06, MaxDL name decremented, replaced by sDLMax
- L50, 112, Length at 50% MaxDL

- …

This is merely an historical hang-over, which once started was hard to change without bothering all current users of the program. Fortunately, an approach to fixing this has now been developed. Should a user begin developing a scenario using this older format the software will automatically rename the *sMaxDL* as *sDLMax* in the 'saudata.csv' file to allow correct usage of population-based conditioning, and it also issues a warning pointing out that *sMaxDL* is deprecated and has been changed. The end result is that users can continue to use older data files without issues arising. This strategy may be applicable to other deprecated instances that have developed from the on-going implementation of expanded options within **aMSE**.

### 8.3.5 Other Code Base Changes

With the advent of population level conditioning some novel model output relating to the population scale have now been included in the standard **aMSE** output. These will enable the individual dynamics within each population to be visualized and examined in more detail. This will be of great value if, for instance, an investigation was made of the implications of changing the Legal minimum length in SAU's that have populations that exhibit large differences in growth and productivity (as is now happening in Tasmania's NorthWest).

Some of these additions are illustrated below from the sauX example, the four populations were set up to have approximately two different levels of productivity.

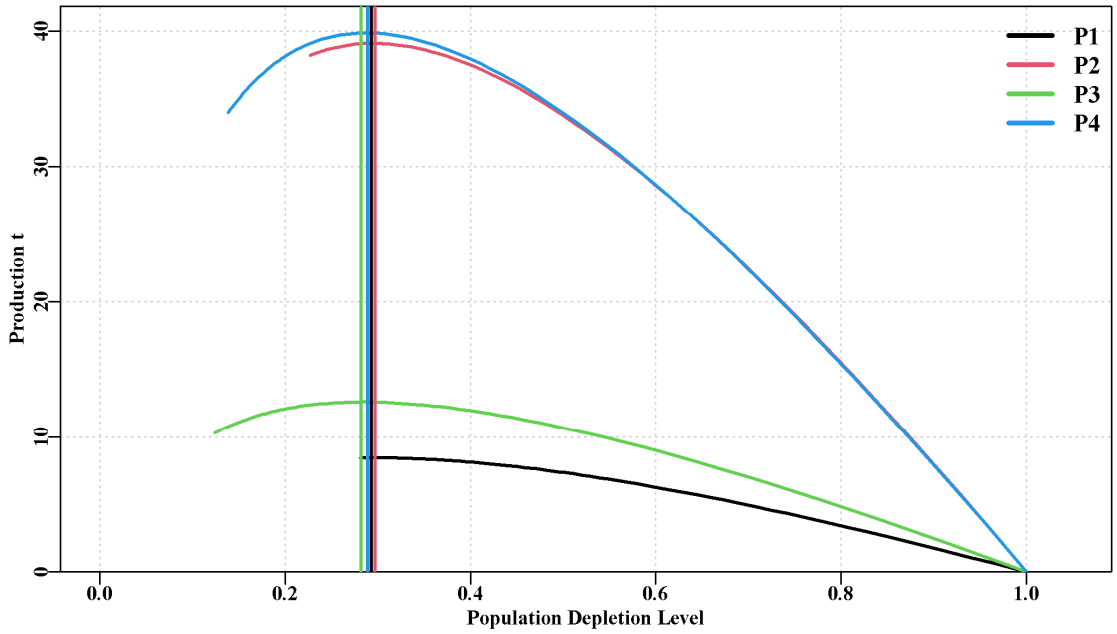


*Figure 8.1: The relationship between production and mature biomass depletion for the four populations in the sauX example SAU.*

Note how in Figure 8.1 all four populations have their maximum productivity at very similar spawning biomass depletion levels despite having very different productivity. In fact, the production is relatively flat between depletion levels of about 25 - 35 %

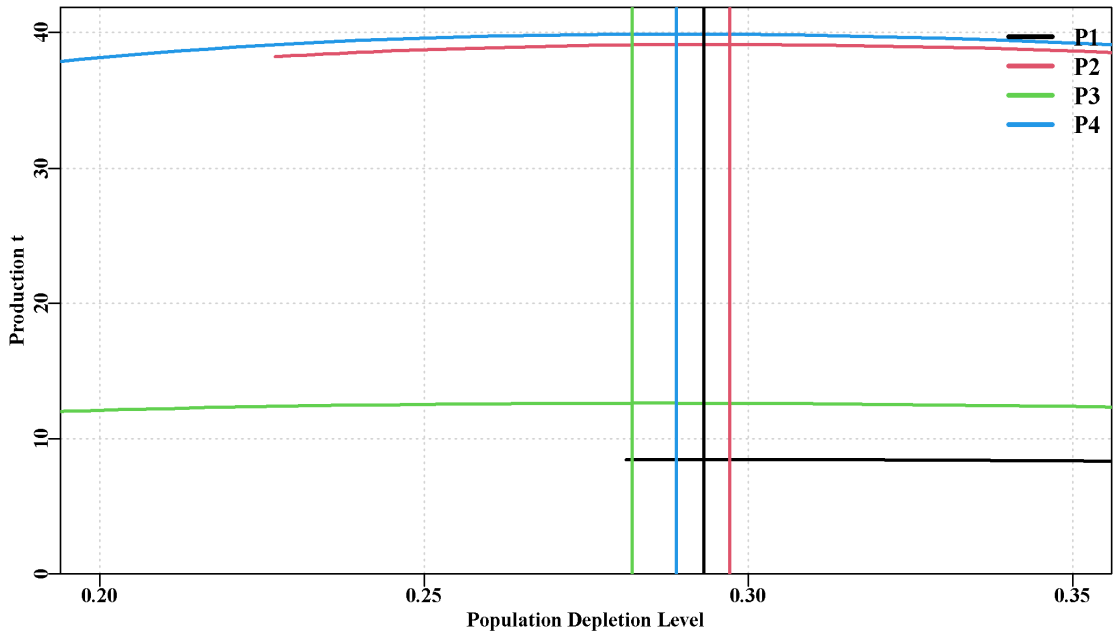


*Figure 8.2: The relationship between production and mature biomass depletion in the sauX example SAU, at an expanded scale.*

Numerous other small but important changes were required in many places throughout the **aMSE** codebase once the population-based conditioning was implemented to ensure that the most recent changes do not alter the functionality of the software when population conditioning is not used.

# 9. Perturbations within Projections

## 9.1 Introduction

Within the simulation framework used in **aMSE**, the population dynamics are not deterministic and variation is introduced in the projections through random variation in the predicted recruitment levels, through variation being introduced in the how the actual catches are distributed among the populations (essentially this is in teh description of the fleet dynamics), and finally, with variation being introduced into the expression of the predicted cpue from the dynamics (this latter is important as all the current harvest strategy performance measures are based upon the commercial cpue).

There are other sources of change and variation that are not accounted for. It is known that there have been marine heatwave events that have led to mortality events observed by divers (exceptional numbers of dead abalone shells washing back and forth in gutters). Such events have been confirmed to correspond to observed oceanographic events entailing large deviations from average sea temperatures. The effects of such heatwaves, destructive storm events, and other external occurrences, such as toxic algal blooms, are not known from direct observations except for the anecdotal observations from divers. These observations inform that there has been an impact but provide no measure of the extent or severity of any such impacts. When we apply a stock assessment model, such as that implemented in the **sizemod** R package, effectively the dynamics as described by such models only identifies the catches and recruitment deviates as the only drivers of the observed dynamics. Without real data on possible impacts due to environmental events their scale of any impact cannot be estimated.

In Australia, there have been heatwave events whose impact is undeniable. In Western Australia, during 2011?, a significant heatwave event occurred down the west coast with one of its impacts being the effective elimination of the *Haliotis roie* stocks north of Perth. Other, less severe heatwave events (and other such perturbations) potentially can have impacts on survivorship of both the emergent and the cryptic animals currently alive. In addition, it is simple to imagine that the physiological shocks experienced by mature animals may well reduce recruitment success rather than induce death (i.e. sublethal effects). Given these considerations a need has therefore been identified for having a facility within **aMSE** for exploring the longer-term influence of possible perturbations to survivorship of currently settled animals but also of perturbations to recruitment success in selected years. The simulation framework is designed to explore the performance of different versions of abalone harvest strategies. It is expected that marine heatwaves and other such perturbations will occur into the future, so it will be valuable to be able to determine how each tentative harvest strategy handles the possible outcomes of significant perturbations to either survivorship or recruitment, or both.

## 9.2 Methods and Results

The capacity to introduce perturbations into the recruitment and the survivorship of post-larval animals has been introduced in such a way that no changes are required if no perturbations are wanted in a set of simulations. Thus, only if perturbations are required are changed required, and the only changes required involve adding a few lines of detail into the control file, which every scenario must have. No changes are required to any other files.

In the control file major sections are identified by capitalized headings such as ‘START’, ZONE’, ‘RECRUIT’, and ‘RANDOM’. The new section can go anywhere but it is proposed

that when it is wanted it gets introduced between the ‘PROJECT’ and the ‘PROJLML’ sections. Thus, in a control file with no defined perturbations one might have:

…
initLML, 140, initial LML for the unfished zone if no historical catches used
PROJECT, 30, number of projection years for each simulation

PROJLML, need the same number as there are projection years
2021, 145, Legal Minimum Length (LML, MLL, MLS) e.g. 140
2022 , 145
…

To add a single year of perturbations we can add:

---

…
initLML, 140, initial LML for the unfished zone if no historical catches used
PROJECT, 30, number of projection years for each simulation

ENVIRON, 1, numbers of years of intervention
eyr, 5, which projection years will have an event
proprec1, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, one for each sau
propNt1, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, one for each sau

PROJLML, need the same number as there are projection years
2021, 145, Legal Minimum Length (LML, MLL, MLS) e.g. 140
2022 , 145
…

---

The new keyword is ‘ENVIRON’. The commas are important as they identify the separate fields that **aMSE** needs to read.

This set up would impose a perturbation in the fifth projection year and the impact will force the recruitment to be only 0.3 of the predicted recruitment level taken off the deterministic stock-recruitment curve for each population, when it only had 10 percent of the usual random recruitment variation that every other year experiences. Survivorship of post-larval animals will be 99 percent, so only a tiny impact on emergent and cryptic animals. If once this section has been introduced, one wants to turn off the perturbation then all that is needed is to set the ‘ENVIRON’ number = 0. That number must be greater than zero for there to be an effect.

With almost no impact on settled individuals the outcome on the dynamics only becomes apparent after a delay of between 5 - 8 years.

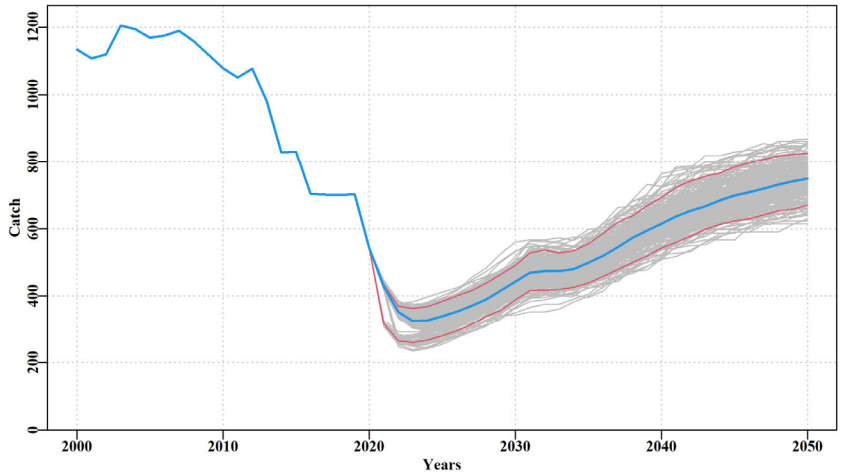


*Figure 9.1: Predicted catches across the whole Tasmanian western zone when a perturbation is introduced in the fifth projection year (2025). The time-lag required for animals that should have settled that year mean that its effect only becomes apparent 5 - 9 years after the event, in the low 2030s.*

To add two years in which perturbations occur we can add:

---

…
initLML, 140, initial LML for the unfished zone if no historical catches used
PROJECT, 30, number of projection years for each simulation

ENVIRON, 2, numbers of years of intervention
eyr, 5, 8, which projection years will have an event
proprec1, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, one for each sau
proprec2, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, one for each sau
propNt1, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, one for each sau
propNt1, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, one for each sau

PROJLML, need the same number as there are projection years
2021, 145, Legal Minimum Length (LML, MLL, MLS) e.g. 140
2022 , 145
…

---

which gives rise to:

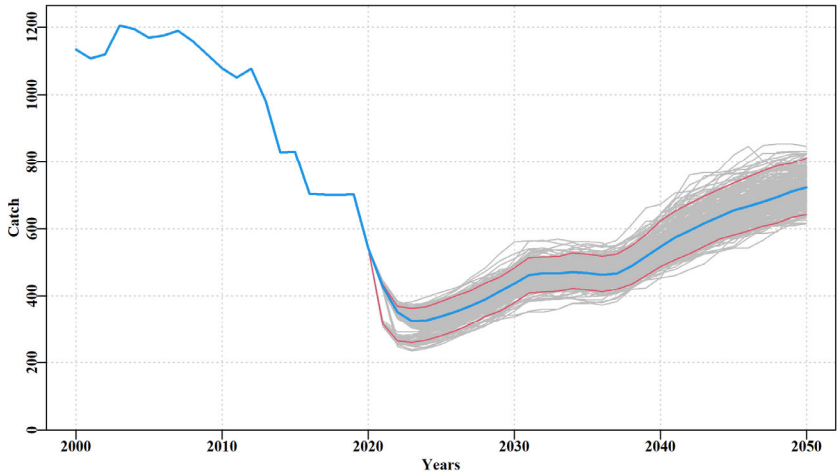


*Figure 9.2: Predicted catches across the Tasmanian western zone when identical perturbations are introduced in both the fifth and eighth projection years (2025 and 2028). The time-lag required for animals that should have settled that year mean its effect only becomes apparent 5 - 9 years after the event, in the low 2030s, but this time extends for much longer to prevent stock and fishery recovery.*

## 9.3 Comparing Scenarios

When the results of these perturbations are compared with the identical harvest strategy with no perturbations then the impacts become clearer.

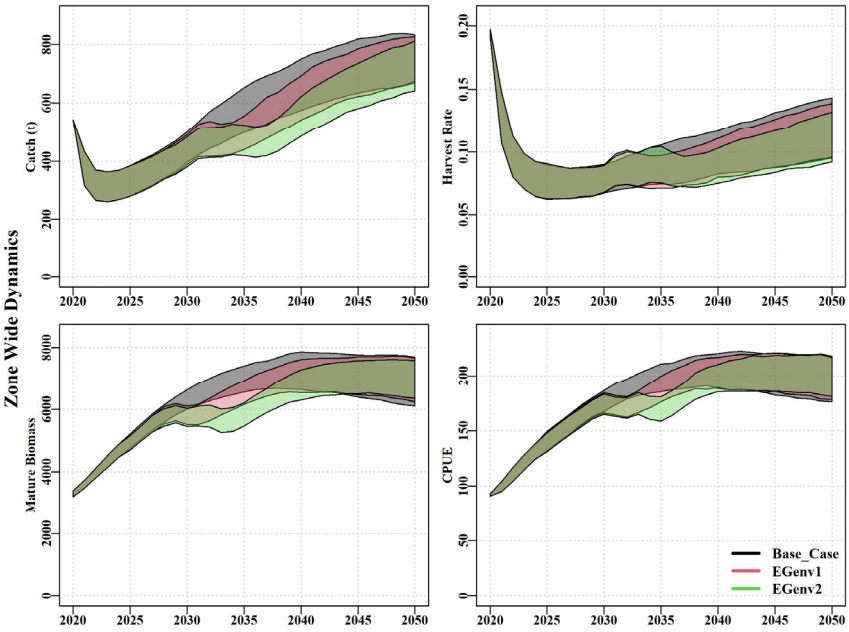


*Figure 9.3: The dynamics of the western Tasmanian zone comparing no perturbations as a base-case (grey), with the effect of a single perturbation in 2025 (red), and a double perturbation in 2025 plus 2028 (green).*

It should be clear that if a sequence of environmentally driven perturbations occur that significantly disrupt the recruitment success, then the stock recovery will be compromised. Given the increasing frequency of such events this is a real possibility that needs to be included in any considerations given to rebuilding Tasmania's abalone stocks.

Given the availability of this facility to introduce perturbations it is now possible to explore how large a perturbation must be to have a significant impact (the example of repeated zone-wide 70% reductions in recruitment success would constitute quite a severe impact). Now it is possible to examine the effect of smaller scale events (at a scale of sau) on the potential risk to stock rebuilding.

# 10. References


Anon (1966) Tasmanian abalone catch increases. *Australian Fisheries Newsletter* **25(4)**:7 & 13.

Beverton, R.J.H. and S.J. Holt (1957) On the dynamics of exploited fish populations. *U.K. Ministry of Agriculture and Fisheries, Fisheries Investigations (Series 2)*, **19**: 1-533.

Beverton, R.J.H. (1992) Patterns of reproductive strategy parameters in some marine teleost fishes. *Journal of Fish Biology* **41** (Supplement B):137-160.

Bradshaw, M. (*ed*) (2018) *Tasmanian Abalone Fishery Sustainable Harvest Strategy* Wild Fisheries Management Branch, Department of Primary Industries, Parks, Water and Environment. 30p.

Butterworth, D. S., and M. O. Bergh (1993) The development of a management procedure for the South African anchovy resource. pp. 83–99. In _Risk Evaluation and Biological Reference Points for Fisheries Management__, eds. S. J. Smith, J. J. Hunt, and D. Rivard. *Canadian Special Publication in Fisheries and Aquatic Sciences*, **120**.

FAO (1985) *FAO. The First 40 Years*. Food and Agricultural Organization of the United Nations, Rome.

FAO (1995) *Code of Conduct for Responsible Fisheries.* Food and Agricultural Organization of the United Nations, Rome. 41p.

FAO (1996) *Precautionary Approach to Capture Fisheries and Species Introductions.* FAO Technical Guidelines for Responsible Fisheries 2. Food and Agricultural Organization of the United Nations, Rome. 55p.

FAO (1997) *Fisheries Management.* FAO Technical Guidelines for Responsible Fisheries 4. Food and Agricultural Organization of the United Nations, Rome. 84p.

Fournier, D.A. and C.P. Archibald (1982) A general theory for analyzing catch at age data. *Canadian Journal of Fisheries and Aquatic Sciences* **39**: 1195-1207.

Francis, R. I. C. C. (1992) Use of risk analysis to assess fishery management strategies: A case study using orange roughy (*Hoplostethus atlanticus*) on the Chatham Rise, New Zealand. *Canadian Journal of Fisheries and Aquatic Science* **49**:922–30.

Garstang, W. (1900) The impoverishment of the sea - a critical summary of the experimental and statistical evidence bearing upon the alleged depletion of the trawling grounds. *Journal of Marine Biological Association of the United Kingdom*, **6**: 1-69.

Gulland, J.A. (1965) Estimation of mortality rates. Annex to Arctic Fisheries Working Group Report (meeting in Hamburg, January 1965). International Council for the Exploration of the Sea, Document 3 (mimeo), Copenhagen. (cited in Megrey, 1989)

Haddon, M. (2007) Fisheries and their management. Pp 515-532. In *Marine Ecology* (*eds*) S.D. Connell & B.M. Gillanders. Oxford University Press. 630 p.

Haddon, M. (2011) *Modelling and Quantitative Methods in Fisheries*. 2nd Ed. CRC/Chapman & Hall. 449p.

Haddon, M. (2021) *Using R for Modelling and Quantitative Methods in Fisheries*. Chapman and Hall/CRC. https://haddonm.github.io/URMQMF

Haddon, M. (2024) *aMSE: A Framework for Abalone Management Strategy Evaluation* An R Package, version 0.3.5. https://github.com/haddonm/aMSE

Haddon, M. (2024a) *aMSEGuide: Management Strategy Evaluation for Abalone Fisheries* A GitBook and PDF at HTTPS://haddonm.github.io/aMSEGuide

Haddon M (2024b). *codeutils: Contains a Series of Utility Functions for File Handling and Programming*. R package version 0.0.15. https://github.com/haddonm/codeutils

Haddon M (2024c). *hplot: Contains utility functions for when using base R graphics*. R package version 0.0.19, https://haddonm.github.io/hplot

Haddon M (2024d). *makehtml: Provides Generic Code for Producing Tabbed HTML for Presenting Plotted Results*. R package version 0.1.2. https://haddonm.github.io/makehtml

Haddon M (2024e). *sizemod: Implemetation of a Size-Based Assessment Model for Single Populations*. R package version 0.2.0, https://github.com/haddonm/sizemod.

Haddon, M., Ziegler, P., Lyle, J. and Burch, P (2005) Using a spatially structured model to assess the Tasmanian fishery for Banded Morwong (*Cheilodactylus spectabilis*). Pp 737-756. In *Assessment and Management of New and Developed Fisheries in Data-Limited Situations.* Lowell Wakefield Symposia; Alaskan Sea-Grant Program.

Haddon, M. and F. Helidoniotis (2013) Legal minimum lengths and the management of abalone fisheries. *Journal of Shellfish Research* **32**:197-208.

Haddon, M. and F. Helidoniotis (2014) *Modelling the Potential for Recovery of Western Victorian Abalone Stocks: The Crags*. Interim Report to 2012/225. Hobart. 61 p.

Haddon, M., Mayfield, S., Helidoniotis, F., Chick, R. and C. Mundy (2013) *Identification and Evaluation of Performance Indicators for Abalone Fisheries*. FRDC Final Report 2007/020. CSIRO Oceans and Atmosphere and Fisheries Research Development Corporation. 295 p.

Haddon M, Mundy C (2024). *TasHS: Provides the Static Functions used in Tasmania's Abalone Harvest Strategy*. R package version 0.1.15. https://github.com/haddonm/TasHS

Haddon M, Mundy C (2024b). *EGHS: Example harvest control rules used with aMSEGuide to illustrate MSE*. R package version 0.0.2. https://github.com/haddonm/EGHS

Haddon, M. and C. Mundy (2016) *Testing abalone empirical harvest strategies for setting TACs and associated LMLs, which include the use of novel spatially explicit performance measures.* FRDC Final Report 2011/028. CSIRO Oceans and Atmosphere and Fisheries Research Development Corporation. Hobart 182p.

Haddon, M., Mundy, C., and D. Tarbath (2008) Using an inverse-logistic model to describe growth increments of blacklip abalone (*Haliotis rubra*) in Tasmania. *Fishery Bulletin* **106**:58-71.

Hastings, N. A. J., and J. B. Peacock (1975) *Statistical distributions*. London: Butterworths and Co.

Helidoniotis, F., Haddon, M., Tuck, G., and D. Tarbath (2011) The relative suitability of the von Bertalanffy, Gompertz and inverse logistic models for describing growth in blacklip abalone populations (*Haliotis rubra*) in Tasmania, Australia. *Fisheries Research* **112**: 13-21.

Hilborn, R. (2012) The evolution of quantitative marine fisheries management 1985 – 2010. *Natural Resource Modeling* **25**: 122-143.

Hilborn, R. and C.J. Walters (1992) *Quantitative Fisheries Stock Assessment: Choice, Dynamics, and Uncertainty.* Chapman & Hall, London.

Jensen, A.L. (1996) Beverton and Holt life history invariants result from optimal trade-off of reproduction and survival. *Canadian Journal of Fisheries and Aquatic Sciences* **53**:820-822.

Larkin, P.A. (1977) An epitaph for the concept of maximum sustainable yield. *Transactions of the American Fisheries Society*, **106**: 1-11.

Ludwig, D., Hilborn, R. and C.J. Walters (1993) Uncertainty, Resource Exploitation, and Conservation: Lessons from History. *Science* **260**: 17 & 36.

Mace, P.M. and M.P. Sissenwine (1993) How much spawning per recruit is enough? p 101-118 in Smith, S.J, Hunt, J.J. & D. Rivard (eds) Risk Evaluation and Biological Reference Points for Fisheries Management. *Canadian Special Publication of Fisheries and Aquatic Sciences* **120**: 1-442.

Magnuson-Stevens Fishery Conservation and Management Act (2007) U.S. Department of Commerce, National Oceanic and Atmospheric Administration, National Marine Fisheries Service. p170. https://www.fisheries.noaa.gov/s3//dam-migration/msa-amended-2007.pdf (at 30/09/2024)

Maunder, M.N. and A.E. Punt (2013) A review of integrated analysis in fisheries stock assessment. *Fisheries Research* **142**:61-74.

Megrey, B.A. (1989) Review and comparison of age-structured stock assessment models from theoretical and applied points of view. *American Fisheries Society Symposium 6*: 8-48.

Methot, R.D. (1989) Synthetic estimates of historical abundance and mortality in northern anchovy. *American Fisheries Society Symposium* **6**:66-82

Methot, R. D. (1990) Synthesis model: an adaptable framework for analysis of diverse stock assessment data. *International Northern Pacific Fishery Commission Bulleton* 50: 259-277.

Methot, R.D. and I.G. Taylor (2011) Adjusting for bias due to variability of estimated recruitments in fishery assessment models. *Canadian Journal of Fisheries and Aquatic Sciences* **68**:1744-1760.

Miller, K.J., Maynard, B.T. and C.N. Mundy (2009) Genetic diversity and gene flow in collapsed and healthy abalone fisheries *Molecular Ecology* **18**: 200-211.

Miller, K.J., Mundy, C.N., and S. Mayfield (2014) Molecular genetics to inform spatial management in benthic invertebrate fisheries: a case study using the Australian Greenlip Abalone *Molecular Ecology* **23**: 4958-4975.

Mundy, C. (2012) *Using GPS Technology to Improve Fishery Dependent Data Collection in Abalone Fisheries.* FRDC Final Report 2006-029. University of Tasmania and Fisheries Research Development Corporation. 122 p.

Mundy, C., Jones, H. and D. Worthington (2018) *Implementing a Spatial Assessment and Decision Process to Improve Fishery Management Outcomes Using Geo-Referenced Diver Data* FRDC Project No 2011-201. University of Tasmania and Fisheries Research Development Corporation. 202 p.

Mundy, C., Haddon, M. and J. McAllister (2023) Assessment of commercial abalone fisheries *Developments in Aquaculture and Fisheries Science* _42:__ 291-330.

Mundy, C., Venables, W., Haddon, M. and C. Dichmont (2024) *Can spatial fishery-dependent data be used to determine stock status in a spatially structured fishery?* FRDC Final Report 201/026. University of Tasmania and Fisheries Research Development Corporation. 140 p.

Orensanz, J.M., Parma, A.M., Jerez, G., Barahona, N., Montecinos, M. and I. Elias (2005) What are the key elements for the sustainability of "S-fisheries"? Insights from South America. *Bulletin of Marine Science* **76**: 527-556.

Parma, A.M., Orensanz, J.M. (Lobo), Elías, I. and G. Jerez (2003) Diving for shellfish and data: incentives for the participation of fishers in the monitoring and management of artisanal fisheries around southern South America, in: Newman, S.J., Gaughan, D.J., Jackson, G., Mackie, M.C., Molony, B., John, J.S., Kailola, P. (Eds.). Australian Society for Fish Biology Workshop Proceedings - *Towards Sustainability of Data-Limited Multi-Sector Fisheries.*

Pikitch, E., Boersma, P.D., Boyd, I.L., Conover, D.O., Cury, P., Essington, T., Heppell, S.S., Houde, E.D., Mangel, M., Pauly, D., Plagányi, É., Sainsbury, K., and Steneck, R.S. (2012) *Little Fish, Big Impact: Managing a Crucial Link in Ocean Food Webs.* Lenfest Ocean Program. Washington, DC. 108 pp.

Pourtois, J.D., Provost, M.M., Micheli, F. and G.A.De Leo (2022) Modelling the effect of habitat and fishing heterogeneity on the performance of a Total Allowable Catch-regulated fishery. *ICES Journal of Marine Science* **79**: 1467–1480. DOI: 10.1093/icesjms/fsac067

Prager, M.H. (1994) A suite of extensions to a nonequilibrium surplus-production model. *Fishery Bulletin* **92**: 374-389.

Punt, A.E., Butterworth, D.S., de Moor, C.L., De Oliveira, J.A.A. and M. Haddon (2016) Management strategy evaluation: best practices. *Fish and Fisheries* **17**: 303-334. DOI: 10.1111/faf.12104

Punt, A.E., Campbell, R.A., and Smith, A.D.M. (2001) Evaluating empirical indicators and reference points for fisheries management: application to the broadbill swordfish fishery off eastern Australia. *Marine and Freshwater Research* **52(6)**: 819-832.

Punt, A. E. and G.P. Donovan (2007) Developing management procedures that are robust to uncertainty: lessons from the International Whaling Commission. *ICES Journal of Marine Science* **64**: 603–612.

Punt, A.E., Huang, T., Maunder, M.N., 2013. Review of integrated size-structured models for stock assessment of hard-to-age crustacean and mollusc species. *ICES Journal of Marine Science* **70**: 16–33. https://doi.org/10.1093/icesjms/fss185

Punt, A.E., Smith, A.D.M., and G. Cui (2001) Review of progress in the introduction of management strategy evaluation (MSE) approaches in Australia's South East Fishery. *Marine and Freshwater Research* **52**:719-726.

R Core Team (2024). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Russell, E.S. (1931) Some theoretical considerations on the ”overfishing” problem. *Journal du Conseil International pour l'Exploration de la Mer* **6**: 3-20.

Schaefer, M.B. (1954) Some aspects of the dynamics of populations important to the management of the commercial marine fisheries. *Bulletin of the Inter-American Tropical Tuna Commission* **1**: 25-56.

Schaefer, M.B. (1957) A study of the dynamics of the fishery for Yellowfin tuna in the Eastern Tropical Pacific Ocean. *Bulletin of the Inter-American Tropical Tuna Commission* **2**: 247-285.

Smith, T.D. (1988) Stock assessment methods: the first fifty years. In: *Fish Population Dynamics: The Implications for Management.* (ed.) J.A. Gulland. pp 1-33. John Wiley & Sons, Chichester.

Smith, A.D.M. (1994) Management Strategy Evaluation: The Light on the Hill. pp 249-253 *in* Hancock, D.A. (*ed*) *Population Dynamics for Fisheries Management* Australian Society for Fish Biology Workshop Proceedings, Perth 24-25 August 1993. Australian Society for Fish Biology. 298p.

Smith, S.J., Hunt, J.J., and D. Rivard (1993) Risk Evaluation and Biological Reference Points for Fisheries Management. *Canadian Special Publication of Fisheries and Aquatic Sciences* **120**: 1:442.

Smith, A.D.M. (1997) Quantification of objectives, strategies and performance criteria for fishery management plans—an Australian perspective. p291–295. *in* Hancock, D.A., Smith, D.C., Grant, A. and J.P. Beumer (*eds*) *Developing and Sustaining World Fisheries Resources. The State of Science and Management.* Proceedings of the 2nd World Fisheries Congress. CSIRO Publishing, Collingwood.

Smith, A.D.M. and A.E. Punt (2001) The gospel of maximum sustainable yield in fisheries management: birth, crucifixion and reincarnation. *Conservation of Exploited Species.* Reynolds, J.D., Mace, G.M., Redford, K.H. and J.G Robinson (eds). The Zoological Society of London, Cambridge University Press. 505 p.

Sullivan, P. J., H.-L. Lai, and V. F. Gallucci (1990) A catch-at-length analysis that incorporates a stochastic model of growth. *Canadian Journal of Fisheries and Aquatic Sciences* **47**:184–98.

Vasconcellos, M., and Cochrane, K. 2005. Overview of world status of data-limited fisheries: Inferences from landings statistics. *Fisheries Assessment and Management in Data-Limited Situations* **21**: 1-20.

Venzon, D.J. and S.H. Moolgavkar (1988) A method for computing profile likelihood-based confidence intervals. Applied Statistics, 37: 87-94.

# 11. Appendix: MSE Output Object Structure

After running the *do_MSE* function with its output pointed at the R list/object named *out*, we can examine the contents of these objects using the base R function `str(object, max.level=1)`. That can sometimes lead to more information than is immediately useful so it is often easier to use a **codeutils** wrapper function `str1(out)` or `str2(out)`, which now omit the attributes (see their respective help pages).

It will be noted that using `str2` expands lists within lists, however, it will also be noted that some of the objects have lists within lists within lists. Obviously the structure of any of these objects can be examined by drilling further down, as for example one could use `str1(out$zoneCP[[1]])` to see the structure of each of the 56 population's constants definition object (as used in the projections; look at *zoneC* to see what was used during the conditioning).

The main simulated results from the MSE are contained in a large object named **out**, which is a list of 28 other objects, many of which are lists themselves.

## 11.1 Main Output R Objects from aMSE

```
List of 29
 $ tottime     : 'difftime' num 2.94
 $ runtime     : POSIXct[1:1], format: "2025-01-07 07:54:54"
 $ starttime   : POSIXct[1:1], format: "2025-01-07 07:51:58"
 $ glb         :List of 19
 $ ctrl        :List of 13
 $ zoneCP      :List of 56
 $ zoneD       :List of 14
 $ zoneDD      :List of 14
 $ zoneDP      :List of 14
 $ NAS         : NULL
 $ projC       :List of 5
 $ condC       :List of 15
 $ sauout      :List of 10
 $ outzone     :List of 13
 $ production  : num [1:71, 1:6, 1:56] 256 244 233 223 213 ...
 $ condout     :List of 2
 $ HSstats     :List of 2
 $ saudat      : num [1:32, 1:8] 21.5 0.3 130 1 54.3 ...
 $ constants   : num [1:33, 1:56] 6 21.5 0.3 130 1 ...
 $ hsargs      :List of 16
 $ sauprod     : num [1:7, 1:8] 438.882 149.874 20.272 0.341 140.632 ...
 $ zonesummary :List of 2
 $ kobedata    : num [1:8, 1:4] 0.33 0.449 0.258 0.242 0.313 ...
 $ outhcr      :List of 8
 $ scoremed    : num [1:30, 1:7] 42.8 35.3 31.8 31.5 33.4 ...
 $ popmedcatch :List of 8
 $ popmedcpue  :List of 8
 $ popmeddepleB:List of 8
 $ pops        : num [1:56, 1:26] 1 2 3 4 5 6 7 8 9 10 ...
```

The structure of each of the objects contained within **out** will be described with the objects used in the running of the MSE described first, followed by outputs from the conditioning, followed by outputs from the projections.

### 11.1.1 glb : List of 19

A globals object with its components used in many places

```
List of 19
 $ numpop   : num 56
 $ nSAU     : num 8
 $ midpts   : num [1:105] 2 4 6 8 10 12 14 16 ...
 $ Nclass   : num 105
 $ reps     : num 250
 $ hyrs     : num 58
 $ pyrs     : num 30
 $ hyrnames : num [1:58] 1963 1964 1965 1966 ...
 $ pyrnames : int [1:30] 2021 2022 2023 2024 2025 2026 2027 2028 ...
 $ saunames : chr [1:8] "sau6" "sau7" "sau8" ...
 $ SAUpop   : num [1:8] 3 3 5 7 9 9 12 8
 $ larvdisp : num 0.01
 $ indexCE  : int 30
 $ envimpact: NULL
 $ warnfile : chr
"c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EG/warnings.txt"
 $ sauLML   : num 0
 $ sauindex : num [1:56] 1 1 1 2 2 2 3 3 ...
 $ move     : num [1:56, 1:56] 0.995 0.005 0 0 0 0 0 0 ...
 $ SAUnum   : num [1:56] 6 6 6 7 7 7 8 8 ...
```

### 11.1.2 ctrl : List of 13

Variables used when controlling the MSE run.

```
List of 13
 $ runlabel   : chr "Base_Case"
 $ datafile   : chr "saudataEG.csv"
 $ controlfile: chr "controlEG.csv"
 $ reps       : num 250
 $ randseed   : num 3543304
 $ randseedP  : num 0
 $ withsigR   : num 0.35
 $ withsigB   : num 0.1
 $ withsigCE  : num 0.1
 $ catches    : num 58
 $ projection : num 30
 $ bysau      : num 1
 $ rundir     : chr "c:/Users/malco/DropBox/A_codeR/aMSEGuide/runs/EG"
```

### 11.1.3 projC : List of 5

Other constants used during the projections. Note that projyrs = glb$pyrs, which is now used more often in the code-base and eventually it will be deprecated in *projC*.

```
List of 5
 $ projLML: num [1:30, 1:2] 145 145 145 145 145 ...
 $ projyrs: num 30
 $ Sel    : num [1:105, 1:56, 1:30] 9.48e-35 2.84e-34 ...
 $ SelWt  : num [1:105, 1:56, 1:30] 5.41e-38 1.44e-36 ...
 $ histCE : num [1:29, 1:8] NA NA NA NA NA ...
```

The NA in the histCE vector reflects the fact that in SAU6 (the 8 SAU are across the columns), like SAU13, has no CPUE data until 2000.

### 11.1.4 condC : List of 7

More constants used during conditioning; *compdat* holds the observed size-composition data from each sau in *lfs*, as well as their relative proportions in *palfs*.

```
List of 15
 $ histCatch   : num [1:58, 1:8] 0 1 2 8 22 31 24 29 39 33 ...
 $ histyr      : num [1:58, 1:2] 1963 1964 1965 1966 1967 ...
 $ histCE      : num [1:29, 1:8] NA NA NA NA NA ...
 $ yearCE      : num [1:29] 1992 1993 1994 1995 1996 ...
 $ initdepl    : num [1:8] 1 1 1 1 1 1 1 1
 $ compdat     :List of 2
  ..$ lfs  : num [1:38, 1:31, 1:8] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ palfs: num [1:31, 1:8] 0 0 0 0 0 0 0 0 0 0 ...
 $ recdevs     : num [1:58, 1:8] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
 $ parsin      : logi FALSE
 $ optpars     : NULL
 $ sizecomp    : num 1
 $ lffiles     : chr "lf_WZ90-20.csv"
 $ poprec      : num [1:56, 1:3] 6 6 6 7 7 7 8 8 8 8 ...
 $ yearFIS     : NULL
 $ fisindexdata: NULL
 $ fissettings : NULL
```

### 11.1.5 condout : List of 2

sauZone contains the conditioning dynamics across hyrs years summed across populations to the sau scale for each of the dynamic variables. This ignores replicates, as found in zonePsau, and is there more for convenience than needed. ssq is for each sau and reflects the difference between the observed cpue and the predicted.

```
List of 2
 $ sauZone:List of 9
  ..$ matB    : num [1:58, 1:9] 439 438 436 429 409 ...
  ..$ expB    : num [1:58, 1:9] 415 414 413 409 395 ...
  ..$ midyexpB: num [1:58, 1:9] 0 431 430 428 421 ...
```

```
  ..$ catch   : num [1:58, 1:9] 0 1 2 8 22 ...
  ..$ recruit : num [1:58, 1:9] 259139 259078 258961 258478 257088 ...
  ..$ harvestR: num [1:58, 1:9] NaN 0.00232 0.00465 0.0187 0.05228 ...
  ..$ cpue    : num [1:58, 1:9] NA 437 436 432 422 ...
  ..$ deplsB  : num [1:58, 1:9] 1 0.998 0.994 0.977 0.933 ...
  ..$ depleB  : num [1:58, 1:9] 1 0.999 0.996 0.985 0.953 ...
 $ ssq    : num [1:8] 385 1801 10159 2444 777 ...
```

### 11.1.6 zoneDD : List of 13

The dynamics object generated during conditioning. Each of the dynamic's matrices are 2-dimensional (years vs populations), the predicted size-composition matrices (*catchN*, *Nt*, and *NumNe*) are 3-dimensional (size-class vs year vs population).

```
List of 14
 $ SAU     : chr [1:56] "sau6" "sau6" "sau6" ...
 $ matureB : num [1:58, 1:56] 311 311 309 304 ...
 $ exploitB: num [1:58, 1:56] 293 293 292 289 ...
 $ midyexpB: num [1:58, 1:56] 0 305 304 303 ...
 $ catch   : num [1:58, 1:56] 0 0.707 1.414 5.655 ...
 $ harvestR: num [1:58, 1:56] 0 0.00232 0.00465 0.01869 ...
 $ cpue    : num [1:58, 1:56] 437 437 436 432 ...
 $ recruit : num [1:58, 1:56] 191443 191398 191312 190957 ...
 $ deplsB  : num [1:58, 1:56] 1 0.998 0.994 0.978 ...
 $ depleB  : num [1:58, 1:56] 1 0.999 0.996 0.985 ...
 $ catchN  : num [1:105, 1:58, 1:56] 0 0 0 0 0 0 0 0 ...
 $ Nt      : num [1:105, 1:58, 1:56] 1.91e+05 7.45e-03 1.50e-01 2.18 ...
 $ NumNe   : num [1:105, 1:58, 1:56] 0 0 0 0 0 0 0 0 ...
 $ outfis  : NULL
```

### 11.1.7 zoneCP : List of 56

The constants for each projected population in a list of lists (see str1(out$zoneCP[[1]])).

- ..$ :List of 21 .. ..- attr(*, "class")= chr "abpop"
- ..$ :List of 21 .. ..- attr(*, "class")= chr "abpop"
- ..$ :List of 21 .. ..- attr(*, "class")= chr "abpop"
- ..$ :List of 21 .. ..- attr(*, "class")= chr "abpop"
- …
- ..- attr(*, "class")= chr "zoneC"

Each *abpop* is a list of 21 objects.

```
List of 21
 $ Me      : num 0.154
 $ R0      : num 192282
 $ B0      : num 311
 $ ExB0    : num 293
 $ MSY     : num 14.5
 $ MSYDepl : num 0.342
```

```
 $ bLML    : num 132
 $ scalece : Named num 1.41
 $ qest    : num 4.76
 $ lambda  : num 0.75
 $ SaM     : num 99.5
 $ popdef  : Named num [1:22] 21.6 130.4 ...
 $ LML     : Named num [1:58] 127 127 127 127 127 ...
 $ G       : 'STM' num [1:105, 1:105] 1.67e-09 4.54e-08 ...
 $ Maturity: num [1:105] 3.02e-10 4.73e-10 ...
 $ WtL     : num [1:105] 0.000571 0.005075 ...
 $ Emergent: num [1:105] 3.74e-15 6.43e-15 ...
 $ Select  : num [1:105, 1:88] 6.81e-45 3.49e-44 ...
 $ SelWt   : num [1:105, 1:88] 3.89e-48 1.77e-46 ...
 $ MatWt   : num [1:105] 1.72e-13 2.40e-12 ...
 $ SAU     : chr "sau6"
```

### 11.1.8 zoneDP : List of 14

The much larger dynamics object generated during the projections. Each of the dynamic's matrices are 3-dimensional having years x populations x replicates. The first year in the model relates to the unfished state or the initial state, hence the zero for mid-year exploitable biomass (midyexpB), catch, acatch, etc.

```
List of 14
 $ SAU     : num [1:56] 6 6 6 7 7 7 8 8 ...
 $ matureB : num [1:88, 1:56, 1:250] 311 311 309 304 ...
 $ exploitB: num [1:88, 1:56, 1:250] 293 293 292 289 ...
 $ midyexpB: num [1:88, 1:56, 1:250] 0 305 304 303 ...
 $ catch   : num [1:88, 1:56, 1:250] 0 0.707 1.414 5.655 ...
 $ acatch  : num [1:88, 1:8, 1:250] 0 1 2 8 ...
 $ harvestR: num [1:88, 1:56, 1:250] 0 0.00232 0.00465 0.01869 ...
 $ cpue    : num [1:88, 1:56, 1:250] 437 437 436 432 ...
 $ cesau   : num [1:88, 1:8, 1:250] 0 437 436 432 ...
 $ catsau  : num [1:88, 1:8, 1:250] 0 1 2 8 ...
 $ recruit : num [1:88, 1:56, 1:250] 191443 191398 191312 190957 ...
 $ deplsB  : num [1:88, 1:56, 1:250] 1 0.998 0.994 0.978 ...
 $ depleB  : num [1:88, 1:56, 1:250] 1 0.999 0.996 0.985 ...
 $ TAC     : num [1:88, 1:250] 0 20 86 290 ...
```

### 11.1.9 NAS : List of 2

The size composition objects following projection. Each array is a 4-dimensional object having size-class x year x population x replicates. Nt is the population structure while catchN is the expected size-composition of the catch by population.

- ..$ Nt : num [1:105, 1:88, 1:56, 1:100] 1.61e+05 1.97e-06 8.07e-05 …
- ..$ catchN: num [1:105, 1:88, 1:56, 1:100] 0 0 0 0 0 0 0 0 0 0 …

When saving the **out** object from a run of the *do_MSE* function, if the *includeNAS* argument is set = FALSE, then the NAS object will be set = *NULL*. This option exists because the NAS object, made up of multiple 4-D arrays can be very large. so the output can increase from a few 100s of Megabytes, or almost 2 Gigabytes.

### 11.1.10 sauout : a List

sauout is one of the major outputs from the MSE. It contains the dynamics for each sau in the zone. Including:

```
List of 10
 $ matureB : num [1:88, 1:8, 1:250] 439 438 436 429 409 ...
 $ exploitB: num [1:88, 1:8, 1:250] 415 414 413 409 395 ...
 $ midyexpB: num [1:88, 1:8, 1:250] 0 431 430 428 421 ...
 $ catch   : num [1:88, 1:8, 1:250] 0 1 2 8 22 ...
 $ acatch  : num [1:88, 1:8, 1:250] 0 1 2 8 22 ...
 $ harvestR: num [1:88, 1:8, 1:250] NaN 0.00232 0.00465 0.0187 0.05228 ...
 $ cpue    : num [1:88, 1:8, 1:250] 0 437 436 432 422 ...
 $ recruit : num [1:88, 1:8, 1:250] 259139 259078 258961 258478 257088 ...
 $ deplsB  : num [1:88, 1:8, 1:250] 1 0.998 0.994 0.977 0.933 ...
 $ depleB  : num [1:88, 1:8, 1:250] 1 0.999 0.996 0.985 0.953 ...
```

If the *includeNAS* argument is set = *TRUE*, then sauout would be a list of 12 objects that would include:

- ..$ catchN : num [1:105, 1:88, 1:8, 1:100] 0 0 0 0 0 0 0 0 0 0 …
- ..$ Nt : num [1:105, 1:88, 1:8, 1:100] 2.59e+05 9.25e-03 1.90e-01 2.81

### 11.1.11 outzone : List of 12

outzone contains the projection dynamics across all years summed across populations and sau to give the zone scale changes for each of the dynamic variables, including catchN and Nt (being only 3-D arrays tends to reduce their size. outzone is another major output from the MSE.

```
List of 13
 $ matureB : num [1:88, 1:250] 14541 14523 14445 14184 ...
 $ exploitB: num [1:88, 1:250] 14587 14578 14530 14361 ...
 $ midyexpB: num [1:88, 1:250] 0 15132 15113 15035 ...
 $ catch   : num [1:88, 1:250] 0 20 86 290 ...
 $ acatch  : num [1:88, 1:8, 1:250] 0 1 2 8 ...
 $ TAC     : num [1:88, 1:250] 0 20 86 290 ...
 $ harvestR: num [1:88, 1:250] 0 0.00132 0.00569 0.01929 ...
 $ cpue    : num [1:88, 1:250] 0 405 399 395 ...
 $ recruit : num [1:88, 1:250] 6705236 6704327 6700497 6687410 ...
 $ deplsB  : num [1:88, 1:250] 1 0.999 0.993 0.975 ...
 $ depleB  : num [1:88, 1:250] 1 0.999 0.996 0.984 ...
 $ catchN  : num [1:105, 1:88, 1:250] 0 0 0 0 0 0 0 0 ...
 $ Nt      : num [1:105, 1:88, 1:250] 6.71e+06 2.00e-02 4.34e-01 6.82 ...
```

### 11.1.12 outhcr

The objects contained in this are determined by a custom function written within each jurisdiction. Each harvest strategy is very different from every other jurisdiction's harvest strategy, and each reaches its outcomes in different ways. The *outhcr* object is produced by a

custom function which is pointed to by the *do_MSE* argument *makeouthcr*. This uses a valuable method in the R language where a function can have a different function as an argument so that different functions can be applied in different circumstances (such as jurisdictions having different harvest strategies). The listing below is for Tasmania but would be different for South Australia and different again for Victoria. Note in Tasmania's case each object is a 3-D array of the projection years (30) x sau (8) X replicates (100; in real-life scenario testing one would use perhaps 250 replicates as 100 might not be sufficient to define the total variability expected).

```
List of 8
 $ g1s      : num [1:30, 1:8, 1:250] 3.62 9.22 9.17 6.66 ...
 $ g4s      : num [1:30, 1:8, 1:250] 6.08 6.49 9.17 9.31 ...
 $ targsc   : num [1:30, 1:8, 1:250] 2.43 3.93 5.83 6.68 ...
 $ finalsc  : num [1:30, 1:8, 1:250] 3.46 5.1 7 7.33 ...
 $ index    : num [1:30, 1:8, 1:250] 4 6 7 8 10 9 8 9 ...
 $ catchmult: num [1:30, 1:8, 1:250] 0.85 1 1.05 1.1 1.2 1.15 1.1 1.15 ...
 $ metaflag : num [1:30, 1:8, 1:250] 0 0 0 0 0 0 0 0 ...
 $ cetarg   : num [1:30, 1:8, 1:250] 122 122 122 122 ...
```

The projected final score (*finalsc*), in Tasmania, is that which is used to select which aspirational catch multiplier (hcr in hsargs), is used in an sau. These data are used to generate the plots seen in the scores tab in the aMSE output (as in *finalscores_sauX.png* files).

### 11.1.13 production 3D array

A 3d array of harvest rates x 6 dynamic variables x numpop, where the dynamic variables are 'ExB' exploitable biomass, 'MatB', mature biomass, 'AnnH', the annual harvest rate, 'Catch' the yield, the maximum of which is the MSY, 'Deplet', the equilibrium depletion that the harvest rate leads to, and 'RelCE', the relative cpue predicted at that equilibrium.

- num [1:81, 1:6, 1:56] 256 244 233 223 213 …
- attr(*, "dimnames")=List of 3 ..$ : chr [1:81] "0" "0.005" "0.01" "0.015" … harvest rates ..$ : chr [1:6] "ExB" "MatB" "AnnH" "Catch" … dynamic variables ..$ : chr [1:56] "1" "2" "3" "4" … population number

### 11.1.14 condout : List of 2

Where sauZone is a list of 9 components 2-D matrices containing the conditioned state of each of 8 sau (in this case 58-year's of historical catches) as well as the complete zone (for Tasmania) in the ninth column.

```
List of 2
 $ sauZone:List of 9
  ..$ matB    : num [1:58, 1:9] 439 438 436 429 409 ...
  ..$ expB    : num [1:58, 1:9] 415 414 413 409 395 ...
  ..$ midyexpB: num [1:58, 1:9] 0 431 430 428 421 ...
  ..$ catch   : num [1:58, 1:9] 0 1 2 8 22 ...
  ..$ recruit : num [1:58, 1:9] 259139 259078 258961 258478 257088 ...
  ..$ harvestR: num [1:58, 1:9] NaN 0.00232 0.00465 0.0187 0.05228 ...
  ..$ cpue    : num [1:58, 1:9] NA 437 436 432 422 ...
  ..$ deplsB  : num [1:58, 1:9] 1 0.998 0.994 0.977 0.933 ...
```

```
  ..$ depleB  : num [1:58, 1:9] 1 0.999 0.996 0.985 0.953 ...
 $ ssq    : num [1:8] 385 1801 10159 2444 777 ...
```

### 11.1.15 HSstats : List of 2

Two HS performance measures, the sum of catches across the first 5 and first 10 years of the projections. In this case, this is for each of SAU used separately, and for the combined zone total.

```
List of 2
 $ sum10: num [1:250, 1:9] 93.1 84 77.4 79.2 92.4 ...
 $ sum5 : num [1:250, 1:9] 33.7 30.9 28.1 28 34.7 ...
```

### 11.1.16 saudat : is an array of constants

This is a copy of the input constants read in from *saudata_Scenario.csv* in case comparisons are wanted.

```
 num [1:32, 1:8] 21.5 0.3 130 1 54.3 ...
```

### 11.1.17 constants : an array of constants

The actual biological and fishery constants used to define each population.

```
 num [1:33, 1:56] 6 21.5 0.3 130 1 ...
```

### 11.1.18 hsargs a copy of the hsargs

This is a copy of the hsargs used in the example scenario and will depend upon the HS used. For Tasmania it is a List of 16:

```
List of 16
 $ mult      : num 0.1
 $ wid       : num 4
 $ targqnt   : num 0.55
 $ maxtarg   : num [1:8] 150 150 150 150 150 150 150 150
 $ pmwts     : num [1:3] 0.65 0.25 0.1
 $ hcr       : num [1:10] 0.25 0.75 0.8 0.85 0.9 1 1.05 1.1 1.15 1.2
 $ hcrm3     : num [1:10] 0.25 0.75 0.8 0.85 0.9 1 1.1 1.2 1.25 1.3
 $ startCE   : num 2000
 $ endCE     : num 2019
 $ metRunder : num 0
 $ metRover  : num 0
 $ decrement : num 1
 $ pmwtSwitch: num 0
 $ stablewts : num [1:3] 0.8 0.15 0.05
 $ hcrname   : chr "constantrefhcr"
 $ printmat  : NULL
```

### 11.1.19 sauprod : a 7 x nsau matrix

A matrix of productivity characteristics for each sau. These include the $B0$, $B_{MSY}$, $MSY$, $D_{MSY}$ (depletion at $MSY$ and $B_{MSY}$), and $CE_{MSY}$, the predicted cpue at $MSY$.

*Productivity properties by SAU*

| | sau6 | sau7 | sau8 | sau9 | sau10 | sau11 | sau12 | sau13 |
|---|---|---|---|---|---|---|---|---|
| B0 | 438.88 | 872.17 | 524.48 | 2438.88 | 2071.96 | 4152.79 | 3126.26 | 915.91 |
| Bmsy | 149.87 | 272.80 | 166.07 | 748.36 | 616.67 | 1257.99 | 951.36 | 303.99 |
| MSY | 20.27 | 43.74 | 24.94 | 122.49 | 102.78 | 207.63 | 156.60 | 41.74 |
| Dmsy | 0.34 | 0.31 | 0.32 | 0.31 | 0.30 | 0.30 | 0.30 | 0.33 |
| CEmsy | 140.63 | 115.89 | 194.81 | 198.17 | 142.31 | 140.29 | 91.11 | 75.48 |
| Hmsy | 0.20 | 0.20 | 0.17 | 0.18 | 0.17 | 0.18 | 0.17 | 0.19 |
| Bexmsy | 91.32 | 191.66 | 126.23 | 600.53 | 536.12 | 1018.44 | 792.97 | 193.29 |

### 11.1.20 scoremed

This contains arrays of the outputs from the HS for each sau and each replicate, along with the median values for each of the arrays.

```
num [1:30, 1:7] 42.8 35.3 31.8 31.5 33.4 ...
```

# 12. The JurisdictionHS File or Package

## 12.1 Use an R source File or an R Package?

The abalone harvest strategies that have been implemented in different jurisdictions around Australia differ markedly from each other and do so in multiple ways. Initially, in the early planning stage of the MSE R package (**aMSE**), it was envisaged that each harvest strategy would be included as a series of R functions within the package. However, it quickly became apparent that such an approach would fix each harvest strategy in a single configuration, which would defeat one of the major advantages of an MSE framework. Thus, it was decided that a better approach would be:

1. Each jurisdiction to develop and maintain a separate R package that encapsulates all the functions required by their respective harvest strategies and their interaction with **aMSE**. The use of an R package is likely to be the most efficient option and would help ensure maintenance and modification of the HS within the MSE would adhere to good practices with respect to documentation and transparency. This would also aid each jurisdiction in simplifying the task of becoming more transparent, defensible, and open about each harvest strategy.

However, a viable option, especially when under development, would be:

2. To have each jurisdictions harvest strategy (HS) defined as a series of functions and constants in a separate R source file that could be *source*'d into the R environment prior to running the MSE,

The Tasmanian implementation of the MSE started by using a *source* R file but has moved the functions developed into its own defined R package **TasHS**. The list of constants used by the TasHS (defined as *hsargs* within **aMSE** see later) must still be defined as a global variable when running the MSE.

### 12.1.1 Important Caveat

It must be emphasized here that any statement in this document concerning the structure and operation of the Tasmanian harvest strategy, which will be used as an example, must not be taken as a formal statement of the HS. That can be found in the actual harvest strategy document (Bradshaw, 2018), which is currently undergoing changes as a result of a formal review as well as suggested changes as a result of the MSE testing. Thus, whatever description is given here must not be taken as "the" Tasmanian harvest strategy at any future time. For details of the implementation of the Tasmanian harvest strategy, read Bradshaw (2018) and the separate documentation to the **TasHS** R package.

This chapter/section is going to be more technical than others as it involves a description of the interaction between the HS functions and the internal functions running the replicate projections within the **aMSE** R package.

### 12.1.2 Where the Harvest Strategy is used in aMSE

Running a scenario within **aMSE** entails first conditioning the Operating Model within the MSE using biological information relating to maturity, growth, natural mortality, and other productivity related factors. Then any historical fishery data involving catches, catch rates, survey results, and catch sampling results can be used to further condition the model so that

its dynamics more closely match the observed dynamics of the fishery being explored. The conditioning period is taken to be that period over which the harvest strategy being tested was not applied (within the **aMSE** code, this is denoted as a period of *hyrs*, as in history years). Thus, the assumption is that the harvest strategy begins in the first year of the projections (*hyrs + 1*) and will continue to be applied in each year of the projections. In Tasmania, the harvest strategy was used informally by the Institute of Marine and Antarctic Sciences (IMAS) to recommend aspirational catches for each statistical block (sau) since 2016 and used formally to identify the block aspirational catches since 2020. Thus, if using data up to the end of 2020, one could start the predictions from 2020 or from 2021. For comparability with other jurisdictions 2021 is used so the assumption is that the HS is used from the start of projections.

The stock dynamics, within the model, are stored in an object called *zoneDP* (as in zone Dynamics + Projections; see the *R_object_structure* chapter). For each year (and replicate) within each population, these dynamics are updated to contain another year of the dynamics, which involves the cycle of half of natural mortality, growth, fishing mortality, the second half of natural mortality, and recruitment (with larval movement).

At the start of each year of the projections, within the function *doprojections()*, data required by the particular harvest strategy being applied is sampled from the MSE's *zoneDP* object.

Depending on the HS there may be a need to obtain samples of one or more of:

1. the commercial cpue,
2. any available fishery independent index of abundance,
3. the numbers-at-size in the commercial catch, and
4. the numbers-at-size predicted for a fishery independent survey, if available.

In addition, the actual catches and the aspirational catches are also required, though these are expected to be the same in South Australia. Currently, no other data streams are supported by the **aMSE**'s operating model. All these are included in the MSE function *doprojections()* using the following code.

```
  hcrout <- makeouthcr(glb,hsargs)  # make an object ready to be filled
# Within aMSE, in the function doprojections is the following loop
  for (year in startyr:endyr) {   # startyr = hyrs+1, endyr = hyrs + pyrs
    if (verbose) cat(year,"   ")  # indicate status of run by year countdown
    for (iter in 1:reps) { # reps = number of replicate projections used
      hcrdata <- getdata(sampleCE,sampleFIS,sampleNaS, # generate the data
                         sauCPUE=zoneDP$cesau[,,iter], # needed by the HS by
                         sauacatch=zoneDP$acatch[,,iter], # calling getdata
                         sauNAS=list(Nt=zoneDP$Nt[,,,iter],
                         catchN=zoneDP$catchN[,,,iter],
                         NumNe=zoneDP$NumNe[,,,iter]),
                         year=year,decrement=hsargs$decrement)
      hcrout <- hcrfun(hcrdata,hsargs,saunames=glb$saunames) # run the hcr
      popC <- calcpopC(hcrout,exb=zoneDP$exploitB[year-1,,iter], # hcr output
                       sauindex,sigmab=sigmab) #= acatch by SAU or TAC by zone
      # calcpopC has a fleet dynamics model that subdivides the acatch or TAC
      # among the populations within each SAU
    # ...  other code
    }
    # ...  other code
}
```

The functions relating to the harvest strategy (HS) used in the projections within **aMSE** have specific names. However, this is not a constraint on the user as some of the arguments of the function *doprojections()* are the names given to the functions representing *sampleCE()*, *sampleFIS()*, and *sampleNAS()*, each of which can be named as the user wishes (see the help file for *doprojections()* or *do_MSE()*). The same thing goes for the *getdata()* function, which calls the three sampling functions. In **TasHS**, the function that does the data sampling is called *tasdata()* and, hence, one of the arguments of *do_MSE()* and then *doprojections()*, which is inside *do_MSE()*, is therefore set as *'getdata = tasdata'*.

Also included in the HS package or source file is the hcrfun, which in Tasmania is the *mcdahcr()* function this takes the input data from *getdata()*, runs the implementation of the HS and outputs, at least, the predicted TAC by zone or aspirational catch by SAU, or both. This output is then put into the *calcpopC()* function (in Tas this is *calcexpectpopC()*), which uses a relatively simple model of the fleet dynamics (how the divers distribute the quota they have available to them) to determine how the aspirational TAC or *acatches* by SAU are subdivided among the populations within each SAU. Ideally, the predicted catch by SAU, which is all that can be observed in the real fishery, should approximate how the catches are distributed among the real SAU. This is less difficult to arrange in Tasmania now that there have been meta-rules included in the HS that constrain how far the divers may deviate from the proposed aspirational catch per SAU (statistical block in Tas). That also allows for diagnostic plots of the predicted deviations to be generated from the projections that are used to monitor the MSE simulation performance.

The idea being used is that the *getdata()* function has arguments that define each of these functions, and also has arguments that reference particular data fields from the dynamic object (*zoneDP*) so that the data required by the harvest strategy can be sampled or generated by each of the three 'sample' functions. Within the *doprojections()* function the structure of *zoneDP* consists of:

..$ SAU : num [1:56] 6 6 6 7 7 7 8 8 8 8 …
..$ matureB : num [1:88, 1:56, 1:100] 322 321 320 315 301 …
..$ exploitB: num [1:88, 1:56, 1:100] 281 305 303 298 284 …
..$ midyexpB: num [1:88, 1:56, 1:100] 303 330 329 328 323 …
..$ catch : num [1:88, 1:56, 1:100] 0 0.719 1.441 5.763 15.85 …
..$ acatch : num [1:88, 1:8, 1:100] 0 1 2 8 22 …
..$ harvestR: num [1:88, 1:56, 1:100] 0 0.00236 0.00475 0.01932 …
..$ cpue : num [1:88, 1:56, 1:100] 359 390 389 385 373 …
..$ cesau : num [1:88, 1:8, 1:100] 0 386 385 381 369 …
..$ catsau : num [1:88, 1:8, 1:100] 0 1 2 8 22 …
..$ recruit : num [1:88, 1:56, 1:100] 161023 160996 160943 160725 …
..$ deplsB : num [1:88, 1:56, 1:100] 1 0.998 0.994 0.978 0.934 …
..$ depleB : num [1:88, 1:56, 1:100] 0.853 1.084 1.079 1.061 1.011 …
..$ Nt : num [1:105, 1:88, 1:56, 1:100] 1.61e+05 1.97e-06 …
..$ catchN: num [1:105, 1:88, 1:56, 1:100] 0 0 0 0 0 0 0 0 0 0 …
..$ NumNe : num [1:105, 1:88, 1:56, 1:100] 1.61e+05 1.97e-06 …
..$ TAC : num [1:88, 1:100] 0 20 86 290 775 …

Obviously, if one looks at this after a given run the values seen will differ from these, but the structure remains the same (in this case the 88 will reflect the hyrs + pyrs, the 56 the number of populations used, the 100 the number of replicates used, and the 105 the number of size classes used).

The 4-dimensional arrays holding the numbers-at-size arrays are removed from *zoneDP* and put into *NAS* after being output from *do_MSE()*. However, within *doprojections()* each object can be individually referenced by population or iteration. Note, in the code above the data pushed into *getdata* is *zoneDP$cesau*, *zoneDP$acatch*, and *zoneDP$NAS* (a combination of *Nt* and *catchN*).

### 12.1.3 Inclusion of a Jurisdiction's HS

If using a source file then the constants needed by the HS functions could be included in the source file. If using a library (R package) then either a source file only containing the constants could be used or the *hsargs* list could be entered explicitly in the R code used to run an MSE scenario in **aMSE**. For example:

```
# hsfile <- "TasHS1_Tas.R"
# source(paste0(rundir,"/",hsfile)) # if using a source file of R functions
library(TasHS)                    # if using an independent R library the HS
# constants in a list still need to be made into a global list object
hsargs <- list(mult=0.1, # expansion factor for cpue range when calc the targqnt
               wid = 4, # number of years in the grad4 PM
               targqnt = 0.55, # quantile defining the cpue target
               maxtarg = c(150,150,150,150,150,150,150,150), # max cpue Target
               pmwts = c(0.65,0.25,0.1),  # relative weights of PMs
               hcr = c(0.25,0.75,0.8,0.85,0.9,1,1.05,1.1,1.15,1.2),# multipliers
               hcrm3 = c(0.25,0.75,0.8,0.85,0.9,1,1.1,1.15,1.2,1.25),
               startCE = 2000, # used in constant reference period HS
               endCE = 2011,   # used in constant reference period HS
               metRunder = 0,  # should the metarules be used. o =
               metRover = 0,   # use metarules
               decrement=1, # use fishery data up to the end of the time series
               pmwtSwitch = 0, # number of years after reaching the targCE to
               stablewts = c(0.4, 0.5, 0.1), # replace pmwts with stablewts
               hcrname="mcdahcr",     # the name of the HCR used
               printmat=NULL)   # An option required in some jurisdictions
# This approach would place the functions making up the Tasmanian Harvest Strategy
# into the main R environment ready for use in a Tasmanian setting.
```

Each of the members of *hsargs* fulfils a specific task within the harvest strategy. In the case of the Tasmanian HS the items within *hsargs* have the following intentions:

1. *mult* the multiplier on the performance measure bounds to expand them both upwards and downwards. default value = 0.1 = 10 percent increase and decrease.
2. *wid* the number of years over which to calculate the gradient, default value = 4, meaning four years.
3. *targqnt* what quantile of the distribution of cpue to use as the target, default value = 0.55.
4. *maxtarg* is the maximum cpue target for each sau, this will vary depending on which sau one is working with.
5. *pmwts* what weights to give to each of the performance measures. Their order is targetCE, grad4, and grad1 with default values = c(0.65, 0.25,0.1).
6. *hcr* is used to translate the overall score between 0 - 10, into a multiplier for the previous aspirational catch.
7. *hcrm3* multipliers used instead of hcr when meta-rule 3 is active.
8. *startCE* is the starting year for CPUE used in Tasmania, the default = 1992. Also used as the start year when using *constrefhcr()* instead of *mcdahcr()*
9. *endCE* the final year of CPUE used in Tasmania when using the constrefhcr, otherwise it is ignored.
10. *metRunder* = 2 meta rule 1, when the cpue is below the targCPUE how many consecutive years must cpue rise before a reduction is NOT made. If set to zero then meta-rule 1 is not used.

11. *metRover* = 2 meta rule 2, how many consecutive years of increase above the targCPUE must cpue occur before an increase can be made. If set to zero then meta-rule 2 is not used.
12. *decrement* = 1 working in the year after the data are available, if decrement = 1 means use all data up to the latest year even if only partial (as in TAS). Using decrement = 2 means omit the final (partial) year of data from the assessment (as in SA).
13. *pmwtSwitch* = 4 how many consecutive years of increase above the targCPUE must occur before switching the performance measure weights from pmwts to stablewts and switching the *acatch* multipliers from hcr to hcr3. If *pmwtSwitch* = 0 then no change is made to the PM weights or to the *acatch* multipliers.
14. *stablewts* = c(0.4,0.5,0.1), what performance measure weights should be used once *pmwtSwitch* is triggered.
15. *hcrname* = "constantrefhcr" the name of the harvest control rule used. Alternatives in Tasmania could be *consthcr()* (a constant catch HS) and *mcdahcr()*, that used a constant reference period defined by startCE and endCE.
16. *printmat* = NULL an option to print out a matrix during development, only used in some jurisdictions.

## 12.2 What Must be Included in the HS File or Package

All harvest strategies (HS) have arguments, settings and other constants that can be altered to influence the performance of the HS (hence *hsargs*). The Tasmanian HS uses the list as described in the code sequence just above. If, somehow, the HS does not require such constants then *hsargs* should be set = *NULL* as *hsargs* is still required as a global variable by the **aMSE** code.

The **TasHS** package currently contains 23 functions but, in terms of interacting with the MSE only eight are used directly. The other fifteen are used by these interacting functions. Within the HS file all of these functions can be called whatever the programmer wishes as they are referenced as arguments for functions within **aMSE**.

The eight required functions are always needed, even if the data they are supposed to generate is not used. In such cases then a simple function returning *NULL* will suffice. For example, if no FIS data is currently used in an HS then one might include a function:

```
tasFIS <- function() { # currently no FIS data is used in TAS
  return(NULL)          # though this may change
} # end of tasFIS
```

The first three of the required functions are the sampling functions that take output from the operating model and sample the respective data for input into the selected jurisdictions HS. The fourth and fifth functions use the first three to sample simulated fishery data from the MSE projections and then run the harvest control rule. The sixth required function uses the outputs from hcrfun to generate the expected or aspirational catches by *sau* and by *zone*, the sum of the *sau* catches should equal the total *zone* catch and both are the same as the expected TAC. The last two functions relate to extracting the internal components of the harvest control rule (the scores and weights, etc) so that the operation of the *hcr* can be

monitored. The *makeouthcr()* function is designed to capture the *hcr* information as it is generated, while the *HSPMs()* is used to reconstruct the *hcr* components after all projections are completed (which is likely to be more efficient computationally, especially if it is not required!).

The eight function names in the listing below are the names as defined in the code base of **aMSE**. The names used in each jurisdiction can be anything. The start of the call to *do_MSE()* illustrates how each of the following is an argument in the *do_MSE()* function and how this provides the opportunity to allocate each of these functions that are internal to **aMSE** to jurisdiction specific functions residing in an external source file or R package. The eight functions after *hsargs*, from *hcrfun* downwards, all need a definition within the jurisdiction's R package or R source file, even if they only return *NULL*. Details are given below but are also available in the help for *?do_MSE*.

```
out <- do_MSE(rundir,controlfile, # needs a global definition
              hsargs=hsargs,    # defined as global object, see above
              hcrfun=mcdahcr,   # the main HS function from TasHS
              sampleCE=tasCPUE, # processes cpue data, from TasHS
              sampleFIS=tasFIS, # processes FIS data (see above), from TasHS
              sampleNaS=tasNaS, # processes Numbers-at-Size data, from TasHS
              getdata=tasdata,  # extracts data from zoneDP objects, from TasHS
              calcpopC=calcexpectpopC, #spreads SAU acatch across populations
              makeouthcr=makeouthcr, # generates updateable HS stats object
              fleetdyn=NULL, # an optional function defining the fleet dynamics
              scoreplot=plotfinalscores, # plots hcr and total scores from HS
              plotmultflags=plotmultandflags,#plots TAC/acatch multipliers and
                                             # meta rule flags
              ...
              ...

)
```

1. *sampleCE()* is used to sample or select data from the cpue predicted by the MSE. It should also include uncertainty (variability) from the predicted cpue by sau. The output from sampleCE should be the projected cpue data used by the HS function (see *hcrfun*).
2. *sampleNaS()* is used to sample numbers-at-size data from the commercial catch if it is used in the HS. If such data is not used in the HS a function is still required but it can simply return *NULL*. The output should be the numbers-at-size data expected to be used by the HS function (see *hcrfun*).
3. *sampleFIS()* is used to sample both cpue and numbers-at-size as if they came from fishery independent surveys. Again, if such data is not used then a function is still required but it can simply return *NULL* (as in *tasFIS()* immediately above). The output should be any FIS data used by the HS function (see *hcrfun*).
4. *getdata()* calls the three sampling functions as arguments and must expect to receive for each sau, the projected years in each replicate of: 1) the predicted time series of cpue, the time series of expected catches, and a large object containing the expected distribution of the numbers-at-size in the commercial catches, the numbers-at-size in the population prior to fishing mortality, and the numbers-at-size in the population

after fishing mortality. The latter two would be needed if a fishery independent survey were to be used. The output from *getdata()* should be a list containing the data needed by the jurisdictions HS. It should take on the format that suits the programmer as it is they who will be writing the *getdata* and *hcrfun* functions. For example, in **TasHS**, the output of *getdata()* is a list of the cpue data used, the years used, and the aspirational catches in each sau.

5. *hcrfun()* this is the function that represents the harvest control rule. It should take in the output from *getdata()* and whatever else it requires. It will likely use other functions from the HS file to conduct the calculations required to implement the harvest strategy and its harvest control rule. Its outputs must include, at least, a TAC for the simulated zone in the following year of projections, as well as the expected or aspirational catches for each sau. The Tasmanian HS generates aspirational catches for each *sau* and sums those to produce a TAC for the simulated zone. The South Australian HS only generates a predicted TAC so it could set the aspirational catches = *NULL*. Nevertheless, the simulation requires that the total catch is distributed among the available sau and this will likely require the dive fleet dynamics to be modelled so that how effort and subsequent catch is distributed can be estimated. Even where aspirational catches are estimated directly, when a fishery operates there is invariably noise associated with such caches and the actual catches by sau differ from the aspirational catches. The final required function is used to model what catches will actually be taken from each sau in the following projection year. In the outputs from the MSE, this is the difference between the *acatch* (aspirational catches per sau) and the *catch* (the actual catch per sau).
6. *calcpopC()* a function that takes the output from the hcrfun (at least the aspirational catches and TAC) and estimates the actual catches per sau. This may entail application of some dive fleet dynamics to decide the distribution of catches/effort, as well as the application of noise to include uncertainty in the simulations. Even if management in a jurisdiction only generates a zone-wide TAC the projections require a method for predicting the actual catches taken from each *sau* (which then need to be distributed across each *sau*'s populations).
7. *makeouthcr()* function (which by chance has an identical name in **TasHS**) should be designed to harvest the *hcr* scores while the projections are proceeding. This option sounds sensible but has the disadvantage that it will lead to large R objects, needed to store each iteration's *hcr* outputs, being passed back and forth between **aMSE** and the external functions. If these objects become very large, which is very possible if the data includes the size-composition data, this will become very inefficient.
8. *fleetdyn()* a function that defines the fleetdynamics used by aMSE to describe how the aspirational catches are then distributed across the *sau*. If not used, as is currently the case in Tasmania, then set this to *NULL*.
9. *scoreplot()* plots the hcr scores and final scores. This will be unique to each jurisdiction's harvest strategy and should be included in either the same source file or package as the harvest strategy itself, or a separate source file to be included when defining each scenario.
10. *plotmultflags()* like *scoreplot* this will be unique to each jurisdiction's harvest strategy and should be included in either the same source file or package as the harvest strategy itself.

## 12.3 Outputs from Each Harvest Strategy

### 12.3.1 aMSE Implementation

A typical scenario run of the **aMSE** software, after the initial setup of directories, and populating *hsargs*, might appear as follows:

```
out <- do_MSE(rundir,controlfile,hsargs=hsargs,hcrfun=mcdahcr,
              sampleCE=tasCPUE,sampleFIS=tasFIS,sampleNaS=tasNaS,
              ...)
```

The important part is the generation of the *out* object (a long list of result objects), which can then be used for plots and eventually within comparisons with other scenarios.

The primary object relating to the harvest control rule performance is termed *outhcr*, as in 'output of the harvest control rule'. The different harvest strategies implemented in South Australia, Tasmania, and Victoria, all have different outputs from their harvest strategies.

### 12.3.2 Tasmania

outhcr for Tasmania is a list of eight 3D arrays of dimension projection_years x SAU x replicates.

- g1s : num [1:30, 1:8, 1:250] 3.612 8.985 … gradient 1 scores
- g4s : num [1:30, 1:8, 1:250] 6.074 6.171 … gradient 4 scores
- targsc : num [1:30, 1:8, 1:250] 2.442 3.759 … target cpue scores
- finalsc : num [1:30, 1:8, 1:250] 3.467 4.885 … final combined scores
- index : num [1:30, 1:8, 1:250] 4 5 1 2 2 4 … TAC mult index from hsargs
- catchmult: num [1:30, 1:8, 1:250] 0.85 0.9 … TAC multiplier value
- metaflag : num [1:30, 1:8, 1:250] 0 0 0 0 0 0 … which metarule occurred
- cetarg : num [1:30, 1:8, 1:250] 121 121 120 … the CPUE target

### 12.3.3 South Australia

*outhcr* for South Australia is a list of 12 3D arrays of dimension projection_years x SAU x replicates. The *cesau* and *catch* are included in their *outhcr* for convenience.

- cesau : num [1:30, 1:8, 1:250] 100.5 94.4 113.9 34.9 45.2 …
- CPUE_score : num [1:30, 1:8, 1:250] 0.607 0 2.519 0 0 …
- FIS_score : num [1:30, 1:8, 1:250] NA NA NA NA NA NA NA NA NA NA …
- Combined_score : num [1:30, 1:8, 1:250] 0.607 0 2.519 0 0 …
- Score_carried : num [1:30, 1:8, 1:250] 0 0 0 0 0 0 0 0 0 0 …
- catch : num [1:30, 1:8, 1:250] 16 7 0.619 0.621 2.292 …
- prop_TAC : num [1:30, 1:8, 1:250] 0.0282 0.0267 0.0255 0.026 0.027 …
- Weighted_SAU_score: num [1:30, 1:8, 1:250] 0.0171 0 0.0643 0 0 …
- Zone_score : num [1:30, 1:8, 1:250] 0.972 0.298 1.077 0 0 …
- Adjustment : num [1:30, 1:8, 1:250] 0.02 0.02 0.117 0.02 0.02 …
- Base_TACC : num [1:30, 1:8, 1:250] 1141 1141 1141 1141 1141 …
- TACC : num [1:30, 1:8, 1:250] 22.2 22.7 81.4 21 21.3 …

### 12.3.4 Victoria

*outhcr* for South Australia is a list of 28 3D arrays of dimension projection_years x SAU x replicates.

- Limit : num [1:30, 1:8, 1:250] 80 80 80 …
- Thres : num [1:30, 1:8, 1:250] 120 120 …
- Target : num [1:30, 1:8, 1:250] 170 170 …
- Mean.CPUE.5 : num [1:30, 1:8, 1:250] 101.3 82.3 …
- Mean.CPUE.4 : num [1:30, 1:8, 1:250] 82.3 78 88 …
- Mean.CPUE.3 : num [1:30, 1:8, 1:250] 78 88 96.8 …
- Mean.CPUE.2 : num [1:30, 1:8, 1:250] 88 96.8 …
- Mean.CPUE.1 : num [1:30, 1:8, 1:250] 96.8 100.5 …
- Mean.CPUE.0 : num [1:30, 1:8, 1:250] 100.5 94.4 …
- CurrentStatus : chr [1:30, 1:8, 1:250] “Limit to Threshold” …
- Years.At.Current.Status: num [1:30, 1:8, 1:250] 3 4 5 1 2 …
- CCROld : chr [1:30, 1:8, 1:250] “1” “1” …
- CCR : chr [1:30, 1:8, 1:250] “1” “1” …
- yr4Gradient : num [1:30, 1:8, 1:250] 9.62 2.49 …
- PrimaryIndicator : chr [1:30, 1:8, 1:250] “Increasing” …
- yr2ratio : num [1:30, 1:8, 1:250] 3.8 -6.1 16.7 …
- SecondaryIndicator : chr [1:30, 1:8, 1:250] “Stable” …
- PrimaryCategory : chr [1:30, 1:8, 1:250] “Increasing” …
- FinalCategory : chr [1:30, 1:8, 1:250] “Increasing” …
- OT : num [1:30, 1:8, 1:250] 7 7.84 7.42 …
- OT.lower : num [1:30, 1:8, 1:250] 7 6.66 7.05 …
- OT.upper : num [1:30, 1:8, 1:250] 8.05 7.45 …
- acatch : num [1:30, 1:8, 1:250] 7.84 7.42 …
- TAC : num [1:30, 1:8, 1:250] 473 382 364 …
- boundup : num [1:30, 1:8, 1:250] -5 -5 -5 -5 …
- boundown : num [1:30, 1:8, 1:250] 5 5 5 5 5 5 …
- boundratioup : num [1:30, 1:8, 1:250] -5 -5 -5 -5 …
- boundratiodown : num [1:30, 1:8, 1:250] 5 5 5 5 5 5 5 …