Testing VPA assumptions

Jan Jaap Poos
<janjaap.poos@wur.nl>
&
Ernesto Jardim
<ernesto@ipimar.pt>

9. November 2010

Table of Contents

1 Introduction

In this tutorial we show how to use **FLR** to simulation-test stock assessment methods and the various assumptions that underpin them. The structure of **FLR** was designed to accommodate new assessment methods to be used alongside more common ones. Any new stock assessment should (even must) be tested for robustness to both the assumptions that are fundamental to the method and to the level and structure of observation error within the data themselves - precision and/or bias need to be understood.

In essence we want to be able to use **FLR** as an observation error model: capable of generating population and fishery observations (indices of abundance, tag releases/recaptures etc.) along with the relevant level and structure of observation error/bias that one might expect/observe. The following script demonstrates how one can use **FLR** and **R** to test the main assumptions of tuned VPA methods: constant selectivity and catchability and also shows how to generate observation error in the available tuning indices.

We will work from an example. Before we can start, we first load the relevant libraries and make the North Sea plaice dataset available.

```
> library(FLCore)
> library(FLXSA)
> data(ple4)
```

2 The example

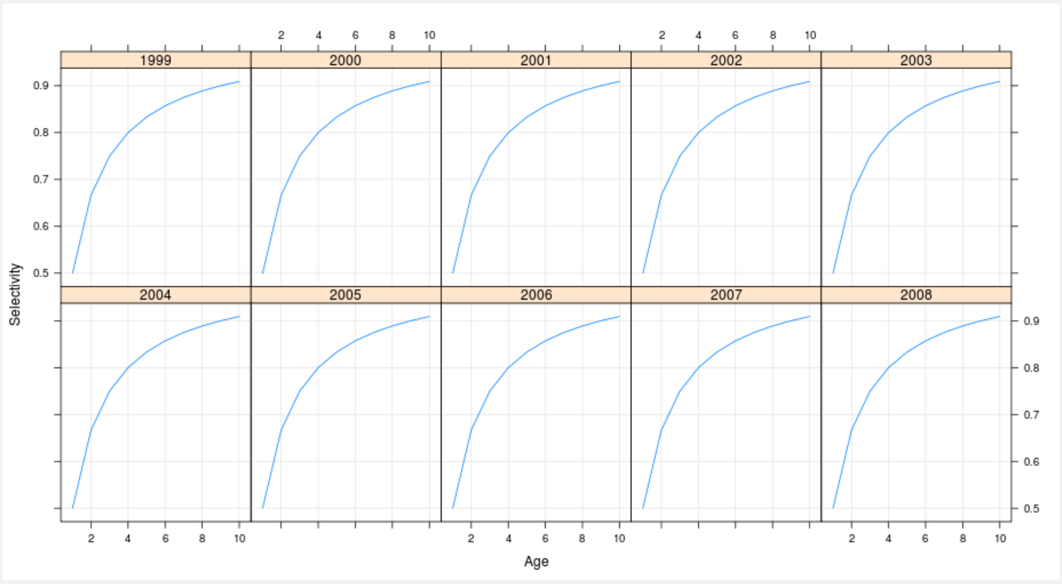
Now that the `ple4` data is available, we can simulate a smaller chunk of data for a survey in summer, using the stock numbers `stock.n(ple4)`, the natural mortality `m(ple4)`, the fishing mortality `harvest(ple4)`.

Next, we will set up the observation error model. In this case we generate data for which the observation error increases with the age of the fish. This is done by increasing the cv of the observations as the fish grow older. The resulting `FLQuant` is called `cv.flq`.

```
> amin <- 1
> amax <- 10
> ymin <- 1999
> ymax <- 2008
> startf <- 0.5
> endf <- 0.6
> tmp.n <- trim(stock.n(ple4) * exp(-mean(c(startf, endf)) * (harvest(ple4) +
+     m(ple4)))), year = ymin:ymax, age = amin:amax)
> cv.min <- 0.25
> cv.max <- 0.35
> cv.age <- seq(cv.min, cv.max, length = length(amin:amax))
> cv.flq <- FLQuant(cv.age, dimnames = dimnames(catch.n(ple4)))
```

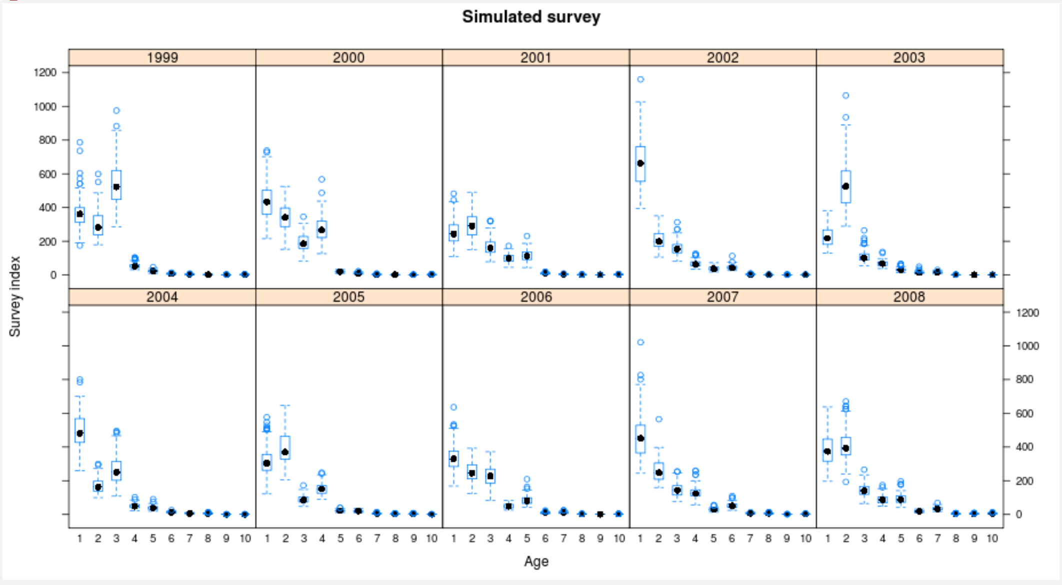
The catchability of the survey is a smooth function of age. Mimicking a typical trawl survey, the catchability increases with size, and thus age. Here, a simple function is used, just for demonstration purposes. Of course more complex functions can be created, for example bell shaped curves, typical for gillnet fisheries, or trawl selection curves where the right hand side slopes down.

```
> sel.flq <- FLQuant((amin:amax)/(1 + (amin:amax)), dimnames = dimnames(tmp.n))
> print(xyplot(data ~ age | as.factor(year), sel.flq, xlab = "Age",
+     ylab = "Selectivity", type = c("g", "l"), layout = c(5, 2),
+     as.table = TRUE))
```



Next, we will put everything together to create 100 surveys. To this end, we use the adjusted stock numbers (in the `tmp.n` object), the selectivity (in the `sel.flq` object), and the catchability (in the `q` object that we'll create).

```
> nrep <- 100
> q <- 0.001
> q.flq <- sel.flq * q
> surv.flq <- rlnorm(nrep, log(q.flq * tmp.n), sqrt(log(1 + cv.flq^2)))
> print(bwplot(data ~ as.factor(age) | as.factor(year), surv.flq,
+     xlab = "Age", ylab = "Survey index", main = "Simulated survey",
+     layout = c(5, 2), as.table = TRUE))
```



For the 100 surveys that were created in the previous step, an equal amount of `FLstock` objects need to be set to store the data later. Each of these stocks will have 100 identical iterations of the stock data. The combined stocks and surveys will be used in a stock assessment method.

```
> ple4Simsurv <- ple4
> stock.n(ple4Simsurv) <- propagate(stock.n(ple4), iter = nrep,
+     fill.iter = F)
> harvest(ple4Simsurv) <- propagate(harvest(ple4), iter = nrep,
+     fill.iter = F)
```

The stock assessment method used here is XSA, that is available through the `FLXSA` library. The settings for the stock assessment are passed on to the assessment by means of and object of the `xsa.control` class. Check the tutorial on tuned VPA methods for more information.

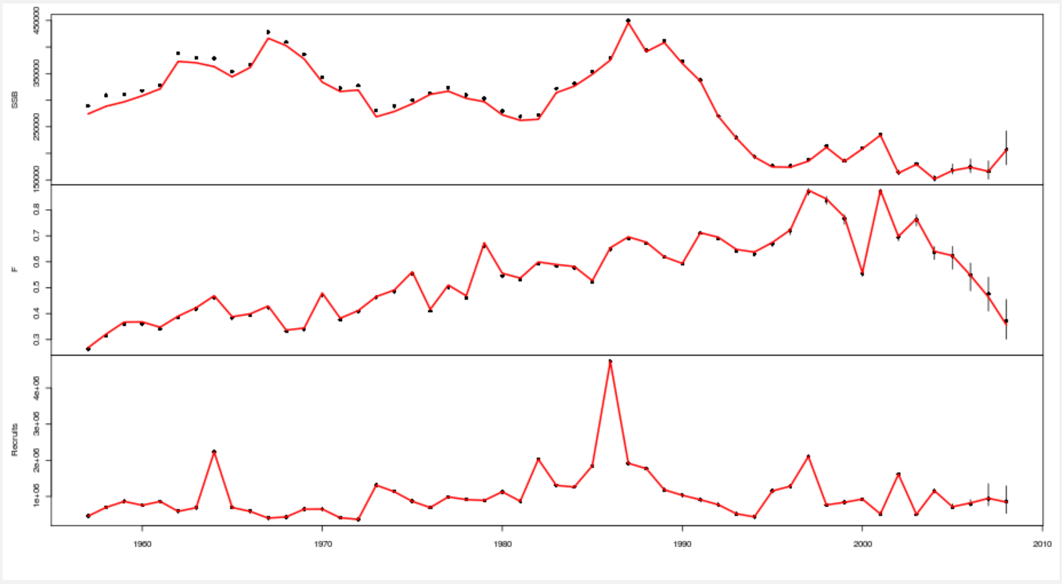
```
> xsaControl <- FLXSA.control(tol = 1e-09, maxit = 60, min.nse = 0.3,
+     fse = 2, rage = -1, qage = 6, shk.n = TRUE, shk.f = TRUE,
+     shk.yrs = 5, shk.ages = 2, window = 100, tsrange = 99, tspower = 0)
```

Finally, the actual assessments are done on the simulated survey data in a loop. The data are stored in the `ple4Simsurv` object.

```
> for (n in 1:nrep) {
+     tmpFLIndex <- FLIndex(index = iter(surv.flq, n))
+     range(tmpFLIndex)[["startf"]] <- startf
+     range(tmpFLIndex)[["endf"]] <- endf
+     name(tmpFLIndex) <- "simsurv"
+     tmpFLIndices <- FLIndices(tmpFLIndex)
+     tmpXSA <- FLXSA(ple4, tmpFLIndices, xsaControl)
+     iter(stock.n(ple4Simsurv), n) <- stock.n(tmpXSA)
+     iter(harvest(ple4Simsurv), n) <- harvest(tmpXSA)
+ }
```

To see how well the assessment with the generated data performs, the results in terms of Spawning Stock Biomass and Fishing mortality can be plotted and contrasted against the actual known values from the stock that was used from the start.

```
> ssbTrue <- ssb(ple4)
> ssbSim <- ssb(ple4Simsurv)
> recTrue <- rec(ple4)
> recSim <- rec(ple4Simsurv)
> fbarTrue <- quantMeans(trim(harvest(ple4), age = 3:6))
> fbarSim <- quantMeans(trim(harvest(ple4Simsurv), age = 3:6))
> par(mfrow = c(3, 1), mar = c(0, 4, 0, 1), oma = c(5, 0.5, 1,
+     0.5))
> yrs <- as.numeric(dimnames(m(ple4))[["year"]])
> qtls <- quantile(ssb(ple4Simsurv), c(0.025, 0.5, 0.975))
> plot(yrs, iter(qtls, 2), ylab = "SSB", pch = 16, axes = FALSE)
> arrows(x0 = yrs, y0 = iter(qtls, 1), x1 = yrs, y1 = iter(qtls,
+     3), length = 0)
> lines(yrs, as.vector(ssb(ple4)), col = "red", lwd = 2, lty = 1)
> axis(2)
> box()
> qtls <- quantile(fbar(ple4Simsurv), c(0.025, 0.5, 0.975))
> plot(yrs, iter(qtls, 2), ylab = "F", pch = 16, axes = FALSE)
> arrows(x0 = yrs, y0 = iter(qtls, 1), x1 = yrs, y1 = iter(qtls,
+     3), length = 0)
> lines(yrs, as.vector(fbar(ple4)), col = "red", lwd = 2, lty = 1)
> axis(2)
> box()
> qtls <- quantile(rec(ple4Simsurv), c(0.025, 0.5, 0.975))
> plot(yrs, iter(qtls, 2), ylab = "Recruits", pch = 16, axes = FALSE)
> arrows(x0 = yrs, y0 = iter(qtls, 1), x1 = yrs, y1 = iter(qtls,
+     3), length = 0)
> lines(yrs, as.vector(rec(ple4)), col = "red", lwd = 2, lty = 1)
> axis(2)
> axis(1)
> box()
```

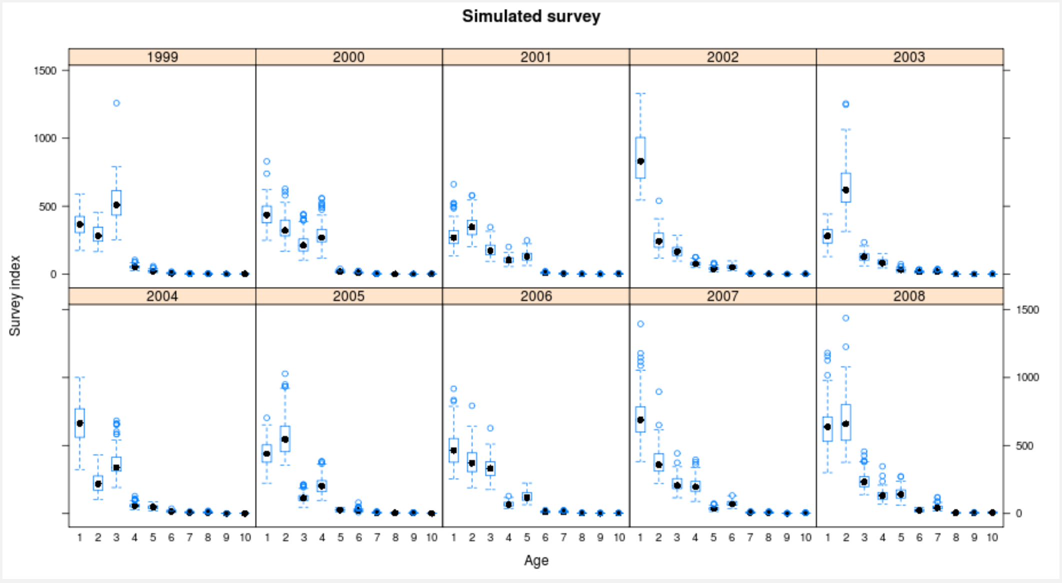


In this case, the stock assessment seems to be able to track the stock developments fairly well. As you can see the uncertainty in the assessment increases in the most recent period. The reason for this is that the stock assessment method is a VPA-like in the historic part, and thus fully determined by the catch-at-age matrix, which is assumed to be without error.

3 Violating the constant catchability assumption

It is quite quite straightforward to violate some of the assumptions in the stock assessment and see if the assessment is still able to follow the “true” trends. One example is to violate the constant catchability assumption (that many assessments make. To this end, we recreate the `surv.flq` object, but we add an increasing component as a function of the years.

```
> nrep <- 100
> q <- 0.001
> q.flq <- sel.flq * q
> surv.flq <- rlnorm(nrep, sweep(log(q.flq * tmp.n), 2, seq(1,
+     1.09, 0.01), "*"), sqrt(log(1 + cv.flq^2)))
> print(bwplot(data ~ as.factor(age) | as.factor(year), surv.flq,
+     xlab = "Age", ylab = "Survey index", main = "Simulated survey",
+     layout = c(5, 2), as.table = TRUE))
```

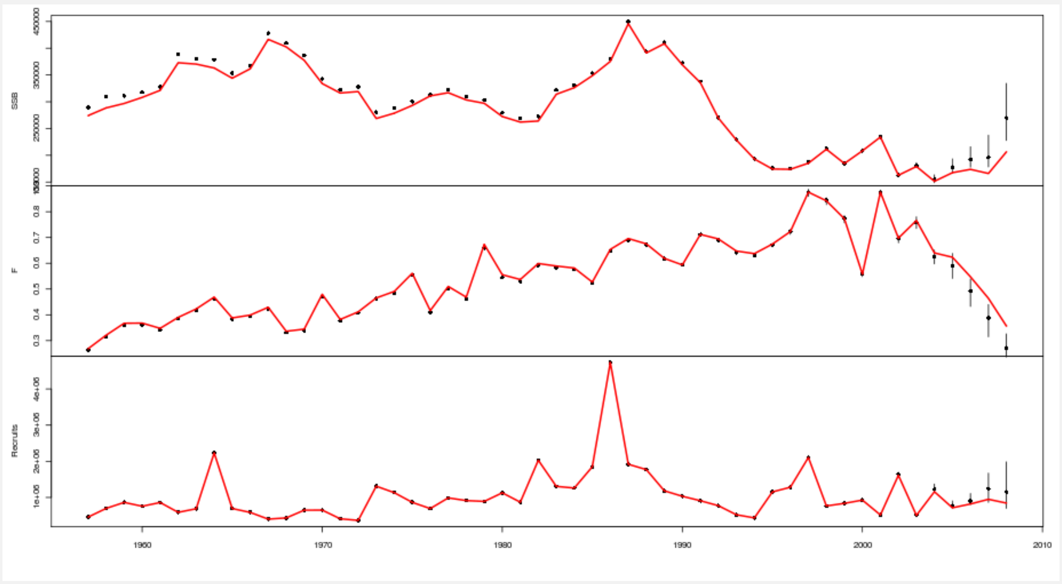


As you can see, `sweep()` was used to multiply a vector with the matrix containing the survey estimates. This vector increases the survey estimates over time (independent of the stock trends).

```
> for (n in 1:nrep) {
+     tmpFLIndex <- FLIndex(index = iter(surv.flq, n))
+     range(tmpFLIndex)[["startf"]] <- startf
+     range(tmpFLIndex)[["endf"]] <- endf
+     name(tmpFLIndex) <- "simsurv"
+     tmpFLIndices <- FLIndices(tmpFLIndex)
+     tmpXSA <- FLXSA(ple4, tmpFLIndices, xsaControl)
+     iter(stock.n(ple4Simsurv), n) <- stock.n(tmpXSA)
+     iter(harvest(ple4Simsurv), n) <- harvest(tmpXSA)
+ }
```

Clearly, this violation of the constant catchability assumption throws off the assumption, especially because there is no other source of information in the assessment that informs about the most recent stock trends. What is striking is that the overestimation of SSB in the final year is much larger than the 10% that the survey was increased.

```
> ssbSim <- ssb(ple4Simsurv)
> recSim <- rec(ple4Simsurv)
> fbarSim <- quantMeans(trim(harvest(ple4Simsurv), age = 3:6))
> par(mfrow = c(3, 1), mar = c(0, 4, 0, 1), oma = c(5, 0.5, 1,
+     0.5))
> yrs <- as.numeric(dimnames(m(ple4))[["year"]])
> qtls <- quantile(ssb(ple4Simsurv), c(0.025, 0.5, 0.975))
> plot(yrs, iter(qtls, 2), ylab = "SSB", pch = 16, axes = FALSE)
> arrows(x0 = yrs, y0 = iter(qtls, 1), x1 = yrs, y1 = iter(qtls,
+     3), length = 0)
> lines(yrs, as.vector(ssb(ple4)), col = "red", lwd = 2, lty = 1)
> axis(2)
> box()
> qtls <- quantile(fbar(ple4Simsurv), c(0.025, 0.5, 0.975))
> plot(yrs, iter(qtls, 2), ylab = "F", pch = 16, axes = FALSE)
> arrows(x0 = yrs, y0 = iter(qtls, 1), x1 = yrs, y1 = iter(qtls,
+     3), length = 0)
> lines(yrs, as.vector(fbar(ple4)), col = "red", lwd = 2, lty = 1)
> axis(2)
> box()
> qtls <- quantile(rec(ple4Simsurv), c(0.025, 0.5, 0.975))
> plot(yrs, iter(qtls, 2), ylab = "Recruits", pch = 16, axes = FALSE)
> arrows(x0 = yrs, y0 = iter(qtls, 1), x1 = yrs, y1 = iter(qtls,
+     3), length = 0)
> lines(yrs, as.vector(rec(ple4)), col = "red", lwd = 2, lty = 1)
> axis(2)
> axis(1)
> box()
```



4 Final thoughts

Testing the effects of violating assessment assumptions and looking at the results may not be the best way of moving stock assessment methodology forward. A more constructive way forward is to create stock assessments that can relax some of the typical assessment assumptions, by making creative methods to fit models to data.