User Manual for IOTC Yellowfin and Bigeye Tuna MSE software

Version 2.0 – August 2018

Paavo Jumppanen (Contact: Paavo.Jumppanen@csiro.au)

Dale Kolody

CSIRO Oceans & Atmosphere  
GPO Box 1538, Hobart, TAS, 7001, Australia

Table of Contents

[1 Introduction 1](#_Toc523486863)

[1.1 Acronyms and definitions used in this document 1](#_Toc523486864)

[1.2 Documentation conventions 3](#_Toc523486865)

[2 Obtaining and Installing the Software 3](#_Toc523486866)

[3 Software Organization 4](#_Toc523486867)

[3.1 OM Conditioning with Stock Synthesis 7](#_Toc523486868)

[3.2 MSE Control and Projection software 9](#_Toc523486869)

[3.3 C++ Projection Code 12](#_Toc523486870)

[3.3.1 General Coding Details Regarding the Mseom C++ Library 13](#_Toc523486871)

[3.3.2 R DLL and Interface Files 16](#_Toc523486872)

[3.3.3 Loading the Library 16](#_Toc523486873)

[3.3.4 Creating and Destroying an Object Instance 17](#_Toc523486874)

[3.3.5 The MSY projection 17](#_Toc523486875)

[3.3.6 The Managed Catch / Effort Forward Projection 19](#_Toc523486876)

[3.4 Some issues to be aware of: 20](#_Toc523486877)

[4 R Classes 29](#_Toc523486878)

[OMd 29](#_Toc523486879)

[Description 29](#_Toc523486880)

[Attributes 29](#_Toc523486881)

[Name 29](#_Toc523486882)

[Label 29](#_Toc523486883)

[Date 29](#_Toc523486884)

[Author 29](#_Toc523486885)

[Notes 29](#_Toc523486886)

[PrimarySource 29](#_Toc523486887)

[CppMethod 29](#_Toc523486888)

[UseCluster 29](#_Toc523486889)

[Npop 30](#_Toc523486890)

[Nfleets 30](#_Toc523486891)

[SSRootDir 30](#_Toc523486892)

[SBlim 30](#_Toc523486893)

[Flim 30](#_Toc523486894)

[OMList 30](#_Toc523486895)

[nsimPerOMFile 30](#_Toc523486896)

[proyears 30](#_Toc523486897)

[targpop 30](#_Toc523486898)

[seed 30](#_Toc523486899)

[recentPerFirst 30](#_Toc523486900)

[recentPerLast 30](#_Toc523486901)

[seasonCEDist 30](#_Toc523486902)

[nsubyears 31](#_Toc523486903)

[lastSeas 31](#_Toc523486904)

[firstSeas 31](#_Toc523486905)

[firstSSYr 31](#_Toc523486906)

[firstCalendarYr 31](#_Toc523486907)

[lastCalendarYr 31](#_Toc523486908)

[firstMPYr 31](#_Toc523486909)

[MPDataLag 31](#_Toc523486910)

[catchBridge 31](#_Toc523486911)

[catchBridgeCV 31](#_Toc523486912)

[indexFisheries 31](#_Toc523486913)

[ReccvTin 32](#_Toc523486914)

[ReccvRin 32](#_Toc523486915)

[RecACTin 32](#_Toc523486916)

[NInitCV 32](#_Toc523486917)

[NInitCVdecay 32](#_Toc523486918)

[selExpRange 32](#_Toc523486919)

[selAgeRange 32](#_Toc523486920)

[selWLRange 32](#_Toc523486921)

[TACEcv 32](#_Toc523486922)

[Ccv 33](#_Toc523486923)

[Icv 33](#_Toc523486924)

[IACin 33](#_Toc523486925)

[Cbcv 33](#_Toc523486926)

[nCALobs 33](#_Toc523486927)

[Ibeta 33](#_Toc523486928)

[ITrendin 33](#_Toc523486929)

[OMss 33](#_Toc523486930)

[Description 33](#_Toc523486931)

[Initializer Arguments 33](#_Toc523486932)

[OMd 33](#_Toc523486933)

[Report 34](#_Toc523486934)

[UseMSYss 34](#_Toc523486935)

[MSE 34](#_Toc523486936)

[Description 34](#_Toc523486937)

[Initializer Arguments 34](#_Toc523486938)

[OM 34](#_Toc523486939)

[MPs 34](#_Toc523486940)

[Interval 34](#_Toc523486941)

[Report 34](#_Toc523486942)

[CppMethod 34](#_Toc523486943)

[UseCluster 35](#_Toc523486944)

[EffortCeiling 35](#_Toc523486945)

[rULim 35](#_Toc523486946)

[TACTime 35](#_Toc523486947)

[5 Operating Model Equations 35](#_Toc523486948)

[5.1 Notation 35](#_Toc523486949)

[5.2 Population Dynamics 37](#_Toc523486950)

[5.3 Management Implementation Errors 43](#_Toc523486951)

[5.4 Observation Errors 43](#_Toc523486952)

[6 Candidate Management Procedures 44](#_Toc523486953)

[7 Performance Measures and Reference Points 46](#_Toc523486954)

[8 Acknowledgements 49](#_Toc523486955)

[9 References 50](#_Toc523486956)

# Introduction

This user manual provides a roadmap to the IOTC yellowfin (YFT) and bigeye (BET) Management Strategy Evaluation (MSE) software, including the mathematical equations for the dynamics, candidate MPs, and R scripts for demonstrating the *niMSE-IO-BET-YFT* software*.*  The initial software development work was jointly funded by the EU (through IOTC/FAO) and CSIRO. The user manual focuses on technical implementation details and options within the software. This manual, which is living document, is updated from time to time as features and contributions are added.

This project was initiated with an over-arching design objective of keeping the code accessible to IOTC member scientists through the use of freely available software, familiar to the fisheries science community, and ideally re-using and extending established MSE code to the extent possible. Most of the software is R-based (The R Foundation for Statistical Computing), and depends on packages freely available in the CRAN repository. The core of the MSE platform was adapted from the Carruthers et al (2014) application developed for Atlantic Bluefin Tuna (ABT) (and funded by the ICCAT Atlantic-Wide Research Programme on Bluefin Tuna). Whilst the first implementation of this software closely mimicked the structure of the Carruthers code, that approach suffered from significant performance issues when running large sets of models. As such, the software was completely re-designed (but still adhering to the same model implementation) and restructured with a view to minimising memory use and extending the use of cluster processing to reduce computation time.

The overall package is relatively simple to install and run for anyone familiar with R, though software modifications might not be easy for novice R users. There are two main components that are not implemented in R: (i) Stock Synthesis (SS) assessment software (Methot and Wetzel 2013) is used for Operating Model (OM) conditioning, and (ii) an optional C++ projection component called from R (detailed below).

## Acronyms and definitions used in this document

ABT Atlantic Bluefin Tuna.

BET Bigeye Tuna.

CCSBT Commission for the Conservation of Southern Bluefin Tuna.

CPUE Catch per Unit Effort – usually assumed to be standardized into a relative abundance index for fish vulnerable to a particular fishery.

HCR Harvest Control Rule – the numerical algorithm for recommending a management action (e.g. providing a TAC given a biomass estimate). In this document, the term is generally not intended to encompass data collection and analysis or fitting a stock assessment model, as these may be considered to be separate components of a complete MP.

ICCAT International Commission for the Conservation of Atlantic Tuna

IOTC Indian Ocean Tuna Commission.

IWC International Whaling Commission.

OM Operating Model – this usually refers to the combination of the generic projection software and suite of model specifications used to simulation test the performance of candidate MPs. We often refer to the OM projection software and OM parameterization separately.

MP Management Procedure – the simulation-tested combination of pre-agreed data collection methods, supporting analysis, and Harvest Control Rule. The term is often used interchangeably with MSE, however the *sensu stricto* MP definition (as used in the IWC and CCSBT) explicitly requires a very high level of pre-specification (i.e. of the data requirements and supporting analyses), to preclude the inherent risk of assessment groups failing to reach consensus during the application of an HCR. MSE is a broader term that does not necessarily imply the same degree of pre-specification.

MSE Management Strategy Evaluation – the process (or final product) of simulation testing a fishery management strategy (see MP).

MSY Maximum Sustainable Yield.

SS Stock Synthesis assessment model software.

TAC Total Allowable Catch – the catch quota set by an MP (it could be fishery-specific or the aggregate across fisheries, depending on context).

TAE Total Allowable Effort – a fishery effort constraint set by an MP. In this context it is manifested as an effort multiplier applied to recent estimates of fishery-specific fishing mortality from an assessment model. For the simulation testing, there is an assumption that effort regulations will translate directly into fishing mortality regulations. In practice, it may be very difficult to define effort is such a way that this can be achieved.

WPM IOTC Working Party on Methods.

WPTT IOTC Working Party on Tropical Tunas

YFT yellowfin tuna.

## Documentation conventions

We use the following style conventions in this document:

filename.ext specific file or directory names or functions using character style FileNames.

Variables variable and parameter names using character style Variables.

Code source code using paragraph style Code.

Menu Commands menu and/or button presses in dialogs and programs shown using Menu character style.

# Obtaining and Installing the Software

The software can be downloaded from Github (https://github.com/pjumppanen/niMSE-IO-BET-YFT). This software was developed under Windows 7 (Service Pack 1), 64 bit Operating System. The code and documentation is publicly accessible (independent developers should fork their own version of the code and propose extensions to be re-integrated into the master via the normal GitHub pull request mechanism). The following steps are required before an MSE can be set-up and run:

1. Install R from <https://cloud.r-project.org/>. We recommend that new R-users should consider an R tutorial, and install an Interactive Development Environment (e.g. Rstudio or TINN-R) to navigate the source files and scripts.
2. Install the following R packages from CRAN:
   * *keep*
   * *stringr*
   * *r4ss*
   * *ggplot2*
   * *reshape2*
   * *parallel*
   * *abind*
   * *data.table*
   * *mseviz*
   * *ggstance*
   * plus any others that are noted to be missing when running the software
3. Go to <https://github.com/pjumppanen/niMSE-IO-BET-YFT> and click on the clone or download button, then press the download zip button.
4. Extract the niMSE-IO-BET-YFT-master.zip file to a drive / directory of your choice.

The R code and control files, scripts, etc. are ASCII files and can simply be edited with standard text editors (or within an R IDE), and submitted to an active R session to update the MSE. There are also R callable C++ routines implementing the Baranov catch equation projection code in a DLL. We provide the source code, including the CSIRO-developed ADT software (Automatic Differentiation with Tapenade) and ancillary libraries, and the fully functional DLL. Any change to the C++ code involving the objective functions will require updated differentials. Although the C++ code is freely modifiable, the step of creating differentials through the use of TAPENADE (INRIA, France) requires a license (as of Sep 2016, an annual license is once again free for academic use).

# Software Organization

Figure 1 shows the class hierarchy in the niMSE-IO-BET-YFT software. The MseDefinition class is a placeholder for all the parameters required to define an instance of the MseFramework class. This includes information naming the Stock Synthesis derived assessment model outputs to use, how to weight them and how many simulations to run, operating model stochasticity, year range of projection and starting year of MP management. The MseFramework class in turn holds one instance of the StockSynthesisModel class per Stock Synthesis assessment result set. The StockSynthesisModel class contains one instance of the StockSynthesisModelData class, which serves as a storage location for the model data used in projections. It exists to allow the data needed to run a projection to be marshalled across process boundaries (when using cluster processing) in a memory efficient manner. The single instance of the ReferenceVars class in StockSynthesisModel contains the MSY projection data needed to produce the normalised reporting statistics for MP projections. The ManagementVars class, on the other hand, contains the performance data from a model run. StockSynthesisModel contains one instance of ManagementVars for the historic data from Stock Synthesis, and one instance per MP projection that contains the projected performance data. The Projection class is a helper class that centralises the code implementing the projection process, and is used to perform the MSY and MP projections for a given Stock Synthesis model.

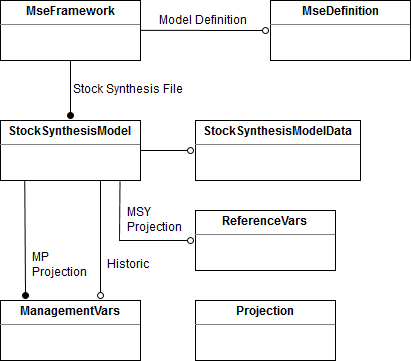


Figure 1. Class diagram for niMSE-IO-BET-YFT software.

Figure 2 shows the sequence of operations in setting up and running a complete MSE using the IOTC yellowfin and bigeye MSE software. Functionally, it is divided into two parts, (1) Conditioning and (2) Model setup and running.

The *“Operating Model Conditioning”* phase involves fitting a suite of Stock Synthesis assessment models to estimate parameters and states that are consistent with the historical data and general understanding of the fishery through the stock assessment process. In this application, conditioning is carried out using Stock Synthesis. The operating model implemented in niMSE-IO-BET-YFT is structurally consistent with the Stock Synthesis implementation. This ensures that when the identified model parameters from conditioning are transferred into the niMSE-IO-BET-YFT operating model, the observed dynamics will be consistent with Stock Synthesis.

Simple R functions automate the process of setting up an array of stock synthesis analyses using template SS control and data files. Batch files are produced to automate the running of the SS analyses, and additional R functions are provided for exploring and evaluating the suite of assessment results. These tools form a template to help automate parts of the OM conditioning process, but the end-user must decide which models to retain or reject from the ensemble using their own judgement, and modify the template files accordingly. These functions are ad hoc tools that worked for the specific cases described and are included because they may be useful, but they were never intended to be robust all-purpose tools. Users will need to know enough about R to modify them for their own purposes.

In the case of the *“Setup and run MSE”* phase R scripts are used to, firstly define the MSE, then with that definition, create an MseFramework object instance, which then forms the basis for tuning and testing MP’s. The runMse() method is used to invoke both MP tuning and MP testing and a number of other methods are available to query statistics and plot performance data for those runs.

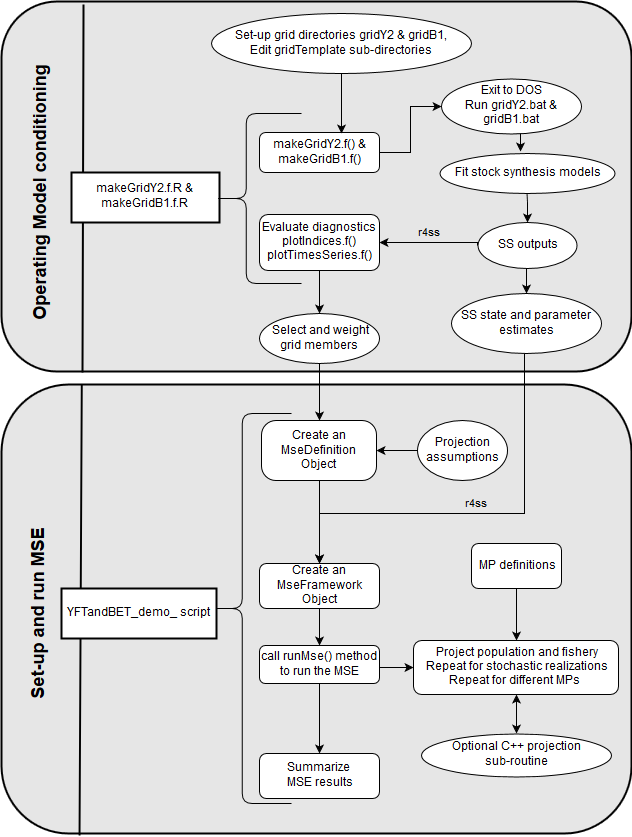


Figure 2. Flowchart illustrating the sequence of processing performed in running the IOTC yellowfin and bigeye MSE. Square boxes represent R-based code.

## OM Conditioning with Stock Synthesis

Conditioning is the process of estimating OM parameters that are consistent with (conditional on) historical data and assumptions about the dynamics of the fishery and population. The approach to conditioning used here involves fitting a suite of Stock Synthesis (SS) assessment models (Methot and Wetzel 2013) that encompass a range of parameter, data and/or structural uncertainties relative to the most recent stock assessment. Note that this step can be skipped if the desired OM has already been created and saved. In that case, one can jump to the next section and simply load() the OM from within R (as indicated in **YFTandBET\_demo\_script.R**)

We refer to the combination of models as a *"grid"*, which is usually a balanced factorial design of all possible assumption interactions (e.g. 3 levels of steepness, 3 levels of *M* and 3 alternative catch history scenarios would form a grid of 3 X 3 X 3 = 27 models). We usually refer to the combination of models comprising an OM as an ensemble, which could be the same as a grid, or include models from multiple grids, or may exclude some models from a grid (e.g. if they are judged to be biologically implausible, or redundant).

Table 1 lists the main scripts and functions used for the conditioning for the demonstration case yellowfin and bigeye OMs.

Table 1. Main scripts and functions used to conduct the SS conditioning for the demonstration case Indian Ocean yellowfin and bigeye operating models.

|  |  |
| --- | --- |
| R Script |  |
| make\_YFTOMGridScript.R | YFT conditioning |
| make\_BETOMGridScript.R | BET conditioning |
| seasAsYrToDecYr.f.R | Functions for converting date formats |
|  |  |
| R Function |  |
| makeGridY2.f() | Make YFT grid |
| makeGridB1.f() | Make BET grid |
| importGrid.f() | Import grid of SS output files into R |
| seasAsYrToMSEYrSeas.f() | Function for converting between SS seasons-defined-as-years and calendar years (in file seasAsYrToDecYr.f.R) |
| plotIndices.f() | Plot some summary SS output statistics partitioned by grid assumptions |
| timeSeriesPlots.f() | Plot time series of some SS output statistics as summary distribution percentiles |
|  |  |

For the demonstration cases, we used SS version 3.24Y (included in the bundled files as ss3.24Y.exe). Earlier versions of SS may fail because of a change to the growth parameterization, or provide incorrect results, due to a bug in the spawning biomass calculations (discussed in the phase 1 project report). The file make\_YFTOMGridScript.R reproduces the yellowfin results from the final report which is the example case discussed below (make\_BETOMGridScript.R is the analogous file for bigeye). Key steps (following software installation described in section 2):

1. Prior to running make\_YFTOM\_script.R, it is necessary to create the root dir gridY3, which must contain a gridTemplate folder and SS executable (see: ...\MSE-IO-BET-YFT\OMconditioning\YFT\gridY3\gridTemplate). gridTemplate includes all the files required to conduct an SS model fitting (e,g, control, data and forecast files). The files templateYFT.dat and/or templateYFT.ctl need to be modified to include all assessment model options required for the grid. This is accomplished with character string option flags. For example, to represent three levels of stock recruit steepness, templateYFT.ctl contains the following lines:

# steepness switches

# xxx h70 0.2 1 0.7 0.7 0 0.2 -1 # SR\_steep

# xxx h80 0.2 1 0.8 0.8 0 0.2 -1 # SR\_steep

# xxx h90 0.2 1 0.9 0.9 0 0.2 -1 # SR\_steep

Each line starts with a character string identifier e.g. # xxx h70, where the first character is the Stock Synthesis comment character # that causes SS to ignore the remainder of the line.

2) The function makeGridY3.f()thencreates a file structure in which a separate SS model is assigned to its own directory, which contains all of the required SS inputs. A model sub-folder is created for every combination of model options and the options are captured in the folder name. That is, folder GridY3 contains 54 model configurations, across 5 dimensions, each given a sub-folder with a name resembling R4MvEst\_h70\_M06\_t00\_q0. In this case, the first dimension R4MvEst refers to spatial and population structure (4 regions, movement estimated) and is actually identical for all models in this grid. The other 4 dimensions have 2-3 levels each (see the phase 1 project report for details), where h70 refers to the stock recruit steepness option (*h* = 0.7), while the other options include h80 and h90 (*h =* 0.8 and 0.9 respectively).

makeGridY3.f()copies over the requisite template files, and modifies them by stripping out the relevant option flags (removing the # xxx h70 comment above activates the first line of the .ctl file above, while the other options remain as inactive comments).

The function also creates a DOS batch file within each new directory to run the individual SS analysis. There are two batch file options, with or without the inverse Hessian calculation. Including the calculation can greatly increase the SS run time, but may provide useful information about parameter estimation uncertainty or convergence problems. A master batch file at the GridY3 root, gridY3.bat is also created, which sequentially calls all of the individual batch files. By default, 4 smaller non-overlapping batch files are also created that can be run instead of the master batch file (in parallel to take advantage of multiple CPUs).

3) After running makeGridY3.f(), pause in the make\_YFTOM\_script.R script, open a DOS command windows and call gridY3.bat. This runs a SS analysis (fits an assessment model) for each element of the grid, and could take many hours, depending on the size of the grid and complexity of the models.

4) Function importGrid.f()imports each SS model output as an individual R object.

5) Function plotIndices.f() plots some simple summary distribution statistics across the grid to rapidly inspect for outlier behaviour in terms of convergence, gross fit between predictions and observations, recruitment trends, general stock status inferences, etc. Each index is disaggregated according to assessment option, and the distribution (boxplot) is plotted with the other options marginalized, (e.g. see the phase 1 final report). Function plotTimeSeries.f() can be used to plot the distribution of some of the standard time series, e.g. *Bt/BMSY* (see phase 1 final report).

The procedure outlined in this R-script does not constitute a comprehensive stock assessment model evaluation, but provides a useful way to set-up and review a substantial number of models. The demonstration grids for both bigeye and yellowfin are very well-behaved, and reasonably consistent with the inferences from the recent stock assessments. However, a large grid with interesting contrast in a large number of dimensions can be expected to produce implausible dynamics, and subjective decisions about what options to include in the grid, and how to weight (or reject) individual models are always required, and may involve several iterations. Detailed results from some individual models can and should be examined using the r4ss package (version 1.24.0, from CRAN) functions SS\_outputs() and SS\_plots(), as in a traditional SS stock assessment**.**

In this MSE software, differential weighting of SS models within the OM ensemble is achieved by the number of stochastic simulations for each set of SS parameters (i.e. number of simulations is proportional to the weight). Most OM model specifications are adopted directly from the SS outputs, including the initial numbers-at-age for the first year of the projections, key biological parameters (M, stock recruit steepness, size-at-age, etc.) and fishery selectivity. However, a number of additional parameters are required to fully specify an OM as detailed in the MseDefinition and MseFramework objects below.

The full suite of SS output files required to populate the demonstration case OMs in the phase 1 report are not provided on GitHub (because of the large number and size of files). They can be recreated using the scripts above, however, the full demonstration case OMs are supplied as R objects that can be loaded to run the MSE. The SS output files are only provided for a couple of models to demonstrate the OM loading.

## MSE Control and Projection software

Once the conditioned assessment models have run and the grid of YFT and BET model results created, the demo script YFTandBET\_demo\_script.R can be run. To do so, run Rgui (64 bit), then change the working directory to the folder YFT-MSE (which is a subfolder of the MSE-IO-BET-YFT-master project that you downloaded and extracted from GitHub). You can do this by selecting the File/Change Dir… menu item and then navigating to and selecting the YFT-MSE folder and pressing OK. Now the demonstration cases can be run by selecting the File/Source R Code… menu, then navigating to the RScripts subfolder to select the **YFTandBET\_demo\_script.R** and pressing the OK button. It may be preferable to step through the R script by submitting the code line by line or in small blocks. Using an IDE for R is a preferable way to work with the R script files.

Table 2 lists the object classes, methods and function classes of the MSE software. Depending on the level at which one wishes to engage with the MP process, the software implementation may be largely irrelevant.

Table 2. The object classes, methods and function classes of the YFT/BET MSE software

|  |  |
| --- | --- |
| Object Classes |  |
| OMd | Operating Model definition - includes list of conditioned SS models and projection assumptions including process, observation and implementation error assumptions, and projection sub-routine options (R/C+, single-threaded, multi-threaded) |
| OMss | Operating Model - an ensemble of models including the parameters and initial states imported from the SS output files, and other specifications defined by the corresponding OMd |
| MSE | Management Strategy Evaluation - summary of MSE results (generated from the internal MSE projection function code or C++ projection subroutine) by MP |
|  |  |
| Methods (core) |  |
| new(OMss) | Creates an instance of an OMss object - |
| new(MSE) | Creates an instance of an MSE object |
|  |  |
| Methods (ancillary) |  |
| tableMSE.f(MSE) | Tabulates the performance measures for the selected MSE object(s) |
| tableMSEbyRF.f(MSE) | Tabulates the selected performance measures by region/fishery for the selected MSE object(s) |
| plotTO.f(MSE) | WPM standard bi-variate trade-off plots. |
| plotUS.f(MSE) | WPM standard confidence interval plots. |
| plotTS.f(MSE) | WPM standard time series plots. |
| Function classes |  |
| IO\_MP | Management Procedure - an algorithm that imports observed data and produces an aggregate TAC and fishery-specific effort multiplier. |
|  |  |

The R script YFTandBET\_demoScript.R demonstrates several MSE applications (lower portion of Figure 1) and assumes that the grid of SS models for the YFT and BET cases have been created. Internally, it calls scripts to create operating model definition objects, and from those a corresponding Operating Model. For instance,

source("Rscripts/Build OMyft2r108.R")

print(system.time(OMyft2r108<-new("OMss",OMd, Report=F)))

creates the operating model OMyft1r108 from the definition in the Build OMyft1r108.R script file. It can take many minutes up to a couple hours to create an OMss object, however, this should only need to be done once, as these objects can be saved and subsequently loaded as shown in YFTandBET\_demoScript.R. Note that print(system.time(...))is only included to record how long a function call takes and is not required. With this operating model it then performs a management strategy evaluation by creating an instance of the MSE class with:,

MPL1 <- c("CC050","CC400","PT41.100.2","IT2.50")

print(system.time(mseOMyft1r216.r<-new("MSE",

OMyft1r216,

MPs=MPL1,

interval=3,

Report=F,

UseCluster=0)))

In this case the management procedures to be used in the evaluation are named in the list MPL1 argument. The interval=3 argument indicates that the MP is to be applied every 3 years, Report=F indicates that various runtime diagnostic outputs will be suppressed, and UseCluster=0 indicates that multi-threading will not be used. Results can then be plotted with one of the plot functions provided in the mseGraphics.R file (e.g. Table 2), which includes; confidence interval plots, bivariate trade-off plots or time series plots (see examples in the phase 1 final report). For example,

plotTS.f(mseObj=mseOMyft2r108.MPL1)

makes time series plots for the MSE object mseOMyft2r108.MPL1.

The OMd object instance defines the characteristics of the OM. This includes a list of SS models from the grid, a vector of weighting factors (of length equal to the number of models in the list), and various parameters that control the projection characteristics (e.g. including the number of years to project, and observation, process and implementation errors). The full list of OMd parameters is listed in Table 3. Some of these parameters support additional options which are not required in the examples shown here. Some have not been tested in the new code, but were retained from the original ABT application, in recognition that it may be desirable for future expansion (e.g. to allow non-stationary or region-specific variability in parameters).

The OMss object instance creates an operating model from the OMd named in the instantiation call. This process involves reading and interpreting the SS model data and massaging it into an appropriate format for the model, creating and initialising state variables, creating placeholders for the model simulation results and calculating MSY estimates for each SS model (but not for each replicate). This can take some time (in the order of 30 seconds per SS specification for the YFT models), although the OM can be saved and reloaded quickly for future re-use. OM parameters are listed in Table 4. In a regular MSE there should be no need for the user to access the OM contents directly, as they are populated by the OMd object and the SS output files.

MP objects need to be defined before the MSE simulations can be run (with a template of options defined in MPs.R). The MP imports key historical data (i.e. actual data used in conditioning, plus new data from the projections, with simulated observation error), and outputs an aggregate annual catch quota, plus a vector of fleet-specific effort multipliers (for those fleets not managed as part of the quota system). This is detailed in the following section on MPs.

The MSE object is the computational workhorse of the MSE software, projecting the population and fishery forward in time, subject to management interventions, and repeating the process for the specified number of simulations and for each MP requested. The option exists here to either use the R-based projection code, or a C++ based sub-routine (detailed below). The completed MSE object is primarily an archive of historical realizations and observations.

The original specification of the MSE code maintained the full array structure of the Carruthers et al. (2014) ABT application, and proved problematic for the demonstration case YFT application described here. While it was functional on a powerful cluster, it was unnecessarily slow (despite the provision for parallel computing) and impractical in terms of memory management for standard laptop or desktop computers. Two approaches were taken to improve the efficiency of the code. The first involved removing the year dimension of the main OM data structures. The second involved a re-implementation of the R projection code using more efficient C++ code to implement a complete Baranov solution, as discussed in the following section. A practical comparison and critique of the two approaches is provided in the phase 1 final report.

## C++ Projection Code

In addition to the computational efficiency gains to be expected by using C++ based projection code for the main projection calculations, there were two other perceived benefits: i) parallel implementation by a second programmer provided an independent check on the integrity of the R-based projection code, and ii) the C++ code was implemented with the standard Baranov catch equations. The main problem with the Baranov equation is that there is no closed form solution for fishing mortality, given a known catch (e.g. when simulating a TAC extraction). An iterative, computationally intensive algorithm is required to reach a solution (to a user-defined degree of precision).

Because of computational efficiency limits, the R-based MSE projection code uses an approximation to the catch equation (described in section 5). While there is some debate about whether the Baranov catch equation is the most appropriate to describe the natural and fishing mortality processes (e.g. Liu and Heino 2014), it is the most commonly used approach in fisheries assessments, including the Stock Synthesis models used for recent yellowfin and bigeye assessments and OM conditioning. The phase 1 final report provides a comparison of the two approaches for YFT and BET demonstration case OMs using several MPs with a range of constant catch objectives, from which it is evident that:

* C++ allows the computationally-demanding Baranov equations to be implemented with about the same run time as the R approximation, such that there is no obvious speed advantage to either.
* When fishing mortality rates were low enough that the prescribed quotas could be attained, the numerical results were essentially equivalent and should not make a noticeable difference to the MP selection process.
* The BET results were also equivalent when quotas were too high to be attained. However, the YFT results can potentially deviate substantially when fishing mortality is very high. The problem appears to arise in YFT because of the spatial structure, when exploitation rates in some areas can be extremely high, while other areas remain rather less exploited. In these cases, the deviation between Baranov and the approximation was lowest when the catch was extracted at the beginning of the time-step (rather than the mid-point), and when the proportional harvest rate constraint was activated at 0.3 rather than 0.5 (see section 5). However, this observation is not the result of a systematic analysis, and may not be true in all circumstances.
* The choice between R and C++ may have implications for choosing between some MPs, but we expect that neither approach would be very accurate under the relevant circumstances, because it is likely that fishery behaviour would change if depletion was that high. And presumably those MPs would not be consistent with the Commission objectives.

The C++ based projection code represents the solution that is the most consistent with the SS conditioning (and most fisheries applications), but it comes with a cost in terms of programmer skill requirements. CSIRO developed an adjunct tool, ADT (Automatic Differentiation with Tapenade), that when combined with TAPENADE (*TRansformations et Outils Informatiques Pour le Calcul Scientifique*), allows the development of C++ code that can make use of forward and reverse mode differentials created through code translation (as opposed to operator overloading) and higher order differentials. TAPENADE (<http://www-sop.inria.fr/tropics/tapenade.html>) was developed by INRIA, National Institute for Research in Computer Science and Control, France). We developed the C++ portion of the MSE tool with the understanding that TAPENADE would be freely available to provide high quality differential codes. Part way through the phase 1 project, INRIA changed their licensing arrangement, in a manner that undermined the intended distribution options for the software. As a consequence, we maintained fully functional R and C++ implementations of the MSE projection code, and made it simple to use either interchangeably. As of Sep 2016, INRIA have reverted to their original license agreement, such that TAPENADE annual licenses are again available free of charge for academic use.

### General Coding Details Regarding the Mseom C++ Library

The Mseom library provides functional support to do MSY projections and managed catch / effort projections. It is implemented in C++ and relies upon ADT to create R callable interface code and differentiated versions of the respective objective functions used in minimisation. The latter part requires the TAPENADE tool. Here we give a brief overview of how the library is structured in an ADT based project but for an in depth discussion refer to the users-manual.pdf in the ADT software distribution, which is available from GitHub (https://github.com/pjumppanen/ADT). Note that the Mseom library source code is bundled with ADT as a sample project.

ADT based libraries are built by deriving a class from AdtArrays and adding the functionality required. The AdtArrays class is provided by the ADlib library and provides support for multi-dimensional C style arrays with arbitrary indexing and contiguous memory block allocation. Any code that needs to be interfaced with R, or needs to be differentiable, must be coded in a subset of C++ and use only intrinsic data types and the array types provide by ADlib. This is required because ADT parses and translates the source code, and the complete language definition is hideously complex. The code generally also requires special comment fields that tell ADT about the sizes of arrays and other details regarding the generation of interface code. An ADT make file tells ADT which files to parse and process. The processing involves parsing the code, performing the prescribed operations on the code and producing new methods and possibly interface code added to a class derived from the parsed one.

In this case, the base class is OperatingModelBase and is declared in the Mseom.hpp header file. A subset of the class definition is shown below.

class OperatingModelBase : public AdtArrays

{

protected:

/\* AD\_LIBNAME Mseom \*/

/\* AD\_ALIAS OmB=D\_OperatingModelBase \*/

/\* AUTOINIT \*/

int nsim;

int npop;

int nages;

int nsubyears;

int nareas;

int nfleets;

ARRAY\_1I Recsubyr/\* nsubyears \*/;

/\* AUTODEC \*/

int SpawnPerYr;

ARRAY\_4D EforYear/\* nfleets, nareas, nsubyears, nsim \*/;

The comment AD\_LIBNAME tells ADT the name of the library generated in compilation. It is needed so that ADT can generate the correctly named registration function to register R callable C functions with R.

The AD\_ALIAS comment tells ADT the name of the class it will be generating interface code on and its alias. The long name is the actual class name whereas the alias is used to generate functional prefixes for the R callable interface.

The AUTOINIT comment instructs ADT that any class attributes appearing after this comment should be initialised via the constructor. This remains in force until the AUTODEC comment, which instructs ADT to only generate code to instantiate those attributes but not initialise them from R via constructor arguments. Therefore, in the sample shown the attributes nsim, npop, nages, nsubyears, nareas, nfleets and Recsubyr will be initialised via constructor arguments whereas SpawnPerYear and EforYear will not.

Note the two array declarations and the corresponding comments. Arrays are multi-dimensional and currently Adlib provides support for up to 10 dimension arrays. The type name for arrays always begins with an ARRAY\_ prefix followed by the number of dimensions followed by a type specifier. I refers to a signed integer type whereas D refers to a double type. The comment appearing after the attribute names the dimensions for the array, the general format of which is,

/\* {from index:}to index \*/

with the from index being optional. When not specified the from index defaults to 1 in deference to the array indexing in R. The EforYear array therefore has the dimensions nfleets, nareas, nsubyears and nsim. Note that in a class declaration context all dimension variables must be declared as class attributes in an AUTOINIT context and in a class method context any dimensions not appearing as class attributes must be passed in the method argument list. It should be noted that arrays in R are organised in column major format whereas the C/C++ language is row major. Therefore, all the array dimensions in the C++ code appear in reverse order to that of R to avoid the need to re-map the memory when using R arrays in C++.

Another feature of ADlib made use of in this library is parellisation, which is handled transparently with the parallelFor() function, which, as the name suggests, runs a for loop in parallel. A typical example in the code is,

ParallelForContext Context(this,

nReport,

ECurrent,

qy,

R0,

M,

mat,

Idist,

Len\_age,

Wt\_age,

sel,

mov,

h,

Recdist,

SRrel,

N,

NBefore,

SSN,

C,

SSBA,

ntargets,

targpop,

run\_years,

MSY,

BMSY,

SSBMSY,

SSBMSY\_B0,

MinPar,

maxit);

allocateShallowCopies(this);

parallelFor(&Context, OperatingModelMin::findMSYrefsTask, 1, nsim);

freeShallowCopies();

The function OperatingModelMin::findMSYrefsTask is called from multiple threads with a unique simulation number. As the for loop will generally be acting on the one object there is potential for shared memory access conflicts so to avoid this we need to create unique copies of the object, although not everything requires a full copy. The calls to allocateShallowCopies() and freeShallowCopies() provide this support. Typically, the points where parallel execution are used in the library revolve around performing minimisation. Unfortunately, the stock minimizers that R provides C callable interfaces to are not thread safe so ADlib provides a thread safe implementation. These have the same calling semantics and names as their parent R implementations but with a name prefix of ts\_ to indicate thread safety.

With this knowledge on basic aspects of an ADT built library it should be possible for someone familiar with C/C++ programming to be able to understand the Mseom library source code.

### R DLL and Interface Files

The R DLL and interface files consist of; Mseom.dll, Om\_R\_interface.r, and OmB\_R\_interface.r. The two R files provide human friendly R functions that wrap corresponding .Call invocations. Both must be included in the R code to use the library via source() commands.

### Loading the Library

The dynamic library should be loaded into R via the dyn.load() command. Furthermore, the code should check for previous loading of the DLL and only load it once. Failure to do so usually results in R crashing, although the exact cause of this behaviour is unclear. The supplied code assumes the R dll resides in the lib sub-folder off the YFT-MSE parent folder and is illustrated below. It is generalised to support Linux builds as well as Windows, and includes the corresponding R interface code. It should be noted that we only provide a 64 bit build of the DLL so it will be necessary to run under a 64 bit version of R. For runs with many simulations, this is essential as the 32 bit version of R will typically fail to run owing to array allocations exceeding the imposed minimum of 32 bit R.

#load Mseom module and R interface code

# Load the library

if (version$os == "mingw32")

{

# Running in Windows

LibName <- "Mseom"

LibExtension <- ".dll"

} else

{

# Running in Linux

LibName <- "libMseom"

LibExtension <- ".so"

}

# Only load library if not already loaded.

# Loading more than once results in R mis-behaving

# and crashing

if (is.na(match(LibName, attr(getLoadedDLLs(), "names"))))

{

LibPath <- paste("./lib/", LibName, LibExtension, sep="")

dyn.load(LibPath)

}

source("./lib/OmB\_R\_interface.r")

source("./lib/Om\_R\_interface.r")

### Creating and Destroying an Object Instance

The callable code within the mseom library comprise a number of methods in a given class. Therefore, to be able to make use of the code we must first create an object instance in R. We do so with, for example,

Obj <- Om.create(nsim,

npop,

nages,

nsubyears,

nareas,

nfleets,

OM@Recsubyr)

Internally the class constructor creates a number of working arrays needed for the model code whose dimensions are based on nsim, npop, nages, nsubyears, nareas and nfleets. When we no longer need the object instance it should be discarded by calling,

Om.destroy(Obj)

### The MSY projection

The Mseom library provides methods to return both the MSY projection and the managed catch / effort forward projection. To ensure consistency it implements the population dynamics in common code shared between both these tasks. The population dynamics implementation is functionally identical to R implementation although in the case of the R code there is duplication between the different tasks.

The C++ implementation of the MSY projection is functionally identical to the R based implementation as testing has demonstrated. The essential difference in returned results is attributable to the different behaviour of the minimizer used. In the case of the R code the optimize() function is used for minimisation whereas the C++ code uses the equivalent of the lbfgsb() C callable minimizer which is used internally by the optim() function in R when using the "L-BFGS-B" method.

The MSY projection works by maximising the yield of a forward projection in time using a scaled effort based on the reference catch years. The projection is taken forward a fixed number of years specified in the call and needs to be sufficient in length to ensure steady state behaviour is reached. With a valid object instance the MSY level is found by calling the Om.nt.findMSYrefs() function as with,

Om.nt.findMSYrefs(Obj,

Report,

.Object@ECurrent[SimNums,,,],

.Object@q[SimNums,],

.Object@R0[SimNums,],

.Object@M[SimNums,,,1],

.Object@mat[SimNums,,,1],

.Object@Idist[SimNums,,,],

.Object@Len\_age[SimNums,,,1],

.Object@Wt\_age[SimNums,,,1],

.Object@sel[SimNums,,],

.Object@mov[SimNums,,,,,],

.Object@h[SimNums,],

.Object@Recdist[SimNums,,],

.Object@SRrel,

N,

NBefore,

SSN,

C,

SSBA,

ntargets,

as.integer(.Object@targpop),

as.integer(nyears),

MinPar,

MSY,

BMSY,

SSBMSY,

SSBMSY\_B0,

maxit)

Note that some arguments are output arguments (SSN, C, SSBA, MinPar, MSY, BMSY, SSBMSY, SSBMSY\_B0), some are state variables (N, NBefore) and others input parameters (ECurrent, q, R0, M, mat, Idist, Len\_age, Wt\_age, sel, mov, h, Recdist, SRrel). The output arguments and the state variables are owned by R and must have the correct type and size for the code to function as expected.

### The Managed Catch / Effort Forward Projection

The managed catch / effort forward projection is driven by an overall TAC level and fishery specific effort levels (TAE’s). All fisheries with zero effort levels are assumed to be managed under an overall TAC. This overall TAC is broken down to fleet specific TACs calculated on the basis of the reference catch mass case in CMCurrent (plus implementation error). The projection is carried out over a full year with the fleet specific TACs corresponding to annual figures. Forward projection is solved by minimizing,

where the summation is carried out over all fleets under TAC management. There is no attempt to impose a penalty on extremely high effort in order to obtain a given TAC, other than a maximum effort boundary constraint controlled by the EffortCeiling parameter in the calling R code. This parameter represents the maximum absolute effort for any given controlled fishery. The objective parameters being optimized are simply scalar multipliers applied to the historic effort in the ECurrent array. As such, fisheries with low initial effort will have a higher boundary constraint than heavily fished ones as the parameter multipliers are relative (to ECurrent) rather than absolute parameters. The forward projection takes place in the call to the Om.nt.projection() function call, as in,

Om.nt.projection(Obj,

y,

as.integer(if (Report) 1 else 0),

EffortCeiling,

TAC,

TAEbyF,

TACEError,

ECurrent,

CMCurrent,

q,

R0,

M\_Y,

mat\_Y,

Idist,

Len\_age\_Y,

Wt\_age\_Y,

Wt\_age\_mid\_Y,

selTS,

mov,

h,

Recdist,

Recdevs\_Y,

recSpatialDevs,

OM@SRrel,

N\_Y,

NBefore\_Y,

SSN\_Y,

C\_Y,

SSBA\_Y,

as.integer(100));

As with the MSY projection some of the arguments represent output and state parameters which need to be setup within the calling R session. Furthermore, the Om.nt.projection() function assumes the N and NBefore state variables are initialised to a known state consistent with the point in time it is projecting from. For this reason it is necessary to run a historic projection for the first year to ensure N and NBefore are in the correct known state. This is handled by the call to OmB.nt.runHistoric(). Prior to running either, it is necessary to call OmB.nt.initialiseParameters() to initialise the initial population and to calculate survivorship. It is also necessary to call Om.nt.beginProjection() prior to forward projection. This function call sets up the initial value of the optimisation parameters, which are saved in between annual projections with subsequent year starting point carrying on from the previous year fit. The rationale behind this behaviour is to (hopefully) reduce the number of iterations required to solve the projection step by starting at a point close to the anticipated solution. The assumption is that the effort multiplier series is likely to be highly correlated with previous years.

## Some issues to be aware of:

* R, R packages and Stock Synthesis are all continuously evolving, and newer or older versions than those described here can be expected to cause software instability, e.g. the phase 1 final report discusses problems with Stock Synthesis version 3.24F, which was used in the Langley (2015) assessment; switching to 3.24Y required some model re-parameterization.
* Year-season configuration. Because of a limitation in the way that SS assigns seasonal tag age-classes, at this time it is preferable for the tropical tuna assessment models to define calendar seasons as model years (e.g. Langley 2015). Some functions are provided for converting time units back and forth in seasAsYrToDecYr.f.R, for plotting and importing to the MSE software below. In the OM software, rate processes are defined in annual units (standard IOTC reporting units). Many SS parameters in a calendar-seasons-as-SS-model-years configuration must be re-scaled (e.g. *MOM = 4MSS*, *MSYOM* = 4*MSYSS* ). The conversion is not necessarily intuitive, e.g. *FOM, quarter =1/4 FSS*, because the calendar-seasons-as-SS-model-years configuration includes a fishery duration definition of 0.25, which is internally converted to an "annual" equivalent for reporting, and which must in turn be externally converted back to a true quarterly *F* because the OM operates on a quarterly time-step. Use of a true year-season SS configuration was not explicitly supported in the OM importing software, and will (probably) cause errors if attempted.
* In spatial models, SS reports spatially-aggregated fishing mortality, and not the region-specific fishing mortality, which is required to estimate the local effective effort for effort-based management. In principle, it should be possible to back-calculate fishing mortality by region from the SS numbers-at-age outputs, but initial attempts to do so suggested that the relevant outputs are not intuitively defined. This is important for calculating MSY-based reference points and using effort-based management, but does not affect the quota-based management projections which have been favoured by IOTC MSE initiatives to date.
* The OM software has flexibility to accept SS inputs with different structural assumptions (e.g. number of spatial units, fisheries, age-classes, etc.). However, these structures must be constant for an individual OM definition, i,e, At this time, different spatial structures or numbers of fisheries, etc. must be defined as different OMs. The MSE software could be run on these different OMs independently, and the results subsequently merged (though the need has not yet arisen to develop such functions).
* The software was originally intended to support multiple independent spawning populations, with independent biology (i.e. retaining this feature from the Carruthers et al 2014 ABT application). However, this feature remains untested in the projection code, and the code for importing multi-stock OMs would need modification.

Table 3. OMd (Operating Model definition) object class attributes defined in file Objects-OMss.R (additional attributes may be defined for potential future expansion).

|  |  |
| --- | --- |
| Slot/attribute |  |
| Identifiers |  |
| Name = "character"  Label = "character"  Date = "character"  Author = "character"  Notes = "character"  PrimarySource = "character"  OMList = "list"  SSRootDir = "character"  SBlim = "numeric"  Flim = "numeric"  CppMethod = "numeric"  UseCluster = "numeric"  seed = "integer"  recentPerFirst = "integer",  recentPerLast = "integer",  seasonCEDist = "integer",  catchBridge = "karray",  catchBridgeCV = "numeric",  NInitCV = "numeric",  NInitCVdecay = "numeric",  selExpRange = "numeric",  selAgeRange = "numeric",  selWLRange = "karray", |  |
| Dimensions |  |
| npop = "integer",  proyears = "integer",  targpop = "integer",  nsimPerOMFile = "karray"  nfleets = "integer",  nsubyears = "integer",  firstCalendarYr = "integer",  lastCalendarYr = "integer",  lastSeas = "integer",  firstSeas = "integer",  firstSSYr = "integer", |  |
| MP details |  |
| firstMPYr = "integer",  MPDataLag = "integer", |  |
| Fishery |  |
|  |  |
| Biology |  |
| ReccvTin = "karray",  RecACTin = "karray",  ReccvRin = "numeric", |  |
| Observation and Implementation errors |  |
| indexFisheries = "integer",    TACEcv = "karray",  Ccv = "numeric",  Cbcv = "numeric",  nCAAobs = "numeric",  nCALobs = "numeric",  Lcv = "numeric",  Ibeta = "numeric",  Icv = "numeric",  IACin = "numeric",  ITrendin = "numeric",  Mbcv = "numeric",  Kbcv = "numeric",  t0bcv = "numeric",  Linfbcv = "numeric",  LFCbcv = "numeric",  LFSbcv = "numeric",  FMSYbcv = "numeric",  FMSY\_Mbcv = "numeric",  BMSY\_B0bcv = "numeric",  ageMbcv = "numeric",  Dbcv = "numeric",  Dcv = "numeric",  Btbcv = "numeric",  Btcv = "numeric",  Ftbcv = "numeric",  Ftcv = "numeric",  hbcv = "numeric",  Recbcv = "numeric",  IMSYbcv = "numeric",  MSYbcv = "numeric",  BMSYbcv = "numeric" |  |
|  |  |
|  |  |

Table 4. OMss (Operating Model) object class attributes in file Objects-OMss.R (additional attributes may be defined for potential future expansion)

|  |  |
| --- | --- |
| Slot/attribute |  |
| Identifiers and Reporting specifications |  |
| Name = "character",  Label = "character",  Date = "character",  Author = "character",  Notes = "character",  PrimarySource = "character",  targpop = "integer",  seed = "integer",  recentPerFirst = "integer",  recentPerLast = "integer",  seasonCEDist = "integer",  SSRootDir = "character",  SBlim = "numeric",  Flim = "numeric",  OMList = "list",  OMid = "karray",  CppMethod = "numeric",  UseCluster = "numeric", |  |
| Dimensions |  |
| nsim = "integer",  nages = "integer",  nyears = "integer",  nsubyears = "integer",  nareas = "integer",  npop = "integer",  proyears = "integer",  nsimPerOMFile = "karray",  nCPUE = "integer", |  |
| MP details |  |
| lastCalendarYr = "integer",  yrLabels = "karray",  seasonLabels = "karray",  yrSeasLabels = "karray",  firstMPYr = "numeric",  catchBridge = "karray",  catchBridgeCV = "numeric",  MPDataLag = "numeric",  NInitCV = "numeric",  NInitCVdecay = "numeric",  selExpRange = "numeric",  selAgeRange = "numeric",  selWLRange = "karray",  selTSSign = "karray",  selTSWaveLen = "karray", |  |
| Fishery |  |
|  |  |
| Biology |  |
| SRrel = "integer",  mat = "karray",  h = "karray",  recgrad = "karray",  ReccvT = "karray",  RecAC = "karray",  ReccvR = "karray",  Linfmu = "karray",  Kmu = "karray",  t0 = "numeric",  Ksd = "karray",  Kgrad = "karray",  Linfsd = "karray",  Linfgrad = "karray",  a = "numeric",  b = "numeric",  ageMmu = "karray",  ageM = "karray",  ageMsd = "karray",  ageMgrad = "karray",  mov = "karray",  Mmov = "karray",  movvar = "karray",  movsd = "karray",  movgrad = "karray",  Mmovvar = "karray",  Mmovsd = "karray",  Mmovgrad = "karray", |  |
| Observation and Implementation errors |  |
| indexFisheries = "integer",  TACEcv = "karray",  Ccv = "numeric",  Cbcv = "numeric",  nCAAobs = "numeric",  nCALobs = "numeric",  Lcv = "numeric",  Ibeta = "numeric",  Icv = "numeric",  IAC = "karray",  ITrend = "karray",  Mbcv = "numeric",  Kbcv = "numeric",  t0bcv = "numeric",  Linfbcv = "numeric",  LFCbcv = "numeric",  LFSbcv = "numeric",  FMSYbcv = "numeric",  FMSY\_Mbcv = "numeric",  BMSY\_B0bcv = "numeric",  ageMbcv = "numeric",  Dbcv = "numeric",  Dcv = "numeric",  Btbcv = "numeric",  Btcv = "numeric",  Ftbcv = "numeric",  Ftcv = "numeric",  hbcv = "numeric",  Recbcv = "numeric",  IMSYbcv = "numeric",  MSYbcv = "numeric",  BMSYbcv = "numeric" |  |
| Simulation data |  |
| Recdevs = "karray",  R0 = "karray",  M = "karray",  Z = "karray",  B = "karray",  SSB = "karray",  NBeforeInit = "karray",  C = "karray",  CAA = "karray",  MSY = "numeric"  BMSY = "numeric"  VBMSY = "numeric"  SSBMSY = "numeric"  UMSY = "numeric"  FMSY1 = "numeric"  SSB0 = "numeric"  B0 = "numeric"  SSBMSY\_SSB0 = "numeric"  IMSY = "numeric"  Idist = "karray",  ECurrent = "karray",  CMCurrent = "karray",  EByQtrLastYr = "karray",  Recsubyr = "integer"  Recdist = "karray",  initIDev = "karray",  SSB0ss = "numeric"  B0ss = "numeric"  MSYss = "numeric"  BMSYss = "numeric"  SSBMSYss = "numeric"  FMSYss = "numeric"  F\_FMSYss = "karray",  Frepss = "karray",  Len\_age = "karray",  Len\_age\_mid = "karray",  Wt\_age = "karray",  Wt\_age\_SB = "karray",  Wt\_age\_mid = "karray",  Css = "karray",  CAAFss = "karray",  SSBAss = "karray",  CBss = "karray",  Bss = "karray",  Recss = "karray",  RecYrQtrss = "karray",  NLLss = "karray",  NLLIss = "karray",  CPUEobsMR = "karray",  CPUEobsY = "karray",  CPUEFleetNums = "karray",  CPUEFleetAreas = "karray",  Edist = "karray", |  |
|  |  |
|  |  |

Table 5. MSE object class attributes in file MSE\_Sournce-OMss.R (additional attributes may be defined for potential future expansion).

|  |  |
| --- | --- |
| Slot/attribute |  |
| Identifiers |  |
| OMName = "character",  Label = "character",  Date = "character",  Author = "character",  Notes = "character",  PrimarySource = "character",  seed = "integer",  CppMethod = "numeric",  UseCluster = "numeric",  OMid = "karray",  SBlim = "numeric",  Flim = "numeric",  yrLabels = "karray",  seasonLabels = "karray",  yrSeasLabels = "karray",  firstMPYr = "numeric",  lastCalendarYr = "numeric",  catchBridge = "karray",  catchBridgeCV = "numeric",  MPDataLag = "numeric", |  |
| Dimensions |  |
| nsim = "integer",  npop = "integer",  nages = "integer",  nyears = "integer",  nsubyears = "integer",  nareas = "integer",  nfleets = "integer",  proyears = "integer",  targpop = "integer", |  |
| MP details |  |
| nMPs = "integer",  MPs = "character", |  |
| Fishery |  |
| CM = "karray",  CMbyF = "karray",  C\_MSY = "karray",  B\_B0 = "karray",  SSB\_SSB0 = "karray",  B\_BMSY = "karray",  SSB\_SSBMSY = "karray",  F\_FMSY = "karray",  B = "karray",  SSB = "karray",  TAC = "karray", |  |
| Management quantities |  |
| CM = "karray",  CMbyF = "karray",  C\_MSY = "karray",  B\_B0 = "karray",  SSB\_SSB0 = "karray",  B\_BMSY = "karray",  SSB\_SSBMSY = "karray",  F\_FMSY = "karray",  B = "karray",  SSB = "karray",  TAC = "karray", |  |
| Observation model |  |
| Cimp = "numeric",  Cb = "numeric",  Cerr = "karray",  Iimp = "numeric",  Ibeta = "numeric",  Ierr = "karray",  IAC = "karray",  ITrend = "karray",  CPUEerr = "karray",  nCAAobs = "numeric",  nCALobs = "numeric",  Lcv = "numeric",  Mb = "numeric",  Kb = "numeric",  t0b = "numeric",  Linfb = "numeric",  LFCb = "numeric",  LFSb = "numeric",  FMSYb = "numeric",  FMSY\_Mb = "numeric",  BMSY\_B0b = "numeric",  ageMb = "numeric",  Dimp = "numeric",  Db = "numeric",  Derr = "karray",  Btimp = "numeric",  Btb = "numeric",  Bterr = "karray",  Ftimp = "numeric",  Ftb = "numeric",  Fterr = "karray",  hb = "numeric",  Recbcv = "numeric",  IMSYb = "numeric",  MSYb = "numeric",  BMSYb = "numeric", |  |
|  |  |
|  |  |

# R Classes

OMd

Description

This class forms a container for all the parameters required to uniquely define an operating model within the MSE. It has no explicit initializer. Normal usage involves using the new operator to create a new but empty object instance and then initializing the attributes explicitly. Additional attributes are defined in the OMd file, but not actively used at this time (and may not be completely implemented).

Attributes

Name

A character string naming the operating model definition. This parameter is for future reference purposes only.

Label

A character string labelling the operating model definition. This parameter is used for result presentation purposes.

Date

A character string naming the date of authorship of the operating model definition. This parameter is for future reference purposes only.

Author

A character string naming the person responsible for the model definition preparation. This parameter is for future reference purposes only.

Notes

A character string giving any additional notes pertinent to the model definition. This parameter is for future reference purposes only.

PrimarySource

A character string parameter describing the source of the conditioned model data this definition is based on. This parameter is for future reference purposes only.

CppMethod

A numeric parameter governing whether to use C++ code for model MSY and management forward projection. 0 = use R code, 1 = use C++ code.

UseCluster

A numeric parameter governing whether to use R cluster processing for model MSY and management forward projection. 0 = no cluster processing, 1 = use cluster processing.

Npop

A numeric parameter setting the number of populations in the model definition.

Nfleets

A numeric parameter setting the number of fleets in the model definition.

SSRootDir

A character string parameter containing the path to the folder containing the Stock Synthesis output files that the model will be using.

SBlim

A numeric parameter setting the reference line value for SSB / SSBMSY plots.

Flim

A numeric parameter setting the reference line value for F / FMSY plots.

OMList

A list of Stock Synthesis models to include in the definition. Typically, this will be the list returned from a call to one of the grid creation functions such as makeGridY3.f(), or a subset of that list.

nsimPerOMFile

A numeric parameter setting the number of simulations to perform per model in the OMList.

proyears

A numeric parameter setting the number of projection years to carry out in the MSE.

targpop

A numeric parameter governing which population/s to collected the MSE performance data for. This can be vector of more than one population (assuming the model has more than one), in which case performance measures are summed over the named populations (not tested).

seed

A numeric parameter used to seed the R random number generator.

recentPerFirst

A numeric parameter setting the first season to use in calculating the seasonal catch effort pattern to use in forward projection. This is a value in sub-years relative to the final sub-year of assessment data and counted backwards. A value of 0 means the final sub-year.

recentPerLast

A numeric parameter setting the last season to use in calculating the seasonal catch effort pattern to use in forward projection. This is a value in sub-years relative to the final sub-year of assessment data and counted backwards. A value of 0 means the final sub-year. Combined with recentPerFirst, it provides a window of sub-years to base an average seasonality on.

seasonCEDist

A numeric parameter governing whether to use seasonal catch effort distribution. 0 = no seasonality (all sub-year catch and effort are equal), 1 = seasonal catch and effort based on the average obtained over the recentPerFirst and recentPerLast sub-years (must be a multiple of whole years).

nsubyears

A numeric parameter setting the number of subyears in the model definition. For quarterly models this will have a value of 4.

lastSeas

A numeric value representing the last season (subyear) of historic assessment data. This allows assessment data to end mid-year.

firstSeas

A numeric value representing the first season (subyear) of historic assessment data. This allows assessment data to start mid-year.

firstSSYr

A numeric value governing the first year of stock synthesis results to include in the MSE. This value is in *Stock Synthesis* years, which represent quarters in the context of the SS assessment of YFT and BET.

firstCalendarYr

A numeric value representing the first calendar year of historic assessment data.

lastCalendarYr

A numeric value representing the first calendar year of historic assessment data.

firstMPYr

A numeric value representing the calendar year in which MP management begins. The supplied management procedures will be utilised from this year onward.

MPDataLag

A numeric value setting the delay in years between the current year of operation and the last set of assessment data supplied to the management procedure. This parameter accounts for the fact that data collation takes time and assessment results always typically lag the year in which management decisions are to be implemented.

catchBridge

A numeric vector indicating the known (annualised) catch history between the final year of assessment and the beginning of management under MPs. The vector is of length between 0 and firstMPYr-lastCalendarYr-1 . Years not covered by the catch bridge vector are considered unknown and are assigned catch based on the previous catch, combined with a lognormal random deviate with CV specified by catchBridgeCV.

catchBridgeCV

A numeric value setting the CV for the lognormal random deviate used to arrive at the catch for unknown years in the catch bridge. Unknown catch is arrived at by applying the deviate to the last known year up to but not including the year that the management projection starts.

indexFisheries

A numeric vector of the fishery numbers from which (fishery-selected) CPUE-based relative abundance indices are to be calculated. In the initial application, there must be exactly one index per region, and these are combined into an aggregate index for the MP calculations.

ReccvTin

A numeric vector, one entry per population, representing the temporal CV for recruitment aggregated over regions. If a value is negative then the recruitment CV is calculated from the RMSE of the SS outputs.

ReccvRin

A numeric value setting the CV for the lognormal random deviate used to provide variability in recruitment among areas.

RecACTin

A numeric value setting the recruitment temporal auto-correlation for the recruitment process (applied annually, even though recruitment occurs quarterly).

NInitCV

A numeric value setting the CV for the lognormal random deviate applied to the initial age 1 population numbers. The other ages have this CV reduced by an exponential decay factor, NInitCVdecay.

NInitCVdecay

A numeric value governing the relationship between the initial population CV at age 1 (NInitCV) and the corresponding value at age a. The relationship is an exponential one described by exp(NinitCV \*(a-1)).

selExpRange

A numeric value governing the exponent range of temporal variability in selectivity. The exponent variation is parameterised as a sinusoid with angular frequency governed by selWLRange and amplitude selExpRange. The angular frequency is randomly chosen from a uniform distribution whose range is selWLRange.

selAgeRange

A numeric value governing the age range shift of selectivity as a function of time. The age shift in selectivity is parameterised as a sinusoid with angular frequency governed by selWLRange and age shift amplitude selAgeRange. The angular frequency is randomly chosen from a uniform distribution whose range is selWLRange.

selWLRange

A pair of numeric values governing the range of angular frequencies used to parameterise temporal selectivity change. The angular frequency is randomly chosen from a uniform distribution whose range is selWLRange. The angular frequency is related to the period of variation in years by the relationship,

Where is the angular frequency, and *T* is the wave period in years (either time or ages depending on context of use).

TACEcv

A numeric vector of length nfleets specifying the CV for a lognormal random deviate applied to fleet specific TAC and TAE.

Ccv

A numeric vector of length 2 specifying the range of CV for a lognormal random deviate to be applied to fleet specific catch as observation error. The actual CV for each fleet is determined by sampling a uniform distribution whose range is specified by Ccv.

Icv

A numeric vector of length 2 specifying the range of CV for a lognormal random deviate to be applied to fleet specific relative abundance indices. The actual CV for each fleet is determined by sampling a uniform distribution whose range is specified by Icv.

IACin

A numeric value setting the autocorrelation in relative abundance lognormal error deviates.

Cbcv

A numeric value specifying the catch bias CV. The fleet specific catch bias is obtained by sampling a lognormal distribution with mean 1, whose CV is Cbcv.

nCALobs

A numeric vector of length 2 specifying the range of the number of annual catch at length (CAL) observations. The actual number of observations is determined by sampling a uniform distribution whose range is specified by nCALobs.

Ibeta

A numeric vector of length 2 specifying the log range of the hyper-stability parameter. The actual value of the hyper-stability parameter is determined by sampling a uniform distribution whose range is specified by the log of nCAAobs and then taking the inverse log (not tested to date).

ITrendin

A numeric value controlling trending in abundance index estimates. A negative value means the trend is extracted from the Stock Synthesis assessment model filename (file-specific index trends assume the single digit following a single q in the filename defines the trend). A positive value is the multiplier applied to a 1% per-annum compounded trend. For example if we use 3 for ITrendin then the trend in abundance indices is 3% per annum.

OMss

Description

This class creates a suite of operating models according to the model definition found in the OMd class instance passed to the initialiser for this class. The initializer creates placeholders for state and result information and initialises the models from data found in the SS assessment output. The OMss object instance is then supplied to the initializer of the MSE class to implement the management strategy evaluation.

Initializer Arguments

OMd

Instance of OMd object containing the attributes describing the operating models to be implemented. This argument must be supplied.

Report

A boolean argument specifying whether to output detailed run information on the progress of model initialisation. If not specified it defaults to F (false).

UseMSYss

A boolean argument specifying whether to use MSY projection statistics from Stock Synthesis. If T (true) then MSY reference points are obtained from Stock Synthesis output. If F (false) the MSY reference points are calculated by performing an MSY projection. If not specified it defaults to F (false).

MSE

Description

This class performs an MSE on the supplied suite of operating models using the supplied suite of management procedures. The initializer creates placeholders for the summary statistics and performance results of the MSE projection and fills them by running the projection for all supplied MPs.

Initializer Arguments

OM

Instance of OMss object containing the suite of operating models to be tested. This argument must be supplied.

MPs

A string vector naming the MP classes to be used for the MSE. This argument must be specified. At present the following MPs are defined in the MPs.R source file:

PT41.100.9, PT41.100.5, PT41.100.2, PT41.100.1, PT42.125.2, PT42.125.5, PT61.75.2, PT61.75.5, PT62.125.2, PT62.125.5, PT82.150.2, PT82.150.5, PT82.100.2, PT82.100.5, IT1.00, IT1.50, IT2.00, IT2.50, IT3.00, IT3.50, CC001, CC050, CC100, CC150, CC200, CC250, CC300, CC350, CC400, CC413, CC450, CE1.0, CE0.01, CE0.1, CE0.25, CE0.5, CE0.75, CE1.5, CE2.0, CE5.0, CC100CE1.0yft, CC50CE1.0bet

Interval

A numeric value setting the number of years between MP updates. If not specified this argument defaults to 3.

Report

A boolean argument specifying whether to output detailed run information on the progress of MSE projection. If not specified it defaults to F (false).

CppMethod

A numeric value specifying whether to force use of the CppMethod. Normally this is specified in the model definition but can be over-ridden with this argument. If not specified it defaults to NA in which case it uses the value from the OM definition. If non-zero the C++ method is used and if zero the R method is used.

UseCluster

A numeric value specifying whether to R cluster processing for MSE forward projection. Normally this is specified in the model definition but can be over-ridden with this argument. If not specified it defaults to NA in which case it uses the value in the OM definition. If non-zero then the R cluster is used and if zero the R cluster is not used.

EffortCeiling

A numeric value specifying the maximum absolute effort for any given controlled fishery in the MSE projection when running the C++ projection code. In the case of the C++ code there is no penalty applied to high effort level in the Baranov solution objective function other than an upper boundary constraint. This upper limit is controlled by EffortCeiling.

rULim

A numeric value indicating the maximum proportional harvest rate that can be taken in the R catch equation approximation. If the harvest rate required to reach the TAC exceeds this threshold, then the actual harvest rate is derived from a function that returns a value higher than rULim, and monotonically increasing toward an asymptote less than 1.0.

TACTime

A numeric value between 0 and 1 specifying where in the course of a full year the instantaneous catch extraction in the Baranov approximation will occur. This parameter defaults to 0.5 meaning the extraction occurs in the middle of the time-step (Pope's approximation). This parameter has no effect on the C++ method which implements a full solution to the Baranov catch equation.

# Operating Model Equations

## Notation

We have attempted to maintain consistency with the Carruthers et al. (2014) Atlantic Bluefin MSE naming conventions and presentation style, to facilitate comparison for users involved with both IOTC and ICCAT.

States, parameters and subscripts are summarized in Table 6. For readability, we often omit subscripts where the context should be self-evident (e.g. maturity is not scripted by year, season and region, because it is usually assumed to be invariant, though that could change in future versions). Subscripts denoting multiple stocks have also been omitted as they are not relevant for any of the demonstration cases to date. When seasonality is not explicitly important, we usually substitute the quarterly time subscript *t* to represent the combination of year (*y*) and season (*s*):

.

There is also some redundancy in the specification of fisheries and regions. Because each fishery operates in exactly one region, the subscript *f* implies a unique region *r* (though the reverse is not true, as multiple fisheries can operate in the same region).

We recycle the greek characters *τ* and *ω* to indicate auto-correlated and independent random normal deviates, respectively, and *σ* for variance-related parameters:

1)

2) ,

where *ρ* is the lag(1) correlation co-efficient (seasonal or annual). The OM was only set up for some processes to use auto-correlated errors (and *ρ* = 0 can be specified to remove it).

Table 6. Operating model states, parameters, scripts and superscripts used in this document (but not necessarily in the code).

|  |  |  |
| --- | --- | --- |
| Variables / Parameter | | |
| *N* | *Number in population* |
| *SB* | *Spawning Biomass* |
| *C* | *Catch* |
| *M* | *Natural mortality* |
| *M* | *Maturity (proportion)* |
| *E* | *Fishing effort* |
| *S* | *Fishery selectivity* |
| *F* | *Fishing mortality* |
| *h* | *Beverton-Holt stock-recruit steepness* |
| *P, ϕ* | *Proportion of a distribution* |
| *Τ* | *A lag(1) auto-correlated random normal deviate* |
| *Ω* | *An independent random normal deviate* |
| *Σ* | *Variance-related parameter* |
| *W* | *Mass-at-age* |
| *L* | *Length-at-age* |
| *α, β* | *Mass-length parameters* |
| *Ψ* | *Movement probability* |
| *TAC* | *Related to quota-based management* |
| *TAE* | *Related to effort-based management* |
| *R* | *Recruitment* |
| *R0* | *Virgin Recruitment* |
| *ρ* | *Auto-correlation co-efficient* |
| Subscripts / Superscripts (capitals denote the total/final, i.e. *A* = oldest age class) | | |
| *a, A* | *Age-class (quarters)* |
| *l,L* | *length-class* |
| *y, Y* | *Year* |
| *s, S* | *Season (quarters)* |
| *t, T* | *Time (simplified representation of combined year and season)* |
| *r, R* | *Region* |
| *TAC* | *Related to quota-based management* |
| *TAE* | *Related to effort-based management* |
| *beforeX* | *Before event X* |
| *afterX* | *After event X* |
| *rec* | *Recruitment-related* |
| *recent* | *Relates to a user-defined recent period* |
| *imp* | *Relates to implementation error* |
| *init* | *Relates to initial conditions* |
| *obs* | *Relates to observation error* |
| *SS* | *Relates to Stock Synthesis state/parameter estimates* |
| *vul* | *Vulnerable (fishery-selected)* |
| *mass, numbers* | *Catch units* |

## Population Dynamics

The following equations describe the dynamics for an individual stochastic realization, conditional on a specific SS scenario within an ensemble of models.

The model follows fairly standard fisheries assumptions for the most part: age-structured with years and quarterly seasons, spatially-structured (optional) with multiple fisheries. The option of describing multiple stocks (independent biology including spawning) has been retained from the ABT MSE for potential future implementation, but this option is not fully tested or supported in the Indian Ocean OM conditioning process at this time. The Indian Ocean MSE has simpler dynamics than the ABT application, in that each fishery is assigned to exactly one area and assumed not to move among areas. Accordingly, there is no attempt to model the fishery effort distribution as a function of the fish population. However, the Indian Ocean model is more complicated in the sense that there is potentially simultaneous catch-based and effort-based management. It is up to each MP to define which fisheries are managed by a catch quota, and which fisheries are managed by effort multipliers. The default assumption is that the annual aggregate catch quota is allocated among the quota-managed fisheries in proportion to the recent observed distribution (defined in the OMs). The fishery-specific quotas are in turn extracted quarterly, in proportion to the recently observed quarterly catch distributions. At this time the OM does not have the capacity to simulate IUU fishing, or effort-creep scenarios, but these scenarios could be explored by imposing effort multiplier time series within MPs.

The initial numbers-at-age, *N*, of the population to be projected forward are extracted from the last year of the SS outputs, with some additional user-defined error:

3) .

The parameter *σinit* controls the magnitude of the error at age 2 (the youngest age imported from SS), and *d* describes how fast the error declines with age (i.e. the estimated precision of partially recruited cohorts is usually artificially low because i) most tuna assessments assume stationary selectivity, and ii) recruitment deviations for recent cohorts are often set to zero to avoid dubious (and often inconsequential) estimates.

Recruitment, *Rt* (to age1, corresponding to age 0 in SS notation), is calculated for each year and season, assuming a spatially-aggregated spawning population (at the beginning of the time-step) according to a Beverton-Holt stock-recruitment relationship with auto-correlated log-normal recruitment deviates:

4) .

The spawning stock biomass, *SB*, is the aggregate across all regions at the beginning of the time-step:

5) 

where *m* is the maturity schedule and is the mass-at-age at the beginning of the time-step, related to length with the standard relationship:

6) .

Note that SS distinguishes between size-at-age at the beginning of the time-step (for spawning purposes), and at the mid-point of the time-step (for catch calculations). This is an unnecessary complication for populations which are assumed to have continuous spawning, however, it was added to minimize inconsistencies between the conditioning and projection code. The aggregate recruitment is partitioned among regions according to a stationary distribution:

7) 

Age-specific, season-specific movement occurs instantaneously, after recruitment, but before mortality:

8) 

where *ψ* is the probability of an individual of age *a* moving from region *r*, to region *k* in season *s*. (there was no spatial structure or migration in the BET demonstration case, while the YFT demonstration case had constant migration among seasons).

There are two options for the catch equations. The C++ based projection sub-routine uses the standard Baranov equations (9, 10) to apply continuous natural and fishing mortality (*M* and *F*):

9) ,

and

10) .

An iterative algorithm is required to simultaneously solve for the catch of the effort-managed fleets and the fishing mortality of the quota-managed fleets.

The R-based approach is a hybrid of Baranov and something similar to Pope's approximation, which does not require an iterative solution (described in equations 11-13). Natural mortality and effort-based fishery extraction are first applied up to a point Δ within the time-step (0 ≤ Δ ≤ 1; 0.5 by default):

11) 

where *f* ϵ *TAE* refers to all effort-managed fisheries. This is followed by instantaneous catch extraction for the quota-managed fisheries:

12) 

where *f* ϵ *TAC* refers to all fisheries that are managed by catch quotas. Finally there is a second continuous mortality event covering the remainder of the time-step:

13) .

Total catch for this latter approach is the sum of the TAC extraction (from 12) and the catch corresponding to each of 11 and 13 which is equivalent to the closed form of equation 9 (i.e. in 11 and 13, *F* is a known function of effort for the effort-managed fleets). Note that the dynamics of equations 9-10 and 11-13 are identical if all fisheries are effort-managed.

In equations 9, 11, and 13, fishing mortality is a function of purely age-based selectivity and effective effort, i.e.

14) 

where effort is known for the effort-managed fleets. Each fishery is managed by either quota or effort. The MP is required to provide a non-zero TAE (in the form of an effective effort multiplier) for each effort-managed fishery, and an aggregate TAC for the remaining fisheries. The effective effort multiplier assumes that the fishing mortality will increase or decrease relative to a recently observed level *Frecent*, (defined in the main text for the demonstration cases). This implicitly assumes that fishery catchability is stationary relative to the effort units defined in the MP (note that in recent YFT and BET assessments, this is only assumed to be true for standardized longline effort).

Selectivity for each fishery is represented by a vector of length *A*, scaled to a maximum of 1.0 for the most vulnerable age(s). There are two options for introducing temporal variability to the selectivity in projections. The first shifts the baseline selectivity function (imported from SS) toward younger or older ages by an integer amount, , as a function of time:

15) 

16) 

Where and  fishery-specific amplitude and angular frequency parameters respectively, and *y* is the projection year starting from 1 (i.e. *y* =1 sets the phase angle so that selectivity is unchanged in the first projection year). An analogous option increases or decreases the degree of targeting in the baseline selectivity:

17) 

18) 

A value of >1 disproportionately increases (<1 decreases) the relative vulnerability of the more highly selected age-classes relative to the other age-classes. Separate and values are used for each type of selectivity process error. The amplitude  for the temporal variability is user-defined and identical among fisheries, but the sign of the amplitude is random for each fishery (i.e the initial trend can be in either direction). The angular frequency is a fishery-specific random sample from a user-defined uniform distribution. The demonstration case parameters in operating models OMyft2 and OMbet1 were selected such that:

* The age-shift amplitude, = 1 results in a maximum selectivity shift of +/- 1 (quarterly) age-class
* The selectivity targeting amplitude exponent, = 0.6, was arbitrarily selected (i.e. from a subjective judgement that the value is big enough to have an effect, but not too big to render the fishery unidentifiable).
* The uniform distribution for the angular frequency parameters was [0.0625,0.5], such that the lower bound results in a monotonic directional change in the selectivity for a projection period of 26 years or less, while the upper bound results in 2 full oscillations of the selectivity vector within a 26 year projection period.

Figure 2 illustrates the temporal variability in selectivity observed for a purse seine and longline fishery from OMbet1. Note that temporal variability in selectivity was added primarily as a means of ensuring that the information content of unbiased size composition data was not unrealistically informative, however, it also introduces time series error structure to the CPUE series.

The aggregate TAC is distributed among seasons and TAC-managed fisheries according to the "recent" distribution of effort or catch. In the case of the C++ based Baranov equations (9, 10), this is achieved by removing the total annual TAC assuming the recent effort distribution remains constant:

19)  ,

while *Ff ϵ TAE* is provided by the MP. The function minimizer solves for the effort multiplier, *x*, that satisfies the catch equations for all 4 quarters simultaneously.

In contrast, the R-based Baranov approximation (equation 12) has a closed form solution. For those fisheries that are managed as part of the quota system, it is assumed that the TAC is apportioned according to the recent (user-defined) catch-in-mass distribution by fishery and season:

20) .

The age-specific, region-specific, catch extraction is calculated each season:

21) ,

where *Bvul*, represents the relative biomass that is vulnerable to each fishery via the selectivity function, *S*:

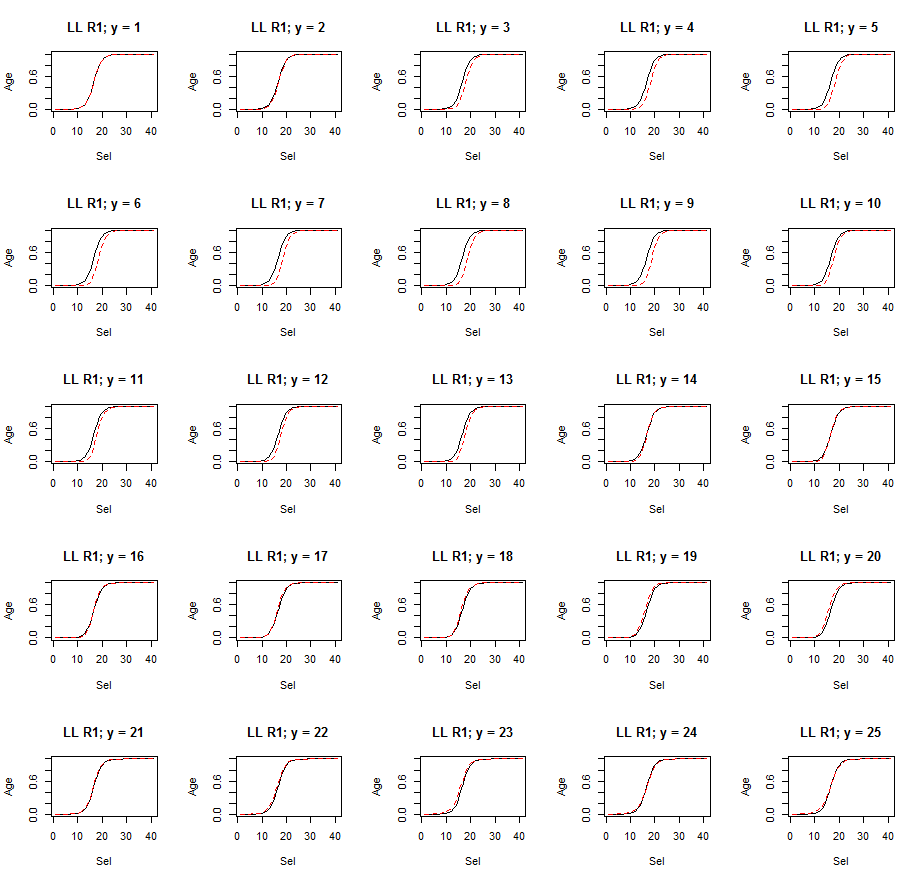
22) ,

The catch in numbers extracted from equation 12 is then:

23) .

Lastly, the age-classes and time-steps of both approaches are incremented from the end of each time-step to the beginning of the next time-step in the standard way (including a plus-group accumulator, in which the characteristics of all fish ≥ age *A* are assumed to be identical):

24) .



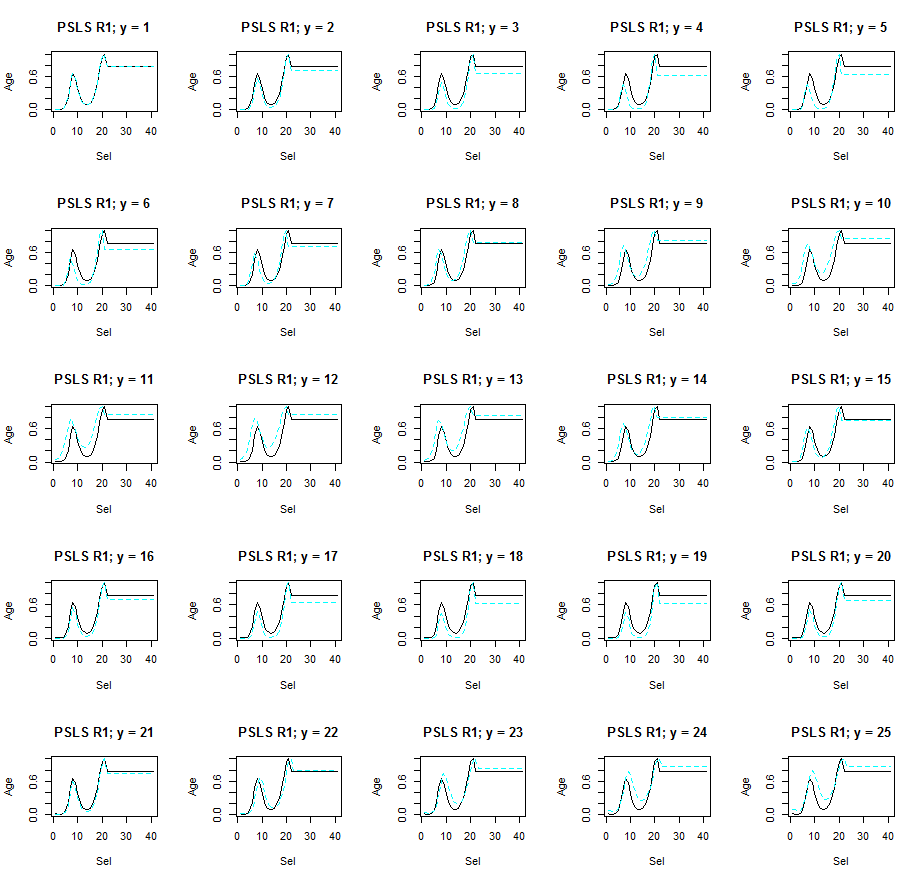


Figure 3. Examples of annual changes in longline selectivity (top panel) and purse seine associated selectivity (bottom panel) over a 25 year projection period for OMbet1. Black line is the stationary selectivity estimated in the SS model, coloured line is the temporally variable selectivity.

## Management Implementation Errors

There is independent implementation error on each fishery-specific quota or effort multiplier (applied equally to all seasons within years):

25) 

26) 

To prevent numerical problems when harvest rates are very high, there is an additional constraint imposed on the approximate catch extraction of equation 12, manifested as a modification to the normal harvest rate:

27) 

28) 

This constraint is required to prevent the catch equation approximation from extracting more than 100% of the population. Harvest rates > *Ulim* continue to increase monotonically toward an asymptote of 1-. It is under these high harvest rate situations, when the TAC cannot be attained, that the projections from the catch approximation can deviate appreciably from the Baranov equations. The final report was completed with values (*Ulim* = 0.5, = 0.12, and TACTime = 0.5), while it appears that the approximation yields results closer the Baranov equation with (*Ulim* = 0.3, = 0.3, and TACTime = 0.01).

## Observation Errors

Currently, the MPs only have access to a subset of relatively aggregated data. Further disaggregation (e.g. all or different data by season and fishery) may eventually prove desirable, but providing these data to the MP without proper consideration of the error structures risks providing unrealistically informative data. e.g. If a relatively high CPUE observation error CV = 0.6 is applied independently each quarter to each of 4 fisheries, the aggregate annual CV ~ 0.15, which is probably unrealistically optimistic for commercial CPUE.

Annual observed catch-in-mass is reported as the product of catch-in-numbers and mass-at-age, summed over ages, seasons and fisheries (fisheries and regions are redundant), with an aggregate log-normal error:

29) 

Observed catch-at-length frequency distributions, *X*, are provided as a multinomial sample from the true catch distributions:

30) ,

where *P* represents the true proportion of fish caught in length class *l*. The sample is unbiased, however, the sample size, *n*, should be set much lower than most real catch-at-length sampling programmes to recognize that samples are not truly random. Furthermore, temporally-structured selectivity can be used (eqn. xxx) to ensure that size data are not unrealistically informative.

An annual aggregate relative abundance index is calculated:

31)

where *rf* ϵ *LL*, indicates the region or fishery (as appropriate) corresponding to the (informative) longline fisheries, *S* is the fishery selectivity, and *q* is the constant of proportionality calculated from the whole of the conditioning assessment model CPUE time period. The CPUE observation consists of:

32)

where βis a hyper-stability / hyper-depletion parameter, and δ is a temporal catchability trend, and the stochastic error and auto-correlation are applied on an annual basis.

# Candidate Management Procedures

The phase 1 final report includes a qualitative description (cartoon schematic) of the IT and PT MPs described below. The parameter values from the main report refer to the MP equations defined in Table 7.

For the PT MPs, the Pella-Tomlinson model dynamics consist of:

33) ,

where *B* is aggregate biomass, *C* is the total catch in mass, *r* and *K* are the estimated population growth rate and carrying capacity, and *p* is the equilibrium yield "shape" parameter (*p = BMSY/B0*, fixed at 0.33 in the demonstration MPs). The model is fit using the least-squares (observation error) objective function:

34) 

where *I* is the aggregate annual CPUE index, and

35) 

The HCR requires estimates of *BY/B0* and

36) .

Using the R-based fitting algorithm, the model appeared to fit some complicated dynamics very poorly (i.e. presumably due to the inability to describe recruitment variability), but the MP generally provided respectable management performance. We did not explore the frequency with which the base R fitting algorithm, optim(), was actually finding the global minimum.

Table 7. MPs included with the initial IOTC MSE release. Additional control parameters are omitted for the IT and PT MPs, including: 1) MP start year, ii) frequency of MP application, 3) data lag (assumed 0 below).

|  |
| --- |
| CCE (constant catch and/or effort) |
| **TAC/TAE calculation:**  *TAEf =*  *TAC = kTAC*  **Control parameters:**  = constant fishery-specific effort-multiplier; constant for all years,  *kTAC* = aggregate *TAC* for fisheries with  = 0; constant for all years. |
| IT (aim for CPUE target) |
| **TAC/TAE calculation:**  *TAEf* = 0      **Data/analysis:**  *Iy* = stanardized longline CPUE in year *y* (aggregated over areas and seasons)  (weighted mean CPUE of last 3 years to reduce volatility)  *m* = CPUE trend (linear regression slope from *Iy-4* to *Iy*)  **Control parameters:**  *IT*= target CPUE  *k1...k4* = responsiveness (gain) parameters  Δmin, Δmax = maximum change constraints |
| PT (Pella Tomlinson with 40:10 type HCR) |
| **TAC/TAE calculation:**  *TAEf* = 0      **Data/analysis:**  *DPT* = Current depletion estimated from fitting Pella-Tomlinson (Observation error) model to catch and CPUE  *MSYPT* = MSY estimate from production model  **Control parameters:**  *k1...k3* = HCR function modifiers  Δmin, Δmax = maximum change constraints |

# Performance Measures and Reference Points

The software reports a standard set of management performance indicators as proposed by the IOTC Working Party on Methods, endorsed by the Scientific Committee (SC 2015), and subsequently refined by the WPM informal sub-group on MSE. Example plots are shown in the phase 1 report.

The IOTC WPM informal MSE working group (April 2016) proposes:

*Time series will be calculated over projection windows of 1, 5, 10 and 20 years, where year 1 is the first year that a TAC or TAE is applied (i.e. as opposed to the first year of projections which might be based on a known or assumed catch because of data and decision process time lags). While it is recognized that MSE is intended to look at medium to longer term performance, one year is included because it is inevitable that industry stakeholders will want to know what the implications of adopting an MP will be in the first year (and they may be particularly disruptive in a rebuilding situation).*

*The performance indicators described in Table [A1] are calculated for each stochastic realization, and then presented as percentiles (10, 25, 50, 75 and 90%) from the distribution of all realizations*

*It was noted that there is currently an inconsistency in the [Commission] identified management objectives, in that achieving target reference points of FMSY and BMSY with near perfect precision would correspond to roughly equal probabilities of being in the green and red Kobe quadrants, while a high probability of being in the green quadrant implies F < FMSY and B > BMSY.*

Standard graphical outputs are described below (and illustrated elsewhere in the user manual and phase 1 project report):

*Time series (quantiles plus some individuals iterations) plots will be used to describe key MSE outputs, including SB/SBMSY (SB/SB0, B/BMSY, B/B0), Catch, F/FMSY, Recruitment and/or CPUE:*

* *Interim reference point reference lines should be included (green target, red limit) when appropriate.*
* *Plots are to indicate the median with a line, and the 25-75th and 10-90th percentiles with shaded ribbons.*
* *When appropriate, 3 individual realizations should be plotted on top, corresponding to the 25, 50, 75th percentiles of SB/SBMSY (or SB/SB0) over the 20 year projection period. These same three individual realizations should then be plotted in all relevant time series plots (i.e. irrespective of which percentile the realization corresponds to in the other plots)*

*Four core trade-off plots, computed for each of 10 and 20 years of projection (i.e. year 1 = first TAC/TAE implementation)*

*1. SB/SBMSY (or SB/SB0 for skipjack) vs. Yield*

*2. Pr(Green Kobe) vs. Yield*

*3. Pr(SB > BLim) vs. Yield*

*4. mean(1 – Cy/Cy-1) vs. Yield*

*Confidence interval plots (double whisker aka udon-soba plots). The summary statistics from [Table xxx] will be summarized graphically by their median, thick confidence interval whiskers for the 25-75th percentiles, and thin whiskers indicating 10-90th percentiles. These plots can compare several MPs for a single performance statistic within a single panel and can pack a lot of information into a small space, but they are less convenient for identifying broad patterns than the other plot types.*

MSY-based reference points are calculated assuming the same "recent" effort distribution (by fishery and season) used for TAC partitioning.

Annual aggregate fishing mortality for reporting purposes is calculated using the SS approach, in which *Z* is first calculated on a spatially aggregated basis, and *F = Z - M* :

37) 

where *s* consists of the 4 seasons within year *y* (and strictly speaking the numerator in the final element of the summation above would be *Ny+1,s=1,a+1,r*).

Table 8. Performance measures (the interim target and limit reference points from Resolution 15/10 are shown for yellowfin and bigeye only; they differ for other IOTC species; Flimit = 1.4FMSY, Ftarget=FMSY, SBlimit=0.4SBMSY, SBtarget=SBMSY)

|  |  |  |
| --- | --- | --- |
| **Status : maximize stock status** | | |
| 1. Mean spawner biomass relative to pristine | *SB*/*SB*0 | Arithmetic mean over years |
| 2. Minimum spawner biomass relative to pristine | *SB*/*SB*0 | Minimum over years |
| 3. Mean spawner biomass relative to *SBMSY* | *SB/SBMSY* | Arithmetic mean over years |
| 4. Mean fishing mortality relative to target | *F*/*Ftar* | Arithmetic mean over years |
| 5. Mean fishing mortality relative to *Fmsy* | *F*/*FMSY* | Arithmetic mean over years |
| 6. Probability of being in Kobe green quadrant | *SB*,*F* | Proportion of years that *SB*≥*SBtar*and *F*≤*Ftar* |
| 7. Probability of being in Kobe red quadrant | *SB*,*F* | Proportion of years that *SB*<S*Btar* and *F*>*Ftar* |
| **Safety : maximize the probability of remaining above low stock status (i.e. minimize risk)** | | |
| 8. Probability of spawner biomass being above 20% of *SB*0 | *SB* | Proportion of years that *SB*>0.2*B0* |
| 9. Probability of spawner biomass being above *B*Lim = 0.4*SBMSY* | *SB* | Proportion of years that *SB*>0.4*SBMSY* |
| **Yield : maximize catches across regions and gears** |  |  |
| 10. Mean catch | *C* | Arithmetic mean over years |
| 11. Mean catch by region and/or gear | *C* | Arithmetic mean over years |
| 12. Mean catch relative to MSY | *C/MSY* | Arithmetic mean over years |
| **Abundance: maximize catch rates to enhance fishery profitability** | | |
| 13. Mean catch rates by region and gear  (for fisheries with meaningful catch-effort relationship) | *I* | Arithmetic mean over years |
| **Stability: maximize stability in catches to reduce commercial uncertainty** | | |
| 14. Mean absolute proportional change in catch | *C* | Arithmetic mean over years of abs(1-*Ct*/*Ct*−1) |
| 15. Variance in catch | *C* | Variance over years |
| 16. Probability of shutdown | *C* | Proportion of years that *C*< 0.1*MSY* |

# Acknowledgements

The first iteration of this work (completed June 2016) was jointly funded by the IOTC (through the European Union and Indian Ocean Commission) and CSIRO. The R-based MSE code was adapted from the Atlantic Bluefin MSE work developed by Tom Carruthers (and funded by the ICCAT GBYP project). Technical oversight and advice was provided by the IOTC Working Party on Methods, notably Iago Mosqueira, Toshihide Kitakado, Nokome Bentley, Gerald Scott, Gorka Merino, Hilario Murua and Rishi Sharma. Adam Langley and Ian Taylor provided helpful insight on the inner workings of Stock Synthesis. Some of the summary result graphics were adopted from the Fisheries Library in R (FLR) code provided by Iago Mosqueira.

# References

Carruthers, T, Kell, L, Davies, C. 2014. Evaluating management strategies for Atlantic bluefin tuna. Report 1 – contract report to support to BFT assessment (GBYP 02/2014) of the Atlantic-wide research programme on bluefin tuna (ICCAT GBYP– Phase 4).

Liu, X, Heino, M. 2014. Overlooked biological and economic implications of within-season fishery dynamics. Can. J. Fish. Aquat. Sci. **71**: 181–188 (2014) dx.doi.org/10.1139/cjfas-2013-0029.

Methot, R.D., Wetzel, C.R., 2013. Stock synthesis: A biological and statistical framework for fish stock assessment and fishery management. Fish. Res. 142: 86– 99.

R Core Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.