Candidate Management Procedure Development Guide

North Atlantic Swordfish MSE

Adrian Hordyk adrian@bluematterscience.com

May 14, 2021

Contents

Introduction

This document describes how to develop Candidate Management Procedures (CMPs) and add them to the North Swordfish MSE (SWOMSE) framework.

`SWOMSE` is an R package designed to evaluate alternative management policies for the North Atlantic Swordfish fishery. The package is built based the `openMSE` framework. The `openMSE` package is automatically installed when `SWOMSE` is installed, and all `openMSE` functions are available to the user when the `SWOMSE` package is loaded.

`openMSE` is an R package that has been developed for conducting fast, flexible, and transparent, MSE for a wide range of fisheries. A non-technical description of `openMSE` and its key features is available on the `openMSE` website.

More information on the SWOMSE framework is available in the SWOMSE user manual and the Trial Specifications Document. All documents relating to the SWOMSE package can be found on the North Atlantic Swordfish MSE homepage[1]

[1] https://iccat.github.io/nswo-mse/

Installation

The SWOMSE framework requires the latest version of the R software installed on your system. The R software can be downloaded for Linux, Mac OS X, and Windows from here.

We also recommend the latest version of RStudio.

The `SWOMSE` R package is available in the ICCAT GitHub repository. The `devtools` package is required to install the `SWOMSE` package. Install the `devtools` package with:

```
install.packages('devtools')
```

Then the `SWOMSE` package can be installed with:

```
devtools::install_github("ICCAT/nswo-mse", auth_token='your_personal_access_token')
```

This will install the `SWOMSE` package and all the dependency packages on your machine.

Note that as this repository is private, you will need to provide a personal access token to download and install from GitHub. See here for information on creating a personal access token. Please contact me (adrian@bluematterscience.com) or ICCAT if you do not have access to the private SWOMSE repository.

Alternatively, you can clone the repository on your machine and build and install the package locally with RStudio.

Once installed, the package can be loaded into the current R session:

```
library(SWOMSE)
#> Loading required package: dplyr
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
#> Loading required package: openMSE
#> Warning: package 'openMSE' was built under R version 4.0.5
#> Loading required package: MSEtool
#> Loading required package: snowfall
#> Loading required package: snow
#> Loading required package: DLMtool
#> Loading required package: SAMtool
#>
#> Attaching package: 'openMSE'
#> The following object is masked from 'package:SAMtool':
#>
#>     userguide
#> The following object is masked from 'package:utils':
#>
#>     demo
```

Operating Models

The operating models (OMs) for the swordfish MSE are still being finalized. Currently, the 288 OMs from the uncertainty grid are available in the `SWOMSE` package. Each OM has 48 simulations. For example:

```
OM_1@Name
#> [1] "M-0.1-sigmaR-0.2-h-0.6-CPUE-0.05-Q-1-ENV-0"
OM_1@nsim
#> [1] 48
```

The OM objects will be updated in the future. The current version is suitable for CMP development and testing.

Fishery Data

The Fishery Data object (`SWOData`) contains the indices of abundance and other data that are passed to the CMPs.

The `SWOData` can be added to the OM using the custom parameters (cpars) feature. This ensures that the historical data in the simulations is exactly matched to the fishery data. The fishery data is also used to condition the observation error for the future projection period.

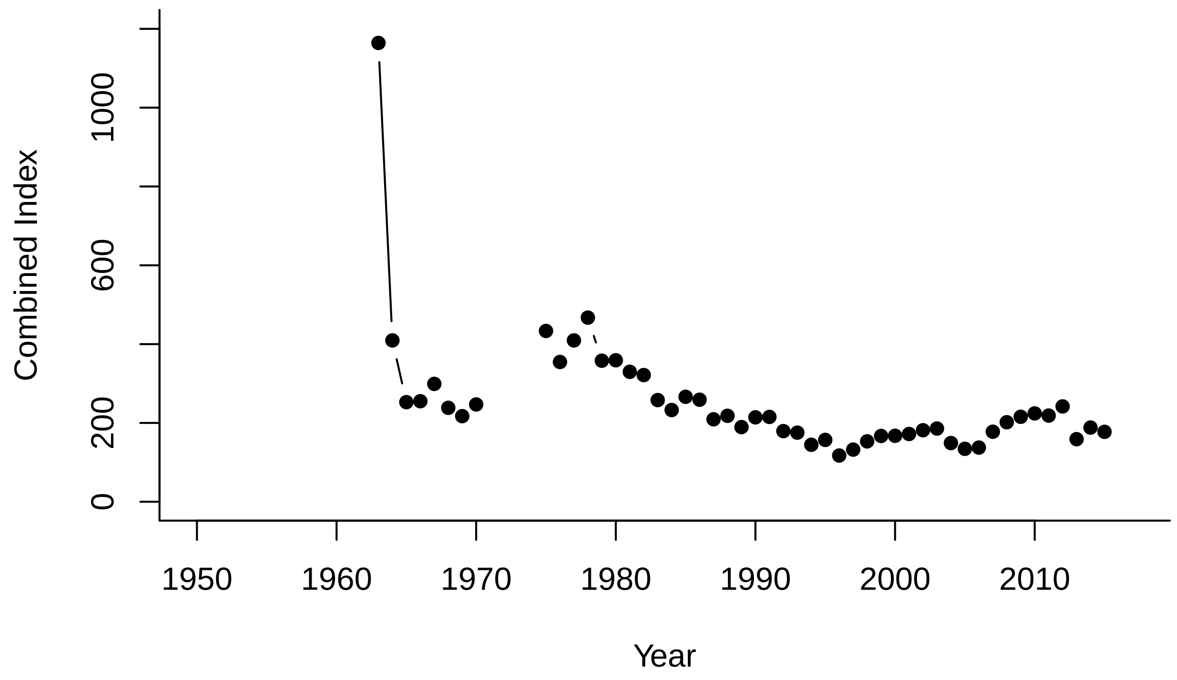
In this example we are adding the `SWOData` object to the first OM from the uncertainty grid (`OM_1`):

```
OM <- OM_1
OM@cpars$Data <- SWOData
```

Index of Abundance

Currently a single index of abundance, Combined Index, is included in the `SWOData` object. The index is located in the `Ind` slot in the `SWOData` object:

```
plot(SWOData@Year, SWOData@Ind[1,], type="b", ylim=c(0, 1200),
     xlab="Year", ylab="Combined Index", bty="l", pch=16)
```



Adding Indices

Additional indices can also be added to the Data object and used by the CMPs. This allows CMPs using different indices to be tested simultaneously. Alternatively, CMPs may be developed that use more than one index of abundance.

More details will be added on this feature at a later date when the additional indices are agreed by the Working Group.

Historical Simulations

When the OM is conditioned with the Data object, the simulated observed catches and indices of abundance for the historical period exactly match those in the Data object.

Here we first run the simulations for the historical period using the `runMSE` function with the argument `Hist=TRUE`:

```
Hist <- runMSE(OM, Hist=TRUE)
#> Loading operating model
#> Valid custom parameters found: M_ageArray Wt_age Len_age LatASD Linf K
#> t0 CAL_bins Mat_age Perr_y Find Fdisc retL SLarray Data
#> M_ageArray has been provided in OM@cpars. Ignoring OM@M and OM@Msd
#> Note: Maximum age ( 25 ) is lower than assuming 1% of cohort survives
#> to maximum age ( 47 )
#> Optimizing for user-specified movement
#> Optimizing for user-specified depletion in last historical year
#> Calculating historical stock and fishing dynamics
#> Calculating MSY reference points for each year
#> Calculating B-low reference points
#> Calculating reference yield - best fixed F strategy
#> Simulating observed data
#> Updating Simulated Data with Real Data from `OM@cpars$Data`
#> Using `OM@cpars$Data@LenCV` (0.1)
#> Updating Simulated Catch from `OM@cpars$Data@Cat`
#> Updating Catch bias from `OM@cpars$Data@Cat`
#> Updating Catch variability from `OM@cpars$Data@Cat`
#> Updating catch observation error from `OM@cpars$Data@Cat`
#> Updating Simulated Total Index from `OM@cpars$Data@Ind`
#> Updating Obs@I_beta from real index
#> Updating Total Index Observation Error based on Observed Data
#> Updating Simulated Catch-at-Length Data from `OM@cpars$Data@CAL`. Note:
#> CAL_ESS is currently NOT updated
#> Using `OM@cpars$Data@Iref` (375)
#> Returning historical simulations
```

We extract the simulated data for the historical period:

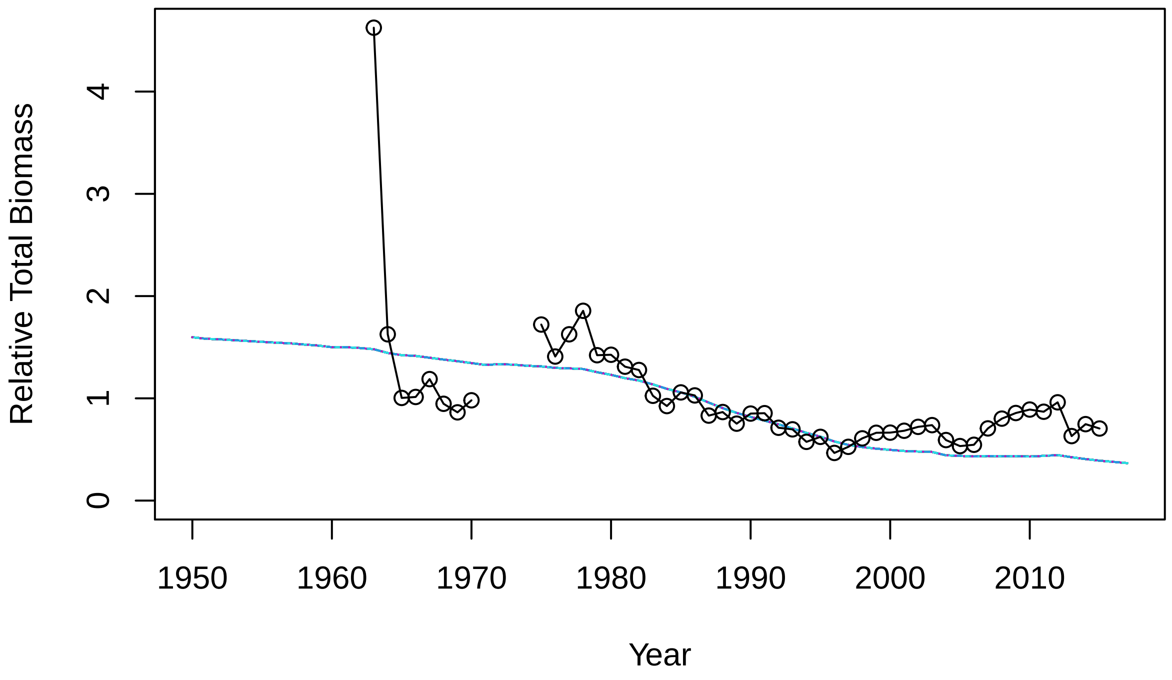
```
SimHistData <- Hist@Data
```

Note that there are now 48 (`OM_1@nsim`) copies of the historical catch and index of abundance in the simulated Data object:

```
# catch data for the first 5 simulations and the first 10 years:
SimHistData@Cat[1:5, 1:10] # catch data is identical for each simulation
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,] 3646 2581 2993 3303 3034 3502 3358 4578 4904  6232
#> [2,] 3646 2581 2993 3303 3034 3502 3358 4578 4904  6232
#> [3,] 3646 2581 2993 3303 3034 3502 3358 4578 4904  6232
#> [4,] 3646 2581 2993 3303 3034 3502 3358 4578 4904  6232
#> [5,] 3646 2581 2993 3303 3034 3502 3358 4578 4904  6232

# index of abundance for the first 5 simulations and the first 30 years
SimHistData@Ind[1:5, 1:30]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
#> [1,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA
#> [2,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA
#> [3,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA
#> [4,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA
#> [5,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA
#>       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
#> [1,] 1164.2 409.4 252.7 255.1 299.1 238.3 217.2 246.9    NA    NA    NA    NA
#> [2,] 1164.2 409.4 252.7 255.1 299.1 238.3 217.2 246.9    NA    NA    NA    NA
#> [3,] 1164.2 409.4 252.7 255.1 299.1 238.3 217.2 246.9    NA    NA    NA    NA
#> [4,] 1164.2 409.4 252.7 255.1 299.1 238.3 217.2 246.9    NA    NA    NA    NA
#> [5,] 1164.2 409.4 252.7 255.1 299.1 238.3 217.2 246.9    NA    NA    NA    NA
#>      [,26] [,27] [,28] [,29] [,30]
#> [1,] 433.4 354.6 409.4 467.2 357.9
#> [2,] 433.4 354.6 409.4 467.2 357.9
#> [3,] 433.4 354.6 409.4 467.2 357.9
#> [4,] 433.4 354.6 409.4 467.2 357.9
#> [5,] 433.4 354.6 409.4 467.2 357.9
```

Here we plot the simulated historical total abundance (generated by the OM) and the index of abundance provided in the Data object. First we standardize both total biomass and the index to make the two comparable:

```r
# sum over areas (default is 2 areas although SWO model doesn't have spatial component)
HistB <- apply(Hist@TSdata$Biomass, 1:2, sum)
RHistB <- HistB/apply(HistB, 1, mean) # standardize biomass to mean 1

RInd <- SimHistData@Ind[1,]/mean(SimHistData@Ind[1,], na.rm=TRUE) # standardize index to mean 1

# plot
matplot(SimHistData@Year, t(RHistB), type="l",
        ylim=c(0, max(c(RInd, RHistB), na.rm=TRUE)),
        xlab="Year", ylab="Relative Total Biomass")
lines(SimHistData@Year, RInd)
points(SimHistData@Year, RInd)
```



Note that although there are 48 simulations of the historical biomass, the lines all fall on top of each other as in this OM the historical simulations are all identical.

Future Projections

The statistical properties of the fit between the index of abundance (i.e., `SWOData@Ind`) and the simulated biomass (determined by the OM) are calculated and used to generate the simulated index of abundance for the future years.

The statistical properties include: hyper-stability (beta), auto-correlation (AC), standard deviation (sd), and correlation (cor) of the deviations between the observed index and the simulated index adjusted for hyper-stability:

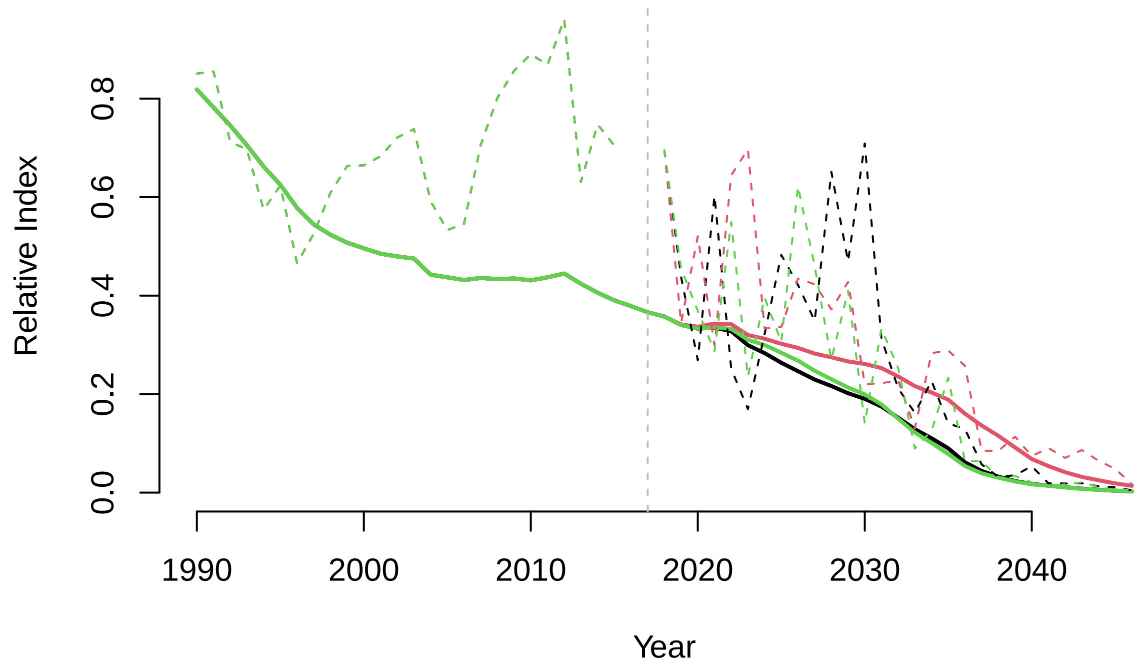
```
Hist@SampPars$Obs$Ind_Stat[1:2,1:4]
#>        beta        AC        sd       cor
#> 1 0.8746291 0.2043197 0.3898256 0.7148464
#> 2 0.8746291 0.2043197 0.3898256 0.7148464
```

These values are used to generate the index for the future projection years. Here we run the model with an simple example CMP to demonstrate the simulated index in the projection years:

```
MSE <- runMSE(OM, MPs="AvC", silent=TRUE) # run MSE with average catch MP
```

Then we re-standardize and plot the simulated biomass (solid line) and the index of abundance (dashed line) for 3 simulations (showing only the data from 1990):

```
HistB <- apply(Hist@TSdata$Biomass, 1:2, sum)
RHistB <- HistB/apply(HistB, 1, mean) # standardize biomass to mean 1
RProjB <- MSE@B[,1,]/apply(HistB, 1, mean)

AllRB <- cbind(RHistB, RProjB)


SimData <- MSE@PPD[[1]] # extract the simulated data
RInd <-  SimData@Ind/apply(SimData@Ind[,1:OM@nyears], 1, mean, na.rm=TRUE) # standardize index to mean

sims <- 1:3
years.plot <- 1990:2046
year.ind <- match(years.plot, SimData@Year)

matplot(SimData@Year[year.ind], t(AllRB[sims,year.ind]), type="l",
        ylim=c(0, max(c(AllRB[sims,year.ind], RInd[sims, year.ind]), na.rm=TRUE)),
        lwd=2, lty=1, bty="n", xlab="Year", ylab="Relative Index")

matplot(SimData@Year[year.ind], t(RInd[sims,year.ind]), type="l",
        lty=2, add=TRUE)

abline(v=OM@CurrentYr, lty=2, col="gray")
```



Candidate Management Procedures

The SWOMSE package includes 4 very simple example MPs:

```
avail('MP', package="SWOMSE")
#> Searching for objects of class MP in package: SWOMSE
#> [1] "AverageC" "ConstC"   "Itarg1"   "Itarg2"
```

For example, `ConstC` is a very simple CMP that sets future TACss equal to the catch in the last historical year:

```
ConstC
#> function(x, Data, reps=100) {
#>   LastTAC <- Data@MPrec[x] # last TAC
#>   TAC <- rep(LastTAC, reps)
#>   TAC <- TACfilter(TAC)
#>   Rec <- new("Rec")
#>   Rec@TAC <- TAC
#>   Rec
#>
#> }
#> <bytecode: 0x000000002eba9180>
#> <environment: namespace:SWOMSE>
#> attr(,"class")
#> [1] "MP"
```

Candidate Management Procedures are functions of class `MP`. All `MP` functions must have at least two arguments `x` - simulation number, and `Data` - the fishery data that the MP operates on, and return an object of class `Rec` that contains the managment recommendations.

The SWOMSE user manual provides more details on the built-in example MPs.

Developing New CMPs

New CMP methods can easily be added to the `SWOMSE` framework.

Example 1 - Model-based CMPs

Model-based CMPs use a stock assessment model to estimate current stock status linked with a harvest control rule (HCR) to set total allowable catch (TAC) limits.

In this example we use a delay-difference stock assessment model from the `SAMtool` package, linked with a 40-10 harvest control rule:

```
CMP_m1 <- function(x, Data, ...) {
  runAssess <- SAMtool::DD_TMB(x, Data) # run the delay-difference model
  Rec <- SAMtool::HCR_ramp(runAssess, LRP=0.1, TRP=0.4) # 40-10 HCR
  Rec # return Rec object
}
class(CMP_m1) <- 'MP'
```

All assessment methods in the `SAMtool` package are available in the `SWOMSE` framework:

```
avail("Assess")
#> Searching for objects of class Assess in package: MSEtool
#> Searching for objects of class Assess in package: SAMtool
#> Searching for objects of class Assess in package: DLMtool
#>  [1] "cDD"      "cDD_SS"   "DD_SS"    "DD_TMB"   "Perfect"  "SCA"
#>  [7] "SCA_Pope" "SCA_RWM"  "SCA2"     "Shortcut" "SP"       "SP_Fox"
#> [13] "SP_SS"    "SSS"
```

Help documentation on the assessment functions is accessed in the usual way, e.g.,:

```
?DD_TMB
```

Custom assessment methods can be included by replacing the call to `SAMtool::DD_TMB` with a code for a stock assessment model. This can include calls to external stock assessment software (e.g., SS3).

Example 2 - Empirical CMPs

Empirical CMPs do not include a stock assessment model, and do not estimate stock status. Instead, empirical CMPs typically modify management recommendations in response to an indicator (or indicators) of the trend in stock abundance.

Here we provide examples of CMPs that compare the mean index of abundance from the last `yrsmth` years, to a target index level specified in the `ITarg` argument. The `mc` argument defines the maximum change in TAC between years:

```
CMP_e1 <- function(x, Data, yrsmth=5, mc=0.05, ITarg=150, ...) {
  ind <- max(1, (length(Data@Year) - yrsmth + 1)):length(Data@Year)
  delta <- mean(Data@Ind[x, ind], na.rm=TRUE)/ITarg # change in TAC
  if (delta < (1-mc)) delta <- 1-mc
  if (delta > (1+mc)) delta <- 1+mc
  LastCatch <- Data@MPrec[x]

  Rec <- new("Rec")
  Rec@TAC <- LastCatch * delta
  Rec
}
class(CMP_e1) <- 'MP'
```

To make a second MP based on `CMP_e1` we can simply change the value of the arguments. For example, to test an MP with a different index target

```
CMP_e2 <- CMP_e1
formals(CMP_e2)$ITarg <- 200
class(CMP_e2) <- 'MP'
```

Now `CMP_e2` is ready to use in the model:

```
CMP_e2
#> function (x, Data, yrsmth = 5, mc = 0.05, ITarg = 200, ...)
#> {
#>     ind <- max(1, (length(Data@Year) - yrsmth + 1)):length(Data@Year)
#>     delta <- mean(Data@Ind[x, ind], na.rm = TRUE)/ITarg
#>     if (delta < (1 - mc))
#>         delta <- 1 - mc
#>     if (delta > (1 + mc))
#>         delta <- 1 + mc
#>     LastCatch <- Data@MPrec[x]
#>     Rec <- new("Rec")
#>     Rec@TAC <- LastCatch * delta
#>     Rec
#> }
#> attr(,"class")
#> [1] "MP"
```

The maximum change and number of years to smooth the index can be changed in a similiar manner:

```
CMP_e3 <- CMP_e1
formals(CMP_e3)$mc <- 0.1
formals(CMP_e3)$yrsmth <- 10
class(CMP_e3) <- 'MP'

CMP_e3
#> function (x, Data, yrsmth = 10, mc = 0.1, ITarg = 150, ...)
#> {
#>     ind <- max(1, (length(Data@Year) - yrsmth + 1)):length(Data@Year)
#>     delta <- mean(Data@Ind[x, ind], na.rm = TRUE)/ITarg
#>     if (delta < (1 - mc))
#>         delta <- 1 - mc
#>     if (delta > (1 + mc))
#>         delta <- 1 + mc
#>     LastCatch <- Data@MPrec[x]
#>     Rec <- new("Rec")
#>     Rec@TAC <- LastCatch * delta
#>     Rec
#> }
#> attr(,"class")
#> [1] "MP"
```

Testing New CMPs

The newly developed example MPs are now available in the `SWOMSE` environment:

```
avail("MP", package="SWOMSE")
#> Searching for objects of class MP in package: SWOMSE
#> [1] "CMP_e1"   "CMP_e2"   "CMP_e3"   "CMP_m1"   "AverageC" "ConstC"   "Itarg1"
#> [8] "Itarg2"
```

Applying CMP to Data

The empirical CMPs can be applied to the swordfish data to generate TAC recommendations:

```
runMP(SWOData, MPs=c('CMP_e1', 'CMP_e2', 'CMP_e3'))
#> Attempting to run 3 MPs:
#> CMP_e1
#> CMP_e2
#> CMP_e3
#>          TAC
#> CMP_e1 13860
#> CMP_e2 12540
#> CMP_e3 14520
```

The model-based CMPs cannot be run on the swordfish data object because they required data (e.g., catch-at-age data) that is not available in the `SWOData` object:

```
runMP(SWOData, MPs=c('CMP_m1'))
#> Attempting to run 1 MPs:
#> CMP_m1
#> Method CMP_m1 failed with error: Error in validObject(.Object) :
#> invalid class "Assessment" object: Error in nlminb(obj$par, obj$fn,
#> obj$gr, h, control = control, lower = low) : NA/NaN gradient evaluation
#> Error in mat[, x] <- unlist(lapply(temp, quantile, probs = perc, na.rm = TRUE)): number of items to
```

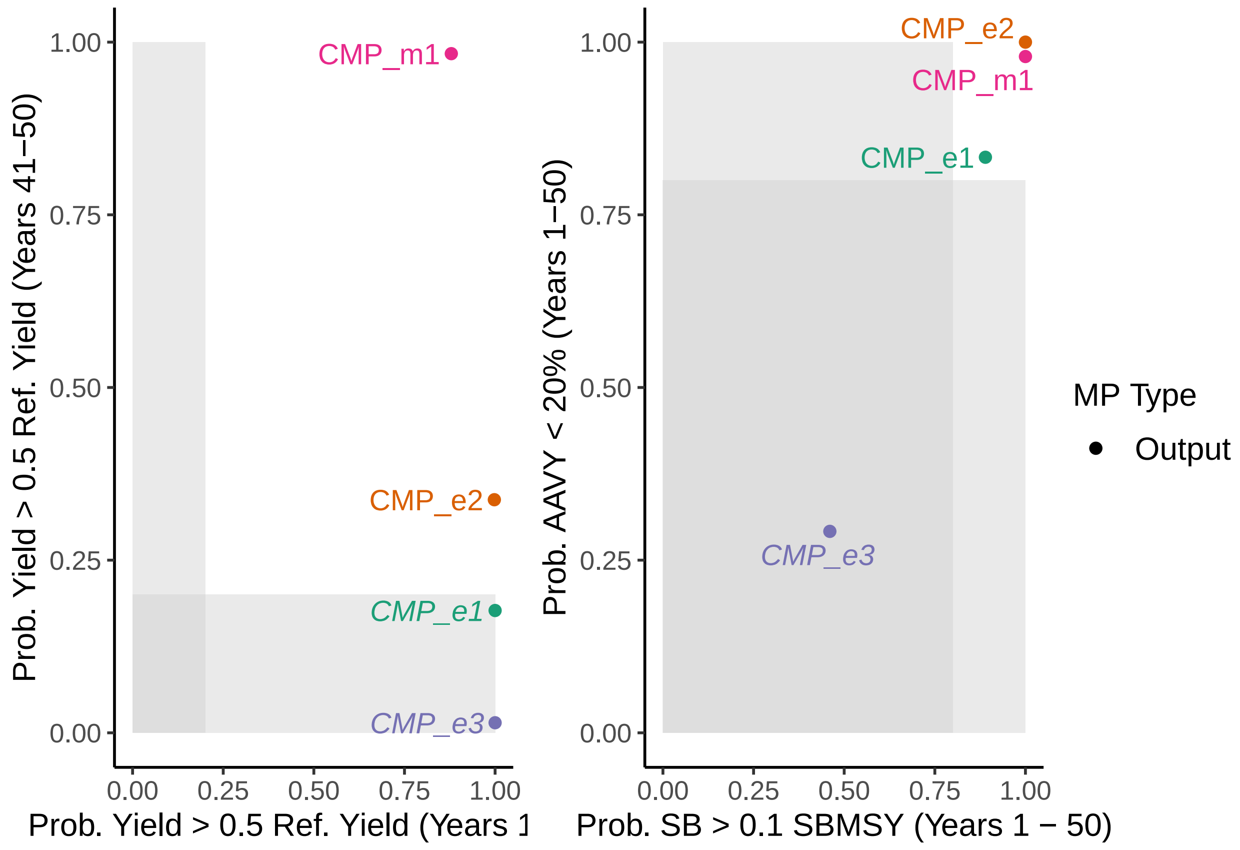
Closed-Loop Simulation

The 5 custom MPs can also be evaluated using closed-loop simulation.

```
MSE <- runMSE(OM, MPs=c(c('CMP_m1', 'CMP_e1', 'CMP_e2', 'CMP_e3')))
#> Checking MPs
#> Loading operating model
#> Valid custom parameters found: M_ageArray Wt_age Len_age LatASD Linf K
#> t0 CAL_bins Mat_age Perr_y Find Fdisc retL SLarray Data
#> M_ageArray has been provided in OM@cpars. Ignoring OM@M and OM@Msd
#> Note: Maximum age ( 25 ) is lower than assuming 1% of cohort survives
#> to maximum age ( 47 )
#> Optimizing for user-specified movement
#> Optimizing for user-specified depletion in last historical year
#> Calculating historical stock and fishing dynamics
#> Calculating MSY reference points for each year
#> Calculating B-low reference points
#> Calculating reference yield - best fixed F strategy
#> Simulating observed data
#> Updating Simulated Data with Real Data from `OM@cpars$Data`
#> Using `OM@cpars$Data@LenCV` (0.1)
#> Updating Simulated Catch from `OM@cpars$Data@Cat`
#> Updating Catch bias from `OM@cpars$Data@Cat`
#> Updating Catch variability from `OM@cpars$Data@Cat`
#> Updating catch observation error from `OM@cpars$Data@Cat`
#> Updating Simulated Total Index from `OM@cpars$Data@Ind`
#> Updating Obs@I_beta from real index
#> Updating Total Index Observation Error based on Observed Data
#> Updating Simulated Catch-at-Length Data from `OM@cpars$Data@CAL`. Note:
#> CAL_ESS is currently NOT updated
#> Using `OM@cpars$Data@Iref` (375)
#> Running forward projections
#> 1 / 4 Running MSE for CMP_m1
#> ..................................................
#> 2 / 4 Running MSE for CMP_e1
#> ..................................................
#> 3 / 4 Running MSE for CMP_e2
#> ..................................................
#> 4 / 4 Running MSE for CMP_e3
#> ..................................................
```

And the results plotted as trade-off plots:

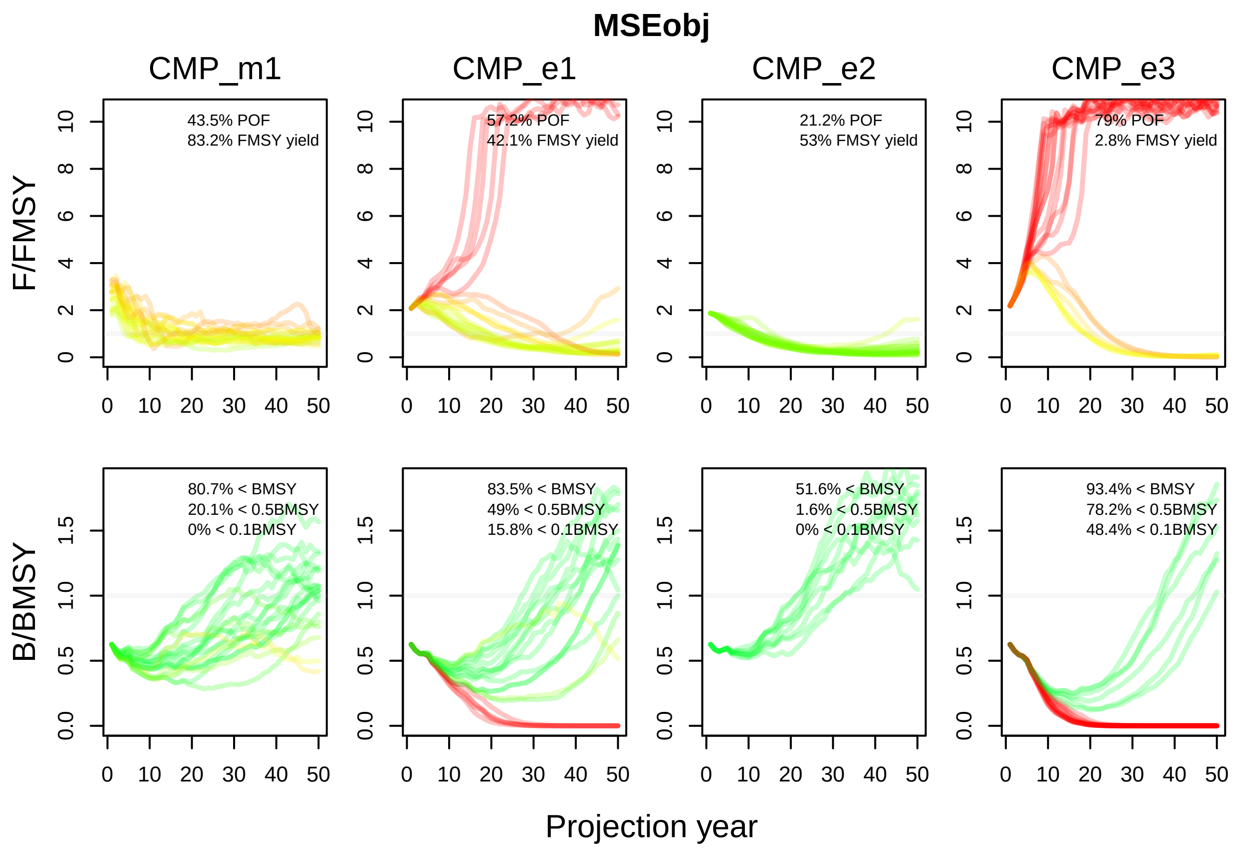
```
TradePlot(MSE)
```



```
#>       MP  STY   LTY  P10 AAVY Satisificed
#> 1 CMP_m1 0.88 0.980 1.00 0.98        TRUE
#> 2 CMP_e1 1.00 0.180 0.89 0.83       FALSE
#> 3 CMP_e2 1.00 0.340 1.00 1.00        TRUE
#> 4 CMP_e3 1.00 0.015 0.46 0.29       FALSE
```

and projection plots:

```
Pplot(MSE)
```


MSEobj
CMP_m1
CMP_e1
CMP_e2
CMP_e3
F/FMSY
B/BMSY
43.5% POF
83.2% FMSY yield
57.2% POF
42.1% FMSY yield
21.2% POF
53% FMSY yield
79% POF
2.8% FMSY yield
80.7% < BMSY
20.1% < 0.5BMSY
0% < 0.1BMSY
83.5% < BMSY
49% < 0.5BMSY
15.8% < 0.1BMSY
51.6% < BMSY
1.6% < 0.5BMSY
0% < 0.1BMSY
93.4% < BMSY
78.2% < 0.5BMSY
48.4% < 0.1BMSY
Projection year