

ASMFC MSE Workshop: Model-based Estimation Model

Gavin Fay

August 2021

Replacing our assessment with an estimation model

In the previous examples we used our survey index as the estimate of abundance.

Here we will instead use a model-based estimator, and fit a Schaefer model to the data from the OM to estimate MSY reference points for use in the control rule. (hint we can reuse the code we developed to condition the OM)

1. Build a function for fitting an estimation model to the index.i time series (don't fit to the future time steps that haven't happened yet)
2. Calculate the TAC based on the last year's (model-estimated) biomass and the estimate of FMSY ($r/2$) from the fitted model.
3. run a FMSY based control rule

Stretch-goal

4. Compare the performance of the EM-based strategy when the OM is the Pella-Tomlinson model.

```
#install.packages('ggplot2')
library(ggplot2)
library(Hmisc)
library(mvtnorm)
```

first load packages

load the functions we used yesterday (sourced from a script to make things easy and reduce clutter in this document)

```
source("first-mse-functions.R")
```

First task is to create a function for the estimation model that will use the index data and call the 'assess()' function.

load data

```
data.years <- 1991:2013
harvest <- c(0.1, 3, 15, 52, 76, 139, 95, 93, 84, 93, 86, 103, 104,
            92, 46, 67, 59, 30, 54, 59, 47, 33, 44)
index <- c(NA, NA, NA, NA, NA, NA, NA, NA, NA, 935, NA, 1057, NA, 678, NA,
          420, NA, 554, NA, 458, NA, 474, NA, 280)
```

Condition the OM

```
set.seed(8675309)
ini.parms <- c(log(1200), log(0.1), log(0.3))
redfish <- assess(harvest, index, calc.vcov=TRUE, ini.parms)
```

```
biomass.mle <- redfish$biomass
print(biomass.mle)
```

```
## [1] 1420.2310 1420.1310 1417.1376 1402.3407 1351.5032 1279.8069 1149.1339
## [8] 1068.5681 992.9793 928.6366 856.7886 793.1563 713.2013 632.5654
## [15] 563.6513 540.0219 495.0459 457.2673 447.6696 413.8427 374.1402
## [22] 345.2745 329.4716 302.1228
```

```
pars.mle <- redfish$pars
#define the number of iterations for the MSE
niter <- 500
#set up a storage matrix for our alternative parameter sets
pars.iter <- matrix(NA, nrow = niter, ncol=3)
colnames(pars.iter) <- c("logK", "r", "sigma")

# generate the sets of parameter values
for (i in 1:niter) {
  pars.iter[i,] <- mvtnorm::rmvnorm(1, mean = redfish$pars,
                                     sigma = redfish$vcov)
}

# Now generate replicate model outputs
biomass.iter <- data.frame()
for (i in 1:niter) {
  #here we calculate the biomass trajectory for each of the above sampled parameter vectors
  biomass.iter <- rbind(biomass.iter,
                        data.frame(year = seq(min(data.years),
                                                max(data.years)+1),
                                   biomass = dynamics(pars.iter[i,], harvest),
                                   iter = i))
}
```

Applying the Management Strategy Define the years for the projection:

```
proj.years <- 2014:2034
```

conduct the evaluation with the model-based estimation method

```
project.emhcr <- em_evaluate(pars.iter, biomass.iter, control.pars,
                             data.years, proj.years, 100)
```

```
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]


```
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
## Warning in rnorm(1, -0.5 * sigma^2, sigma): NAs produced
```

Plot the trajectories:

```
projection.plot(project.emhcr)
```

```
## Warning: Removed 200 rows containing non-finite values (stat_summary).
```

```
## Warning: Removed 200 rows containing non-finite values (stat_summary).
```


