CINAR MSE Workshop: First MSE

Gavin Fay

January 2022

Introduction to MSE

Here we will work through a simple example of applying MSE. Later this week we will take a more modular approach to implementing MSEs, but we walk through steps here to give you a flavor for how the pieces work and can be put together.

This lab is based (heavily) on tutorial by Katell Hamon & Jan-Jaap Poos, published in Chapter 3 of Edwards & Dankel (eds): *"Management Science in Fisheries, An introduction to simulation-based methods"*. All errors below are completely the fault of GF.

We consider a fishery for a population of *Sebastes electronicus*:
* the operating model population dynamics are governed by a logistic (Schaefer) production model.
* Data available from the fishery are the catch (known without error), and a biomass index (e.g. from a survey).
* We will apply a simple empirical harvest control rule to demonstrate the MSE, and use a small set of performance statistics to compare among alternative versions of the HCR.

There are plenty of places where additional complexity can be built in to this example. We encourage you to play around with adding functionality of interest. Some options could include adding a model-based control rule, changing the dynamics of the operating model, applying the control rule every 3 years instead of every year, etc.
* However, you should be able to walk through this tutorial without tweaks if you just want to get a feel for how things work.

We assume you have installed `R` on your computer and have an appropriate text editor or development environment (e.g. `Rstudio`).
First we install some libraries in `R` that we will use later.
(If you do not have these packages installed then see the code on the landing page of the website to 'install.packages()')

```
library(tidyverse)
library(ggdist)
library(Hmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following objects are masked from 'package:base':
```

```
##
##     format.pval, units
```

```
library(mvtnorm)
```

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

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

where B_y is the biomass in year y, C_y is the catch in year y, r is the population intrinsic growth rate, and K is the population carrying capacity.

We assume that the population is at carrying capacity in the first year of our simulation (i.e. $B_1 = K$).

Our first task is to condition our operating model, that we will then use to perform the MSE simulations.

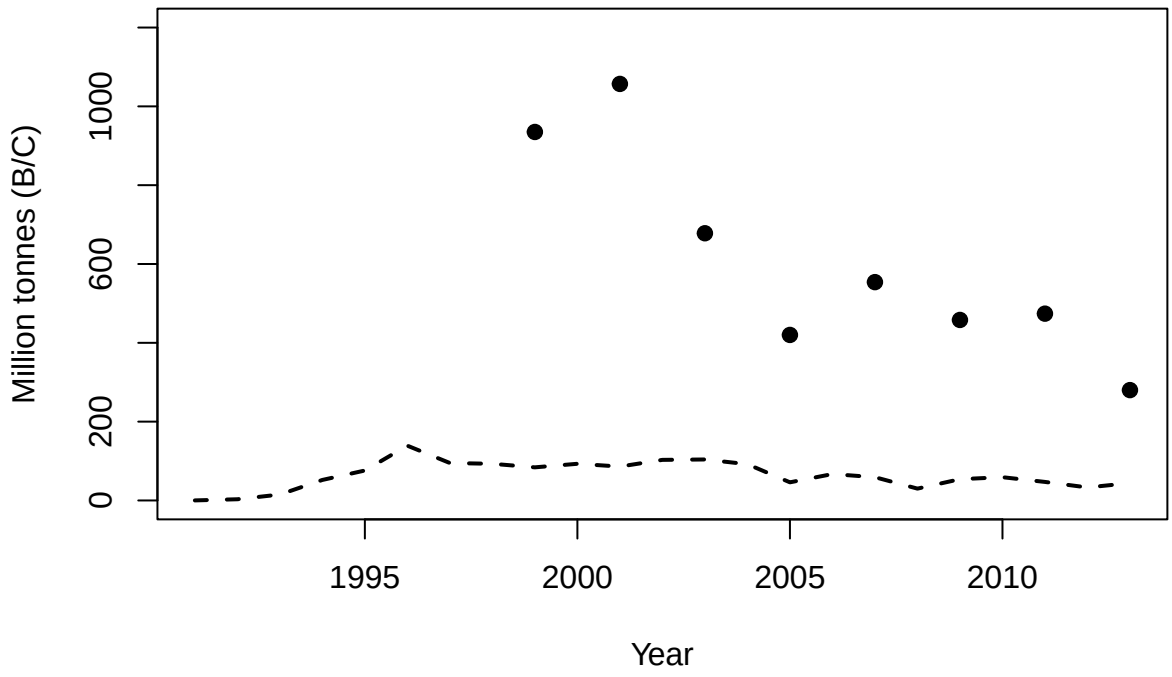
#####Specify input data and associated years

```
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)
```

We create time series of the years, catches (harvest), and biomass index data for our historical period that are already available.

We can plot these:

```
plot(data.years,index, pch=19,xlab="Year",ylab="Million tonnes (B/C)",
     ylim=c(0,1200))
lines(data.years,harvest,lty=2,lwd=2)
```



We see that the biomass index has been declining.

Now we will create some functions to use as we develop the operating model.

First, the logistic production function:

```
schaefer <- function(B,C,K,r) {
  #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) - C
  return(max(0.001,res))  # we add a constraint to prevent negative biomass
}
```

Now a function to do the biomass projection:

```
dynamics <- function(pars,C,yrs) {
  # 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] <- schaefer(B[y-1],C[y-1],K,r)
  }

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

  #end function dynamics
}
```

We are going to condition the operating model by estimating the parameters based on the historical biomass index data.

To do this we make a function that shows how well the current parameters fit the data, we assume that the observation errors around the true biomass are log-normally distributed.

```
# function to calculate the negative log-likelihood
nll <- function(pars,C,U) {  #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)  #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) {
  # 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,hessian=TRUE)

  # store the output from the model fit
  output <- list()
  output$pars <- res$par
  output$biomass <- dynamics(res$par,catch)
  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), log(r), log(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)
redfish
```

```
## $pars
## [1]  7.258575 -2.721061 -1.769252
##
## $biomass
##  [1] 1420.2310 1420.1310 1417.1376 1402.3407 1351.5032 1279.8069 1149.1339 1068.5681
##  [9]  992.9793  928.6366  856.7886  793.1563  713.2013  632.5654  563.6513  540.0219
## [17]  495.0459  457.2673  447.6696  413.8427  374.1402  345.2745  329.4716  302.1228
##
## $convergence
## [1] 0
##
## $nll
## [1] -2.802687
##
## $vcov
##               [,1]          [,2]          [,3]
## [1,]  4.859291e-03 -2.777817e-02 -2.724771e-06
## [2,] -2.777817e-02  1.690107e-01  1.448294e-05
## [3,] -2.724771e-06  1.448294e-05  6.250267e-02
```

Extract the maximum likelihood and parameter estimates

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

```
##  [1] 1420.2310 1420.1310 1417.1376 1402.3407 1351.5032 1279.8069 1149.1339 1068.5681
##  [9]  992.9793  928.6366  856.7886  793.1563  713.2013  632.5654  563.6513  540.0219
## [17]  495.0459  457.2673  447.6696  413.8427  374.1402  345.2745  329.4716  302.1228
```

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

```
## [1] 1.420231e+03 6.580487e-02 1.704604e-01
```

To obtain a set of 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.

```
#define the number of iterations for the MSE
niter <- 200
#set up a storage matrix for our alternative parameter sets
pars.iter <- matrix(NA,nrow = niter, ncol=3)
colnames(pars.iter) <- c("log_K","log_r","log_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))
}
biomass.iter
```

```
##     year   biomass iter
## 1   1991 1390.9170    1
## 2   1992 1390.8170    1
## 3   1993 1387.8244    1
## 4   1994 1373.0505    1
## 5   1995 1322.3431    1
## 6   1996 1251.1212    1
## 7   1997 1121.3371    1
## 8   1998 1042.2653    1
## 9   1999  968.4130    1
## 10  2000  905.9723    1
## 11  2001  836.1224    1
## 12  2002  774.5650    1
## 13  2003  696.7204    1
## 14  2004  618.2054    1
## 15  2005  551.3761    1
## 16  2006  529.7674    1
```

```
## 17  2007  486.8060    1
## 18  2008  450.9971    1
## 19  2009  443.3333    1
## 20  2010  411.4690    1
## 21  2011  373.7045    1
## 22  2012  346.7347    1
## 23  2013  332.8121    1
## 24  2014  307.3676    1
## 25  1991 1525.4662    2
## 26  1992 1525.3662    2
## 27  1993 1522.3710    2
## 28  1994 1507.5207    2
## 29  1995 1456.3798    2
## 30  1996 1383.5750    2
## 31  1997 1250.8095    2
## 32  1998 1166.7193    2
## 33  1999 1087.0114    2
## 34  2000 1018.1469    2
## 35  2001  941.5502    2
## 36  2002  873.0097    2
## 37  2003  788.0985    2
## 38  2004  702.5530    2
## 39  2005  628.9130    2
## 40  2006  600.8192    2
## 41  2007  551.4617    2
## 42  2008  509.5191    2
## 43  2009  495.9579    2
## 44  2010  458.1728    2
## 45  2011  414.7021    2
## 46  2012  382.3305    2
## 47  2013  363.2100    2
## 48  2014  332.6160    2
## 49  1991 1378.4702    3
## 50  1992 1378.3702    3
## 51  1993 1375.3778    3
## 52  1994 1360.6108    3
## 53  1995 1309.9421    3
## 54  1996 1238.8602    3
## 55  1997 1109.3360    3
## 56  1998 1030.6933    3
## 57  1999  957.3318    3
## 58  2000  895.4203    3
## 59  2001  826.1175    3
## 60  2002  765.1173    3
## 61  2003  687.8282    3
## 62  2004  609.8545    3
## 63  2005  543.5356    3
## 64  2006  522.3989    3
## 65  2007  479.9003    3
## 66  2008  444.5258    3
## 67  2009  437.2714    3
## 68  2010  405.8195    3
## 69  2011  368.4451    3
## 70  2012  341.8336    3
```

```
## 71  2013  328.2478    3
## 72  2014  303.1347    3
## 73  1991 1506.7679    4
## 74  1992 1506.6679    4
## 75  1993 1503.6721    4
## 76  1994 1488.8043    4
## 77  1995 1437.5633    4
## 78  1996 1364.3868    4
## 79  1997 1230.9001    4
## 80  1998 1145.5373    4
## 81  1999 1064.2814    4
## 82  2000  993.6469    4
## 83  2001  915.1172    4
## 84  2002  844.4835    4
## 85  2003  757.3566    4
## 86  2004  669.4648    4
## 87  2005  593.3736    4
## 88  2006  562.7556    4
## 89  2007  510.8329    4
## 90  2008  466.2719    4
## 91  2009  450.0410    4
## 92  2010  409.5382    4
## 93  2011  363.2913    4
## 94  2012  328.0812    4
## 95  2013  306.0563    4
## 96  2014  272.4858    4
## 97  1991 1389.5220    5
## 98  1992 1389.4220    5
## 99  1993 1386.4282    5
## 100 1994 1371.6207    5
## 101 1995 1320.7226    5
## 102 1996 1248.8005    5
## 103 1997 1117.6871    5
## 104 1998 1036.3223    5
## 105 1999  959.7490    5
## 106 2000  894.2601    5
## 107 2001  821.1364    5
## 108 2002  756.0821    5
## 109 2003  674.5757    5
## 110 2004  592.2199    5
## 111 2005  521.4104    5
## 112 2006  495.7242    5
## 113 2007  448.6087    5
## 114 2008  408.5519    5
## 115 2009  396.5381    5
## 116 2010  360.2092    5
## 117 2011  317.8486    5
## 118 2012  286.1355    5
## 119 2013  267.3044    5
## 120 2014  236.7667    5
## 121 1991 1426.1401    6
## 122 1992 1426.0401    6
## 123 1993 1423.0466    6
## 124 1994 1408.2494    6
```

```
## 125 1995 1357.4101    6
## 126 1996 1285.7082    6
## 127 1997 1155.0263    6
## 128 1998 1074.4527    6
## 129 1999  998.8610    6
## 130 2000  934.5232    6
## 131 2001  862.6888    6
## 132 2002  799.0825    6
## 133 2003  719.1666    6
## 134 2004  638.5899    6
## 135 2005  569.7593    6
## 136 2006  546.2380    6
## 137 2007  501.3807    6
## 138 2008  463.7412    6
## 139 2009  454.3023    6
## 140 2010  420.6424    6
## 141 2011  381.1277    6
## 142 2012  352.4765    6
## 143 2013  336.9111    6
## 144 2014  309.8174    6
## 145 1991 1400.3586    7
## 146 1992 1400.2586    7
## 147 1993 1397.2667    7
## 148 1994 1382.5168    7
## 149 1995 1331.9448    7
## 150 1996 1261.2203    7
## 151 1997 1132.3797    7
## 152 1998 1054.9478    7
## 153 1999  983.0438    7
## 154 2000  922.7940    7
## 155 2001  855.3075    7
## 156 2002  796.2968    7
## 157 2003  721.1444    7
## 158 2004  645.5014    7
## 159 2005  581.7108    7
## 160 2006  563.2808    7
## 161 2007  523.5783    7
## 162 2008  491.1552    7
## 163 2009  487.0082    7
## 164 2010  458.7599    7
## 165 2011  424.7682    7
## 166 2012  401.7594    7
## 167 2013  391.9861    7
## 168 2014  370.8697    7
## 169 1991 1554.5244    8
## 170 1992 1554.4244    8
## 171 1993 1551.4284    8
## 172 1994 1536.5533    8
## 173 1995 1485.2713    8
## 174 1996 1411.9457    8
## 175 1997 1278.1801    8
## 176 1998 1192.3641    8
## 177 1999 1110.5921    8
## 178 2000 1039.4113    8
```

```
## 179 2001  960.3327    8
## 180 2002  889.1694    8
## 181 2003  801.5520    8
## 182 2004  713.2448    8
## 183 2005  636.8465    8
## 184 2006  606.0420    8
## 185 2007  553.9879    8
## 186 2008  509.3999    8
## 187 2009  493.2425    8
## 188 2010  452.8533    8
## 189 2011  406.8251    8
## 190 2012  371.9653    8
## 191 2013  350.4024    8
## 192 2014  317.3730    8
## 193 1991 1221.4000    9
## 194 1992 1221.3000    9
## 195 1993 1218.3174    9
## 196 1994 1203.8523    9
## 197 1995 1154.8614    9
## 198 1996 1089.8070    9
## 199 1997  971.2347    9
## 200 1998  910.8437    9
## 201 1999  858.1360    9
## 202 2000  818.5393    9
## 203 2001  772.5106    9
## 204 2002  735.9054    9
## 205 2003  683.7967    9
## 206 2004  632.1599    9
## 207 2005  593.2187    9
## 208 2006  600.2994    9
## 209 2007  586.4083    9
## 210 2008  580.4486    9
## 211 2009  603.4426    9
## 212 2010  602.5594    9
## 213 2011  596.6743    9
## 214 2012  602.7705    9
## 215 2013  622.8859    9
## 216 2014  631.9891    9
## 217 1991 1526.5350   10
## 218 1992 1526.4350   10
## 219 1993 1523.4396   10
## 220 1994 1508.5810   10
## 221 1995 1457.3931   10
## 222 1996 1384.4145   10
## 223 1997 1251.3139   10
## 224 1998 1166.6399   10
## 225 1999 1086.2291   10
## 226 2000 1016.5696   10
## 227 2001  939.1136   10
## 228 2002  869.6543   10
## 229 2003  783.7827   10
## 230 2004  697.2380   10
## 231 2005  622.5751   10
## 232 2006  593.4494   10
```

```
## 233 2007  543.0526   10
## 234 2008  500.0663   10
## 235 2009  485.4571   10
## 236 2010  446.6108   10
## 237 2011  402.0721   10
## 238 2012  368.6282   10
## 239 2013  348.4264   10
## 240 2014  316.7342   10
## 241 1991 1387.9624   11
## 242 1992 1387.8624   11
## 243 1993 1384.8702   11
## 244 1994 1370.1098   11
## 245 1995 1319.4783   11
## 246 1996 1248.5340   11
## 247 1997 1119.2736   11
## 248 1998 1041.0995   11
## 249 1999  968.3036   11
## 250 2000  907.0388   11
## 251 2001  838.4446   11
## 252 2002  778.2224   11
## 253 2003  701.7708   11
## 254 2004  624.7129   11
## 255 2005  559.3900   11
## 256 2006  539.3219   11
## 257 2007  497.9292   11
## 258 2008  463.7242   11
## 259 2009  457.7033   11
## 260 2010  427.5254   11
## 261 2011  391.4985   11
## 262 2012  366.3249   11
## 263 2013  354.2638   11
## 264 2014  330.7524   11
## 265 1991 1422.9272   12
## 266 1992 1422.8272   12
## 267 1993 1419.8344   12
## 268 1994 1405.0570   12
## 269 1995 1354.3302   12
## 270 1996 1283.0409   12
## 271 1997 1153.1415   12
## 272 1998 1073.9161   12
## 273 1999  999.9210   12
## 274 2000  937.3681   12
## 275 2001  867.4466   12
## 276 2002  805.8791   12
## 277 2003  728.0933   12
## 278 2004  649.7455   12
## 279 2005  583.2185   12
## 280 2006  562.0508   12
## 281 2007  519.5850   12
## 282 2008  484.3844   12
## 283 2009  477.4359   12
## 284 2010  446.3250   12
## 285 2011  409.4267   12
## 286 2012  383.4672   12
```

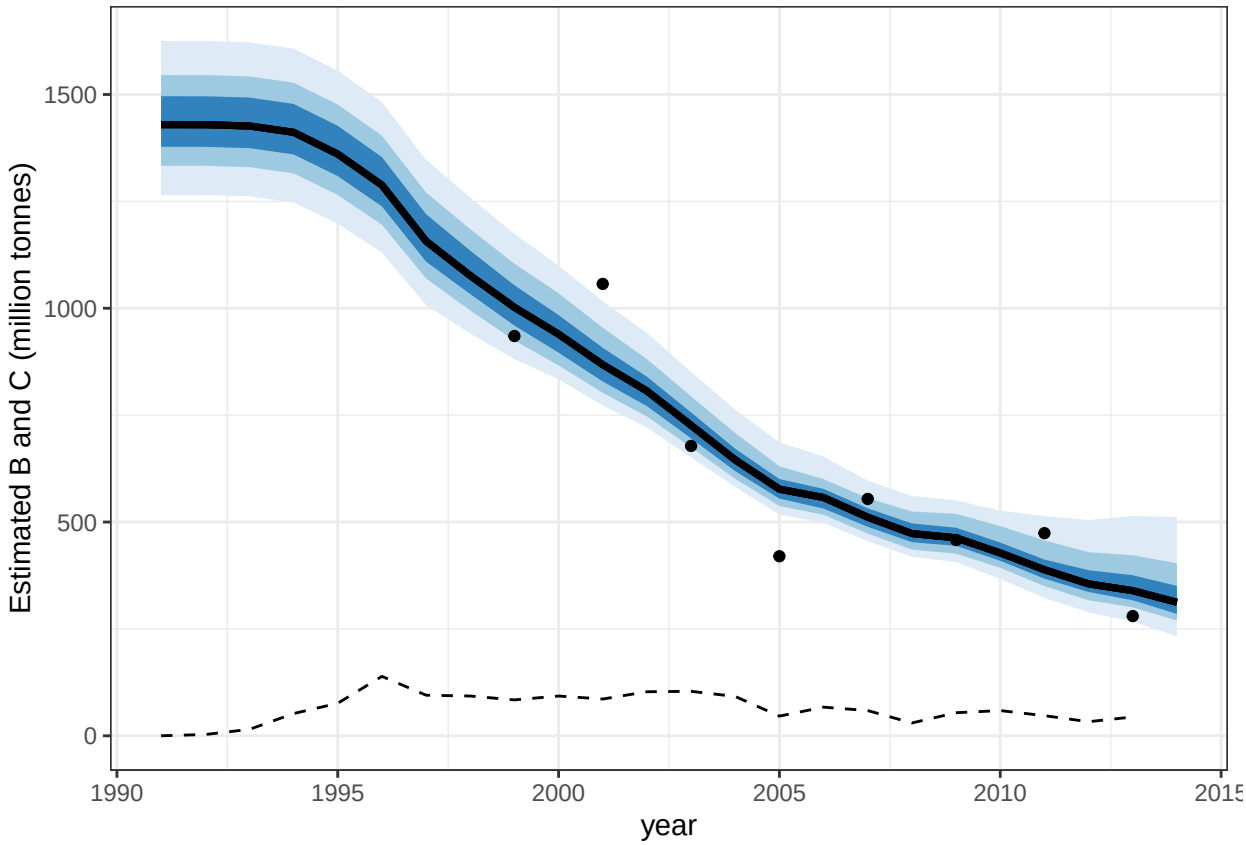
```
## 287 2013  370.6784   12
## 288 2014  346.4559   12
## 289 1991 1495.2185   13
## 290 1992 1495.1185   13
## 291 1993 1492.1229   13
## 292 1994 1477.2597   13
## 293 1995 1426.0454   13
## 294 1996 1352.9668   13
## 295 1997 1219.6667   13
## 296 1998 1134.6201   13
## 297 1999 1053.7371   13
## 298 2000  983.5146   13
## 299 2001  905.4192   13
## 300 2002  835.2345   13
## 301 2003  748.5600   13
## 302 2004  661.1128   13
## 303 2005  585.4441   13
## 304 2006  555.2181   13
## 305 2007  503.6747   13
## 306 2008  459.4653   13
## 307 2009  443.5593   13
## 308 2010  403.3742   13
## 309 2011  357.4177   13
## 310 2012  322.4616   13
## 311 2013  300.6613   13
## 312 2014  267.2980   13
## 313 1991 1336.4252   14
## 314 1992 1336.3252   14
## 315 1993 1333.3350   14
## 316 1994 1318.6369   14
## 317 1995 1268.3557   14
## 318 1996 1198.6819   14
## 319 1997 1071.7801   14
## 320 1998  997.5636   14
## 321 1999  929.3327   14
## 322 2000  873.0540   14
## 323 2001  809.6967   14
## 324 2002  754.9472   14
## 325 2003  684.1133   14
## 326 2004  612.8121   14
## 327 2005  553.3045   14
## 328 2006  539.0543   14
## 329 2007  503.5492   14
## 330 2008  475.2798   14
## 331 2009  475.2696   14
## 332 2010  451.2592   14
## 333 2011  421.5276   14
##  [ reached 'max' / getOption("max.print") -- omitted 4467 rows ]
```

We can now plot the estimated biomass time series

```
biomass.iter %>%
  group_by(year) %>%
  median_qi(biomass, .width = c(.5, .8, .95)) %>%
  ggplot() +
```

```
  geom_lineribbon(aes(x = year, y = biomass, ymin = .lower, ymax = .upper),
                  show.legend = FALSE) +
  scale_fill_brewer() +
  #theme_bw() +
  geom_line(aes(y=harvest,x=year), data = tibble(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(scale = "none")
```

```
## 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. (Uncomment the call to geom_line() to view some 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.

```
##### Data generation
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
}
```

Evaluation function that projects the operating model forward & implements the mgmt procudeure at each time step.
We will first step through this for one iteration to view how things work.

```
# evaluate <- function(pars.iter, biomass.iter,
#                      control.pars, data.years, proj.years,
#                      iterations, ...) {
  # function arguments:
  # pars.iter & biomass.iter, the parameters & historical biomass trajectories of the operating model
  # control.pars, the specifications of the harvest control rule

  # set up some indexing values
  iyr <- length(data.years)+1
  pyr <- length(proj.years)
  yrs <- c(data.years, proj.years, max(proj.years)+1)

  # set up a data frame to store the results
  res <- data.frame()

  # loop over the iterations of the MSE, each iteration conducts a 20 year projection with annual gener
  # observations and appliations of the control rule.
  #for(i in 1:iterations) {
  i = 1

    #extract the parameters for this iteration
    K.i <- exp(pars.iter[i,1])
    r.i <- exp(pars.iter[i,2])
    sig.i <- exp(pars.iter[i,3])

    #set up vectors for time series of interest.
    biomass.i <- c(subset(biomass.iter, iter==i)$biomass, numeric(pyr))
    index.i <- c(index,numeric(pyr))
    catch.i <- c(harvest, numeric(pyr))
```

```
    TAC.i <- numeric(pyr)

    # loop over the projection period.
    for (y in iyr:(iyr+pyr-1)) {
      #generate the data for the most recent year
      index.i[y] <- observe(biomass.i[y] , sig.i)
      #calculate the TAC based on the harvest control rule
      # note that the control rule ONLY sees the index data, not the operating model biomass.
      TAC.i [y]  <- control(index.i[y], control.pars) * index.i[y]
      #find the realized catch after implementation error
      catch.i[y] <- implement(TAC.i[y])

      # update the true biomass of the operating model based on the output of the HCR
      biomass.i[y+1] <- schaefer(biomass.i[y],catch.i[y],K.i,r.i)

      #end projection year loop for iteration i
    }

    #store the results for this iteration
    res <- rbind(res, data.frame(year = yrs[-length(yrs)],
                                 value = index.i, type = "index", iter = i),
                 data.frame(year = yrs[-length(yrs)],
                            value = catch.i, type = "catch", iter=i),
                 data.frame(year = yrs, value = biomass.i,
                            type= "biomass", iter=i))
    #end loop over iterations
  #}
  res
```

```
##     year      value    type iter
## 1   1991         NA   index    1
## 2   1992         NA   index    1
## 3   1993         NA   index    1
## 4   1994         NA   index    1
## 5   1995         NA   index    1
## 6   1996         NA   index    1
## 7   1997         NA   index    1
## 8   1998         NA   index    1
## 9   1999  935.00000   index    1
## 10  2000         NA   index    1
## 11  2001 1057.00000   index    1
## 12  2002         NA   index    1
## 13  2003  678.00000   index    1
## 14  2004         NA   index    1
## 15  2005  420.00000   index    1
## 16  2006         NA   index    1
## 17  2007  554.00000   index    1
## 18  2008         NA   index    1
## 19  2009  458.00000   index    1
## 20  2010         NA   index    1
## 21  2011  474.00000   index    1
## 22  2012         NA   index    1
## 23  2013  280.00000   index    1
## 24  2014  268.34448   index    1
```

```
## 25  2015  251.74592   index    1
## 26  2016  327.03246   index    1
## 27  2017  284.19137   index    1
## 28  2018  293.35511   index    1
## 29  2019  248.39732   index    1
## 30  2020  227.61370   index    1
## 31  2021  253.12306   index    1
## 32  2022  213.68682   index    1
## 33  2023  193.40387   index    1
## 34  2024  197.72370   index    1
## 35  2025  205.08045   index    1
## 36  2026  207.99937   index    1
## 37  2027  142.93710   index    1
## 38  2028  199.08928   index    1
## 39  2029  167.77435   index    1
## 40  2030  149.61077   index    1
## 41  2031  157.95987   index    1
## 42  2032  125.01135   index    1
## 43  2033  149.84648   index    1
## 44  2034  109.20222   index    1
## 45  1991    0.10000   catch    1
## 46  1992    3.00000   catch    1
## 47  1993   15.00000   catch    1
## 48  1994   52.00000   catch    1
## 49  1995   76.00000   catch    1
## 50  1996  139.00000   catch    1
## 51  1997   95.00000   catch    1
## 52  1998   93.00000   catch    1
## 53  1999   84.00000   catch    1
## 54  2000   93.00000   catch    1
## 55  2001   86.00000   catch    1
## 56  2002  103.00000   catch    1
## 57  2003  104.00000   catch    1
## 58  2004   92.00000   catch    1
## 59  2005   46.00000   catch    1
## 60  2006   67.00000   catch    1
## 61  2007   59.00000   catch    1
## 62  2008   30.00000   catch    1
## 63  2009   54.00000   catch    1
## 64  2010   59.00000   catch    1
## 65  2011   47.00000   catch    1
## 66  2012   33.00000   catch    1
## 67  2013   44.00000   catch    1
## 68  2014   26.83445   catch    1
## 69  2015   25.17459   catch    1
## 70  2016   32.70325   catch    1
## 71  2017   28.41914   catch    1
## 72  2018   29.33551   catch    1
## 73  2019   24.83973   catch    1
## 74  2020   22.76137   catch    1
## 75  2021   25.31231   catch    1
## 76  2022   21.36868   catch    1
## 77  2023   19.34039   catch    1
## 78  2024   19.77237   catch    1
```

```
## 79  2025   20.50805   catch    1
## 80  2026   20.79994   catch    1
## 81  2027   14.29371   catch    1
## 82  2028   19.90893   catch    1
## 83  2029   16.77743   catch    1
## 84  2030   14.96108   catch    1
## 85  2031   15.79599   catch    1
## 86  2032   12.50114   catch    1
## 87  2033   14.98465   catch    1
## 88  2034   10.92022   catch    1
## 89  1991 1390.91703 biomass    1
## 90  1992 1390.81703 biomass    1
## 91  1993 1387.82436 biomass    1
## 92  1994 1373.05052 biomass    1
## 93  1995 1322.34314 biomass    1
## 94  1996 1251.12116 biomass    1
## 95  1997 1121.33707 biomass    1
## 96  1998 1042.26534 biomass    1
## 97  1999  968.41296 biomass    1
## 98  2000  905.97234 biomass    1
## 99  2001  836.12238 biomass    1
## 100 2002  774.56495 biomass    1
## 101 2003  696.72036 biomass    1
## 102 2004  618.20543 biomass    1
## 103 2005  551.37611 biomass    1
## 104 2006  529.76739 biomass    1
## 105 2007  486.80595 biomass    1
## 106 2008  450.99711 biomass    1
## 107 2009  443.33331 biomass    1
## 108 2010  411.46898 biomass    1
## 109 2011  373.70452 biomass    1
## 110 2012  346.73470 biomass    1
## 111 2013  332.81206 biomass    1
## 112 2014  307.36756 biomass    1
## 113 2015  298.08207 biomass    1
## 114 2016  290.07214 biomass    1
## 115 2017  274.19474 biomass    1
## 116 2018  261.90986 biomass    1
## 117 2019  248.15528 biomass    1
## 118 2020  238.25807 biomass    1
## 119 2021  229.96752 biomass    1
## 120 2022  218.72296 biomass    1
## 121 2023  210.86376 biomass    1
## 122 2024  204.63475 biomass    1
## 123 2025  197.65360 biomass    1
## 124 2026  189.57312 biomass    1
## 125 2027  180.77339 biomass    1
## 126 2028  178.00667 biomass    1
## 127 2029  169.47427 biomass    1
## 128 2030  163.60425 biomass    1
## 129 2031  159.22339 biomass    1
## 130 2032  153.76106 biomass    1
## 131 2033  151.28334 biomass    1
## 132 2034  146.18034 biomass    1
```

```
## 133 2035  144.84774 biomass    1
```

```
#  return(res)
#  #end function evaluate()
#}
```

Reloading the full function with lines uncommented (code hidden from html to save scrolling time), means we can then run this.

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)
```

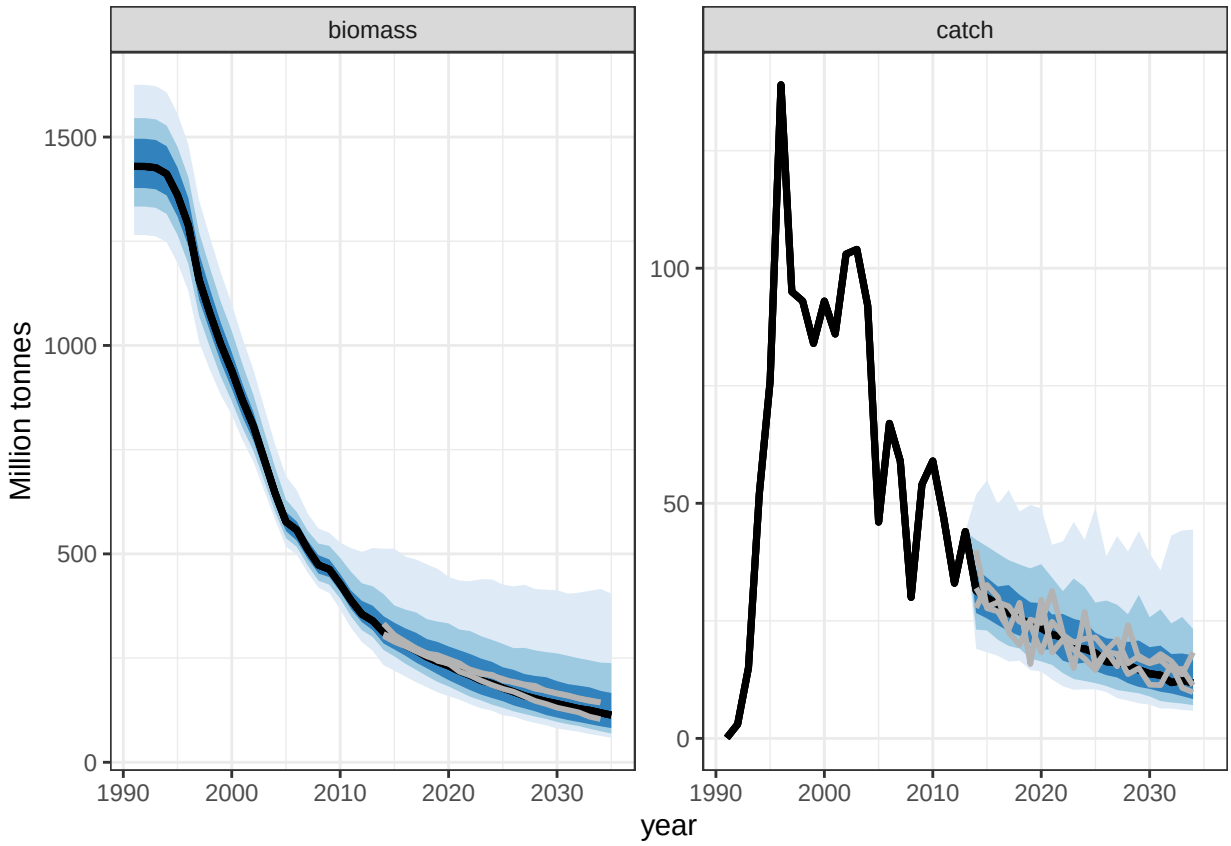
```
##       year     value    type iter
## 26595 2030 129.69299 biomass  200
## 26596 2031 118.10417 biomass  200
## 26597 2032 112.18595 biomass  200
## 26598 2033 105.97432 biomass  200
## 26599 2034 102.78694 biomass  200
## 26600 2035  99.75374 biomass  200
```

We can view the trajectories of catch and operating model biomass from the output.
We will do this again so write a function to repeat the task easily

```
projection.plot <- function(project.results) {
  #Fig2 <- ggplot(data = subset(project.results, type != "index"),
  #             aes(x = year, y = value))
  project.results %>%
    filter(type %in% c("biomass","catch")) %>%
    group_by(type, year) %>%
    median_qi(value, .width = c(.5, .8, .95)) %>%
    ggplot() +
    geom_lineribbon(aes(x = year, y = value, ymin = .lower, ymax = .upper),
                    show.legend = FALSE) +
    scale_fill_brewer() +
    geom_line(aes(y=value,x=year),data = subset(project.results, type != "index" & iter==1 & year %in% p
    geom_line(aes(y=value,x=year),data = subset(project.results, type != "index" & iter==2 & year %in% p
    geom_line(aes(y=value,x=year),data = subset(project.results, type != "index" & iter==3 & year %in% p
#stat_summary(fun.data = "median_hilow", geom = "smooth", col="black",
#                     fill = gray(0.5), lty = 2, aes=0.1) +
#       stat_summary(fun = median, fun.min = function(x)0, geom="line",
#                     data = subset(project.results, type != "index" &  year %in% data.years), lwd=1)
    facet_wrap(~type, scale = "free_y") +
    ylab("Million tonnes") +
    theme_bw()

}
```

Plot the projection:

```
projection.plot(project.fixed)
```



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
}
```

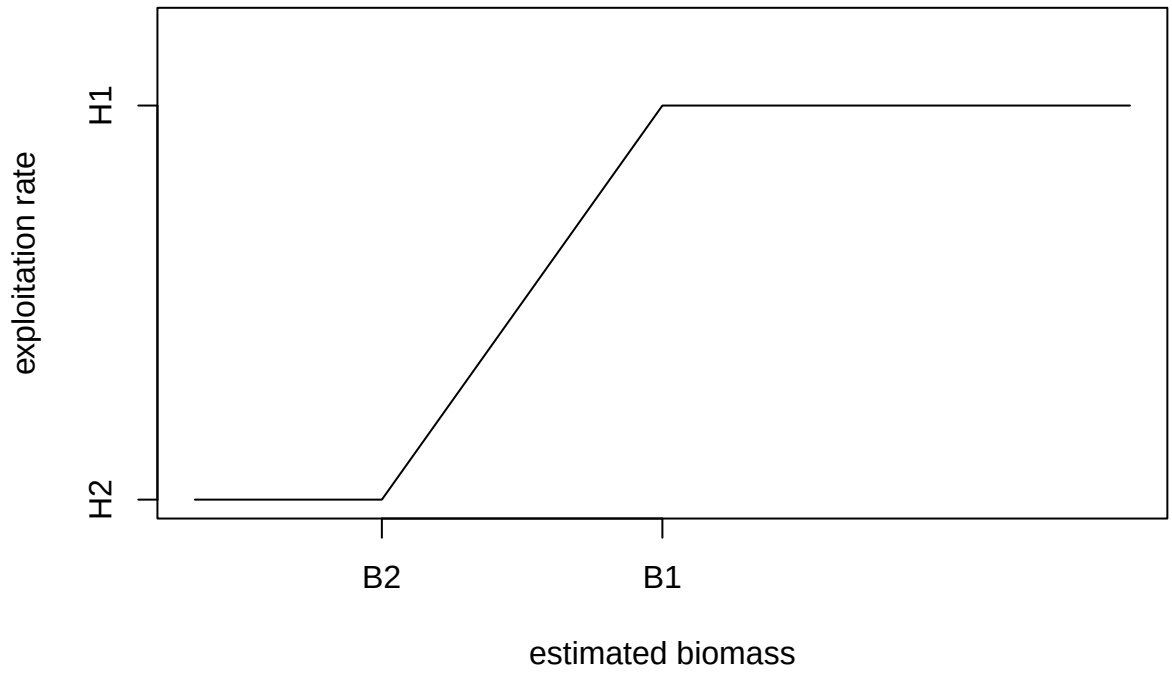
Define control parameters for HCR using reference points
We (arbitrarily) set the threshold and limit biomass reference points as 50% & 20% of the maximum observed survey index value during the historical period.
The target exploitation rate is set at 5%.

```
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()
```

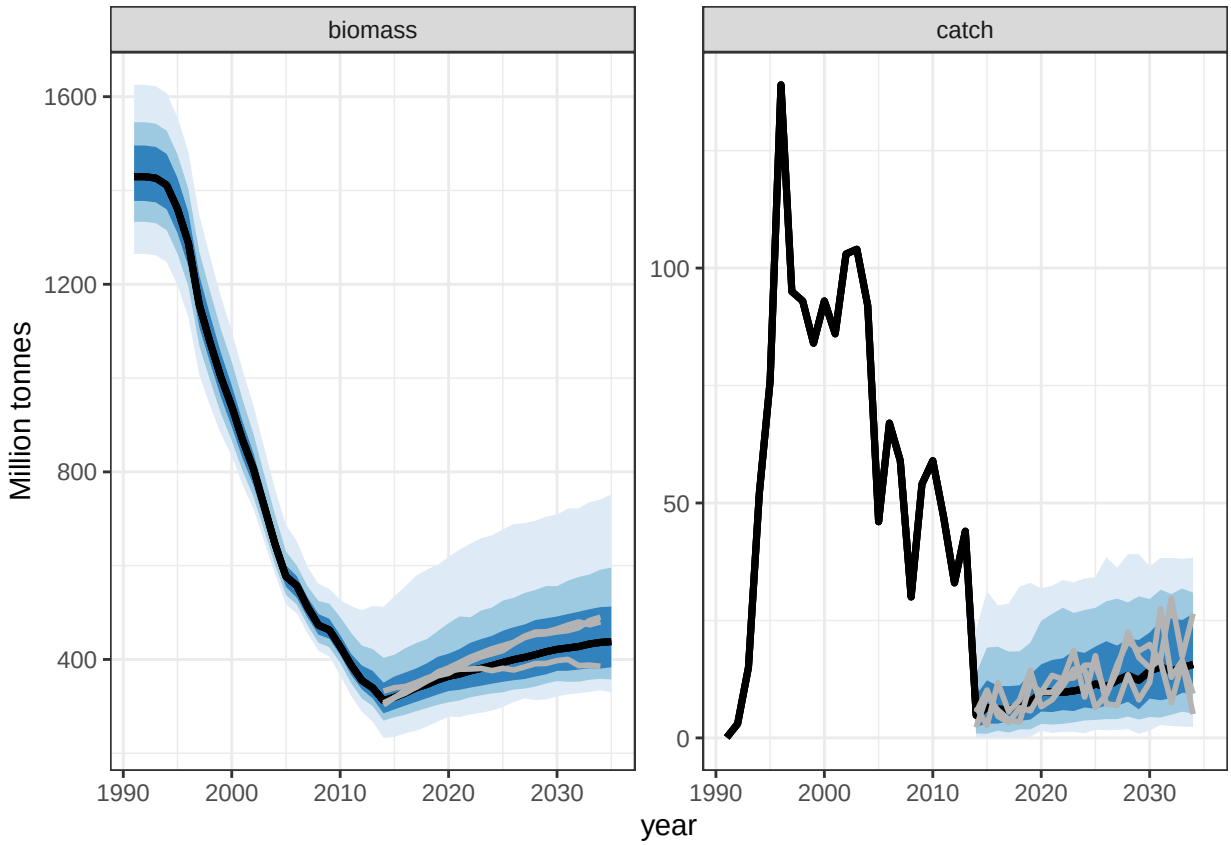


Conduct the evaluation by projecting system forward in time

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

Plot the trajectories:

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



Now let's add potential for overshooting the TAC

```
implement <- function(TAC, overshoot, ...) {
  TAC * (1 + overshoot)
}
```

Comparing different HCRs & accounting for possible TAC overshoot Set the HCR parameters

```
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
```

Conduct the base scenario (no TAC overshoot)

```
proj.hcr1.noerror <- evaluate(pars.iter, biomass.iter,
                              control.pars, data.years,
                              proj.years, niter,
                              overshoot = 0)
```

Now run the HCR with 20% overshoot in TAC

```
proj.hcr1.error <- evaluate(pars.iter, biomass.iter,
                                control.pars, data.years,
                                proj.years, niter,
                                overshoot = 0.2)
```

We will further do two more HCRs where we increase the target harvest rate:

```
control.pars$H1 <- 0.15
```

Run both scenarios with this new target harvest rate

```
proj.hcr2.noerror <- evaluate(pars.iter, biomass.iter,
                              control.pars, data.years,
                              proj.years, niter,
                              overshoot = 0)
proj.hcr2.error <- evaluate(pars.iter, biomass.iter,
                            control.pars, data.years,
                            proj.years, niter,
                            overshoot = 0.2)
```

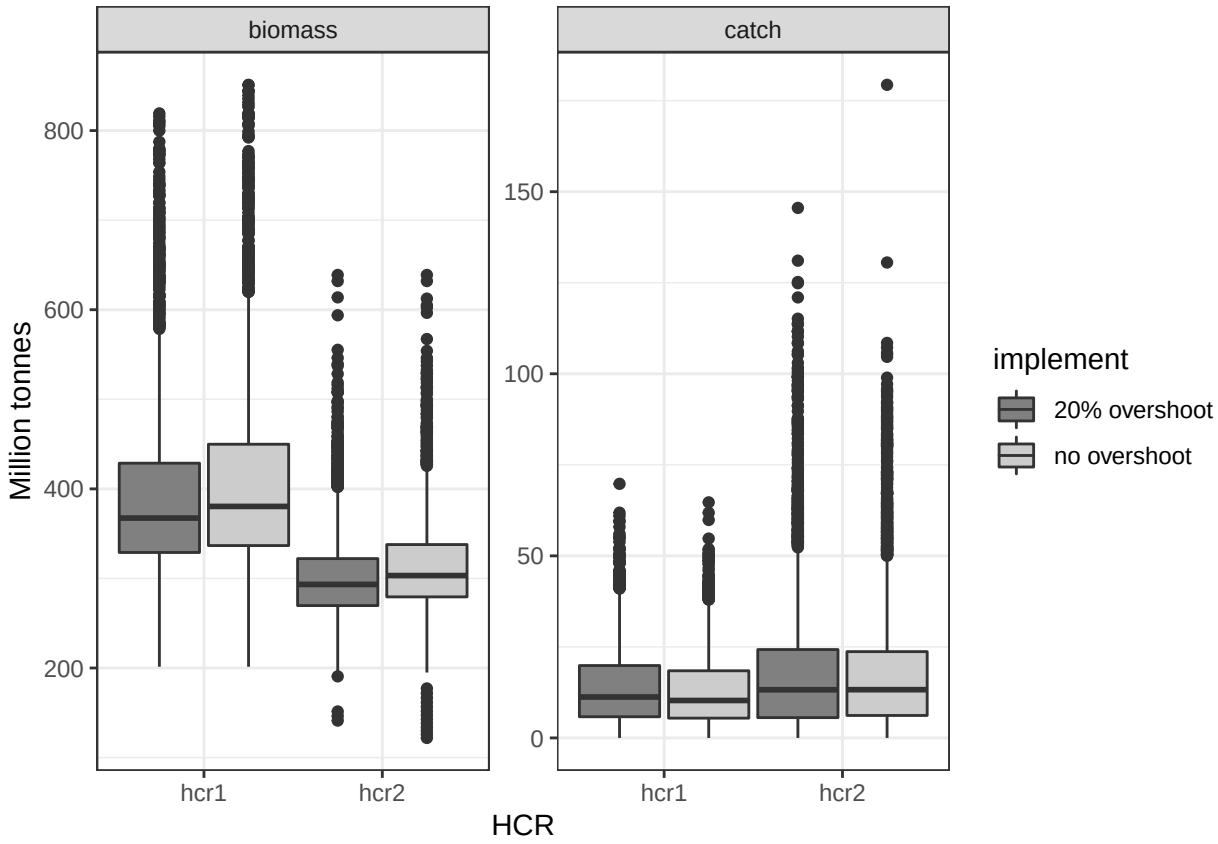
Diagnostics We have run the evaluations for 4 HCRs. We can now compare these.
Create an object containing all the results:

```
MSE <- rbind(cbind(proj.hcr1.noerror, HCR="hcr1",
                   implement = "no overshoot"),
             cbind(proj.hcr1.error, HCR="hcr1",
                   implement = "20% overshoot"),
             cbind(proj.hcr2.noerror, HCR="hcr2",
                   implement = "no overshoot"),
             cbind(proj.hcr2.error, HCR="hcr2",
                   implement = "20% overshoot"))
head(MSE)
```

```
##   year value  type iter  HCR    implement
## 1 1991    NA index    1 hcr1 no overshoot
## 2 1992    NA index    1 hcr1 no overshoot
## 3 1993    NA index    1 hcr1 no overshoot
## 4 1994    NA index    1 hcr1 no overshoot
## 5 1995    NA index    1 hcr1 no overshoot
## 6 1996    NA index    1 hcr1 no overshoot
```

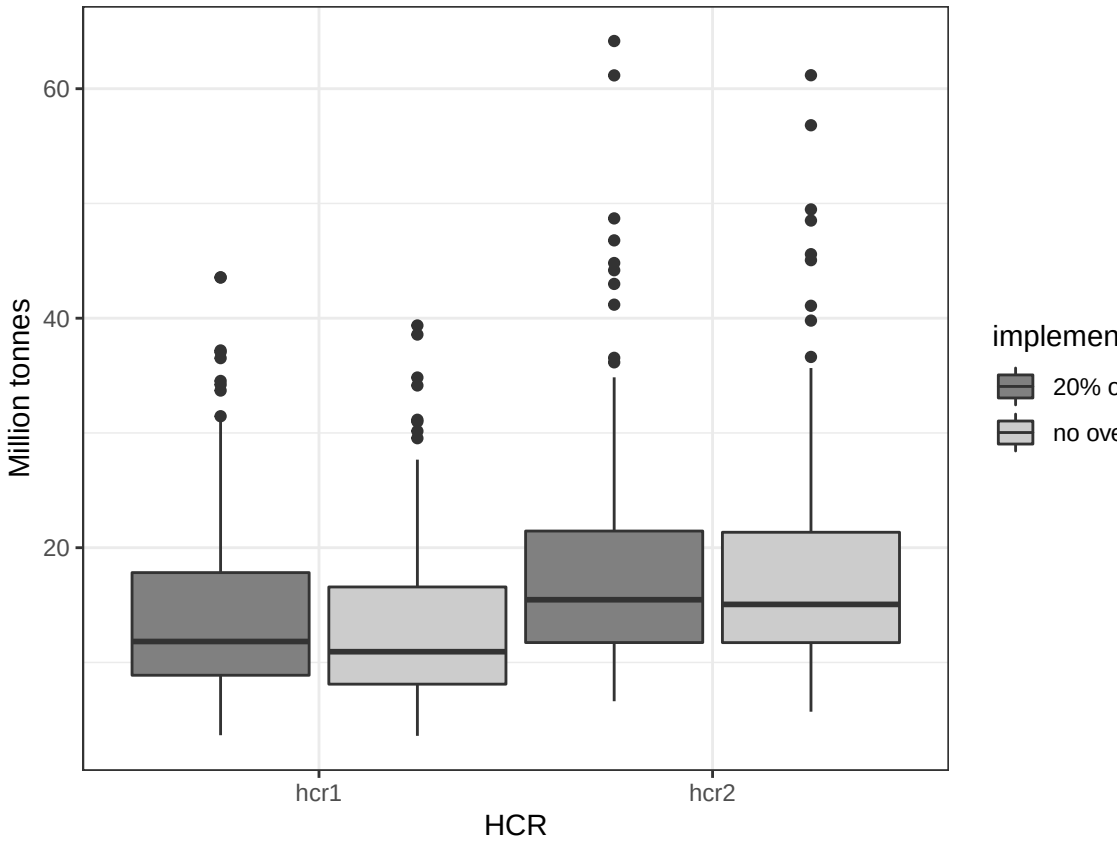
Summarize biomass & catch for all 4 options:

```
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
```



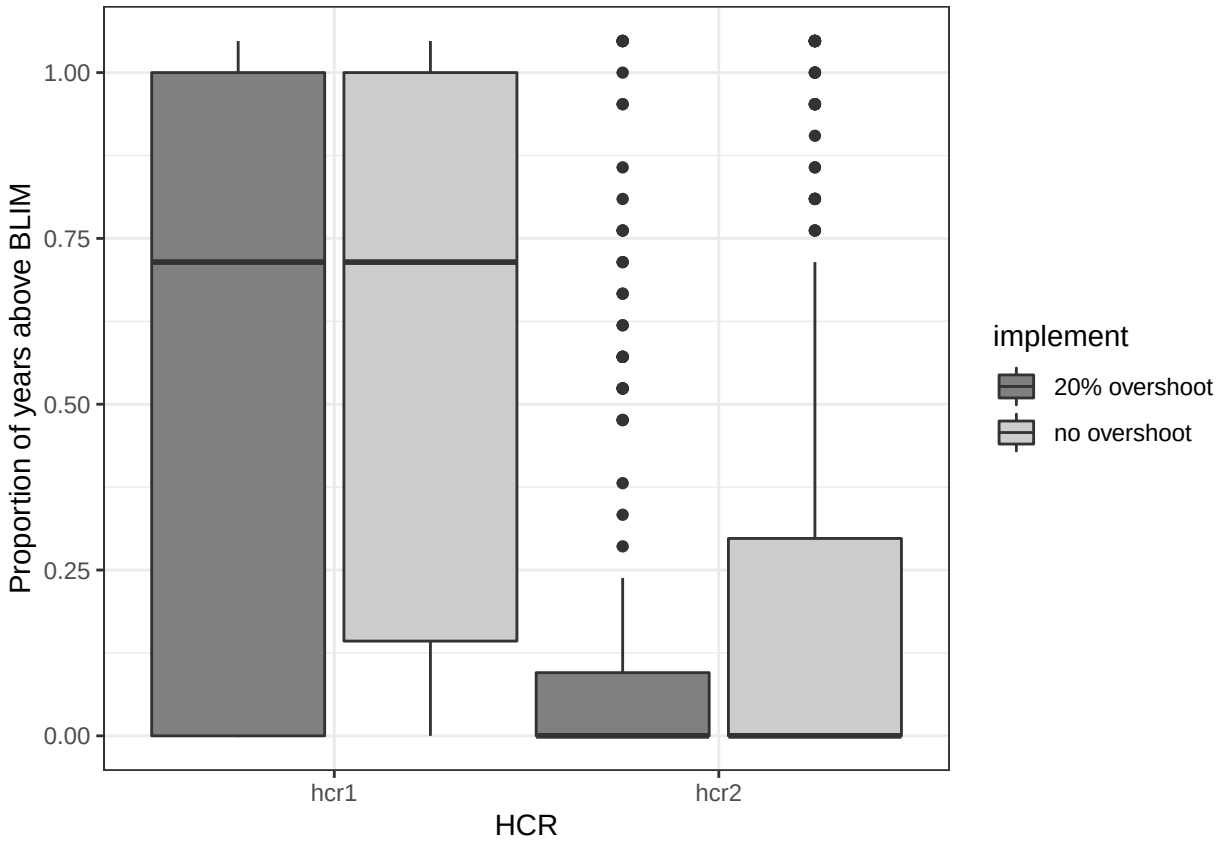
We immediately see a yield-biomass tradeoff - HCR2 gives more catch but leads to lower biomass. There is not much change when the catch is 20% higher than the TAC.

```
#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
```

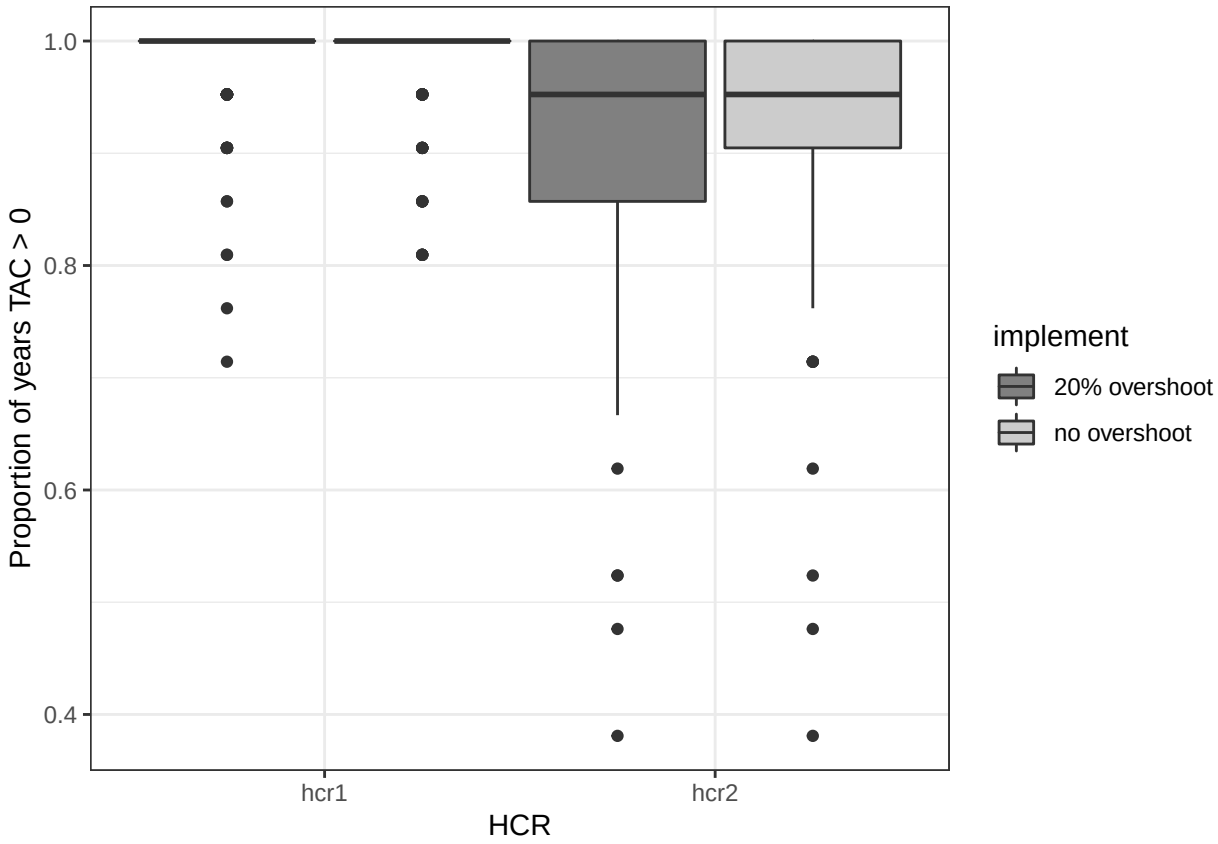


Performance statistics

```
# 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_
```



Next Steps

Your turn to add features!

Suggestions:
1. Produce a trade-off plot (hint: perhaps think about some alternative performance statistics that integrate across iterations)
2. Add a model-based control rule by performing a stock assessment (e.g. production model) each year in the projection period. Then use the catch associated with the estimated FMSY as the TAC. Be careful not to give the assessment model the true parameter values from the operating model.
3. Implement the HCR every 3 yrs rather than every 1.
4. Add a more complicated implementation function (say based on price?)
5. Add environmental variability (process error) into the population dynamics