ASMFC MSE Workshop: Alternative Operating Models

Gavin Fay

August 2021

Operating Model uncertainty

In the previous example we characterized OM uncertainty using paramter uncertainty in the fit to the original data.

Here we will compare the performance of our simple HCRs given operating model structure uncertainty.

1. Build a function for when the operating model dynamics are governed by the Pella-Tomlinson model (i.e. yield function is not symmetric)
2. Compare the performance of our HCRs from day 1 to both OMs

Stretch-goal

3. Develop an OM that includes process error (recruitment variabilty?) in the population dynamics

```
#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")
```

The Operating Model The population dynamics for the operating model (the 'real' dynamics) are governed by the Pella-Tomlinson equation:

$$B_{y+1} = B_y + B_y * r * \left(1 - \left(\frac{B_y}{K}\right)^{m-1}\right) - C_y$$

where B_y is the biomass in year y, C_y is the catch in year y, r is the population intrinsic growth rate, K is the population carrying capacity, and m is a shape parameter that determines the location of BMSY.

First task is to create a function for the P-T dynamics (hint modify the schaefer function that we used yesterday. Remember there is an extra parameter)

```
pellat <- function(B,C,K,r,m) {
  #function schaefer takes the current biomass, a catch,
  #and the model parameters to compute next year's biomass
  res <- B + B * r * (1 - (B/K)^(m-1)) - C
  return(max(0.001,res))  # we add a constraint to prevent negative biomass
}
```

Once you have this, condition your OM as yesterday but with a fixed value for the shape parameter of 1.2 (Hint, you will want to adjust the biomass projection to use your new function)

change the dynamics function

```
#biomas projection
dynamics <- function(pars,C,yrs,m) {
  # dynamics takes the model parameters, the time series of catch,
  # & the yrs to do the projection over

  # first extract the parameters from the pars vector (we estimate K in log-space)
  K <- exp(pars[1])
  r <- exp(pars[2])

  # find the total number of years
  nyr <- length(C) + 1

  # if the vector of years was not supplied we create
  # a default to stop the program crashing
  if (missing(yrs)) yrs <- 1:nyr

  #set up the biomass vector
  B <- numeric(nyr)

  #intialize biomass at carrying capacity
  B[1] <- K
  # project the model forward using the schaefer model
  for (y in 2:nyr) {
    B[y] <- pellat(B[y-1],C[y-1],K,r,m)
  }

  #return the time series of biomass
  return(B[yrs])

  #end function dynamics
}
```

#####Specify input data and associated years

```
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,935,NA,1057,NA,678,NA,
           420,NA,554,NA,458,NA,474,NA,280)
```

```
# function to calculate the negative log-likelihood
nll <- function(pars,C,U,m) {  #this function takes the parameters, the catches, and the index data
  sigma <- exp(pars[3])  # additional parameter, the standard deviation of the observation error
  B <- dynamics(pars,C,m=m)  #run the biomass dynamics for this set of parameters
  Uhat <- B   #calculate the predicted biomass index - here we assume an unbiased absolute biomass esti
  output <- -sum(dnorm(log(U),log(Uhat),sigma,log=TRUE),na.rm=TRUE)   #calculate the negative log-likel
  return(output)
#end function nll
}
```

Function to perform the assessment and estimate the operating model parameters
(i.e. to fit the logistic model to abundance data)

```
assess <- function(catch,index,calc.vcov=FALSE,pars.init,m=2) {
  # assess takes catch and index data, initial values for the parameters,
  # and a flag saying whether to compute uncertainty estimates for the model parameters

  #fit model
  # optim runs the function nll() repeatedly with differnt values for the parameters,
  # to find the values that give the best fit to the index data
  res <- optim(pars.init,nll,C=catch,U=index,m=m,hessian=TRUE)

  # store the output from the model fit
  output <- list()
  output$pars <- res$par
  output$biomass <- dynamics(res$par,catch,m=m)
  output$convergence <- res$convergence
  output$nll <- res$value
  if (calc.vcov)
    output$vcov <- solve(res$hessian)

  return(output)
#end function assess
}
```

Now we have written the functions to do the calculations, we can run them and perform the assessment.

First define initial parameter vector for: log(K), r, sigma

```
ini.parms <- c(log(1200), log(0.1), log(0.3))
```

Fit the logistic model to data:

```
redfish <- assess(harvest,index,calc.vcov=TRUE,ini.parms,m=1.2)
redfish
```

```
## $pars
## [1]  7.275450 -1.483786 -1.764651
##
## $biomass
##  [1] 1444.4012 1444.3012 1441.3057 1426.4459 1375.2542 1302.2992 1169.3534
##  [8] 1085.3237 1005.9985  937.9198  862.5172  795.6821  712.9664  630.2582
## [15]  560.1032  536.0265  490.8874  453.4997  444.7686  411.9385  373.6692
## [22]  346.7472  333.2694  308.4814
##
## $convergence
## [1] 0
##
## $nll
## [1] -2.767724
##
## $vcov
##               [,1]          [,2]         [,3]
## [1,]  4.167971e-03 -2.616297e-02 8.263051e-06
## [2,] -2.616297e-02  1.783338e-01 1.475391e-05
## [3,]  8.263051e-06  1.475391e-05 6.253185e-02
```

Extract the maximum likelihood and parameter estimates

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

```
##  [1] 1444.4012 1444.3012 1441.3057 1426.4459 1375.2542 1302.2992 1169.3534
##  [8] 1085.3237 1005.9985  937.9198  862.5172  795.6821  712.9664  630.2582
## [15]  560.1032  536.0265  490.8874  453.4997  444.7686  411.9385  373.6692
## [22]  346.7472  333.2694  308.4814
```

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

```
## [1]  7.275450 -1.483786 -1.764651
```

To obtain plausible alternatives for the parameters of the operating model, we will use the statistical uncertainty from the estimation by sampling parameter sets from the estimated variance-covariance matrix.

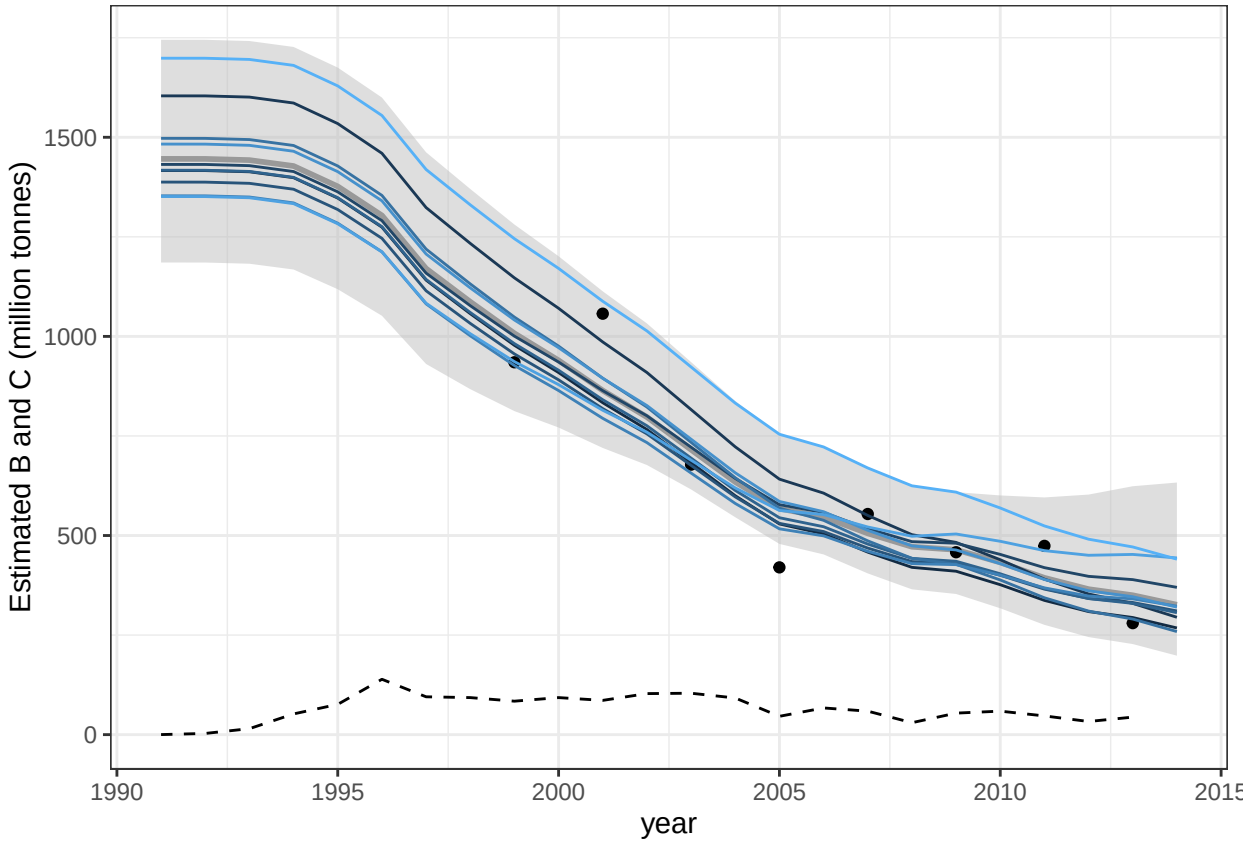
```
m=1.2
#define the number of iterations for the MSE
niter <- 500
#set up a storage matrix for our alternative parameter sets
pars.iter <- matrix(NA,nrow = niter, ncol=3)
colnames(pars.iter) <- c("logK","r","sigma")

# 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,m=m),
                                   iter = i))
}
```

We can now plot the estimated biomass time series

```
Fig1 <- ggplot(data=biomass.iter,aes(x=year,y=biomass))
Fig1 +
     stat_summary(fun.data = "median_hilow",
                  fun.args = list(conf.int=1),
                  geom = "ribbon" ,
                  alpha=0.5,
                  fun.min = function(x)0,
                  fill = "gray")  +
     stat_summary(fun.data = "median_hilow",
                  geom = "smooth" ,
                  color = gray(0.6))  +
     geom_line(aes(y=harvest,x=year), data = data.frame(harvest = harvest,
               year = data.years),lty=2) +
     geom_point(aes(y=index, x=year), data = data.frame(index=index,
```

```
              year = data.years)) +
    geom_line(aes(y=biomass,x=year,group=iter,col=iter),data = subset(biomass.iter,iter%in%1:10)) +
     ylab("Estimated B and C (million tonnes)") +
  theme_bw() +
  guides(col=FALSE)
```

```
## Warning: Removed 15 rows containing missing values (geom_point).
```



The shaded area indicates the range of the biomass time series, with the dark line the median. The lighter lines indicate indiviudal biomass trajectories.

Applying the Management Strategy We have now conditioned our operating model. We will conduct the MSE loop over a 20 year projection period, with the catches set each year by repeated estimation of the current biomass and application of a harvest control rule.

Define the years for the projection:

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

Data generation We write a function to generate the observations (new biomass index values) from the operating model.

```
observe <- function(biomass, sigma) {
  biomass * exp(rnorm(1, -0.5*sigma^2, sigma))
}
```

This function takes the true biomass from the operating model, and generates the data by adding (lognormally distributed) observation error.

Harvest Control Rule We first demonstrate the MSE using a fixed target exploitation rate - the control rule calculates the catch for next year based on a fixed percentage (10%) of the most recent biomass estimate.

```
control.pars <- list()
control.pars$Htarg <- 0.1
control <- function(estimated.biomass, control.pars) {
  control.pars$Htarg
}
```

We assume perfect implementation of the strategy - in that the realized catch is the same as the TAC.

```
implement <- function(TAC,...) {
  TAC
}
```

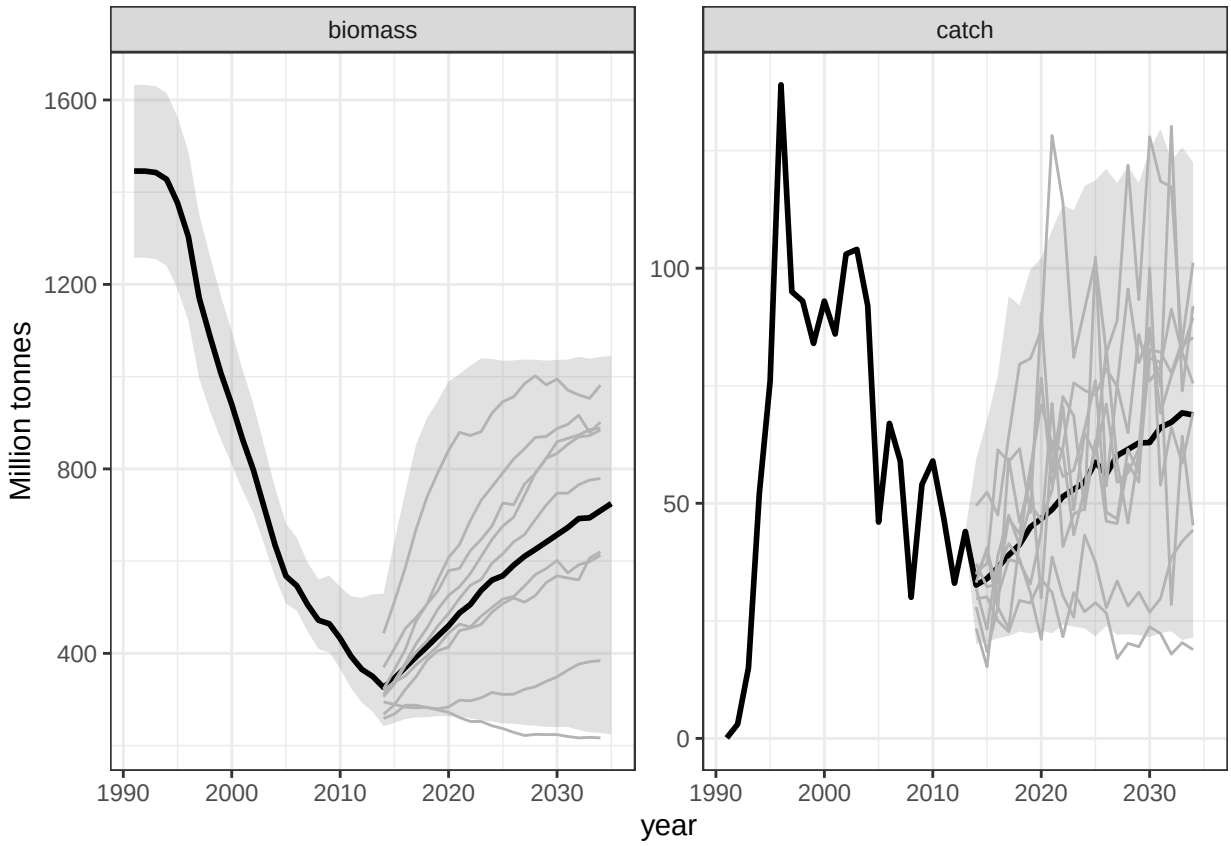
Project with fixed 10% exploitation rate of estimated biomass for all iterations & 20 yrs

```
project.fixed <- evaluate(pars.iter, biomass.iter, control.pars, data.years,
                          proj.years, niter)
tail(project.fixed)
```

```
##       year    value type iter
## 88495 2029 35.92181  tac  500
## 88496 2030 39.22611  tac  500
## 88497 2031 41.76298  tac  500
## 88498 2032 40.64805  tac  500
## 88499 2033 49.77911  tac  500
## 88500 2034 48.33598  tac  500
```

Plot the projection:

```
projection.plot(dplyr::filter(project.fixed, type != "tac"))
```



Management using alternative harvest control rules Define a HCR that converts estimated biomass into a harvest rate using a functional form determined by values in 'control.pars'.

```
control <- function(estimated.biomass, control.pars) {
  H1 <- control.pars$H1
  H2 <- control.pars$H2
  Bmax <- control.pars$Bmax
  B2 <- control.pars$B2
  B1 <- control.pars$B1

  harv <- ifelse(estimated.biomass >= B1, H1,
                 ifelse(estimated.biomass < B2, H2,
                        (H1-H2)/(B1-B2)*(estimated.biomass - B2) + H2))

  return(harv)

#end function control
}
```

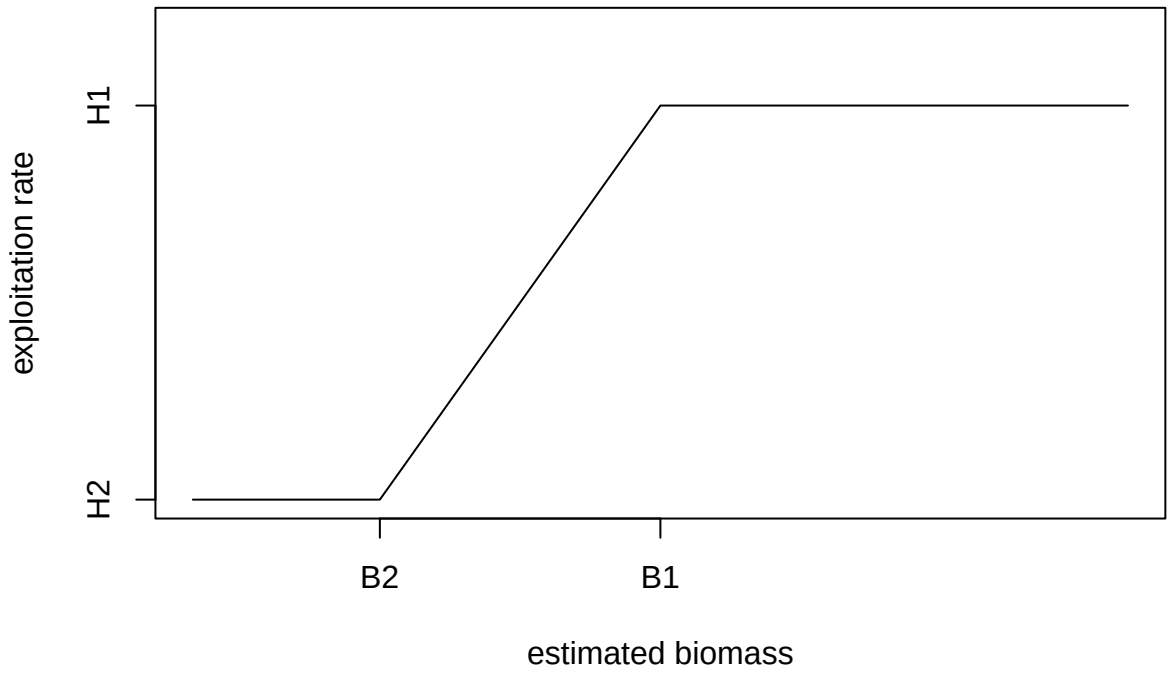
Define control parameters for HCR using reference points
We (arbitrarily) set the threshold and limit biomass reference points as 50% & 20% of the maximum observed survey index value during the historical period.
The target exploitation rate is set at 5%.

```
control.pars <- list()
control.pars$H1 <- 0.05
control.pars$H2 <- 0
control.pars$Bmax <- max(index, na.rm =TRUE)
```

```
control.pars$B2 <- 0.2*control.pars$Bmax
control.pars$B1 <- 0.5*control.pars$Bmax
```

Plot the HCR shape:

```
plot(c(0,control.pars$B2,control.pars$B1,control.pars$Bmax),
     c(control.pars$H2,control.pars$H2,control.pars$H1,control.pars$H1),
     type='l',axes=F,xlab="estimated biomass",ylab="exploitation rate",
     ylim=c(0,1.2*control.pars$H1))
axis(1,at=c(control.pars$B2,control.pars$B1),labels=c("B2","B1"))
axis(2,at=c(control.pars$H2,control.pars$H1),labels=c("H2","H1"))
box()
```



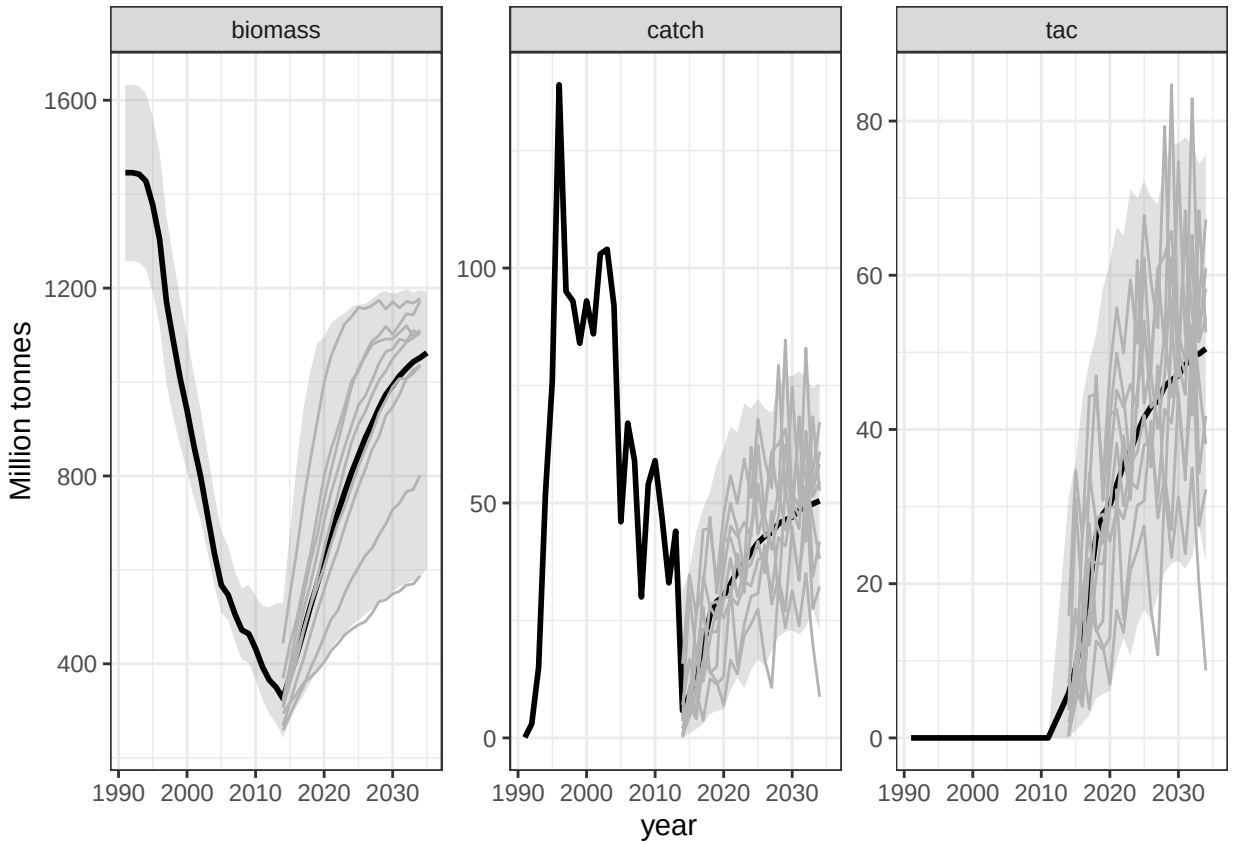
Conduct the evaluation by projecting system forward in time

```
project.hcr.sf <- evaluate(pars.iter, biomass.iter, control.pars,
                           data.years, proj.years, niter, m = 2)
```

Plot the trajectories:

```
projection.plot(project.hcr.sf)
```

```
## Warning: Removed 1000 rows containing non-finite values (stat_summary).

## Warning: Removed 1000 rows containing non-finite values (stat_summary).
```



Conduct the evaluation by projecting system forward in time

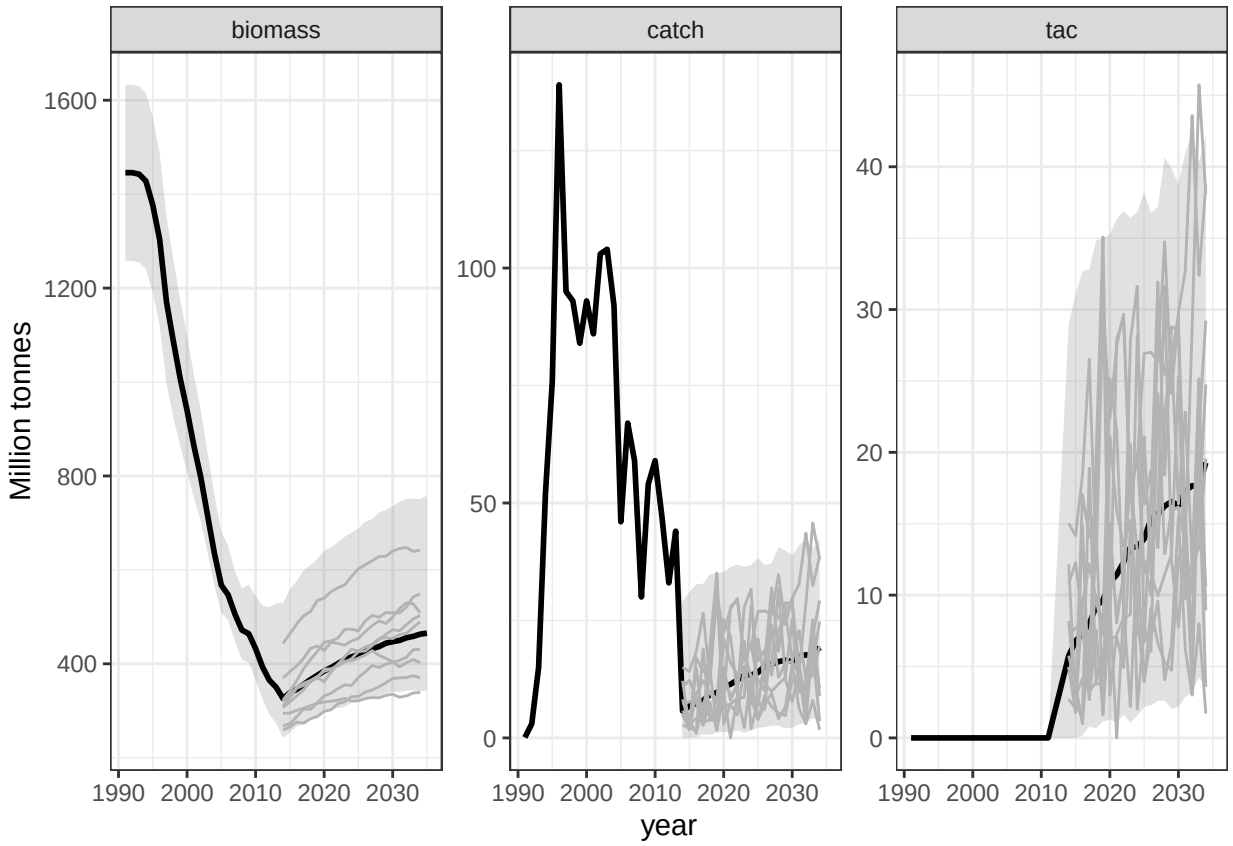
```
project.hcr.pt <- evaluate(pars.iter, biomass.iter, control.pars,
                           data.years, proj.years, niter, m = 1.2)
```

Plot the trajectories:

```
projection.plot(project.hcr.pt)
```

```
## Warning: Removed 1000 rows containing non-finite values (stat_summary).
```

```
## Warning: Removed 1000 rows containing non-finite values (stat_summary).
```



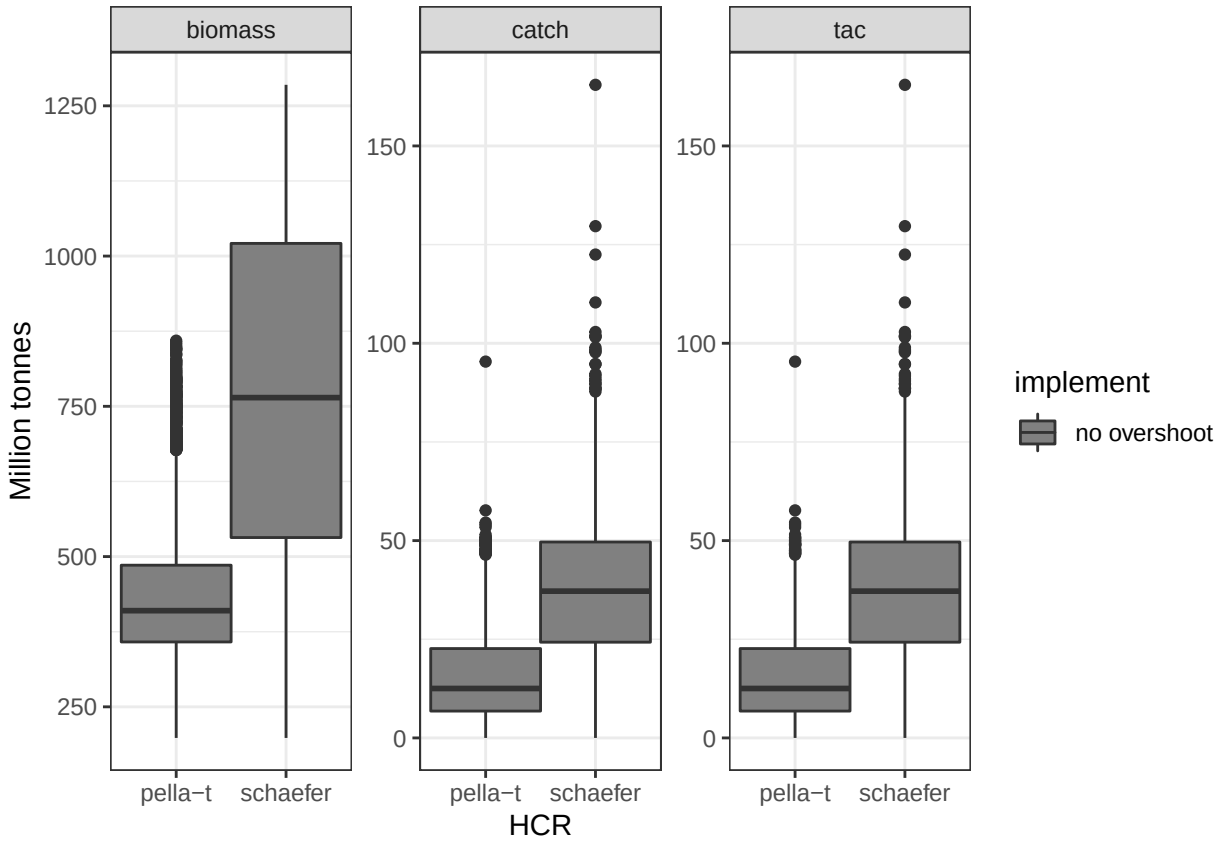
Diagnostics We have run the evaluations for 4 HCRs. We can now compare these.
Create an object containing all the results:

```
MSE <- rbind(cbind(project.hcr.sf, HCR="schaefer",
                   implement = "no overshoot"),
             cbind(project.hcr.pt, HCR="pella-t",
                   implement = "no overshoot"))
head(MSE)
```

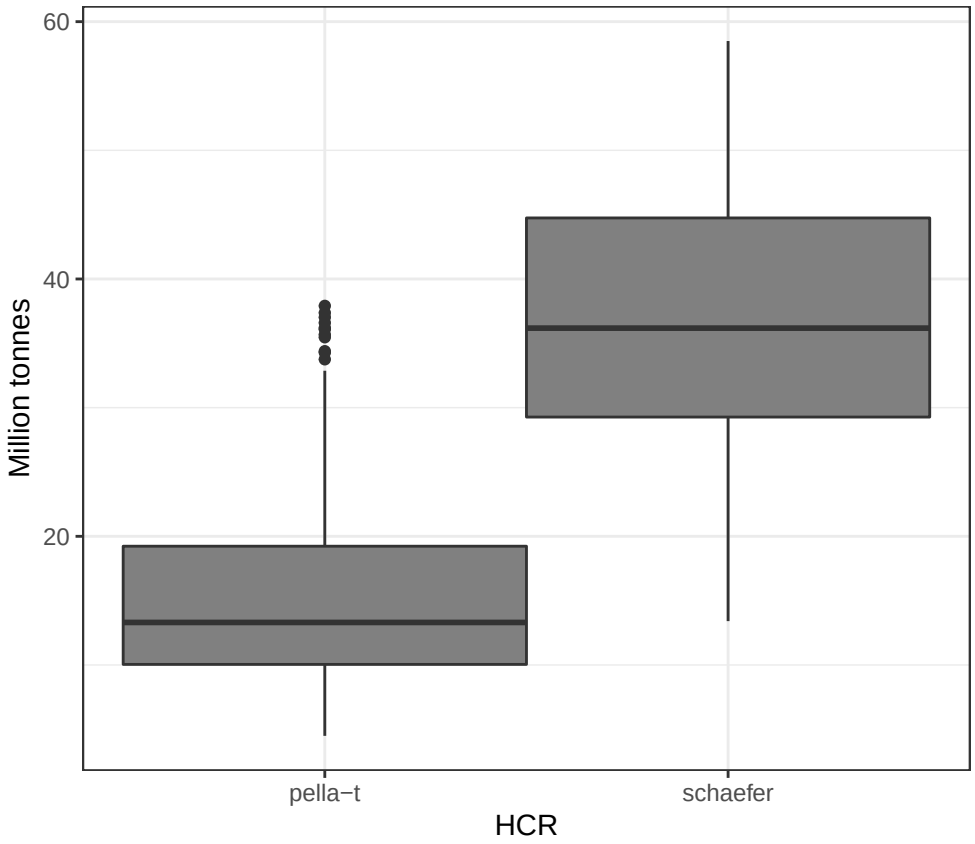
```
##   year value  type iter      HCR    implement
## 1 1991    NA index    1 schaefer no overshoot
## 2 1992    NA index    1 schaefer no overshoot
## 3 1993    NA index    1 schaefer no overshoot
## 4 1994    NA index    1 schaefer no overshoot
## 5 1995    NA index    1 schaefer no overshoot
## 6 1996    NA index    1 schaefer no overshoot
```

Summarize biomass & catch for all 4 options:

```
Fig5 <- ggplot(data=subset(MSE, type !="index" &
                           year %in% proj.years),
               aes(x=HCR, y=value, ymin=0))
Fig5 + geom_boxplot(aes(fill=implement), width = 1) + facet_wrap(~type, scale="free_y") + ylab("Million
```

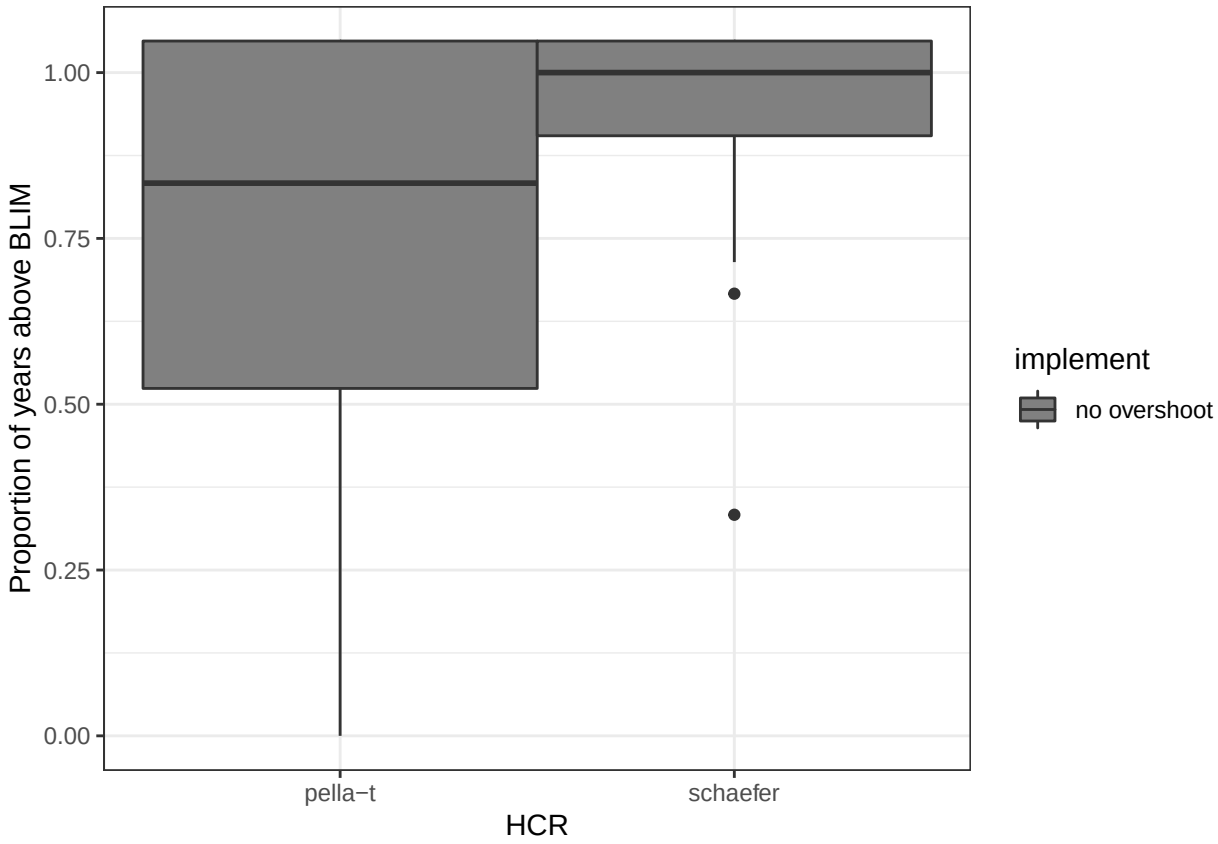



```
#Yield based metrics  (e.g. average annual catch)
#Stock Biomass metrics (e.g. distribution for B/BMSY, P(B>BLIM), etc.)
#Inter-annual stability of catch advice (e.g. how often the control rule closes the fishery)

aac2 <- with(MSE[MSE$year>max(data.years) & MSE$type=="catch",],
             aggregate(value,by=list(iter=iter,HCR=HCR,implement=implement),FUN=mean,na.rm=TRUE))


Fig6 <- ggplot(data=subset(aac2),
               aes(x=HCR, y=x, ymin=0))
Fig6 + geom_boxplot(aes(fill=implement), width = 1)  + ylab("Million tonnes") + scale_fill_grey(start=0
```

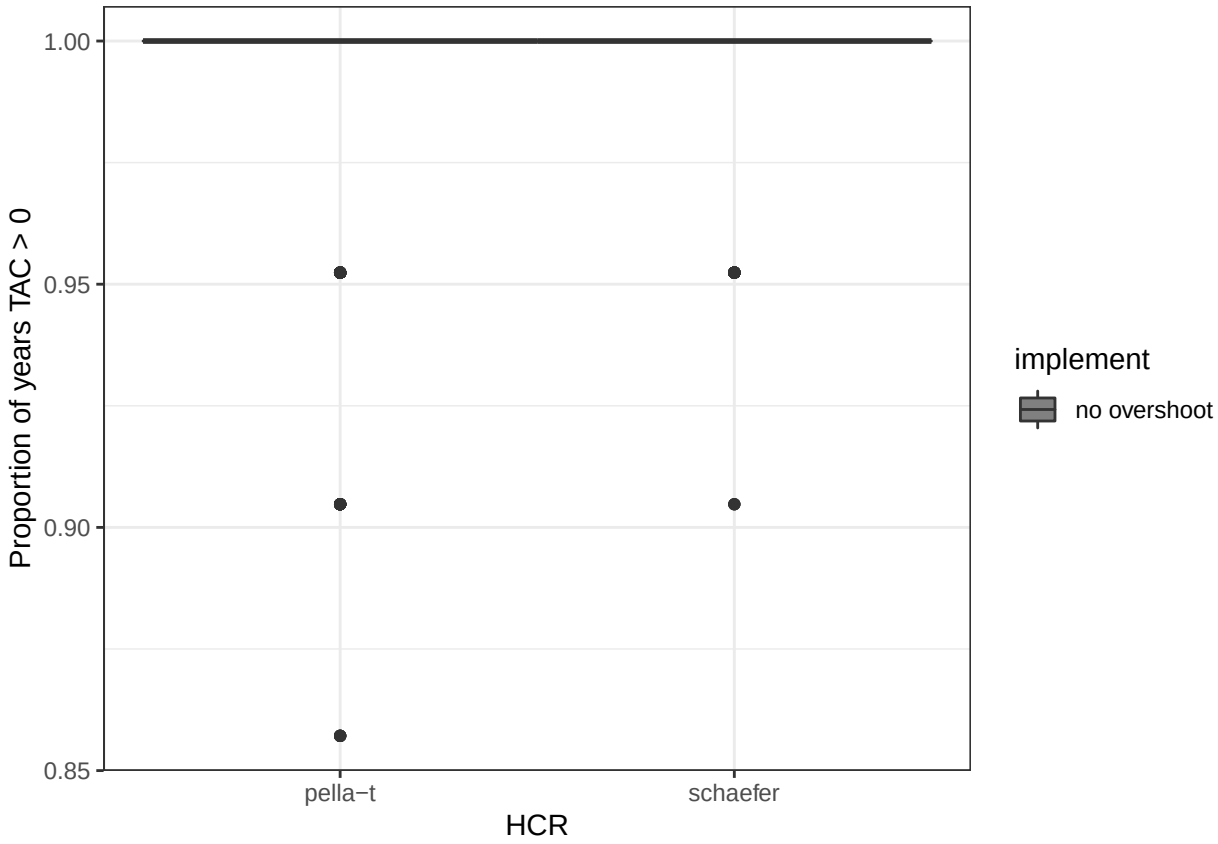


Performance statistics

```
# years B > BLIM
# BLIM = 0.25*K  (we specify BLIM for our performance as half BMSY)
blim <- 0.25*exp(pars.iter[,1])

num.above <- function(vec,threshold) {
 length(vec[vec>threshold])/length(vec)
}
MSE$blim <- blim[MSE$iter]
MSE$above.blim <- ifelse(MSE$value>MSE$blim,1,0)

above.blim <- with(MSE[MSE$year>max(data.years) & MSE$type=="biomass",],
            aggregate(above.blim,by=list(iter=iter,HCR=HCR,implement=implement),FUN=sum,na.rm=TRUE))
above.blim$x <- above.blim$x/length(proj.years)

Fig7 <- ggplot(data=subset(above.blim),
               aes(x=HCR, y=x, ymin=0))
Fig7 + geom_boxplot(aes(fill=implement), width = 1)  + ylab("Proportion of years above BLIM") + scale_f
```




```
# num years fishery is open
not.closed <- with(MSE[MSE$year>max(data.years) & MSE$type=="catch",],
             aggregate(value,by=list(iter=iter,HCR=HCR,implement=implement),FUN=num.above,threshold=0))

Fig8 <- ggplot(data=subset(not.closed),
               aes(x=HCR, y=x, ymin=0))
Fig8 + geom_boxplot(aes(fill=implement), width = 1)  + ylab("Proportion of years TAC > 0") + scale_fill_
```


1.00
0.95
0.90
0.85
Proportion of years TAC > 0
pella−t
schaefer
HCR
implement
no overshoot