Some notes on setting up HCR simulations using the fishvise package

Einar Hjorleifsson

November 26, 2013

- R version 3.0.1 (2013-05-16), `x86_64-redhat-linux-gnu`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: data.table 1.8.8, ggplot2 0.9.3.1, knitr 1.2, lubridate 1.3.0, plyr 1.8, RColorBrewer 1.0-5, reshape2 1.2.2, scales 0.2.3, stringr 0.6.2
- Loaded via a namespace (and not attached): colorspace 1.2-2, dichromat 2.0-0, digest 0.6.3, evaluate 0.4.4, formatR 0.8, grid 3.0.1, gtable 0.1.2, labeling 0.2, MASS 7.3-27, munsell 0.4.2, proto 0.3-10, tools 3.0.1

```
[1] "This document was created in knitr"
```

Contents

1 Preamble

some nice stuff

2 A quick overview of the operation of the program

Here a quick overview is provided on how the functional routines are set up (besides the error structure). The HCR evaluation of the Icelandic cod HCR rule is here taken as an example. The needed input files for that stock are available as part of the fishvise package.

1. Read in the control file:

   ```
   suppressPackageStartupMessages(require(fishvise))
   file <- paste(path.package("fishvise"), "extdata/hcr_iCod.ctr", sep = "/")
   ctr <- hcr_read_ctr(file)
   ```

2. Set the sweep of management harvest rate vectors. For a demonstration because of consideration of speed only run a 0, 10, 20 and 30% rule is setup. Besides creating the vector of management harvest rate vectors one needs to pass it to the control file (that was read in the last chunk of code).

   ```
   HRATE <- c(1e-05, 0.1, 0.2, 0.25)
   ctr$HRATE <- HRATE
   ```

3. Set up the structure. Creates a long list of array objects stored in an object (list) here named monsterStructure. This contains the nuts and bolts of any HCR evaluation, so the details are provided later in the document (on ToDo-list).

   ```
   monsterStructure <- hcr_set_dimensions(ctr)
   ```

4. Read in the starting year information (details also on the documentation ToDo-list) and pass it, the control file data and the monster structure to hcr_starting_conditions function wich will set up a an object in the workspace (working name X) containg needed information for the stock in question to start the HCR-evaluation (also on the documentation ToDo-list).

   ```
   file <- paste(path.package("fishvise"), "extdata/hcr_iCod.dat", sep = "/")
   dat <- hcr_read_startfile(file)
   hcr_set_starting_conditions(dat, monsterStructure, ctr)
   ```

5. Here we start the actual evaluation by inititialize the first year. One should think of this as the assessment year (current year). The way that the hcr-evaluation is set up is that the TAC has already been set for the assessment year (effectively set last year based on last years assessment).

   (a) Given that the TAC in the current year was based on last years assessment that of course includes observation error we need to calculate the realized harvest rate that is assiciated with that decision. We hence start by estimating the mortality multiplier that is associated with the set TAC. This is done using the hcr_TAC_to_Fmult function:

   (b) In the second step we apply the realized mortality to the true stock in numbers. This is often referred to as the operating model, called by the hcr_operating_mode function: Within the operating model resides a recruitment model to estimate incoming recruits (age 0) in the assessment year. Once the stock in numbers of age zero numbers are available the population is carried though to the end of the year based on the conventional stock equation. On the programming To-Do list is to implement this through the switch function.

   In the actual coding we may have something like the following for the first year:

```
y <- 1
for (h in 1:length(HRATE)) {
    Fmult <- hcr_TAC_to_Fmult(y, h)
    hcr_operating_model(y, h, ctr, Fmult, nR = 1)
}
```

6. Now for main year loop, starting in the second year. Which one can think of as the current assessment year. What is done in the fishvise framework is to run all the iterations within a year in one sweep. I.e. we do not have a for-loop. Hence we loop through each management decision mortality within the first year. This is actually what is done in all consequitve steps.

   (a) Modelling implementation error is done in the first step. This model is something that is mentioned in the Bible's but rarely used. In principle the actual removals in forms of catch are set differently than that dictated in the actual HCR. If that is the case one really should question if it is worth while doing a HCR evaluation at all.

   Whatever the case, here a space suggested in the script for adding implementation error model. Here one makes use the switch function of R-base to control the implementation model being run. More on that later. As of now the the hcr_implementation_model_1 is set in the first slot as a dummy.

   (b) The second step is the same as used in the first year, i.e. calculation of the realized mortality multiplier relative to the TAC that was set (done last year) using the hcr_TAC_to_Fmult function.

   (c) The third step, the operating model has also been mentioned, i.e. the hcr_operating_model function

   (d) In the fourth step, often referred to as the observation model one adds an error term to the true observation value. This may be any index one can think of to base decision on, such as mortality signals, survey indices, rate of changes etc. Here we have only set up observation error model for Fmort, spwaning stock biomass and reference biomass, this being called by the function hcr_observation_error. More on this later in the document (documentation ToDo-list and using switch on the programming ToDo-list).

   (e) The decision model actually contains the HCR rule. Here the decision rule on next years TAC is (or whatever other management measure used to put some contraint on fishing mortality) is implemented. Of course based on the values from the observation model.

      i. A space for a rule that is not implemented. This space is left for those that think one can hit the target F each and every year. So awaits some ingenuity ...

      ii. The conventional ICES F-based rules, by function hcr_management_fmort (not implemented yet):

      iii. The conventional Icelandic biomass-based rules, by function hcr_management_bio.

   The code may look something like this:

```
for (h in 1:length(HRATE)) {
  for(y in 2:(ctr$y2 - ctr$y1)) {

    switch(ctr$i_number,
           dummy <- hcr_implementation_model_1(),
           stop("implementation model 2 is available for developement"),
           stop(" and whatever more ..."))

    Fmult <- hcr_TAC_to_Fmult(y, h)
    hcr_operating_model(y, h, ctr, Fmult, nR=1)

    hat <- hcr_observation_error(y,h,Fmult,ctr)

    switch(ctr$h_number,
           stop("Not implemented year"),
            stop("hcr_management_fmort(hat$,hat$ssb,ctr)"),
            hcr_management_bio(y,h,hat$bio,hat$ssb,ctr))

  }
}
```

7. Now update the final year. Here we only do the same as the first year, i.e. calculate the stock in numbers to the end of the final year in the simulation.

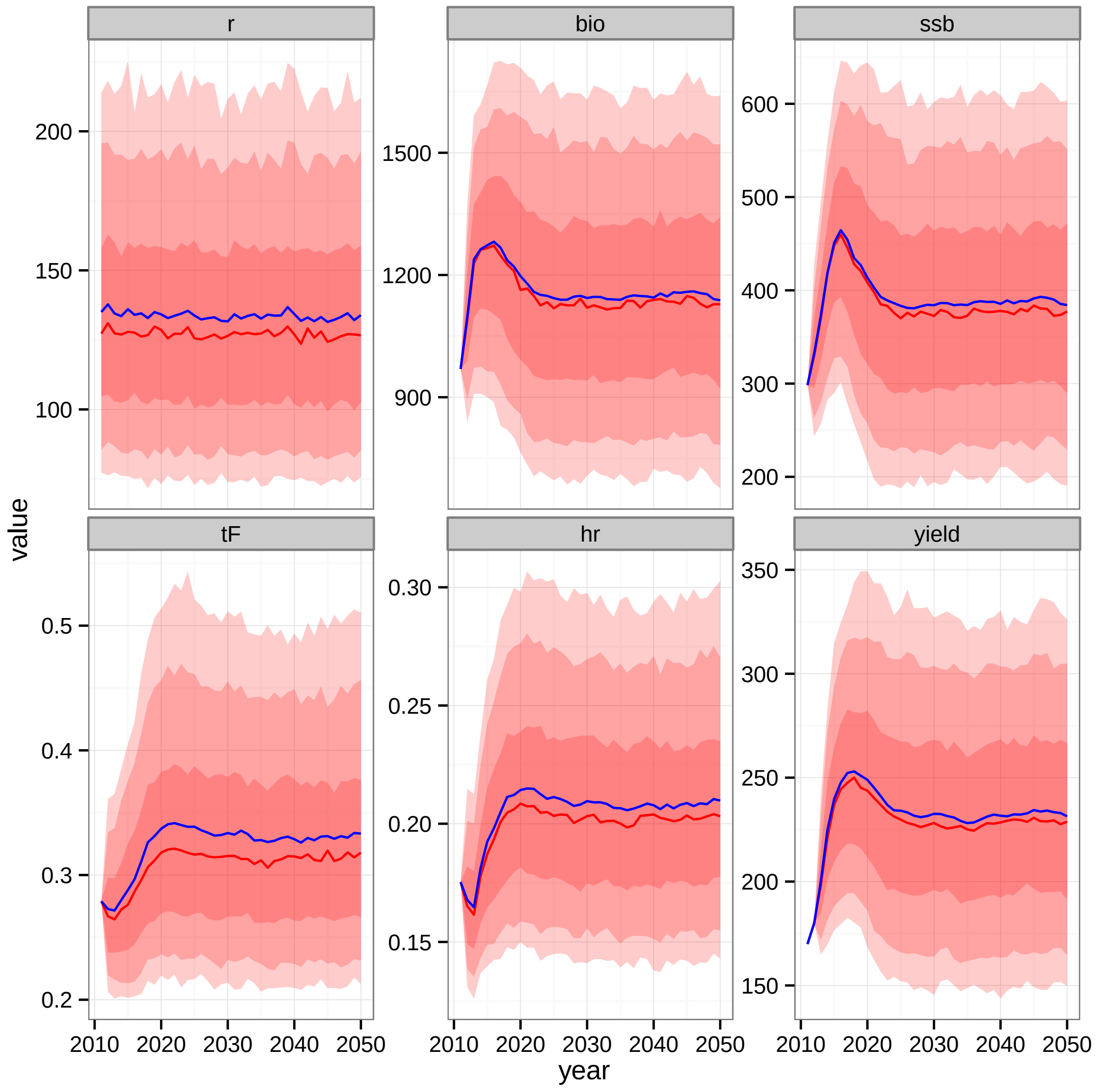
```
y <- ctr$y2 - ctr$y1 + 1
for (h in 1:length(HRATE)) {
    Fmult <- hcr_TAC_to_Fmult(y, h)
    hcr_operating_model(y, h, ctr, Fmult, nR = 1)
}
```

The result data for now resides in an object in the working directory called (for now) X. This is a list containing arrays of various elements. To just obtain estimates of catches one can simply do:

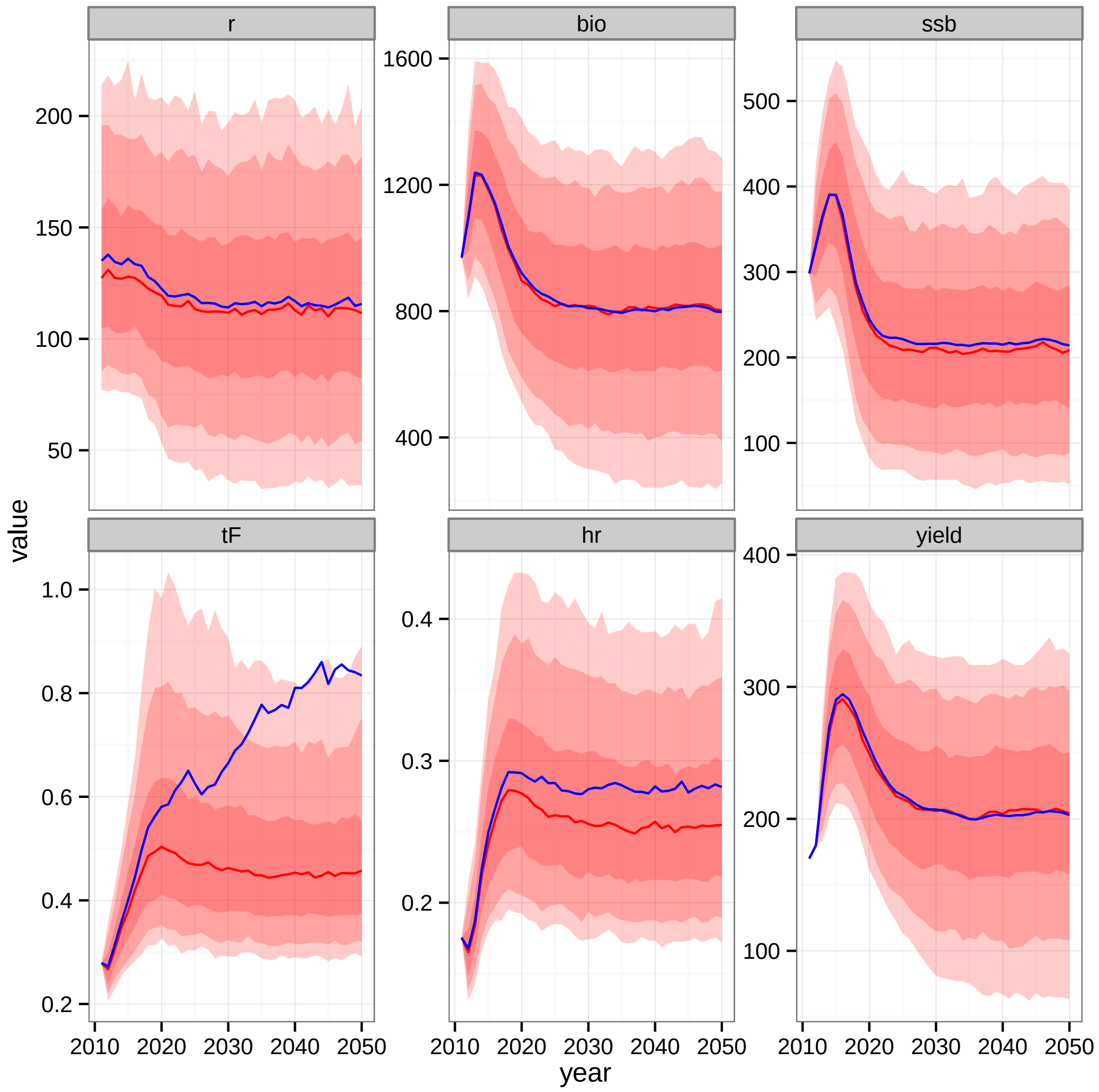
```
dat <- hcr_summarise_data(X)
head(dat)

##   year iter target      r    bio    ssb       tF       hr  yield
## 1 2011    1  1e-05 116.87  969.3  298.3 0.278994 0.175322 169.94
## 2 2012    1  1e-05 160.53 1327.9  411.3 0.214158 0.135496 179.92
## 3 2013    1  1e-05  95.72 1544.1  511.7 0.087403 0.058223  89.90
## 4 2014    1  1e-05  93.22 1809.0  712.5 0.032973 0.024807  44.88
## 5 2015    1  1e-05  58.90 1795.3  822.7 0.015067 0.012459  22.37
## 6 2016    1  1e-05 182.50 2188.7 1084.6 0.005953 0.005064  11.08
```

```
i <- dat$target %in% c(0.2)
d <- melt(dat[i, ], c("year", "iter", "target"))
ggplot(d, aes(year, value)) + stat_summary(fun.data = i90, fill = "red", geom = "ribbon",
    alpha = 0.2) + stat_summary(fun.data = i80, fill = "red", geom = "ribbon",
    alpha = 0.2) + stat_summary(fun.data = i50, fill = "red", geom = "ribbon",
    alpha = 0.2) + stat_summary(fun.y = median, colour = "red", geom = "line") +
    stat_summary(fun.y = mean, colour = "blue", geom = "line") + facet_wrap(~variable,
    scale = "free_y")
```



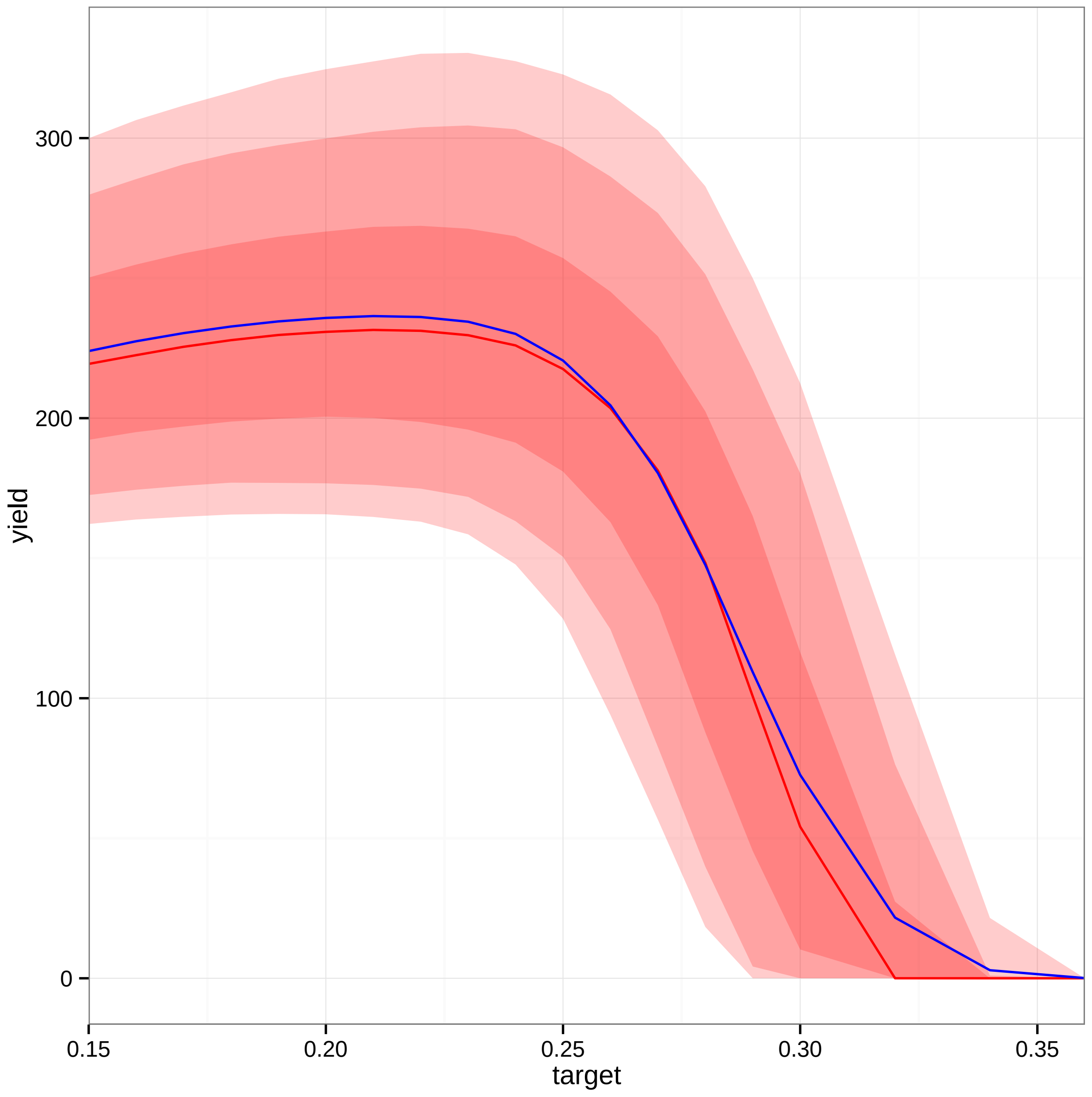
```
i <- dat$target %in% c(0.25)
d <- melt(dat[i, ], c("year", "iter", "target"))
ggplot(d, aes(year, value)) + stat_summary(fun.data = i90, fill = "red", geom = "ribbon",
    alpha = 0.2) + stat_summary(fun.data = i80, fill = "red", geom = "ribbon",
    alpha = 0.2) + stat_summary(fun.data = i50, fill = "red", geom = "ribbon",
    alpha = 0.2) + stat_summary(fun.y = median, colour = "red", geom = "line") +
    stat_summary(fun.y = mean, colour = "blue", geom = "line") + facet_wrap(~variable,
    scale = "free_y")
```



Lets stop for now with this. In the next two pages some plots are provided of output based on the exactly the code above.

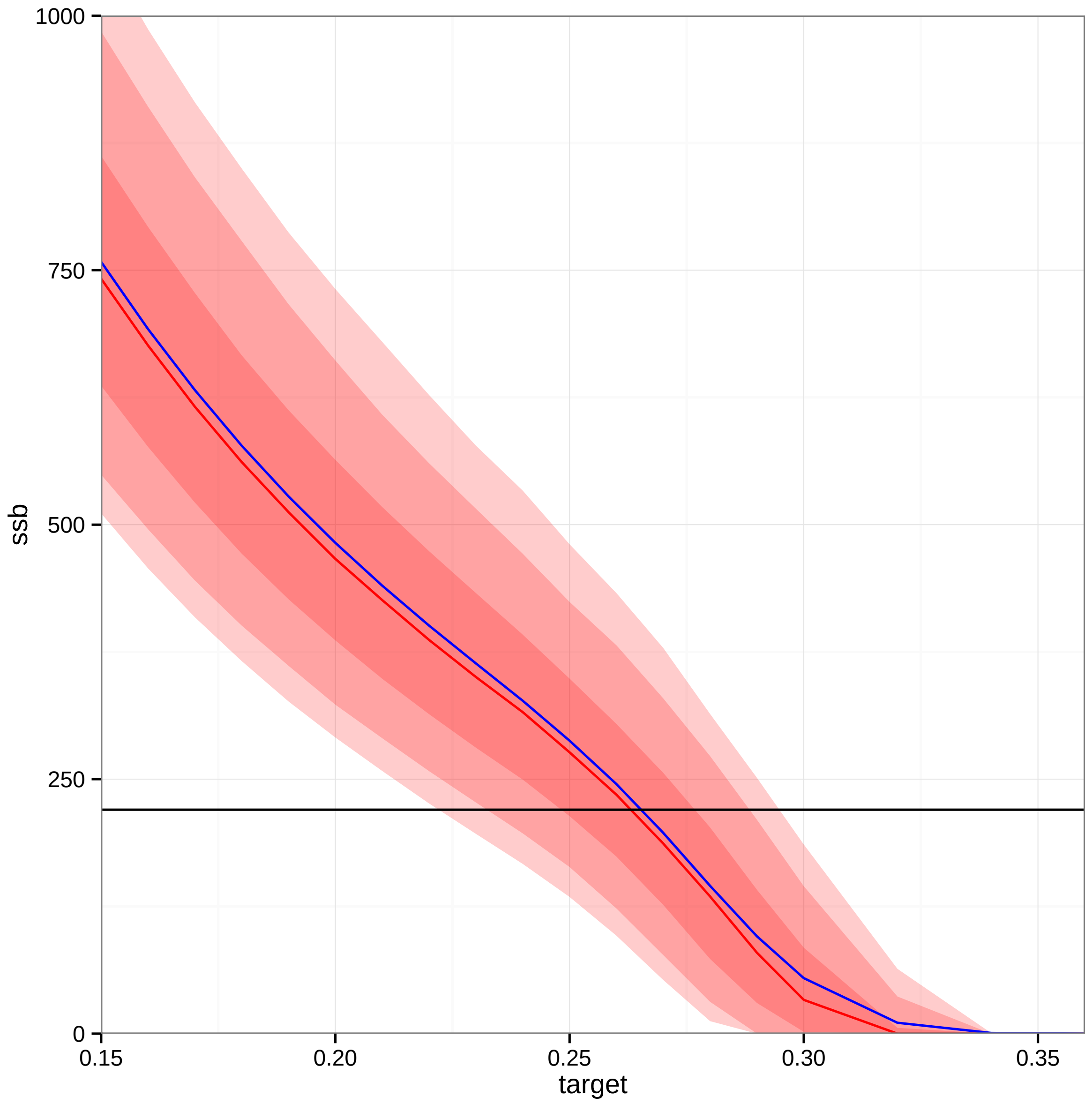
And out of this one can plot the illusive MSY (details provided later) and think about the appropriate fishing mortality to be used in the advisory context:

```r
ggplot(hcr_iCod,aes(target,yield)) +
  stat_summary(fun.data = i90, fill = "red", geom="ribbon", alpha=0.2) +
  stat_summary(fun.data = i80, fill = "red", geom="ribbon", alpha=0.2) +
  stat_summary(fun.data = i50, fill = "red", geom="ribbon", alpha=0.2) +
  stat_summary(fun.y = median, colour = "red", geom="line") +
  stat_summary(fun.y = mean, colour = "blue", geom="line") +
  coord_cartesian(xlim=c(0.15,0.36))
```



If one is concerned with the 5th percentile vs some biomass reference value one has SSB profile:

```
ggplot(hcr_iCod,aes(target,ssb)) +
  stat_summary(fun.data = i90, fill = "red", geom="ribbon", alpha=0.2) +
  stat_summary(fun.data = i80, fill = "red", geom="ribbon", alpha=0.2) +
  stat_summary(fun.data = i50, fill = "red", geom="ribbon", alpha=0.2) +
  stat_summary(fun.y = median, colour = "red", geom="line") +
  stat_summary(fun.y = mean, colour = "blue", geom="line") +
  geom_hline(yintercept = 220) +
  coord_cartesian(xlim=c(0.15,0.36),ylim=c(0,1000))
```



3 On coding error structures

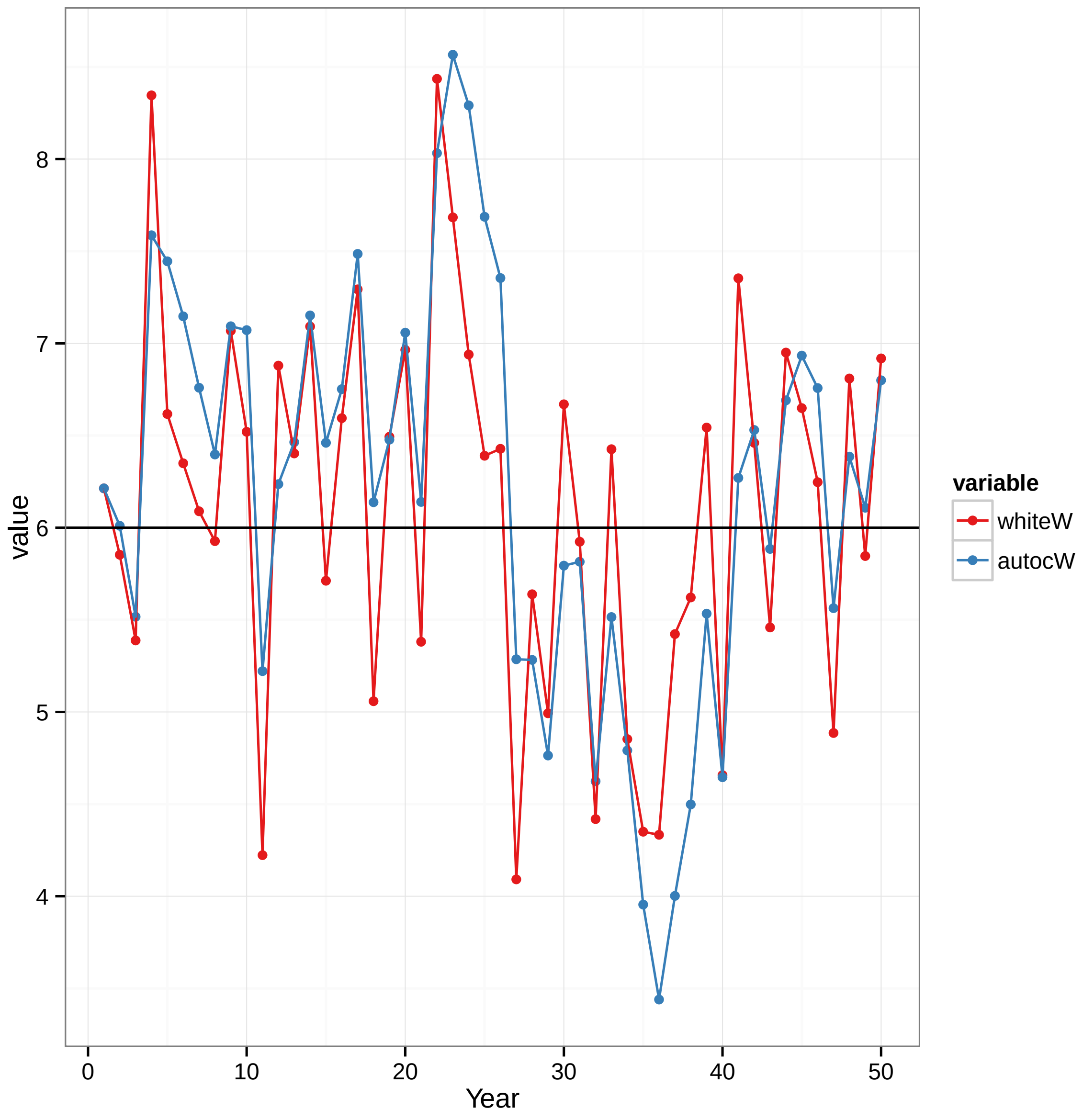
In the current version error structures are only available for recruitement, weights at age and assessment error. For those, the error structure can be either normal og lognormal. All provide a switch option to include autocorrelation, something that one should consider as default for all stocks unless proven to be of pure random nature.

Since no stock dependency is assumed/allowed in the current version all the error structure can be set up in the initialization process. The following example illustrates how the error code is set up in the fishvise code, an example were the

```
n_years <- 50
cv <- 0.15
rho <- 0.6

x1 <- rnorm(n_years)
x2 <- x1
for (y in 2:n_years) {
    x2[y] <- rho * x2[y - 1] + sqrt(1 - rho^2) * x2[y]
}
x1 <- x1 * cv
x2 <- x2 * cv

cW <- rep(6, n_years)
d <- data.frame(Year = c(1:n_years), whiteW = cW * (1 + x1), autocW = cW * (1 +
    x2))
d2 <- melt(d, id.vars = "Year")
ggplot(d2, aes(Year, value, colour = variable)) + geom_point() + geom_line() +
    geom_hline(yintercept = cW) + scale_colour_brewer(palette = "Set1")
```


value
Year
variable
whiteW
autocW