Stock assessment and management advice with a4a methods DRAFT

Ernesto Jardim[1], Colin Millar[1], and Finlay Scott[1]

[1]European Commission, Joint Research Centre, IPSC / Maritime Affairs Unit, 21027 Ispra (VA), Italy
*Corresponding author ernesto.jardim@jrc.ec.europa.eu

April 23, 2014

Contents

1 Introduction

- Objectives
- a4a concepts
  - life history considers parameters to have distributions, it's a kind of Bayesian posteriors informed estimates, but if one runs a Bayesian analysis to estimate growth parameters the posteriors can be used
- Workflow diagram

```
# ==============================================================================
# Libraries and constants
# ==============================================================================
library(FLa4a)
library(XML)
library(reshape2)
data(rfLen)
data(ple4)
data(ple4.indices)
```

```
# ==============================================================================
# Some functions for transforming the data
# ==============================================================================

# quant 2 quant
qt2qt <- function(object, id = 5, split = "-") {
    qt <- object[, id]
    levels(qt) <- unlist(lapply(strsplit(levels(qt), split = split), "[[", 2))
    as.numeric(as.character(qt))
}

# check import and massage
cim <- function(object, n, wt, hrv = "missing") {
    v <- object[sample(1:nrow(object), 1), ]
    c1 <- c(n[as.character(v$V5), as.character(v$V1), 1, as.character(v$V2)] ==
        v$V6)
    c2 <- c(wt[as.character(v$V5), as.character(v$V1), 1, as.character(v$V2)] ==
        v$V7)
    if (missing(hrv)) {
        c1 + c2 == 2
    } else {
        c3 <- c(hrv[as.character(v$V5), as.character(v$V1), 1, as.character(v$V2)] ==
            v$V8)
        c1 + c2 + c3 == 3
    }
}
```

2 Reading files and building FLR objects

For this document we'll use the plaice in ICES area IV dataset, provided by FLR, and a length-based simulated dataset based on red fish, using Gadget (http://www.hafro.is/gadget), provided by Daniel Howell (Institute of Marine Research, Norway).

In this section we read in the Gadget data files, and transform them into FLR objects.

First we read in the files as data frames.

```r
# ==============================================================================
# Read files
# ==============================================================================

# catch
cth.orig <- read.table("data/catch.len", skip = 5)

# stock
stk.orig <- read.table("data/red.len", skip = 4)

# surveys
idx.orig <- read.table("data/survey.len", skip = 5)
idxJmp.orig <- read.table("data/jump.survey.len", skip = 5)
idxTrd.orig <- read.table("data/tend.survey.len", skip = 5)

# ==============================================================================
# Recode the length categories into something usable
# ==============================================================================

# catch
cth.orig[, 5] <- qt2qt(cth.orig)

# stock
stk.orig[, 5] <- qt2qt(stk.orig)

# surveys
idx.orig[, 5] <- qt2qt(idx.orig)
idxJmp.orig[, 5] <- qt2qt(idxJmp.orig)
idxTrd.orig[, 5] <- qt2qt(idxTrd.orig)
```

Then we reshape the data frames into six dimensional arrays.

```r
# ==============================================================================
# Cast the data into arrays
# ==============================================================================

# catch
cth.n <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V6", data = cth.orig)
cth.wt <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V7", data = cth.orig)
hrv <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V8", data = cth.orig)

# stock
stk.n <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V6", data = stk.orig)
stk.wt <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V7", data = stk.orig)

# surveys
idx.n <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V6", data = idx.orig)
idx.wt <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V7", data = idx.orig)
idx.hrv <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V8", data = idx.orig)

idxJmp.n <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V6", data = idxJmp.orig)
idxJmp.wt <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V7", data = idxJmp.orig)
idxJmp.hrv <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V8", data = idxJmp.orig)

idxTrd.n <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V6", data = idxTrd.orig)
idxTrd.wt <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V7", data = idxTrd.orig)
idxTrd.hrv <- acast(V5 ~ V1 ~ 1 ~ V2 ~ 1 ~ 1, value.var = "V8", data = idxTrd.orig)
```

We take the arrays and make *FLQuant* objects from them.

```
# ==============================================================================
# Make FLQuant objects
# ==============================================================================

# catch
dnms <- dimnames(cth.n)
names(dnms) <- names(dimnames(FLQuant()))
names(dnms)[1] <- "len"
cth.n <- FLQuant(cth.n, dimnames = dnms)
cth.wt <- FLQuant(cth.wt, dimnames = dnms)
hrv <- FLQuant(hrv, dimnames = dnms)
units(hrv) <- "f"

# stock
dnms <- dimnames(stk.n)
names(dnms) <- names(dimnames(FLQuant()))
names(dnms)[1] <- "len"
stk.n <- FLQuant(stk.n, dimnames = dnms)
stk.wt <- FLQuant(stk.wt, dimnames = dnms)

# stock
dnms <- dimnames(idx.n)
names(dnms) <- names(dimnames(FLQuant()))
names(dnms)[1] <- "len"
idx.n <- FLQuant(idx.n, dimnames = dnms)
idx.wt <- FLQuant(idx.wt, dimnames = dnms)
idx.hrv <- FLQuant(idx.hrv, dimnames = dnms)

dnms <- dimnames(idxJmp.n)
names(dnms) <- names(dimnames(FLQuant()))
names(dnms)[1] <- "len"
idxJmp.n <- FLQuant(idxJmp.n, dimnames = dnms)
idxJmp.wt <- FLQuant(idxJmp.wt, dimnames = dnms)
idxJmp.hrv <- FLQuant(idxJmp.hrv, dimnames = dnms)

dnms <- dimnames(idxTrd.n)
names(dnms) <- names(dimnames(FLQuant()))
names(dnms)[1] <- "len"
idxTrd.n <- FLQuant(idxTrd.n, dimnames = dnms)
idxTrd.wt <- FLQuant(idxTrd.wt, dimnames = dnms)
idxTrd.hrv <- FLQuant(idxTrd.hrv, dimnames = dnms)
```

Some sanity checks to check that the resulting objects have matching dimensions.

```
# ==============================================================================
# Check dims match
# ==============================================================================

# catch
cim(cth.orig, cth.n, cth.wt, hrv)

## [1] TRUE

# stock
cim(stk.orig, stk.n, stk.wt)
```

```
## [1] TRUE

# surveys
cim(idx.orig, idx.n, idx.wt, idx.hrv)

## [1] TRUE

cim(idxJmp.orig, idxJmp.n, idxJmp.wt, idxJmp.hrv)

## [1] TRUE

cim(idxTrd.orig, idxTrd.n, idxTrd.wt, idxTrd.hrv)

## [1] TRUE
```

Finally, we make FLR objects from the data.

```
# ==============================================================================
# FLR objects
# ==============================================================================

rfLen.stk <- FLStockLen(stock.n = stk.n, stock.wt = stk.wt, stock = quantSums(stk.wt *
    stk.n), catch.n = cth.n, catch.wt = cth.wt/cth.n, catch = quantSums(cth.wt),
    harvest = hrv)
m(rfLen.stk)[] <- 0.05
mat(rfLen.stk)[] <- m.spwn(rfLen.stk)[] <- harvest.spwn(rfLen.stk)[] <- 0
mat(rfLen.stk)[38:59, , , 3:4] <- 1

rfTrawl.idx <- FLIndex(index = idx.n, catch.n = idx.n, catch.wt = idx.wt, sel.pattern = idx.hrv)
effort(rfTrawl.idx)[] <- 100

rfTrawlJmp.idx <- FLIndex(index = idxJmp.n, catch.n = idxJmp.n, catch.wt = idxJmp.wt,
    sel.pattern = idxJmp.hrv)
effort(rfTrawlJmp.idx)[] <- 100

rfTrawlTrd.idx <- FLIndex(index = idxTrd.n, catch.n = idxTrd.n, catch.wt = idxTrd.wt,
    sel.pattern = idxTrd.hrv)
effort(rfTrawlTrd.idx)[] <- 100

# ==============================================================================
# Save the objects for future use
# ==============================================================================

save(rfLen.stk, rfTrawl.idx, rfTrawlJmp.idx, rfTrawlTrd.idx, file = "data/rfLen.rdata")
```

3 Converting length data to age

The stock assessment framework is based on age dynamics. To use length information it must be pre-processed before used for assessment. The rationale is that the pre-processing should give the analyst the flexibility to use a range of sources of information, *e.g.* literature or online databases, to grab information about the species growth and the uncertainty about the model parameters.

Within the a4a framework this is handled using the *a4aGr* class. In this section we introduce the *a4aGr* class and look at the variety of ways that parameter uncertainty can be included.

3.1 a4aGr - The growth class

The convertion of length data to age is performed through the use of a growth model. The implementation is done through the *a4aGr* class.

```
showClass("a4aGr")

## Class "a4aGr" [package "FLa4a"]
##
## Slots:
##
## Name:      grMod  grInvMod    params      vcov     distr      name
## Class:   formula   formula    FLPar     array character character
##
## Name:       desc     range
## Class: character   numeric
##
## Extends: "FLComp"
```

A simple construction of *a4aGr* objects requires the model and parameters to be provided. Check the help file for more information.

Here we show an example using the von Bertalanffy growth model. It is possible to use other growth models.

```
# ==============================================================================
# An example using the von Bertalanffy growth model
# ==============================================================================

# Create the a4aGr object by passing in: the model (length ~ time) the
# inverse of the model (time ~ length) the parameters
vbObj <- a4aGr(grMod = ~linf * (1 - exp(-k * (t - t0))), grInvMod = ~t0 - 1/k *
    log(1 - len/linf), params = FLPar(linf = 58.5, k = 0.086, t0 = 0.001, units = c("cm",
    "ano-1", "ano")))
```

The predict method allows the transformation between age and lengths.

```
# ==============================================================================
# Predicting ages from lengths and vice-versa
# ==============================================================================

# First we check the model and its inverse
lc = 20
predict(vbObj, len = lc)

##    iter
##         1
##   1 4.866

predict(vbObj, t = predict(vbObj, len = lc)) == lc

##    iter
##        1
##   1 TRUE

# Example of converting a vector of lengths or ages
predict(vbObj, len = 5:10 + 0.5)
```

```
##    iter
##         1
##   1 1.149
##   2 1.371
##   3 1.596
##   4 1.827
##   5 2.062
##   6 2.301
```

```
predict(vbObj, t = 5:10 + 0.5)
```

```
##    iter
##         1
##   1 22.04
##   2 25.05
##   3 27.80
##   4 30.33
##   5 32.66
##   6 34.78
```

3.2 Adding parameter uncertainty with a multivariate normal distribution

Uncertainty is introduced through the inclusion of parameter uncertainty. There are several ways of doing this. The simplest method is to assume that the parameters are represented using a multivariate normal distribution. The implementation for *a4aGr* makes use of the `vcov` slot to set the parameter's covariance matrix.

Here we set the variance-covariance matrix with some made up numbers.

```
# ==============================================================================
# Setting the variance-covariance matrix
# ==============================================================================

# Make an empty vcov matrix
mm <- matrix(NA, ncol = 3, nrow = 3)
# Fill it up with made up values
diag(mm) <- c(100, 0.001, 0.001)
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.1, 0.1, 3e-04)
# Create the a4aGr object as before but now we also include arguments for
# multivariate normal uncertainty
vbObj <- a4aGr(grMod = ~linf * (1 - exp(-k * (t - t0))), grInvMod = ~t0 - 1/k *
    log(1 - len/linf), params = FLPar(linf = 58.5, k = 0.086, t0 = 0.001, units = c("cm",
    "ano-1", "ano")), vcov = mm)


# ==============================================================================
# Simulating parameters using the variance-covariance matrix
# ==============================================================================

# Note that the object we have just created has a single iteration of each
# parameter
vbObj@params
```

```
## An object of class "FLPar"
## params
##   linf      k     t0
```

```
## 58.500  0.086  0.001
## units:  cm ano-1 ano

dim(vbObj@params)

## [1] 3 1

# We simulate from the a4aGr object by calling mvrnorm(). Here we create 100
# iterations.
vbNorm <- mvrnorm(10000, vbObj)
# Now we have 100 iterations of each parameter, randomly sampled from the
# multivariate normal distribution
vbNorm@params

## An object of class "FLPar"
## iters:  10000
##
## params
##               linf                 k                t0
## 58.6305417(9.9006)  0.0861432(0.0321)  0.0011171(0.0313)
## units:  cm ano-1 ano

dim(vbNorm@params)

## [1]     3 10000

# We can now convert from length to ages data based on the 100 parameter
# iterations.  This gives us 100 sets of ages data.  For example, here we
# convert a single length vector:
ages <- predict(vbNorm, len = 5:10 + 0.5)
dim(ages)

## [1]     6 10000

# We show the first ten only as an illustration
ages[, 1:10]

##    iter
##          1     2      3     4     5     6     7      8     9    10
##   1 0.6329 3.551 0.6544 1.827 1.208 1.137 1.015 0.9791 0.843 1.001
##   2 0.7499 4.253 0.7789 2.177 1.446 1.359 1.205 1.1610 1.009 1.201
##   3 0.8687 4.973 0.9058 2.533 1.688 1.584 1.397 1.3456 1.177 1.404
##   4 0.9896 5.712 1.0352 2.896 1.937 1.814 1.592 1.5331 1.348 1.612
##   5 1.1125 6.469 1.1673 3.266 2.191 2.048 1.790 1.7237 1.523 1.824
##   6 1.2375 7.246 1.3021 3.643 2.452 2.286 1.991 1.9172 1.701 2.041
```

The marginal distributions can be seen in Figure 1.

The shape of the correlation can be seen in Figure 2.

Growth curves for the 100 iterations can be seen in Figure 3.

3.3 Adding parameter uncertainty with a multivariate triangle distribution

One alternative to using normal distributions, particularly if you don't believe in asymptotic theory, is to use triangle distributions (http://en.wikipedia.org/wiki/Triangle_distribution). These distri-

```r
par(mfrow = c(3, 1))
hist(c(params(vbNorm)["linf", ]), main = "linf", xlab = "")
hist(c(params(vbNorm)["k", ]), main = "k", prob = TRUE, xlab = "")
# lines(x. <- seq(min(k), max(k), len=100), dnorm(x., mean(k), sd(k)))
hist(c(params(vbNorm)["t0", ]), main = "t0", xlab = "")
```

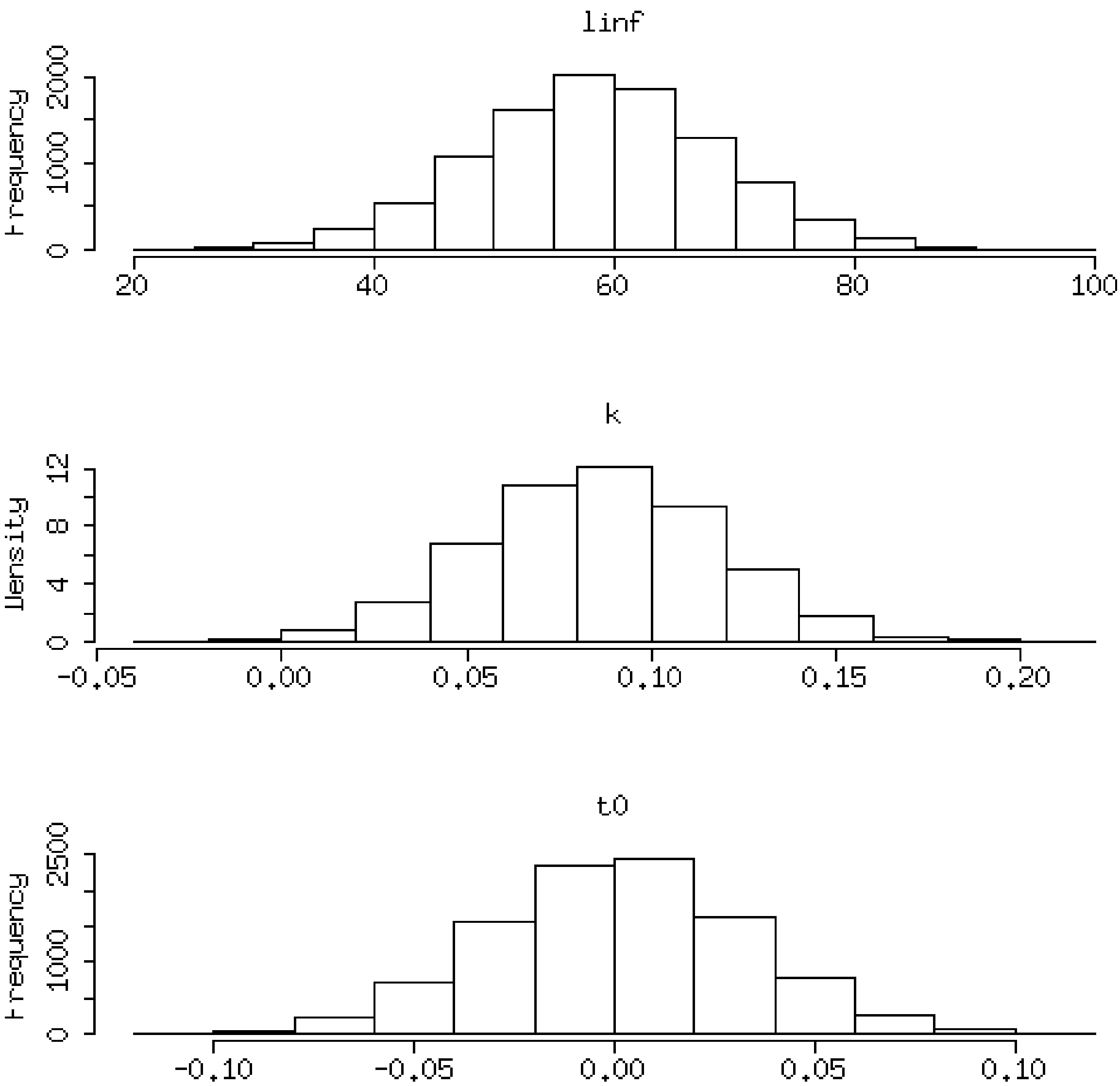


Figure 1: The marginal distributions of each of the parameters from using a multivariate normal distribution.

```r
splom(data.frame(t(params(vbNorm)@.Data)), pch = ".")
```

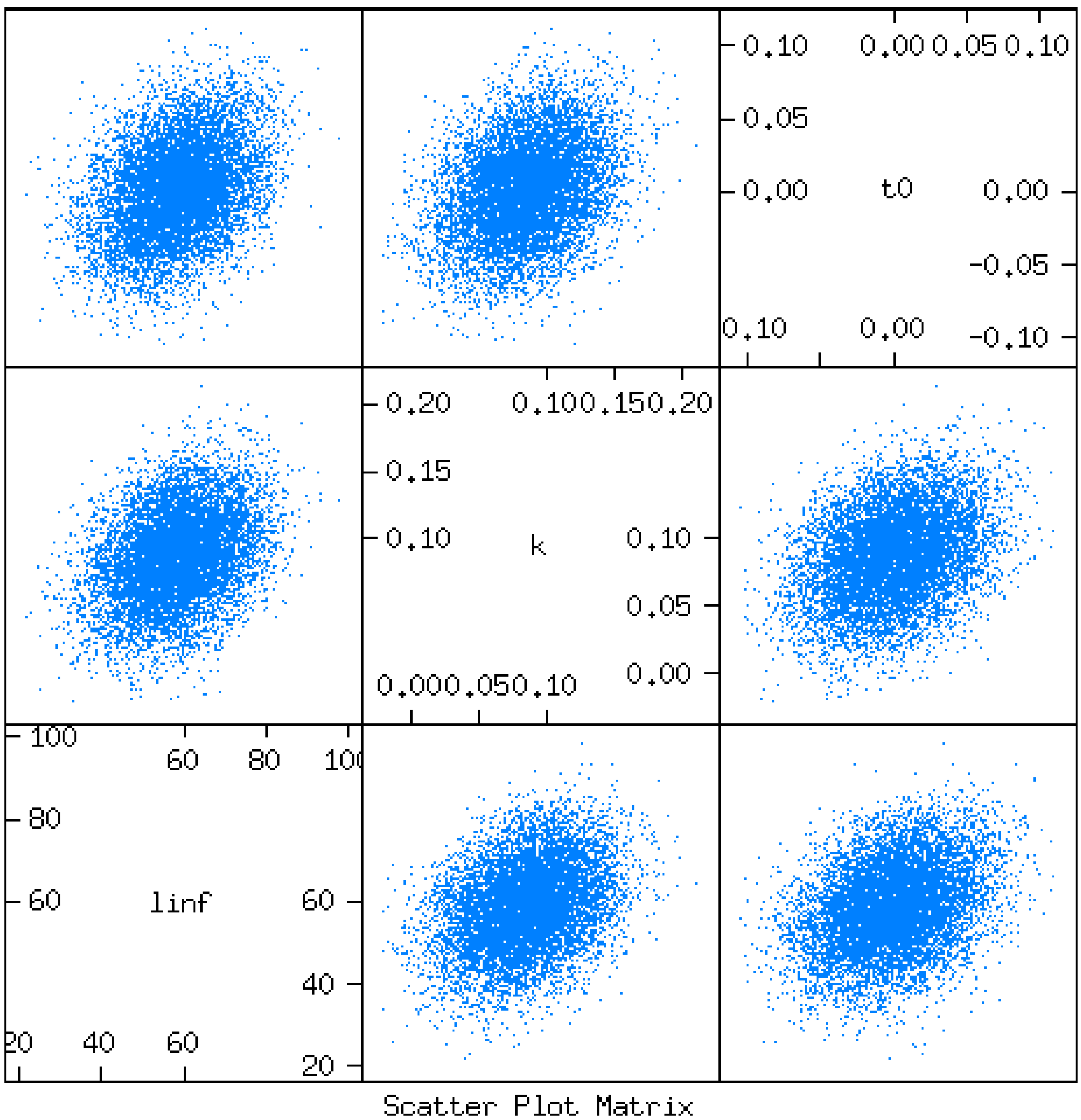


Figure 2: Scatter plot of the 10000 samples parameter from the multivariate normal distribution.

```
# Generating and plotting growth curves
boxplot(t(predict(vbNorm, t = 0:50 + 0.5)))
```

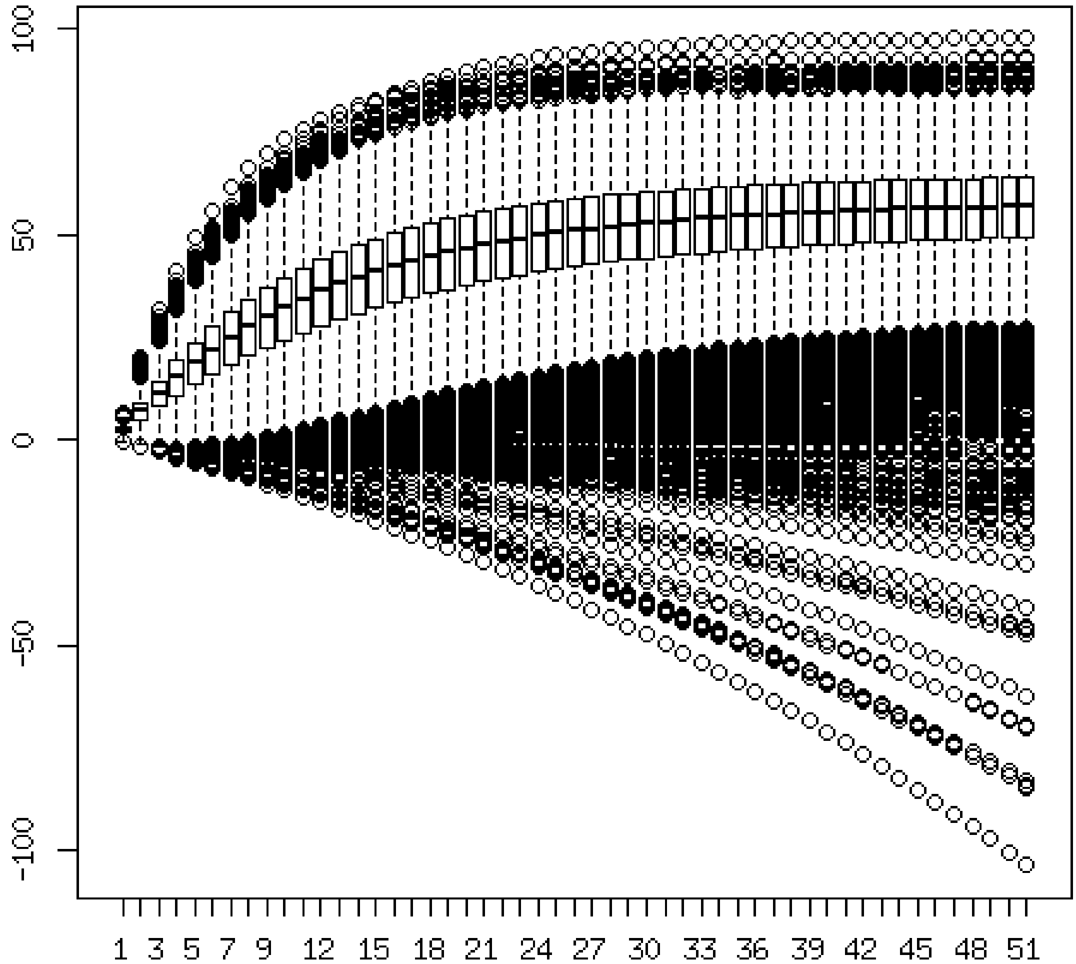


Figure 3: Growth curves using parameters simulated from a multivariate normal distribution.

butions are parametrized using the minimum, maximum and median values. This can be very attractive if the analyst needs to scrape information from the web or literature and perform some kind of meta-analysis.

```
# ==============================================================================
# Example of setting a triangle distribution with values taken from FishBase
# ==============================================================================

# The web address for the growth parameters for redfish (Sebastes
# norvegicus)
addr <- "http://www.fishbase.org/PopDyn/PopGrowthList.php?ID=501"
# Scrape the data
tab <- try(readHTMLTable(addr))
# Interrogate the data table and get vectors of the values
linf <- as.numeric(as.character(tab$dataTable[, 2]))
k <- as.numeric(as.character(tab$dataTable[, 4]))
t0 <- as.numeric(as.character(tab$dataTable[, 5]))
# Set the min, max and median values for each parameters as a list of lists
# Note that t0 has no 'c' (max) value. This makes the distribution
# symmetrical
triPars <- list(list(a = min(linf), b = max(linf), c = median(linf)), list(a = min(k),
    b = max(k), c = median(k)), list(a = median(t0, na.rm = T) - IQR(t0, na.rm = T)/2,
    b = median(t0, na.rm = T) + IQR(t0, na.rm = T)/2))
# Simulate 10000 times using mvrtriangle
vbTri <- mvrtriangle(10000, vbObj, paramMargins = triPars)
```

The marginals will reflect the uncertainty on the parameter values that were scraped from FishBase but, as we don't really believe the parameters are multivariate normal we adopted a more relaxed distribution based on a t copula with triangle marginals. The marginal distributions can be seen in Figure 4.

The shape of the correlation can be seen in Figure 5.

Off course we can still use `predict()` to get see the growth model uncertainty (Figure 6).

Remember that the above examples use a variance-covariance matrix that we made up. If you want to be really geeky, you can scrape the entire growth parameters dataset from FishBase and compute the shape of the variance covariance matrix yourself.

3.4 Adding parameter uncertainty with other copulas

A more general approach to adding parameter uncertainty is to make use of whatever copula and marginal distribution you want. This is possible with `mvrcop()` function. The example below keeps the same parameters and changes only the copula type and family but a lot more can be done. Check the package *copula* for more.

```
vbCop <- mvrcop(10000, vbObj, copula = "archmCopula", family = "clayton", param = 2,
    margins = "triangle", paramMargins = triPars)
```

The shape of the correlation changes (Figure 7).

As well as the resulting growth curves (Figure 8).

3.5 The "l2a" method

After introducing uncertainty on the growth model through the parameters it's time to transform the length-based dataset into an age-based dataset. The method that deals with this process is `l2a()`. The implementation of this method for the *FLQuant* class is the main workhorse. There's two other

```r
par(mfrow = c(3, 1))
hist(c(params(vbTri)["linf", ]), main = "linf", xlab = "")
hist(c(params(vbTri)["k", ]), main = "k", prob = TRUE, xlab = "")
# lines(x. <- seq(min(k), max(k), len=100), dnorm(x., mean(k), sd(k)))
hist(c(params(vbTri)["t0", ]), main = "t0", xlab = "")
```

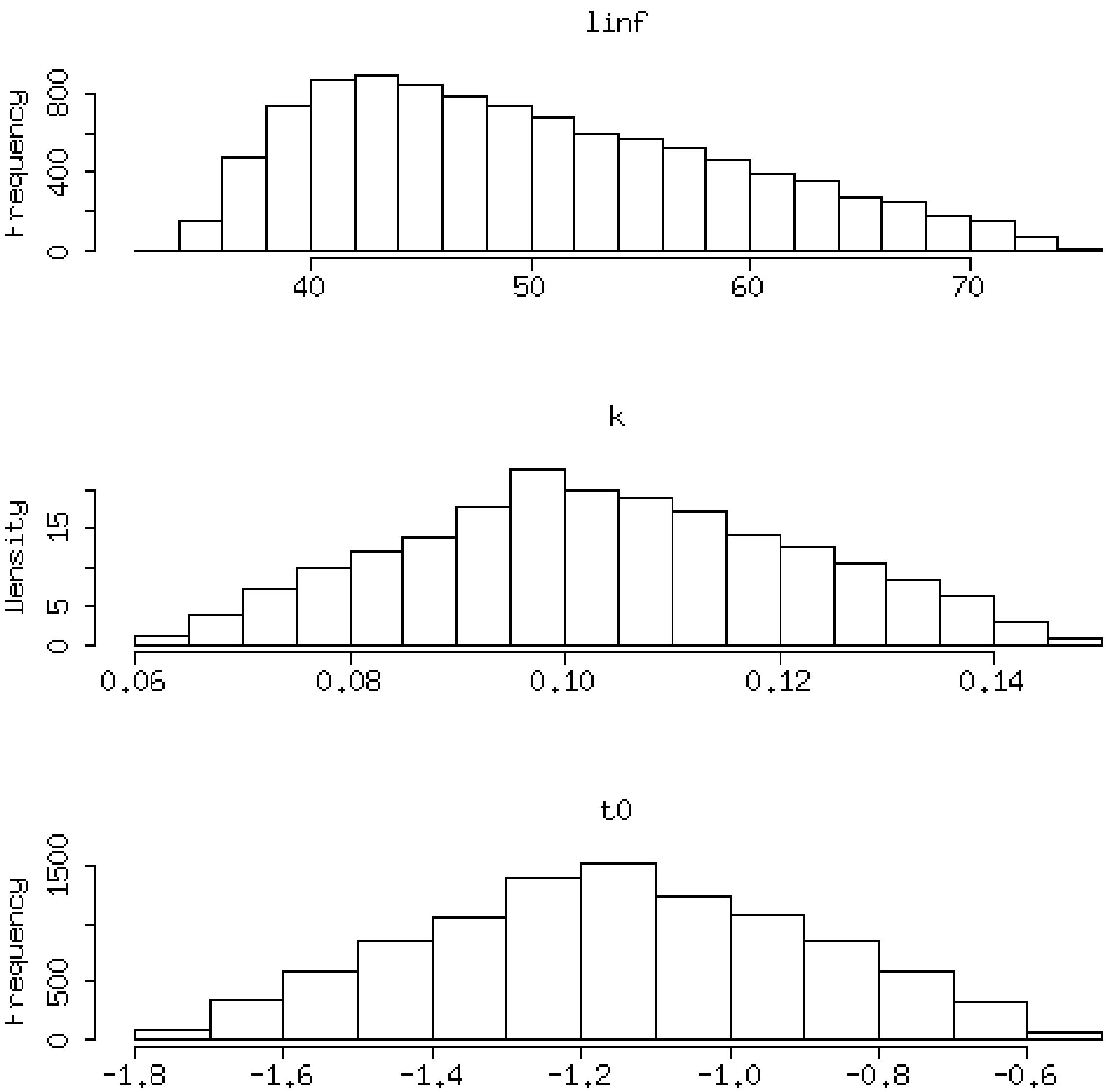


Figure 4: The marginal distributions of each of the parameters from using a multivariate triangle distribution.

```r
splom(data.frame(t(params(vbTri)@.Data)), pch = ".")
```

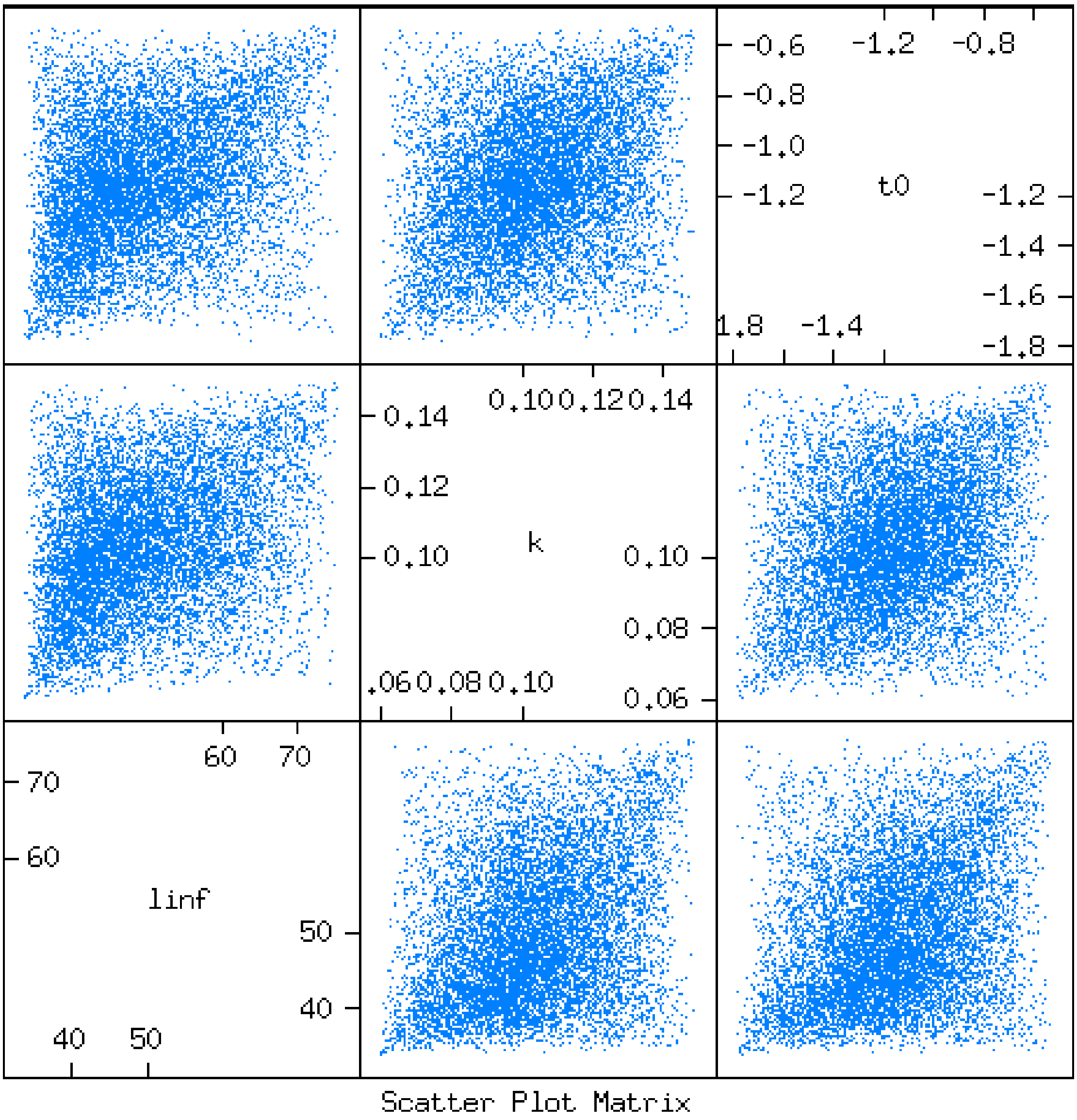


Figure 5: Scatter plot of the 10000 samples parameter from the multivariate triangle distribution.

```
boxplot(t(predict(vbTri, t = 0:20 + 0.5)))
```

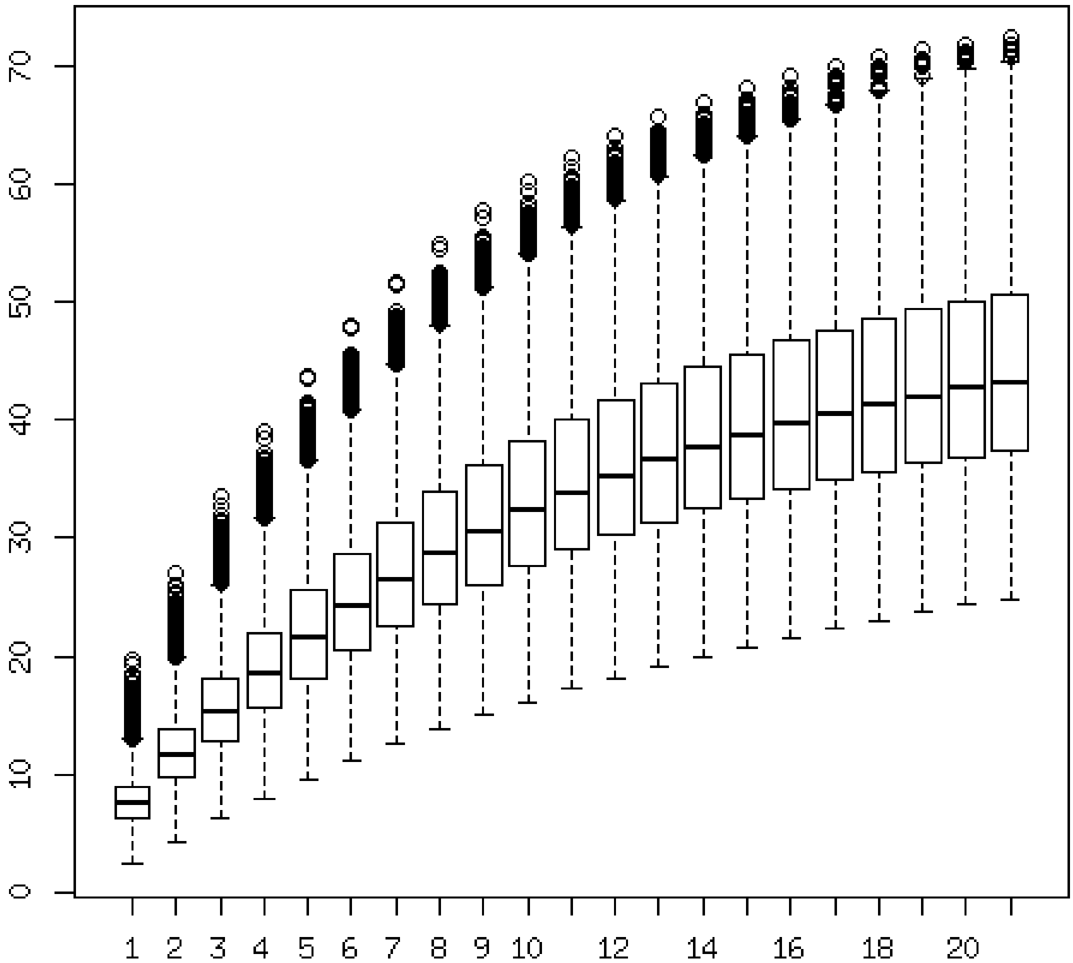


Figure 6: Growth curves using parameters simulated from a multivariate triangle distribution.

```
splom(data.frame(t(params(vbCop)@.Data)), pch = ".")
```

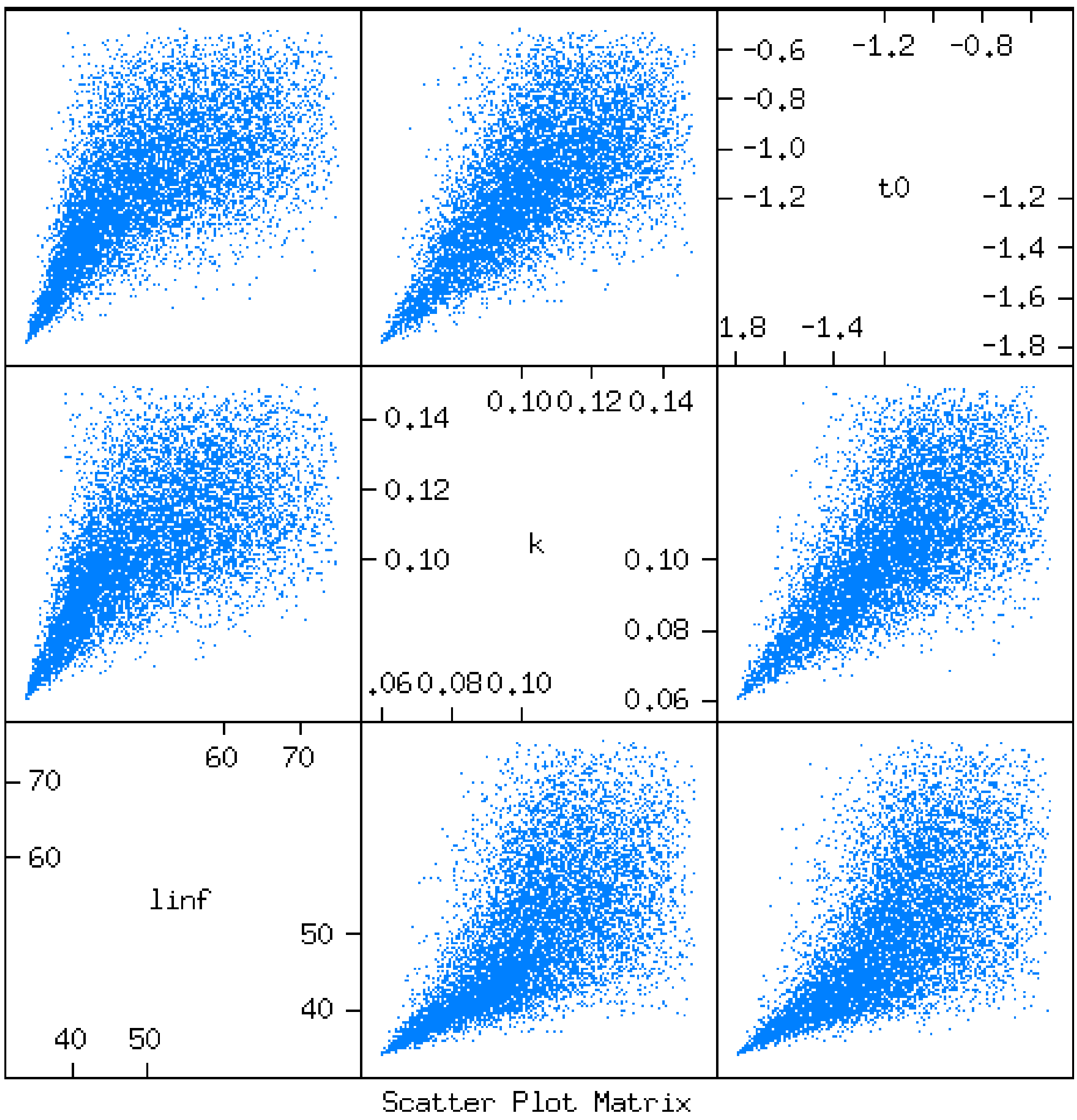


Figure 7: Scatter plot of the 10000 samples parameter from the using a triangle copula.

```
boxplot(t(predict(vbCop, t = 0:20 + 0.5)))
```

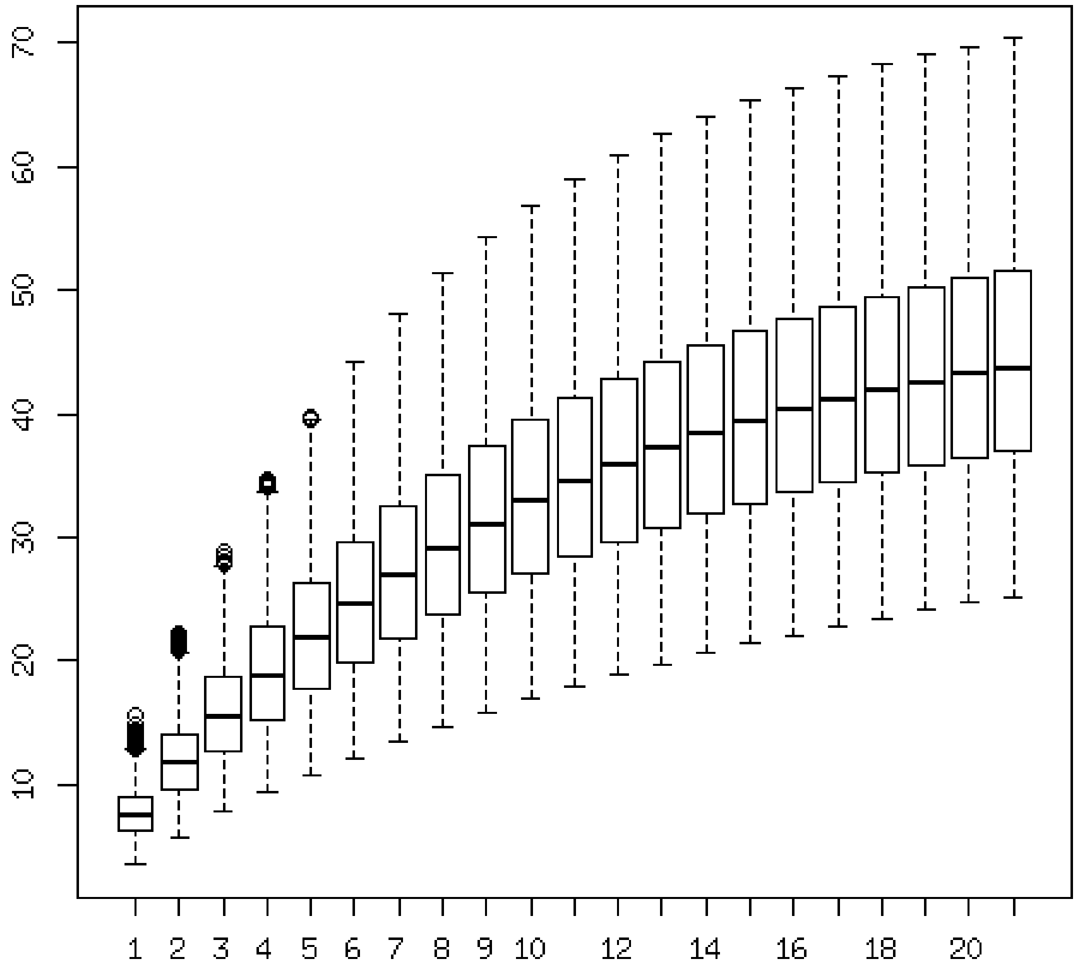


Figure 8: Growth curves from the using a triangle copula.

implementations, for the *FLStock* and *FLIndex* classes, which are mainly wrappers that call the *FLQuant* method several times.

When converting from length-based data to age-based data you need to be aware of how the aggregation of length classes is performed. For example, individuals in length classes, 1-2, 2-3, and 3-4 cm may all be considered as being of age 1 (obviously depending on the growth model). How should the values in those length classes be combined?

If the values are abundances then the values should be summed. Summing other types of values such as weights and rates (like fishing mortality) does not make sense. Instead these values shoud be averaged over the length classes (weighted by the abundance for weights). This is controlled using the `stat` argument which can be either `mean` or `sum` (the default).

```
# ==============================================================================
# Convert length-based FLQuant to age-based FLQuant
# ==============================================================================

# We demontrate the method by converting a catch-at-length FLQuant to a
# catch-at-age FLQuant.  First we make the an a4aGr object with a
# multivariate triangle distribution.  We use 10 iterations as an example.
# More would be better.
vbTriSmall <- mvrtriangle(10, vbObj, paramMargins = triPars)
# We use l2a by passing in the length-based FLQuant and the a4aGr object
cth.n <- l2a(catch.n(rfLen.stk), vbTriSmall)

## Converting lengths to ages ...
```

Instead of converting one *FLQuant* at a time, we can convert entire *FLStock* and *FLIndex* objects. In these cases the individual *FLQuant* slots of those classes are converted from length-based to age-based. As mentioned above, the aggregation method depends on the type of values the slots contain. The abundance slots (`*.n`, such as `stock.n`) are summed. The `*.wt`, `m`, `mat`, `harvest.spwn`, `m.spwn` and `harvest` slots of an *FLStock* object are averaged. The weights are then corrected to ensure that the total catch weight (and landings, discards and stock weights) in the age-based object is the same as that in the length-based object. The `index`, `catch.wt`, `index.var`, `sel.pattern` and `index.q` slots of an *FLIndex* object are averaged.

The method for *FLStock* classes takes an additional argument for the plusgroup.

```
# ==============================================================================
# Convert length-based FLStock and FLIndec to age-based
# ==============================================================================

aStk <- l2a(rfLen.stk, vbTriSmall, plusgroup = 14)

## Processing sum slots
## Converting lengths to ages ...
## Converting lengths to ages ...
## Converting lengths to ages ...
## Converting lengths to ages ...
## Processing mean slots
## Converting lengths to ages ...
## Converting lengths to ages ...
## Converting lengths to ages ...
## Converting lengths to ages ...
## Converting lengths to ages ...
## Processing weighted mean slots
## Converting lengths to ages ...
## Converting lengths to ages ...
## Converting lengths to ages ...
```

```
## Converting lengths to ages ...
## Washing up
## [1] "maxfbar has been changed to accomodate new plusgroup"
```

```
aIdx <- l2a(rfTrawl.idx, vbTriSmall)
```

```
## Converting lengths to ages ...
## Processing mean slots
## Converting lengths to ages ...
## Converting lengths to ages ...
## Converting lengths to ages ...
## Converting lengths to ages ...
## Processing weighted mean slots
## Converting lengths to ages ...
```

When converting with `l2a()` there are a number of defaults of which the user should be aware. As there is no information in the growth model on how to deal with individuals larger than Linf, all lengths above Linf are converted to the maximum age. The variability around Linf is dealt with by the randomization of the parameter Linf and this should be consistent with the variance in the data.

4 Natural mortality

Natural mortality is dealt with as an external parameter to the stock assessment model. The rationale is similar to that of growth: one should be able to grab information from a range of sources and feed it into the assessment.

The mechanism used by a4a is to build an interface that makes it transparent, flexible and hopefully easy to explore different options. In relation to natural mortality it means that the analyst should be able to use distinct models like Gislasson's, Charnov's, Pauly's, etc in a coherent framework making it possible to compare the outcomes of the assessment.

Within the a4a framework, the easiest way to insert natural mortality in the stock assessment is to use an *a4aM* object and run the method `m` to compute the values. The output is a *FLQuant* that can be directly inserted in the *FLStock* object to be used for the assessment.

4.1 `a4aM` - The M class

Natural mortality is implemented in a class named *a4aM*. This class is made up of three models of the class *FLModelSim*. Each model represents one effect: an age effect, a year effect and a time trend, named `shape`, `level` and `trend`, respectively. Check the help files for more information.

```
showClass("a4aM")
```

```
## Class "a4aM" [package "FLa4a"]
##
## Slots:
##
## Name:        shape      level      trend       name       desc      range
## Class: FLModelSim FLModelSim FLModelSim  character  character    numeric
##
## Extends: "FLComp"
```

The *a4aM* constructor requires that the models and parameters are provided. The default method will build each of these models as a constant value of 1. For example the usual "0.2" guessestimate could be set up by

```
mod02 <- FLModelSim(model = ~a, params = FLPar(a = 0.2))
m1 <- a4aM(level = mod02)
m1

## a4aM object:
##   shape: ~1
##   level: ~a
##   trend: ~1
```

More interesting natural mortality shapes can be set up using biological knowledge. The following example uses Jensen's second estimator (Kenshington, 2013) $M = 1.5K$ and an exponential decay to set up the level and shape of M.

```
# ==============================================================================
# Models or shape and level
# ==============================================================================

shape2 <- FLModelSim(model = ~exp(-age - 0.5))
level2 <- FLModelSim(model = ~1.5 * k, params = FLPar(k = 0.4))

# ==============================================================================
# a4aM constructor
# ==============================================================================

m2 <- a4aM(shape = shape2, level = level2)
m2

## a4aM object:
##   shape: ~exp(-age - 0.5)
##   level: ~1.5 * k
##   trend: ~1
```

As an alternative, an external factor may impact the natural mortality. This can be added through the `trend` model. Suppose M depends on the NAO through some mechanism that results in having lower M when NAO is negative and higher when it's positive. The impact is represented by the NAO value on the quarter before spawning, which occurs in the second quarter.

```
# ==============================================================================
# Get NAO
# ==============================================================================

nao.orig <- read.table("http://www.cdc.noaa.gov/data/correlation/nao.data",
    skip = 1, nrow = 62, na.strings = "-99.90")
dnms <- list(quant = "nao", year = 1948:2009, unit = "unique", season = 1:12,
    area = "unique")
# Build an FLQuant from the NAO data
nao.flq <- FLQuant(unlist(nao.orig[, -1]), dimnames = dnms, units = "nao")
# Build covar by calculating mean over the first 3 months
nao <- seasonMeans(nao.flq[, , , 1:3])
# Turn into Boolean
nao <- (nao > 0)

# ==============================================================================
# The trend model M increases 50% if NAO is positive on the first quarter
# ==============================================================================

trend3 <- FLModelSim(model = ~1 + b * nao, params = FLPar(b = 0.5))
```

```
# ==============================================================================
# Constructor
# ==============================================================================

shape3 <- FLModelSim(model = ~exp(-age - 0.5))
level3 <- FLModelSim(model = ~1.5 * k, params = FLPar(k = 0.4))
m3 <- a4aM(shape = shape3, level = level3, trend = trend3)
m3

## a4aM object:
##   shape: ~exp(-age - 0.5)
##   level: ~1.5 * k
##   trend: ~1 + b * nao
```

4.2 Adding multivariate normal parameter uncertainty

Uncertainty on natural mortality is added through uncertainty on the parameters. In the case of *a4aM* class it makes use of the *FLModelSim* 'mvr' methods. A wrapper for `mvrnorm` was implemented, but all the other options must be carried out in each sub-model at the time.

```
# ==============================================================================
# The same exponential decay for shape
# ==============================================================================

shape4 <- FLModelSim(model = ~exp(-age - 0.5))

# ==============================================================================
# For level we'll use Jensen's third estimator (Kenshington, 2013).
# ==============================================================================

# Has two parameters, k and t We include a variance-covariance matrix for
# the parameter interaction
level4 <- FLModelSim(model = ~k^0.66 * t^0.57, params = FLPar(matrix(c(0.4,
    10)), dimnames = list(params = c("k", "t"), iter = 1)), vcov = array(c(0.002,
    0.01, 0.01, 1), dim = c(2, 2)))

# ==============================================================================
# A trend from NAO
# ==============================================================================

# We also include variance
trend4 <- FLModelSim(model = ~1 + b * nao, params = FLPar(b = 0.5), vcov = matrix(0.02))

# ==============================================================================
# Create object and simulate 100 times
# ==============================================================================
m4 <- a4aM(shape = shape4, level = level4, trend = trend4)
m4 <- mvrnorm(100, m4)
m4

## a4aM object:
##   shape: ~exp(-age - 0.5)
##   level: ~k^0.66 * t^0.57
##   trend: ~1 + b * nao

# Look at the level model (for example)
```

```
m4@level

## An object of class "FLModelSim"
## Slot "model":
## ~k^0.66 * t^0.57
##
## Slot "params":
## An object of class "FLPar"
## iters:  100
##
## params
##                k               t
##  0.38961(0.0431) 10.12339(0.9083)
## units:  NA NA
##
## Slot "vcov":
##       [,1] [,2]
## [1,] 0.002 0.01
## [2,] 0.010 1.00
##
## Slot "distr":
## [1] "norm"

# Note the variance in the parameters
```

In this particular case, the **shape** model will not be randomized because it doesn't have a variance covariance matrix. Also note that because there is only one parameter in the **trend** model, the randomization will use a univariate normal distribution.

The same model could be achieved with:

```
m4 <- a4aM(shape = shape4, level = mvrnorm(100, level4), trend = mvrnorm(100,
    trend4))
```

4.3 Adding parameter uncertainty with copulas

We can also use copulas add parameter uncertainty to the natural mortality model, similar to the way we use them for the growth model 3.3. As stated above these processes make use of the methods implemented for the *FLModelSim* class.

In the following example we'll use Gislason's second estimator (REF), $M_l = K(\frac{L_inf}{l})^1.5$.

```
# ===========================================================================
# Using a triangle copula to model parameter uncertainty in natural
# mortality
# ===========================================================================

linf <- 60
k <- 0.4
# vcov matrix (make up some values)
mm <- matrix(NA, ncol = 2, nrow = 2)
# 10% cv
diag(mm) <- c((linf * 0.1)^2, (k * 0.1)^2)
# 0.2 correlation
mm[upper.tri(mm)] <- mm[lower.tri(mm)] <- c(0.05)
# a good way to check is using cov2cor
cov2cor(mm)
```

```
##        [,1]   [,2]
## [1,] 1.0000 0.2083
## [2,] 0.2083 1.0000
```

```
# create object
mgis2 <- FLModelSim(model = ~K * (linf/len)^1.5, params = FLPar(linf = linf,
    K = k), vcov = mm)
# set the lower, upper and (optionally) centre of the parameters
pars <- list(list(55, 65), list(a = 0.3, b = 0.6, c = 0.35))
mgis2 <- mvrtriangle(1000, mgis2, paramMargins = pars)
mgis2
```

```
## An object of class "FLModelSim"
## Slot "model":
## ~K * (linf/len)^1.5
##
## Slot "params":
## An object of class "FLPar"
## iters:  1000
##
## params
##             linf                K
## 60.06872(2.1171)  0.40957(0.0728)
## units:  NA
##
## Slot "vcov":
##       [,1]   [,2]
## [1,] 36.00 0.0500
## [2,]  0.05 0.0016
##
## Slot "distr":
## [1] "un t copula family  triangle"
```

The resulting parameter estimates can be seen in Figure 9.

We now have a new model that can be used for the `shape` model. Use the constructor or the set method to add the new model. Note that we have a quite complex method now for *M*. A length based `shape` model from Gislason's work, Jensen's third based temperature `level` and a time `trend` depending on NAO.

```
m5 <- a4aM(shape = mgis2, level = level4, trend = trend4)
# or
m5 <- m4
level(m5) <- mgis2
```

4.4 The "m" method

The `m` method is the workhorse method for computing natural mortality. The method returns an *FLQuant* that can be inserted in an *FLStock* for usage by the assessment method. Note that if the models use *age* and/or *year* as terms, the method expects these to be included in the call. If they're not, the method will use the range slot to work out the ages and/or years that should be predicted.

```
# ===============================================================================
# Using the m method to make an FLQuant of constant natural mortality
# ===============================================================================
```

```
splom(t(params(mgis2)@.Data))
```

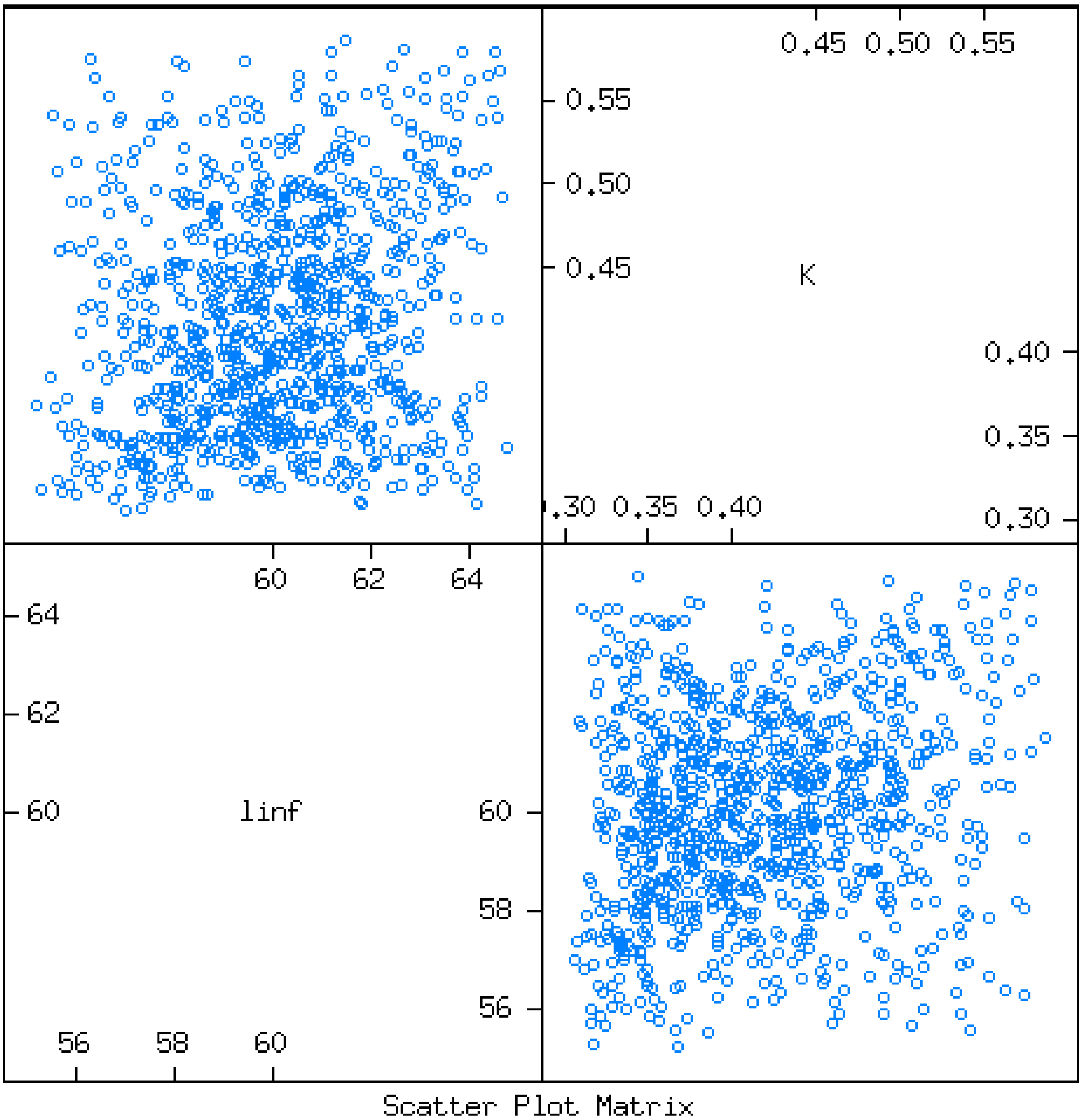


Figure 9: Parameter estimates for Gislason's second natural mortality model from using a triangle distribution.

```
par(mfrow = c(2, 1))
hist(c(params(mgis2)["linf", ]), main = "Linf", xlab = "")
hist(c(params(mgis2)["K", ]), main = "K", xlab = "")
```

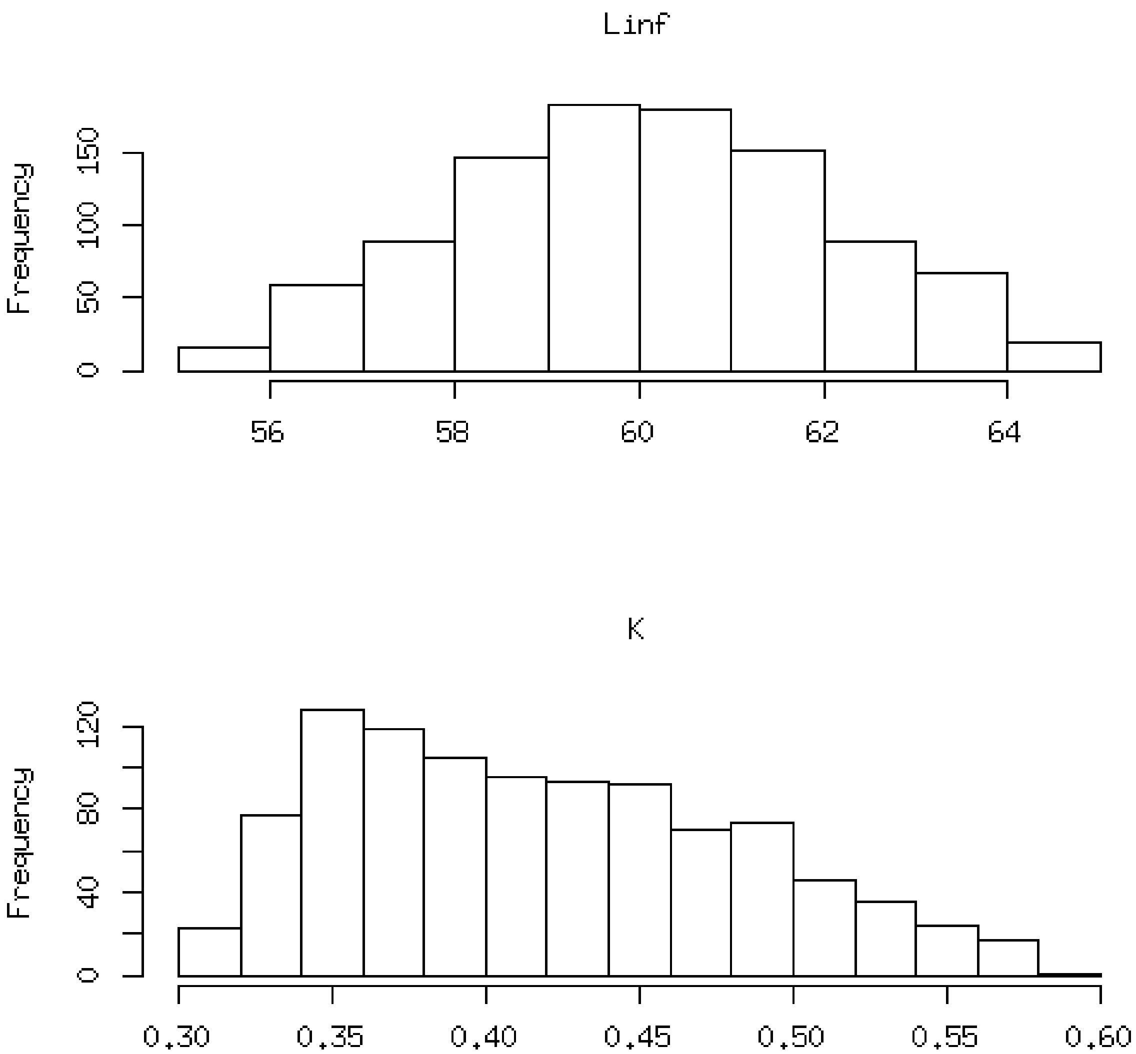


Figure 10: Marginal distributions of the parameters for Gislason's second natural mortality model from using a triangle distribution.

```
# Simple - no ages or years
m(m1)

## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 0
##     0 0.2
##
## units:  NA

# With ages
rngage(m1) <- c(0, 15)  # set the age range
m(m1)

## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 0
##    0  0.2
##    1  0.2
##    2  0.2
##    3  0.2
##    4  0.2
##    5  0.2
##    6  0.2
##    7  0.2
##    8  0.2
##    9  0.2
##    10 0.2
##    11 0.2
##    12 0.2
##    13 0.2
##    14 0.2
##    15 0.2
##
## units:  NA

# with ages and years
rngyear(m1) <- c(2000, 2010)  # set the year range
m(m1)

## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
##    0  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    1  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    3  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    4  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    5  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    6  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    7  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
```

```
##    8  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    9  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    10 0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    11 0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    12 0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    13 0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    14 0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##    15 0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2  0.2
##
## units:  NA
```

The next example has an age based shape. The information on the range of ages can be passed when calling `m`, or else the method will pick it up from the `range` slot. Note that in this case *mbar* becames relevant. It's the range of ages that is used to compute the mean level, which will match the `level` model.

```
# ==============================================================================
# Using the m method to make an FLQuant with age varying natural mortality
# ==============================================================================

# Simple
m(m2)

## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 0
##     0 0.6
##
## units:  NA

# With ages
rngage(m2) <- c(0, 15)
m(m2)

## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 0
##    0  6.0000e-01
##    1  2.2073e-01
##    2  8.1201e-02
##    3  2.9872e-02
##    4  1.0989e-02
##    5  4.0428e-03
##    6  1.4873e-03
##    7  5.4713e-04
##    8  2.0128e-04
##    9  7.4046e-05
##    10 2.7240e-05
##    11 1.0021e-05
##    12 3.6865e-06
##    13 1.3562e-06
##    14 4.9892e-07
##    15 1.8354e-07
##
## units:  NA
```

```
# With ages and years
rngyear(m2) <- c(2000, 2003)
m(m2)

## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 2000       2001       2002       2003
##    0  6.0000e-01 6.0000e-01 6.0000e-01 6.0000e-01
##    1  2.2073e-01 2.2073e-01 2.2073e-01 2.2073e-01
##    2  8.1201e-02 8.1201e-02 8.1201e-02 8.1201e-02
##    3  2.9872e-02 2.9872e-02 2.9872e-02 2.9872e-02
##    4  1.0989e-02 1.0989e-02 1.0989e-02 1.0989e-02
##    5  4.0428e-03 4.0428e-03 4.0428e-03 4.0428e-03
##    6  1.4873e-03 1.4873e-03 1.4873e-03 1.4873e-03
##    7  5.4713e-04 5.4713e-04 5.4713e-04 5.4713e-04
##    8  2.0128e-04 2.0128e-04 2.0128e-04 2.0128e-04
##    9  7.4046e-05 7.4046e-05 7.4046e-05 7.4046e-05
##    10 2.7240e-05 2.7240e-05 2.7240e-05 2.7240e-05
##    11 1.0021e-05 1.0021e-05 1.0021e-05 1.0021e-05
##    12 3.6865e-06 3.6865e-06 3.6865e-06 3.6865e-06
##    13 1.3562e-06 1.3562e-06 1.3562e-06 1.3562e-06
##    14 4.9892e-07 4.9892e-07 4.9892e-07 4.9892e-07
##    15 1.8354e-07 1.8354e-07 1.8354e-07 1.8354e-07
##
## units:  NA

# Note that
predict(level(m2))

##    iter
##       1
##   1 0.6

# is similar to
m(m2)["0"]

## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 2000 2001 2002 2003
##     0 0.6  0.6  0.6  0.6
##
## units:  NA

# that's because mbar is '0'
rngmbar(m2)

## minmbar maxmbar
##       0       0

# changing ...
rngmbar(m2) <- c(0, 5)
quantMeans(m(m2)[as.character(0:5)])
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 2000 2001 2002 2003
##   all 0.6  0.6  0.6  0.6
##
## units:  NA
```

```
# ==============================================================================
# Using the m method to make an FLQuant with a time trend
# ==============================================================================
# Simple
m(m3, nao = 1)
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 0
##     0 0.9
##
## units:  NA
```

```
# With ages
rngage(m3) <- c(0, 15)
m(m3, nao = 0)
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 0
##    0  6.0000e-01
##    1  2.2073e-01
##    2  8.1201e-02
##    3  2.9872e-02
##    4  1.0989e-02
##    5  4.0428e-03
##    6  1.4873e-03
##    7  5.4713e-04
##    8  2.0128e-04
##    9  7.4046e-05
##    10 2.7240e-05
##    11 1.0021e-05
##    12 3.6865e-06
##    13 1.3562e-06
##    14 4.9892e-07
##    15 1.8354e-07
##
## units:  NA
```

```
# With ages and years - passing in the NAO data as numeric (1,0,1,0)
rngyear(m3) <- c(2000, 2003)
m(m3, nao = as.numeric(nao[, as.character(2000:2003)]))
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 2000       2001       2002       2003
##    0  9.0000e-01 6.0000e-01 9.0000e-01 6.0000e-01
##    1  3.3109e-01 2.2073e-01 3.3109e-01 2.2073e-01
##    2  1.2180e-01 8.1201e-02 1.2180e-01 8.1201e-02
##    3  4.4808e-02 2.9872e-02 4.4808e-02 2.9872e-02
##    4  1.6484e-02 1.0989e-02 1.6484e-02 1.0989e-02
##    5  6.0642e-03 4.0428e-03 6.0642e-03 4.0428e-03
##    6  2.2309e-03 1.4873e-03 2.2309e-03 1.4873e-03
##    7  8.2069e-04 5.4713e-04 8.2069e-04 5.4713e-04
##    8  3.0192e-04 2.0128e-04 3.0192e-04 2.0128e-04
##    9  1.1107e-04 7.4046e-05 1.1107e-04 7.4046e-05
##    10 4.0860e-05 2.7240e-05 4.0860e-05 2.7240e-05
##    11 1.5032e-05 1.0021e-05 1.5032e-05 1.0021e-05
##    12 5.5298e-06 3.6865e-06 5.5298e-06 3.6865e-06
##    13 2.0343e-06 1.3562e-06 2.0343e-06 1.3562e-06
##    14 7.4838e-07 4.9892e-07 7.4838e-07 4.9892e-07
##    15 2.7531e-07 1.8354e-07 2.7531e-07 1.8354e-07
##
## units:  NA
```

```
# ==============================================================================
# Using the m method to make an FLQuant with uncertainty
# ==============================================================================

# Simple - no time trend
m(m4, nao = 1)

## An object of class "FLQuant"
## iters:  100
##
## , , unit = unique, season = all, area = unique
##
##      year
## quant 0
##     0 3.1053(0.495)
##
## units:  NA

# With ages
rngage(m4) <- c(0, 15)
m(m4, nao = 0)

## An object of class "FLQuant"
## iters:  100
##
## , , unit = unique, season = all, area = unique
##
##      year
## quant 0
##    0  2.0903e+00(2.33e-01)
##    1  7.6899e-01(8.57e-02)
##    2  2.8289e-01(3.15e-02)
```

```
##     3  1.0407e-01(1.16e-02)
##     4  3.8286e-02(4.26e-03)
##     5  1.4084e-02(1.57e-03)
##     6  5.1814e-03(5.77e-04)
##     7  1.9061e-03(2.12e-04)
##     8  7.0122e-04(7.81e-05)
##     9  2.5797e-04(2.87e-05)
##     10 9.4900e-05(1.06e-05)
##     11 3.4912e-05(3.89e-06)
##     12 1.2843e-05(1.43e-06)
##     13 4.7248e-06(5.26e-07)
##     14 1.7382e-06(1.94e-07)
##     15 6.3943e-07(7.12e-08)
##
## units:  NA
```

```
# With ages and years
rngyear(m4) <- c(2000, 2003)
m(m4, nao = as.numeric(nao[, as.character(2000:2003)]))
```

```
## An object of class "FLQuant"
## iters:  100
##
## , , unit = unique, season = all, area = unique
##
##       year
## quant 2000                 2001                 2002
##     0  3.1053e+00(4.95e-01) 2.0903e+00(2.33e-01) 3.1053e+00(4.95e-01)
##     1  1.1424e+00(1.82e-01) 7.6899e-01(8.57e-02) 1.1424e+00(1.82e-01)
##     2  4.2026e-01(6.70e-02) 2.8289e-01(3.15e-02) 4.2026e-01(6.70e-02)
##     3  1.5460e-01(2.46e-02) 1.0407e-01(1.16e-02) 1.5460e-01(2.46e-02)
##     4  5.6876e-02(9.07e-03) 3.8286e-02(4.26e-03) 5.6876e-02(9.07e-03)
##     5  2.0923e-02(3.34e-03) 1.4084e-02(1.57e-03) 2.0923e-02(3.34e-03)
##     6  7.6973e-03(1.23e-03) 5.1814e-03(5.77e-04) 7.6973e-03(1.23e-03)
##     7  2.8317e-03(4.51e-04) 1.9061e-03(2.12e-04) 2.8317e-03(4.51e-04)
##     8  1.0417e-03(1.66e-04) 7.0122e-04(7.81e-05) 1.0417e-03(1.66e-04)
##     9  3.8323e-04(6.11e-05) 2.5797e-04(2.87e-05) 3.8323e-04(6.11e-05)
##     10 1.4098e-04(2.25e-05) 9.4900e-05(1.06e-05) 1.4098e-04(2.25e-05)
##     11 5.1864e-05(8.27e-06) 3.4912e-05(3.89e-06) 5.1864e-05(8.27e-06)
##     12 1.9080e-05(3.04e-06) 1.2843e-05(1.43e-06) 1.9080e-05(3.04e-06)
##     13 7.0190e-06(1.12e-06) 4.7248e-06(5.26e-07) 7.0190e-06(1.12e-06)
##     14 2.5822e-06(4.12e-07) 1.7382e-06(1.94e-07) 2.5822e-06(4.12e-07)
##     15 9.4992e-07(1.51e-07) 6.3943e-07(7.12e-08) 9.4992e-07(1.51e-07)
##       year
## quant 2003
##     0  2.0903e+00(2.33e-01)
##     1  7.6899e-01(8.57e-02)
##     2  2.8289e-01(3.15e-02)
##     3  1.0407e-01(1.16e-02)
##     4  3.8286e-02(4.26e-03)
##     5  1.4084e-02(1.57e-03)
##     6  5.1814e-03(5.77e-04)
##     7  1.9061e-03(2.12e-04)
##     8  7.0122e-04(7.81e-05)
##     9  2.5797e-04(2.87e-05)
##     10 9.4900e-05(1.06e-05)
##     11 3.4912e-05(3.89e-06)
##     12 1.2843e-05(1.43e-06)
```

```
##    13 4.7248e-06(5.26e-07)
##    14 1.7382e-06(1.94e-07)
##    15 6.3943e-07(7.12e-08)
##
## units:  NA
```

See Figure 11.

5 Running assessments

There are two basic types of assessments available from using a4a: the management procedure (MP) fit and the full assessment fit. The MP fit does not compute estimates of covariances and is therefore quicker to execute, while the full assessment fit returns parameter estimates and their covariances and hence retains the ability to simulate from the model at the expense of longer fitting time.

In the a4a assessment model, the model structure is defined by submodels. These are models for the different parts of a statistical catch at age model that requires structural assumptions, such as the selectivity of the fishing fleet, or how F-at-age changes over time. It is advantageous to write the model for F-at-age and survey catchability as linear models (by working with log F and log Q) becuase it allows us to use the linear modelling tools available in R: see for example gam formulas, or factorial design formulas using lm. In R's linear modelling lanquage, a constant model is coded as ~ 1, while a slope over age would simply be $\sim$ age. Extending this we can write a traditional year / age seperable F model like $\sim$ factor(age) + factor(year).

There are effectively 5 submodels in operation: the model for F-at-age, a model for initial age structure, a model for recruitment, a (list) of model(s) for survey catchability-at-age, and a list of models for the observation variance of catch.n and the survey indices. In practice, we fix the variance models and the initial age structure models, but in theory these can be changed. A basic set of submodels would be

```
fmodel <- ~factor(age) + factor(year)
qmodel <- list(~factor(age))
```

```
library(FLa4a)
library(diagram)
data(ple4)
data(ple4.indices)
source("funs.R")
```

5.1 Quick and dirty

The default settings of the stock assessment model work reasoably well. It's an area of research that will improve with time.

```
data(ple4)
data(ple4.indices)
fit <- sca(ple4, ple4.indices)
```

```
## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997  1
##      BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
```

```
bwplot(data ~ factor(quant) | year, data = m(m4, nao = as.numeric(nao[, as.character(2000:2003)])))
```

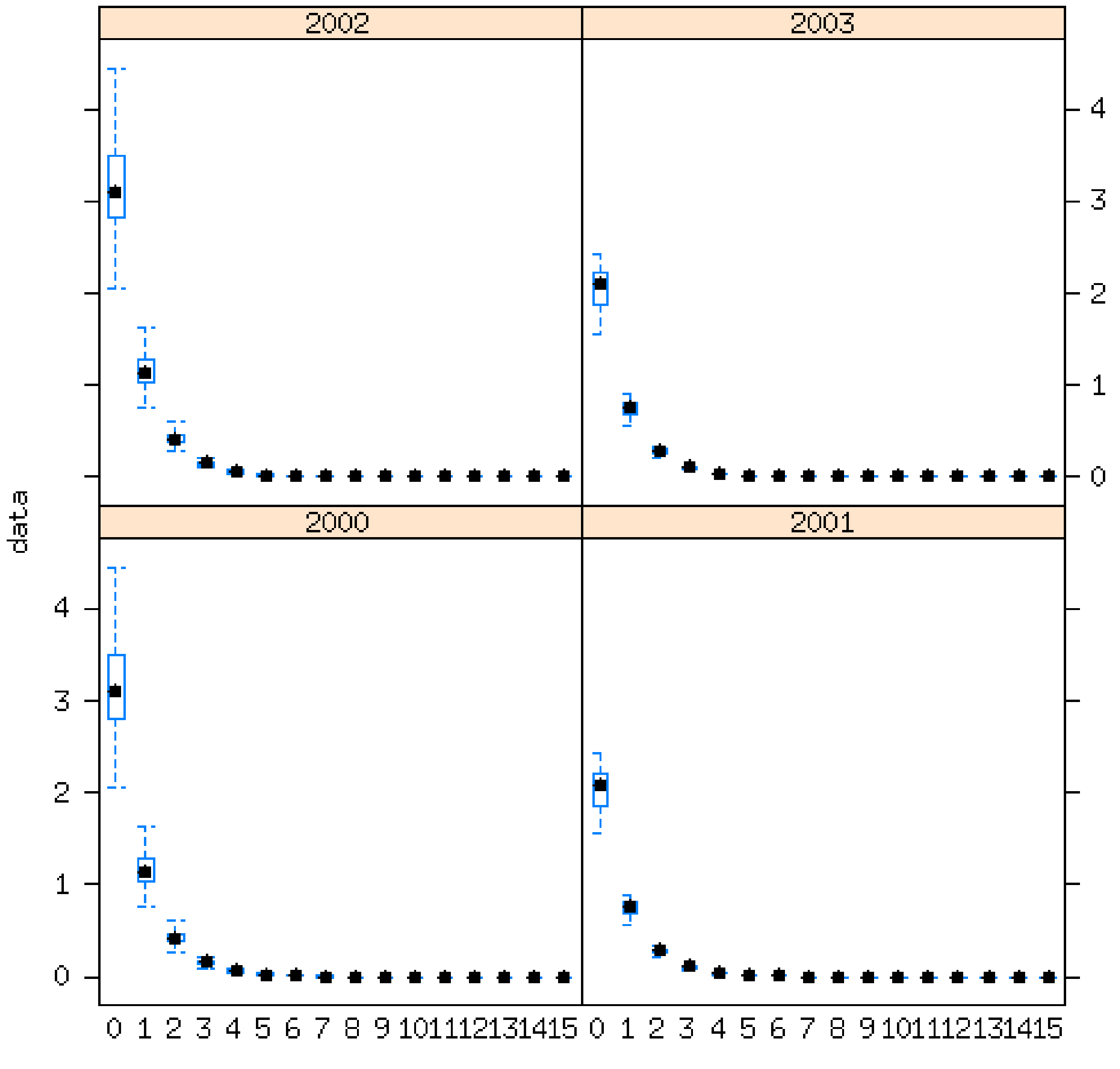


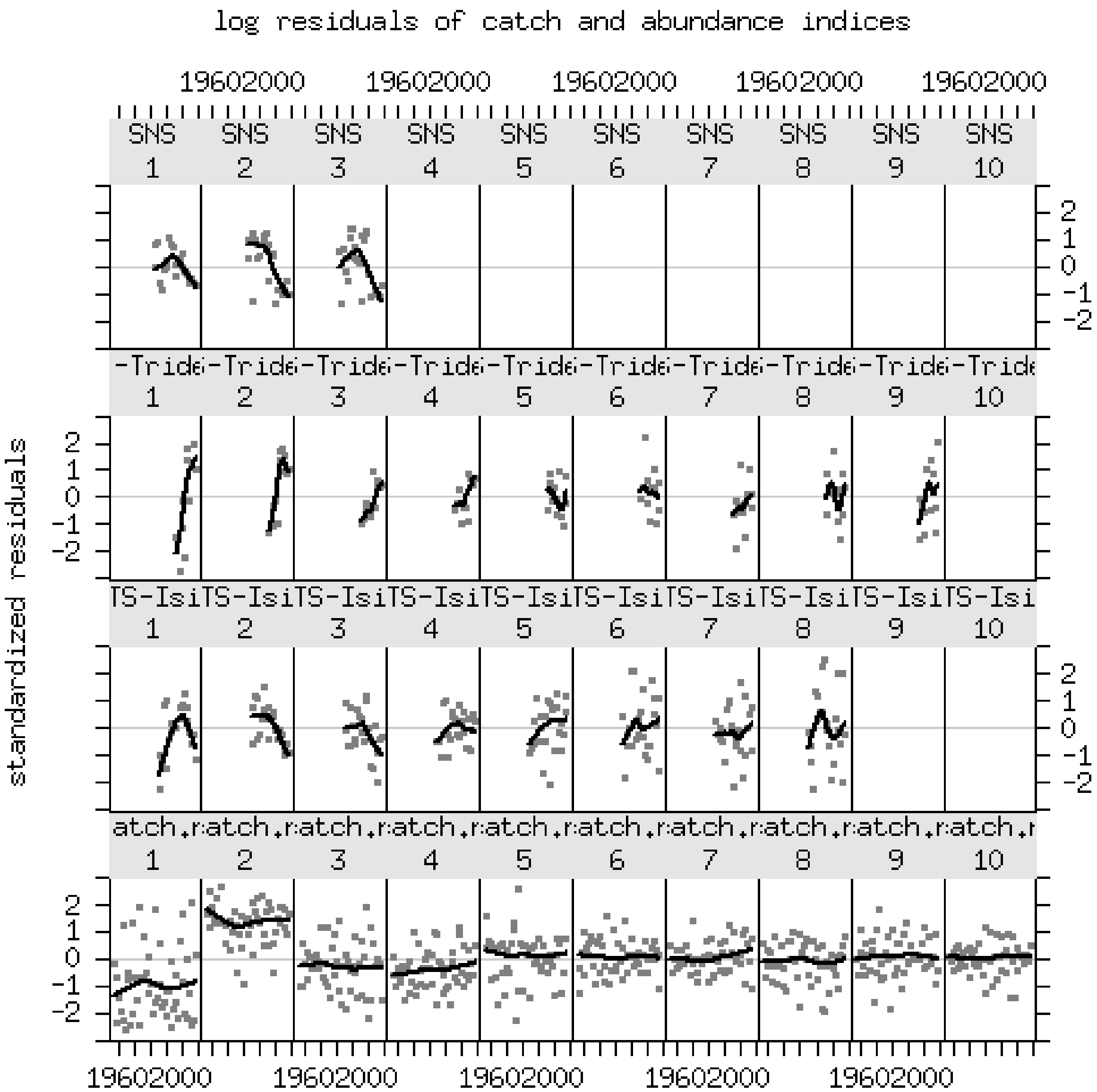
Figure 11: Natural mortality with age and year trend.

```
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##      Predictions will be made for missing observations.
```

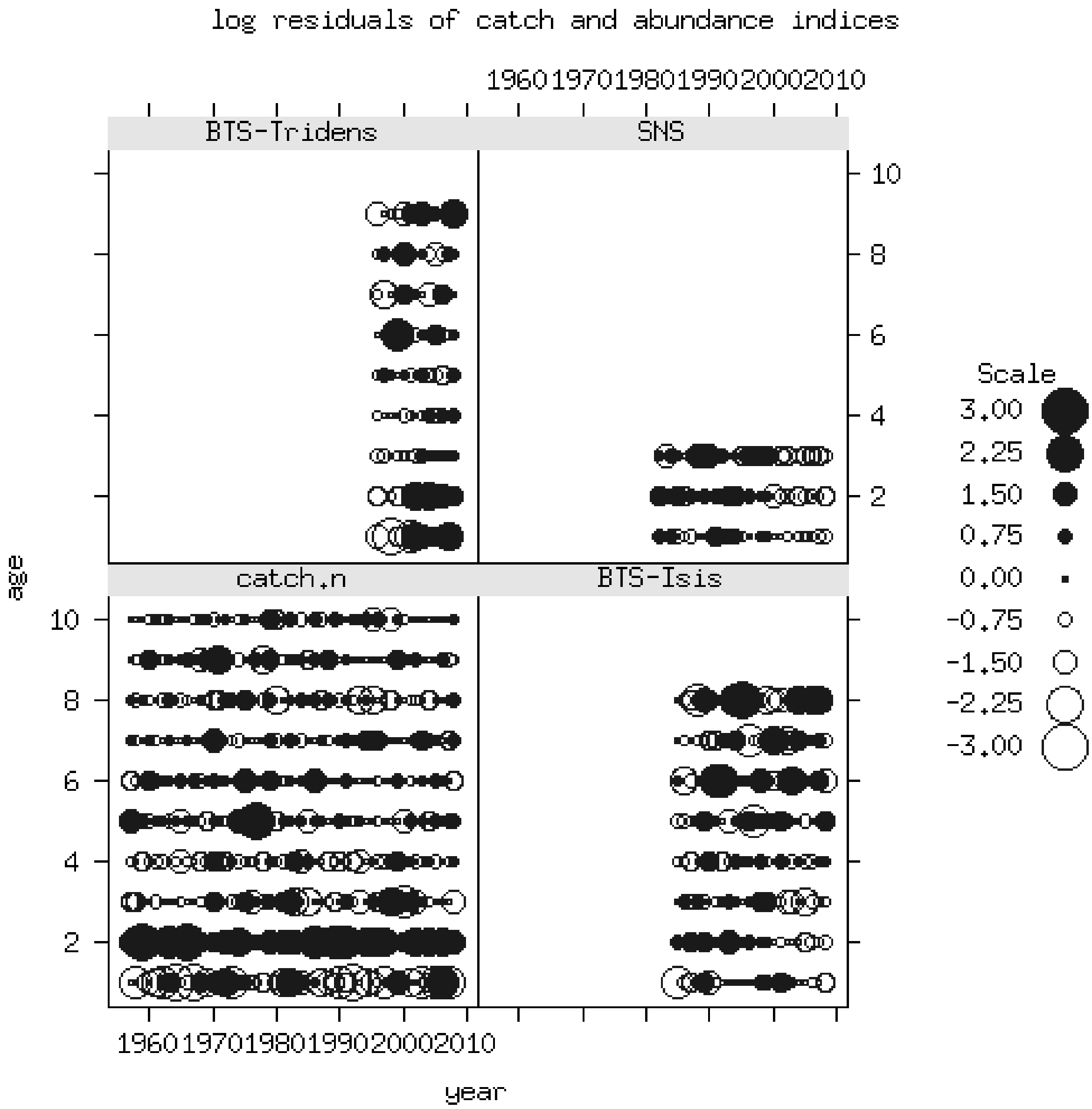
Note that because the survey index for plaice has missing values we get a warning saying that we assume these values are missing at random, and not because the observations were zero.

```
res <- residuals(fit, ple4, ple4.indices)
```

```
plot(res, main = "Residuals")
```



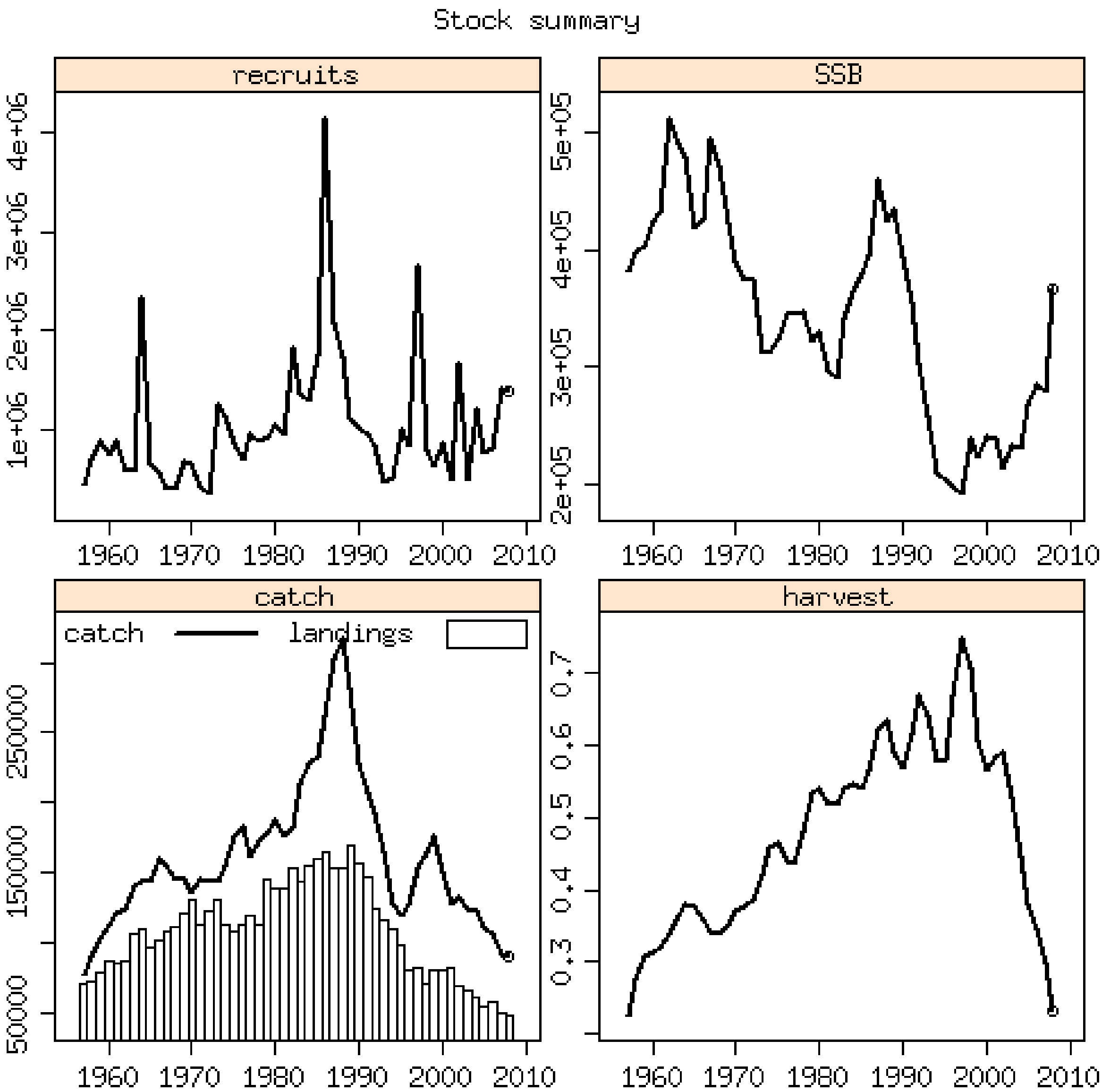

```
bubbles(res)
```



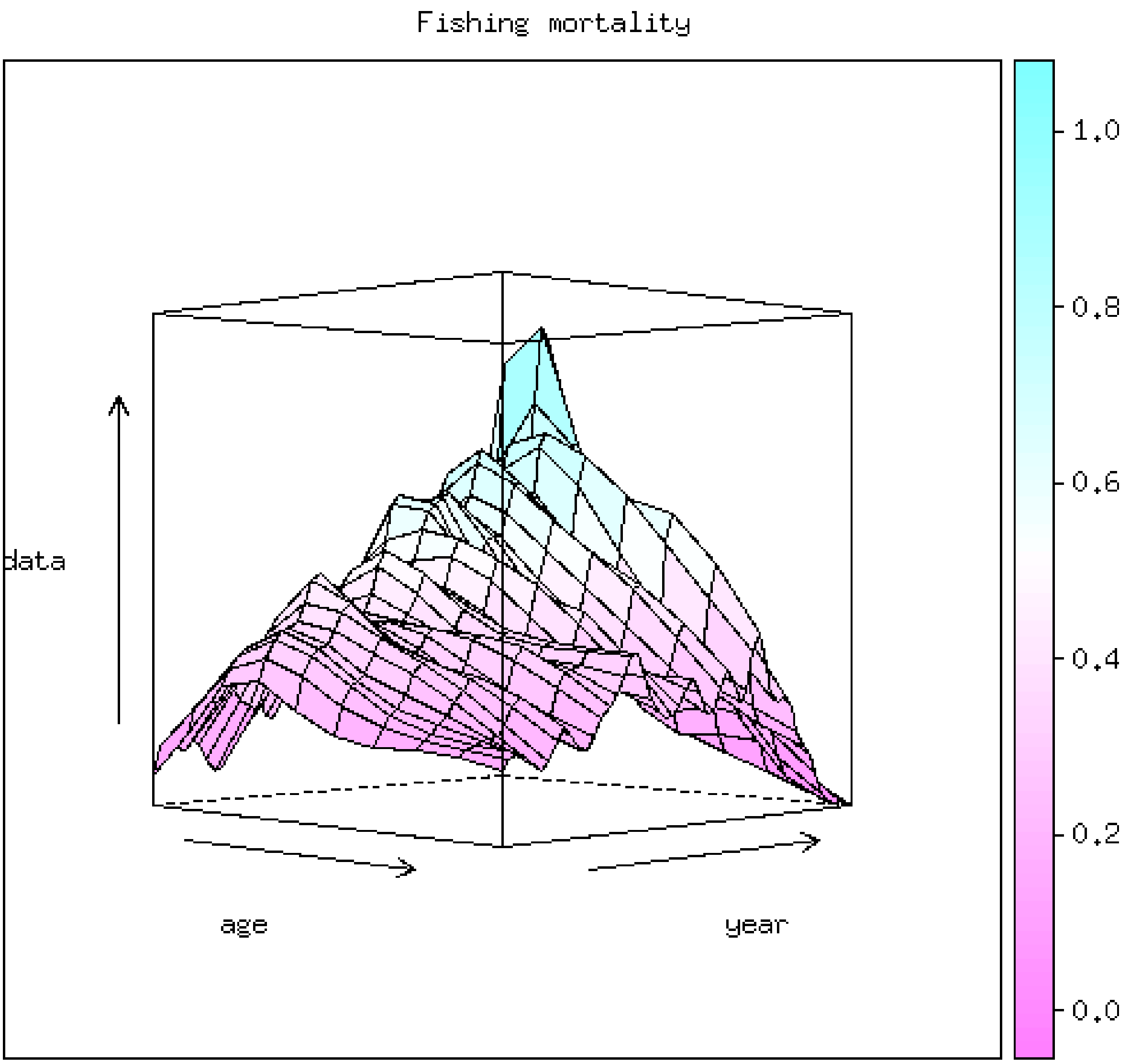

```
qqmath(res)

## Error:  unable to find an inherited method for function 'qqmath' for signature '"formula",
"data.frame"'
```

We can inspect the summaries from this fit my adding it to the origional stock object, for example to see the fitted fbar we can do

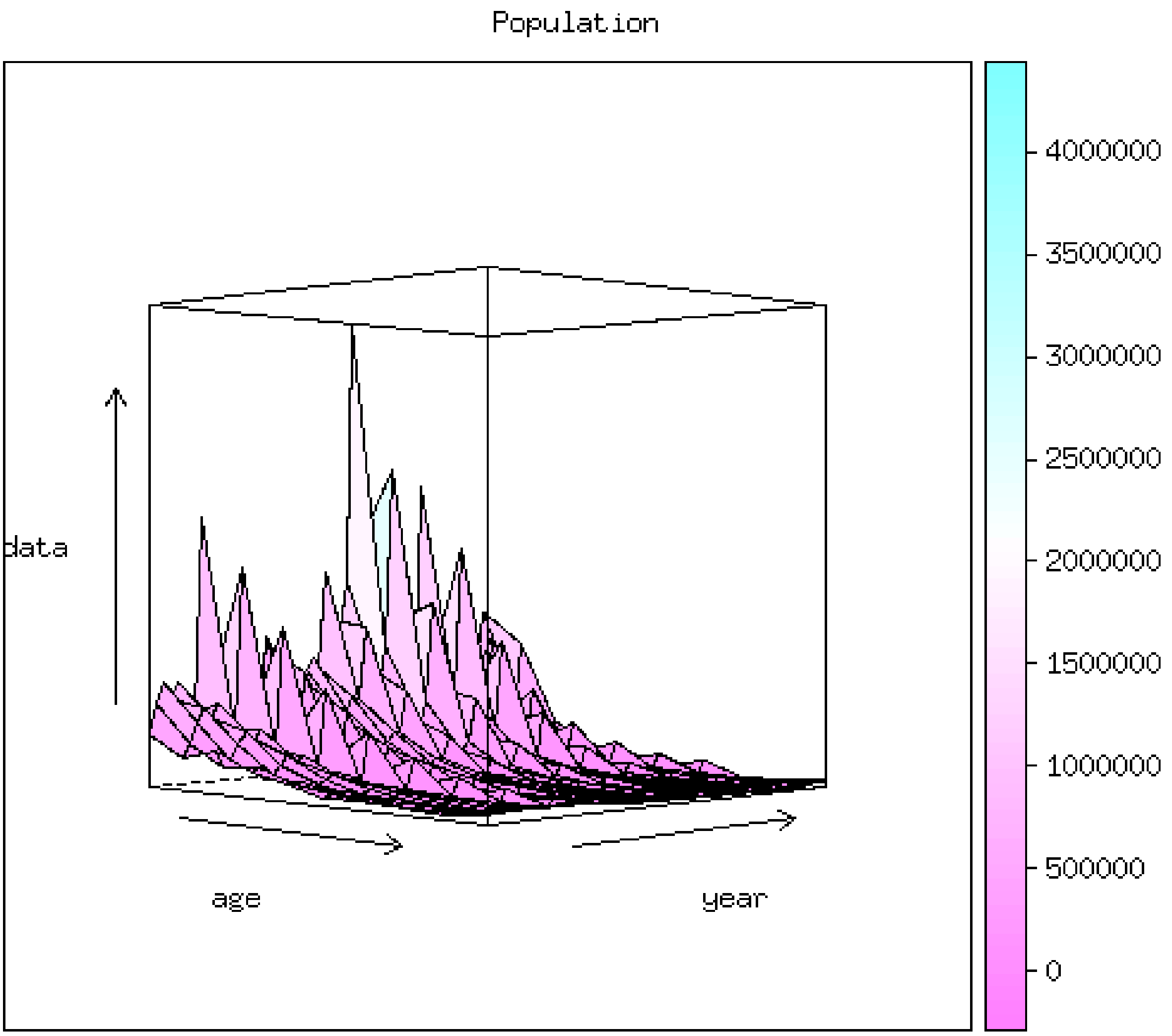
```
stk <- ple4 + fit
plot(stk, main = "Stock summary")

## Warning:  invalid factor level, NA generated
```



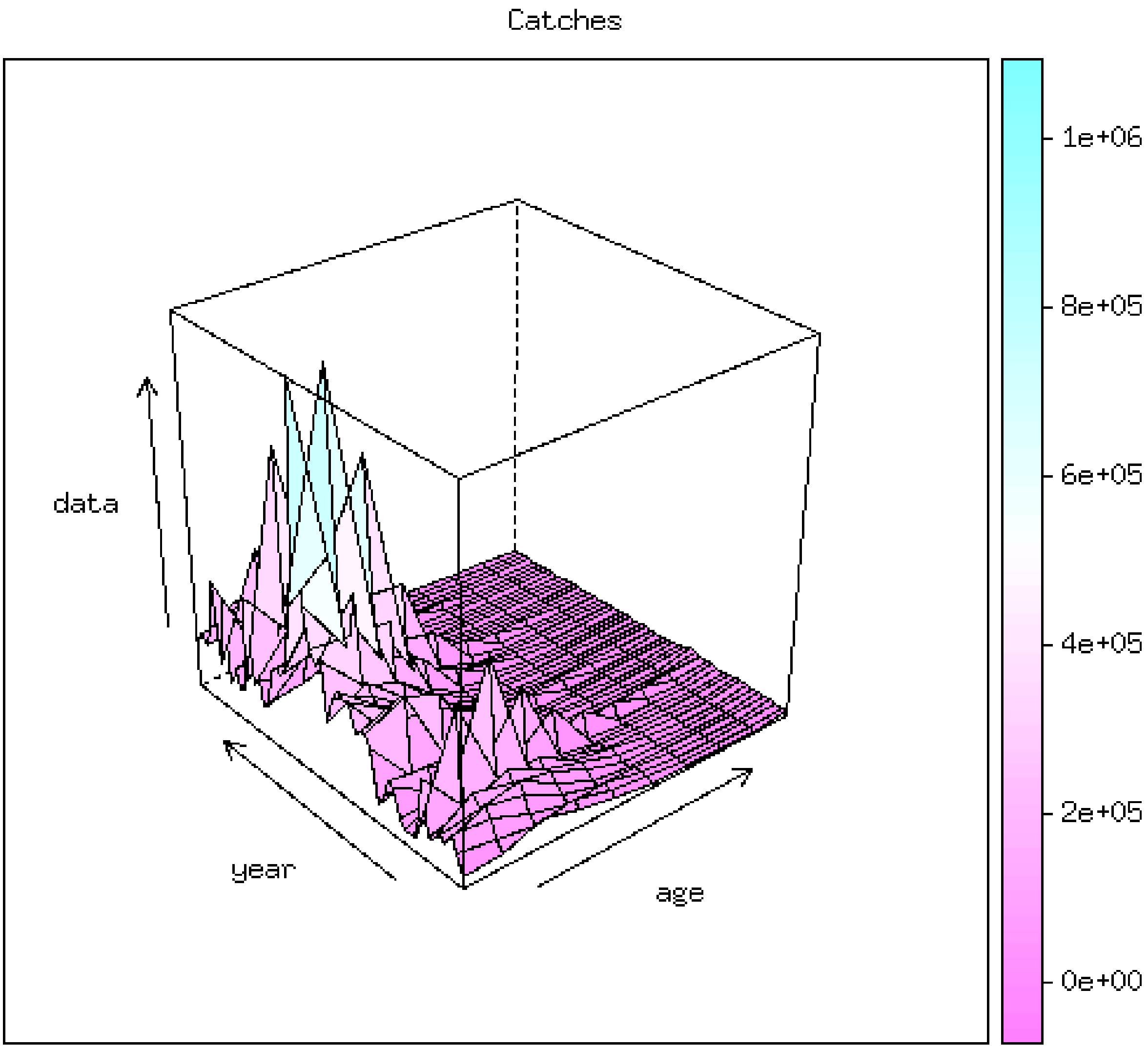

```
wireframe(data ~ age + year, data = as.data.frame(harvest(stk)), drape = TRUE,
    main = "Fishing mortality", screen = list(x = -90, y = -45))
```




```
wireframe(data ~ age + year, data = as.data.frame(stock.n(stk)), drape = TRUE,
    main = "Population", screen = list(x = -90, y = -45))
```




```
wireframe(data ~ age + year, data = as.data.frame(catch.n(stk)), drape = TRUE,
    main = "Catches")
```



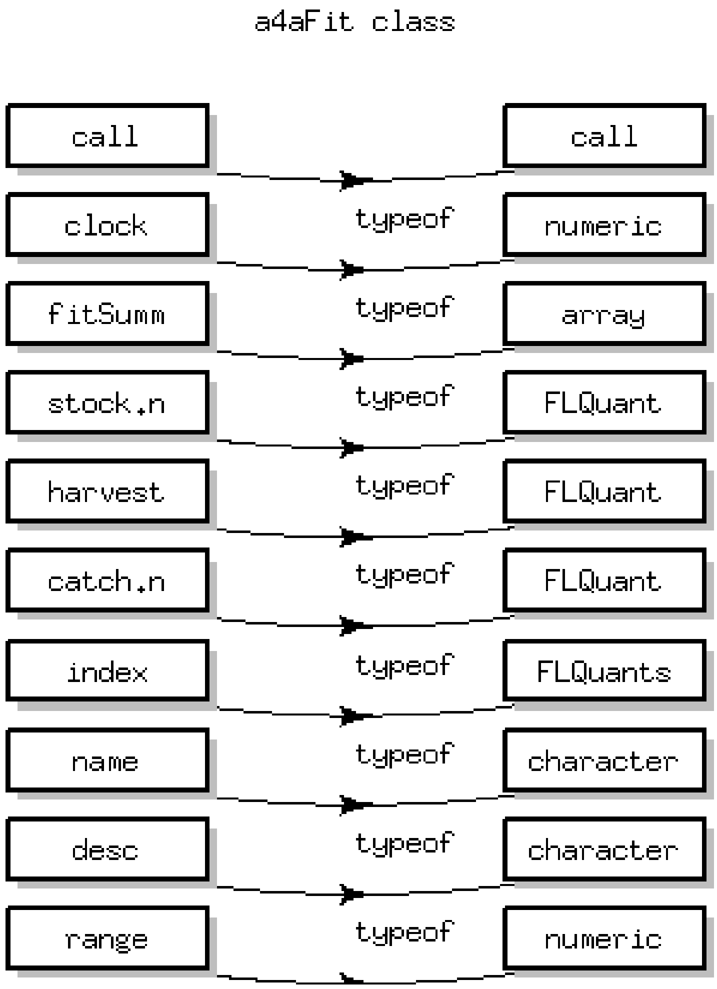
5.2 Data structures

The basic model output is contained in the a4aFit class. This object contains only the fitted values.

```
showClass("a4aFit")
```

```
## Class "a4aFit" [package "FLa4a"]
##
## Slots:
##
## Name:        call     clock    fitSumm   stock.n   harvest   catch.n
## Class:       call   numeric      array   FLQuant   FLQuant   FLQuant
##
## Name:       index      name       desc     range
## Class:  FLQuants character character   numeric
##
## Extends: "FLComp"
##
## Known Subclasses: "a4aFitSA"
```

```
plotS4("a4aFit", main = "a4aFit class", lwd = 1, box.lwd = 2, cex.txt = 0.8,
    box.size = 0.1, box.type = "square", box.prop = 0.3)
```



Fitted values are stored in the `stock.n`, `harvest`, `catch.n` and `index` slots. It also contains information carried over from the stock object used to fit the model: the name of the stock in `name`, any description provided in `desc` and the age and year range and mean F range in `range`. There is also a wall clock that has a breakdown of the time taken o run the model.

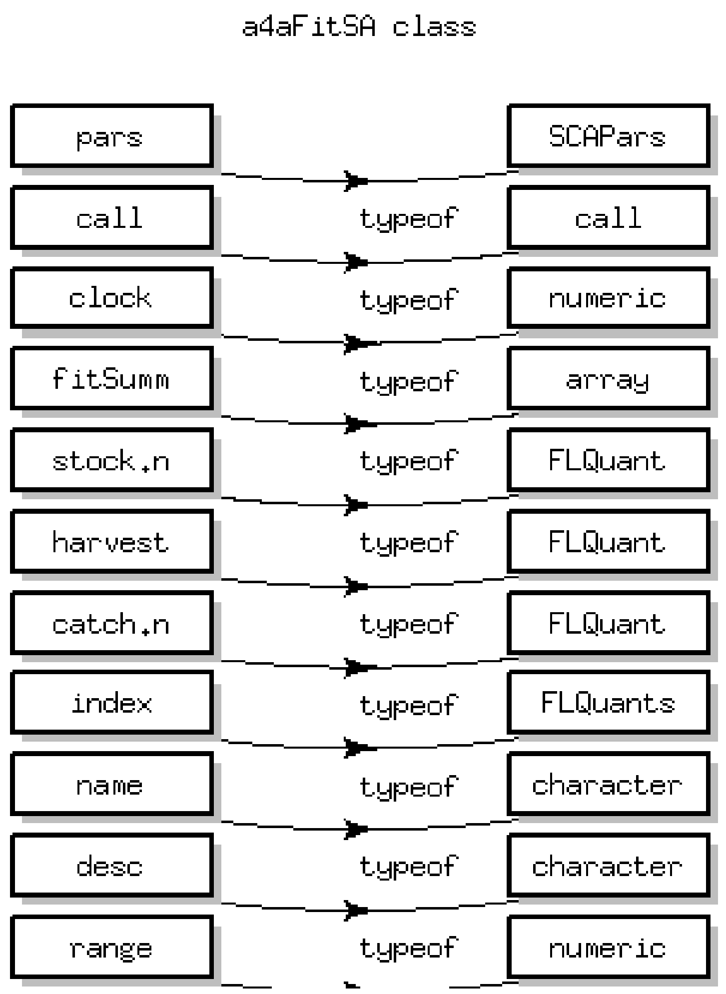
The full assessment fit returns an object of `a4aFitSA` class:

```
showClass("a4aFitSA")
```

```
## Class "a4aFitSA" [package "FLa4a"]
##
## Slots:
##
## Name:        pars      call     clock    fitSumm    stock.n    harvest
## Class:    SCAPars      call   numeric      array    FLQuant    FLQuant
##
```

```
## Name:     catch.n     index       name       desc     range
## Class:    FLQuant  FLQuants character character   numeric
##
## Extends:
## Class "a4aFit", directly
## Class "FLComp", by class "a4aFit", distance 2
```

```
plotS4("a4aFitSA", main = "a4aFitSA class", lwd = 1, box.lwd = 2, cex.txt = 0.8,
    box.size = 0.1, box.type = "square", box.prop = 0.3)
```



The additional slots in the assessment output is the **fitSumm** and **pars** slots which are containers for model summaries and the model parameters. The **pars** slot is a class of type **SCAPars** which is itself composed of sub-classes, designed to contain the information necessary to simulate from the model.

```
showClass("SCAPars")
```

```
## Class "SCAPars" [package "FLa4a"]
##
```

```
## Slots:
##
## Name:       stkmodel        qmodel        vmodel
## Class: a4aStkParams     submodels     submodels
```

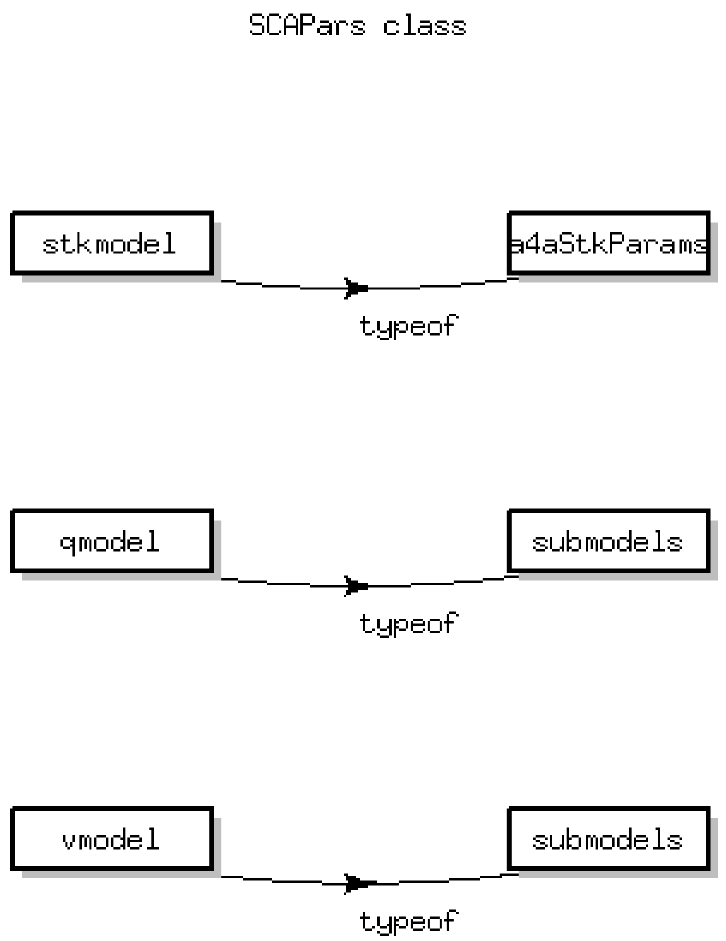
```r
showClass("a4aStkParams")
```

```
## Class "a4aStkParams" [package "FLa4a"]
##
## Slots:
##
## Name:        fMod      n1Mod      srMod     params       vcov centering
## Class:    formula    formula    formula      FLPar      array   numeric
##
## Name:       distr          m      units       name       desc      range
## Class: character    FLQuant character character character    numeric
##
## Extends: "FLComp"
```

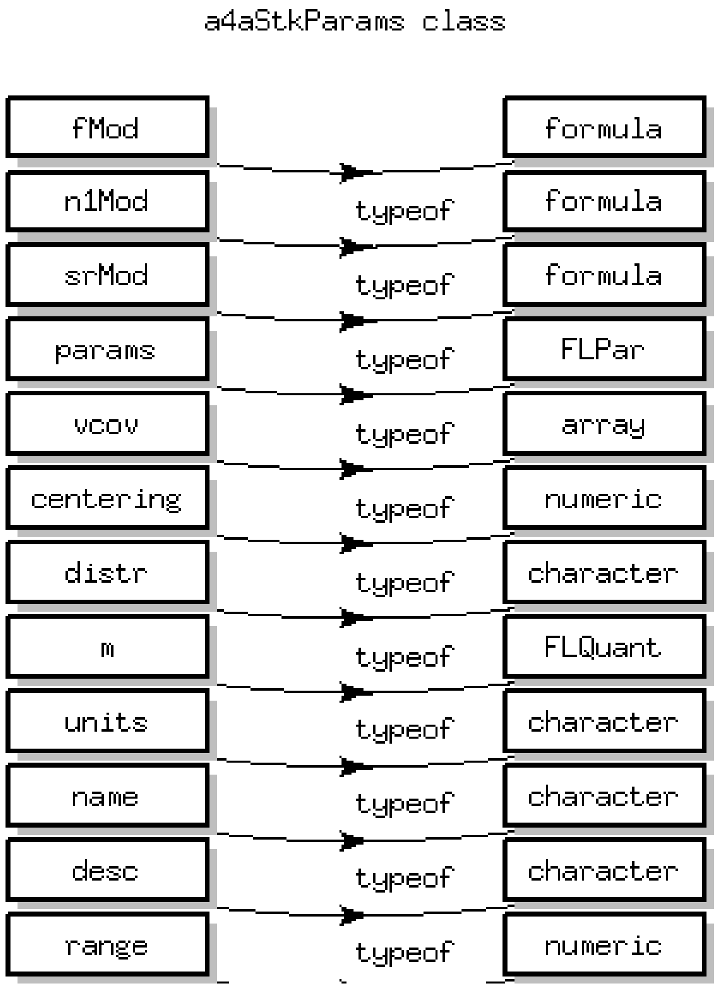
```r
showClass("submodel")
```

```
## Class "submodel" [package "FLa4a"]
##
## Slots:
##
## Name:         Mod     params       vcov centering      distr       name
## Class:    formula      FLPar      array   numeric character character
##
## Name:        desc      range
## Class: character    numeric
##
## Extends: "FLComp"
```

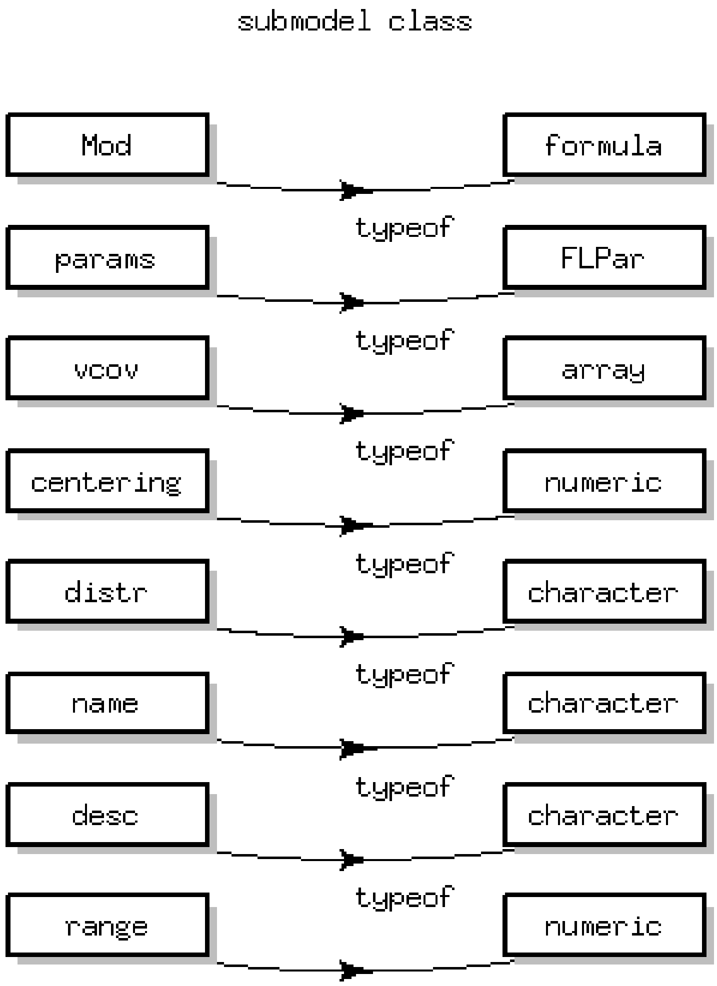
```r
plotS4("SCAPars", main = "SCAPars class", lwd = 1, box.lwd = 2, cex.txt = 0.8,
    box.size = 0.1, box.type = "square", box.prop = 0.3)
```




```
plotS4("a4aStkParams", main = "a4aStkParams class", lwd = 1, box.lwd = 2, cex.txt = 0.8,
    box.size = 0.1, box.type = "square", box.prop = 0.3)
```




```
plotS4("submodel", main = "submodel class", lwd = 1, box.lwd = 2, cex.txt = 0.8,
    box.size = 0.1, box.type = "square", box.prop = 0.3)
```



for example, all the parameters required so simulate a time-series of mean F trends is contained in the `stkmodel` slot, which is a class of type `a4aStkParams`. This class contains the relevant submodels (see later), their parameters `params` and the joint covariance matrix `vcov` for all stock related parameters.

5.3 The sca method - statistical catch-at-age

We will now take a look at some examples for F models and the forms that we can get. Lets start with a separable model in which we model selectivity at age as an (unpenalised) thin plate spline. We will use the North Sea Plaice data again, and since this has 10 ages we will use a simple rule of thumb that the spline should have fewer than $\frac{10}{2} = 5$ degrees of freedom, and so we opt for 4 degrees of freedom. We will also do the same for year and model the change in F through time as a smoother with 20 degrees of freedom.

Lets now investigate some variations in the selectivity shape with time, but only a little... we can do this by adding a smooth interaction term in the fmodel

A further move is to free up the Fs to vary more over time

In the last examples the Fs are linked across age and time. What if we want to free up a specific age class because in the residuals we see a consistent pattern. This can happen, for example, if the spatial distribution of juvenilles is disconnected to the distribution of adults. The fishery focuses on the adult fish, and therefore the the F on young fish is a function of the distribution of the juveniles and could deserve a seperate model. This can be achieved by

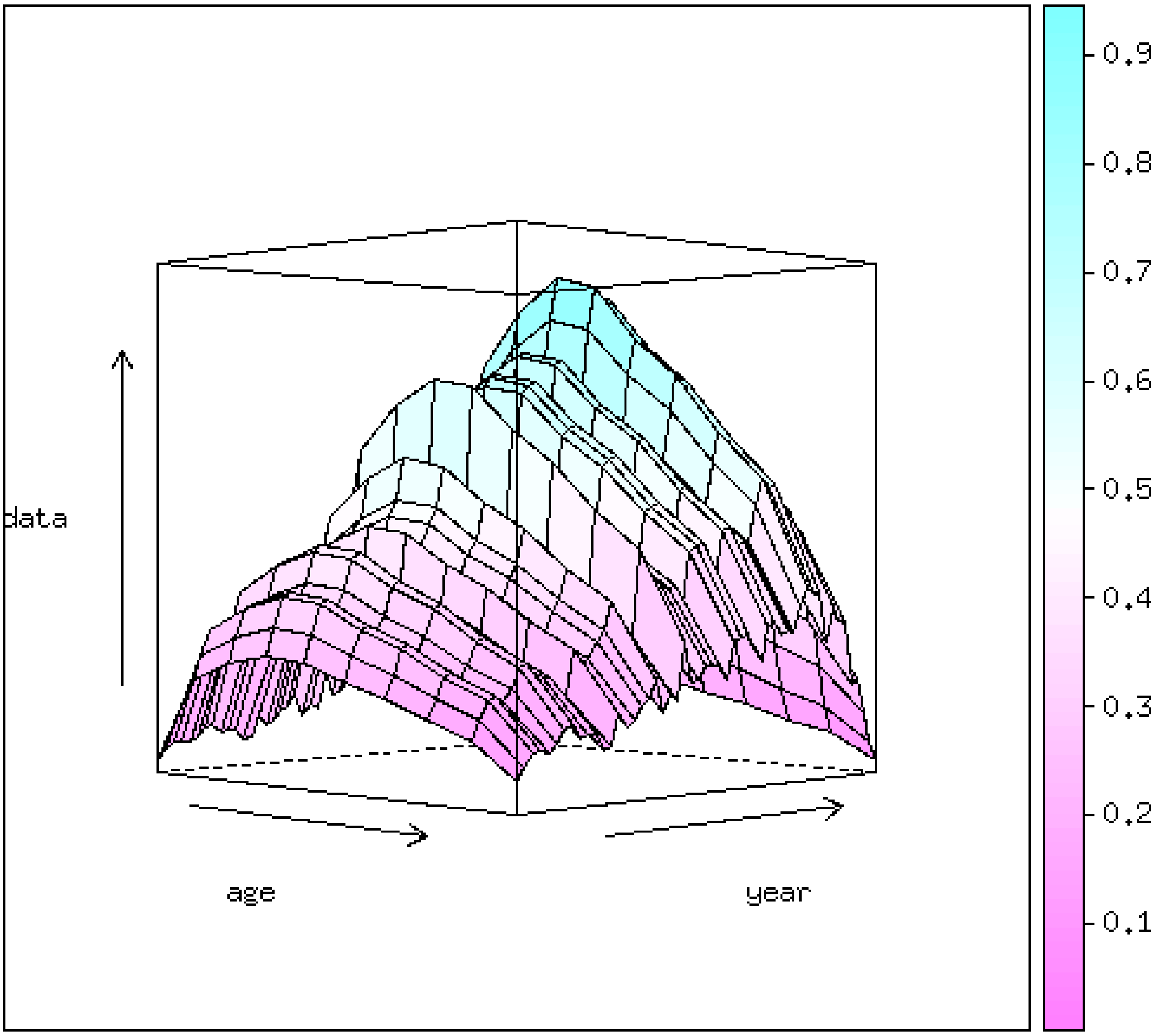
Please note that each of these model *structures* lets say, have not been tuned to the data. The degrees of freedom of each model can be better tuned to the data by using model selection procedures such as AIC or BIC.

5.3.1 Fishing mortality submodel

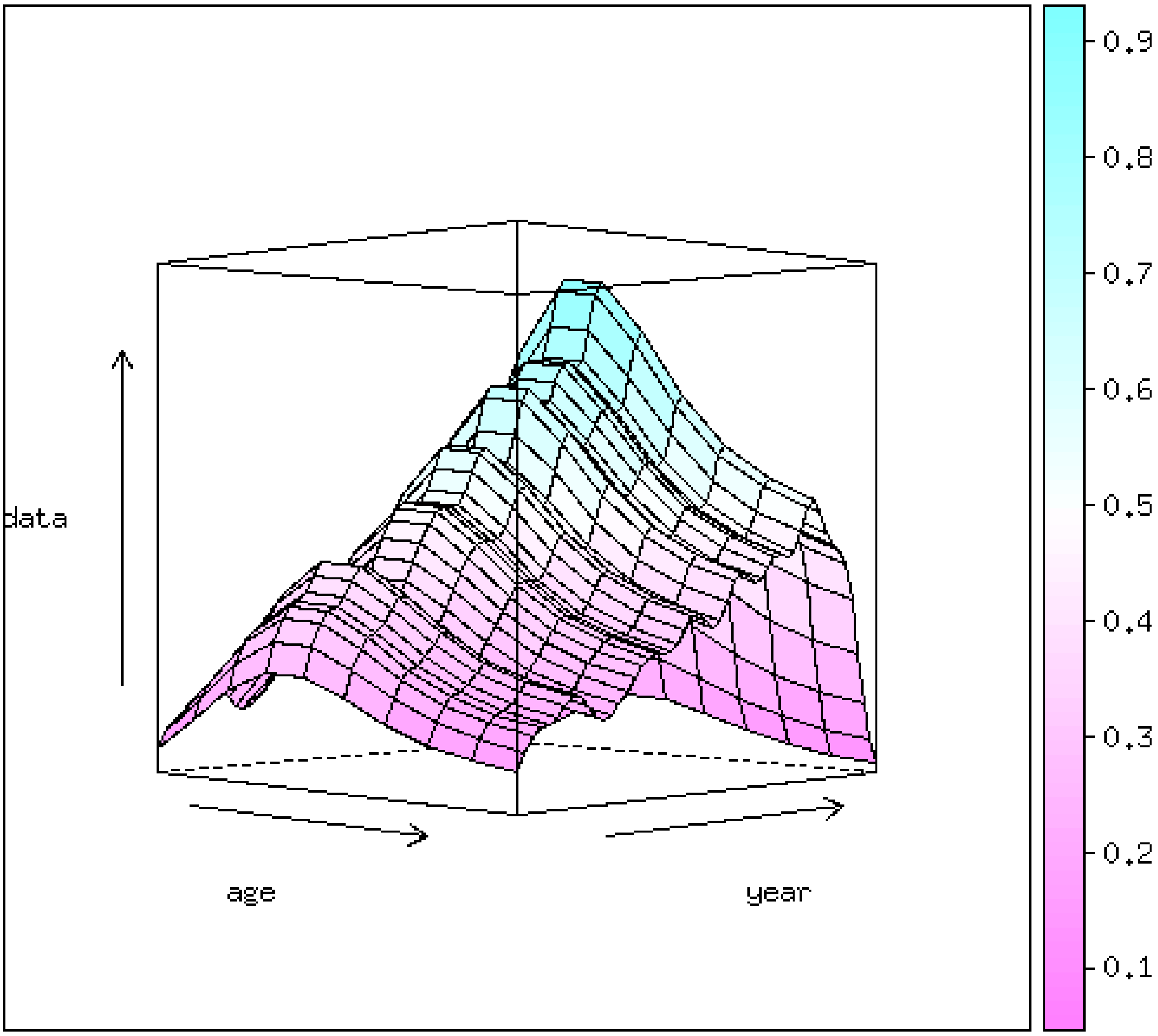
```
qmodel <- list(~factor(age))
fmodel <- ~factor(age) + factor(year)
fit <- sca(stock = ple4, indices = ple4.indices[1], fmodel = fmodel, qmodel = qmodel)
```

```
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##      Predictions will be made for missing observations.
```

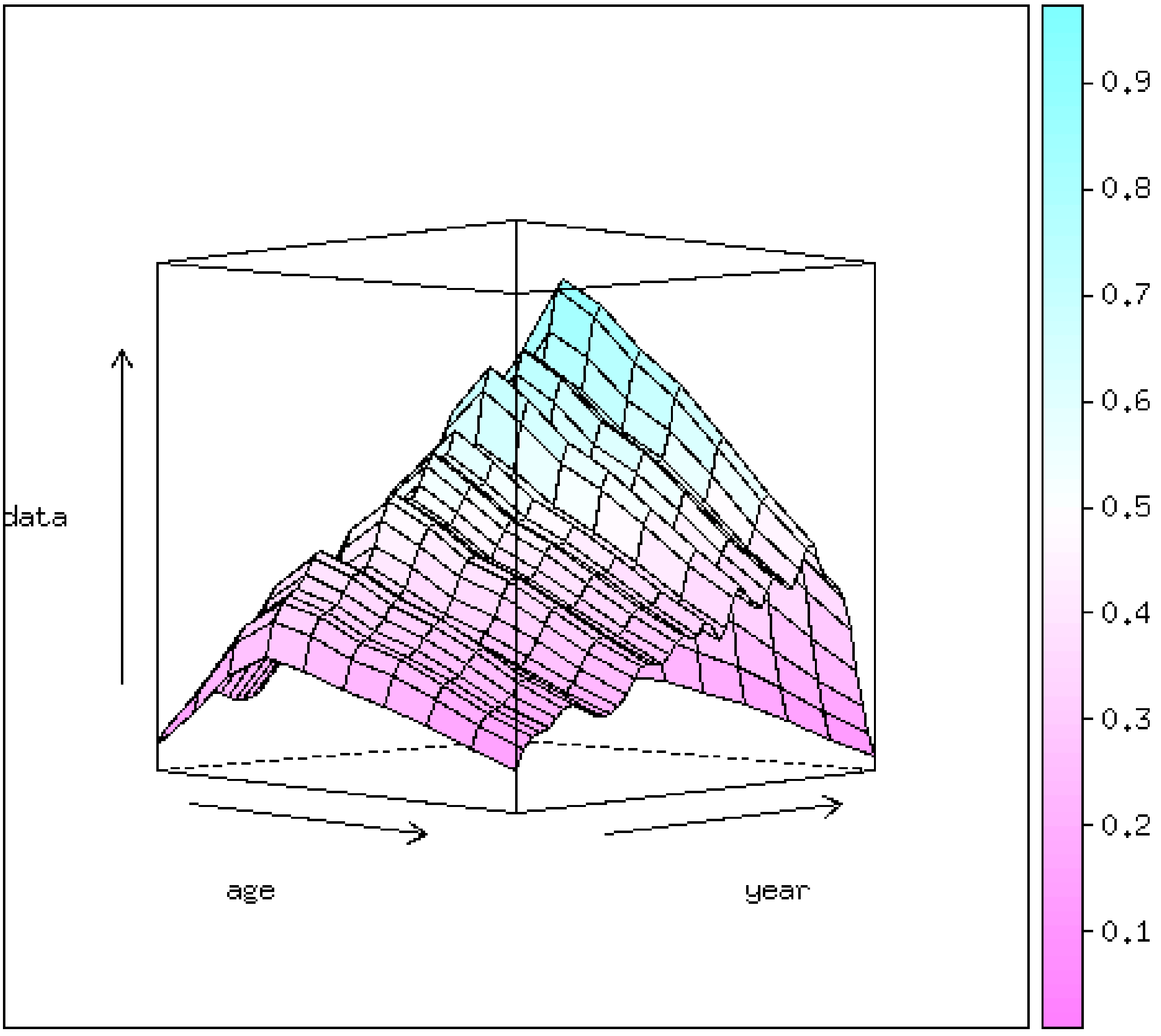
```
wireframe(data ~ age + year, data = as.data.frame(harvest(fit)), drape = TRUE,
    screen = list(x = -90, y = -45))
```



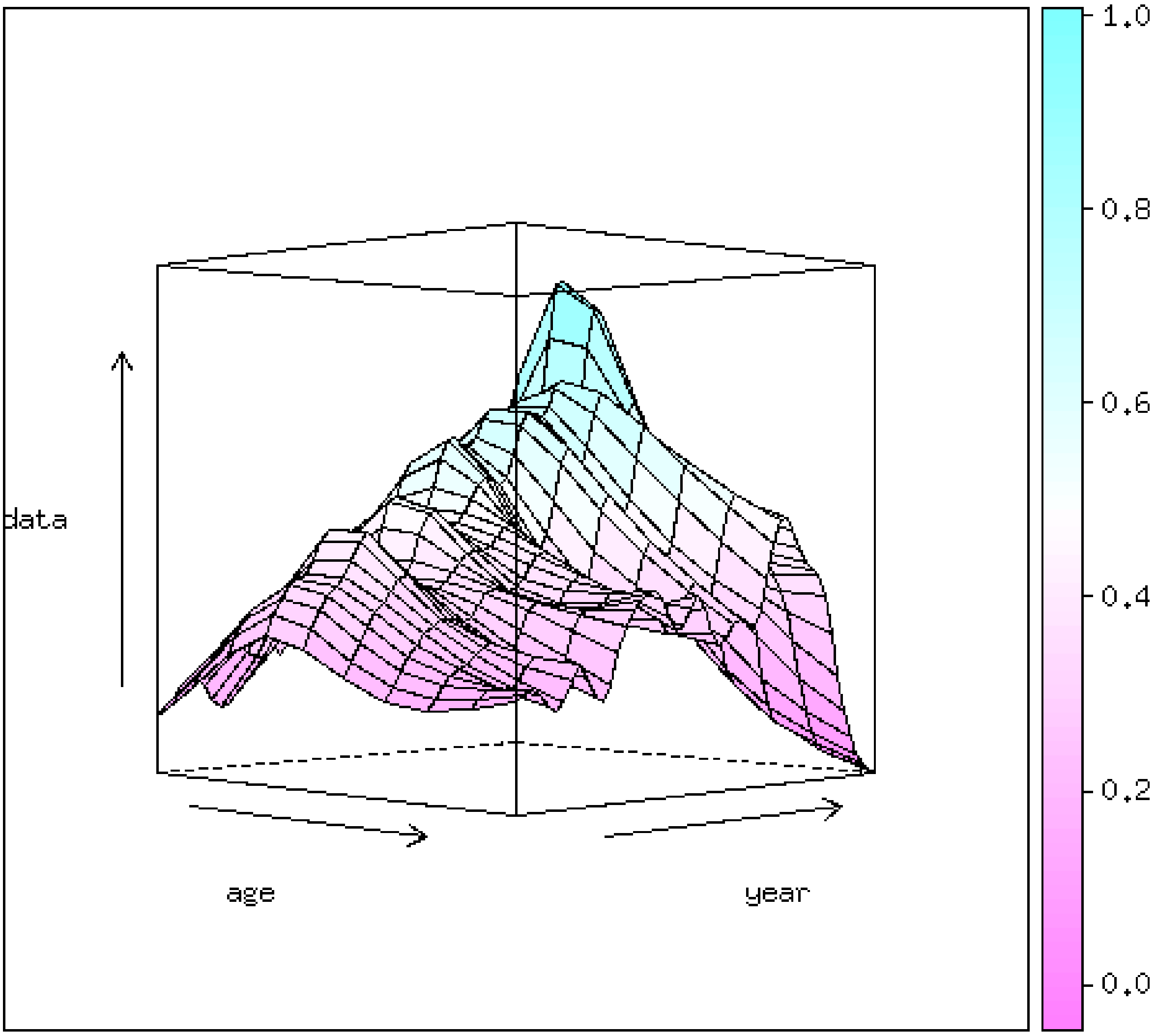

```
fmodel <- ~s(age, k = 4) + s(year, k = 20)
fit1 <- sca(ple4, ple4.indices[1], fmodel, qmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

wireframe(data ~ age + year, data = as.data.frame(harvest(fit1)), drape = TRUE,
    screen = list(x = -90, y = -45))
```



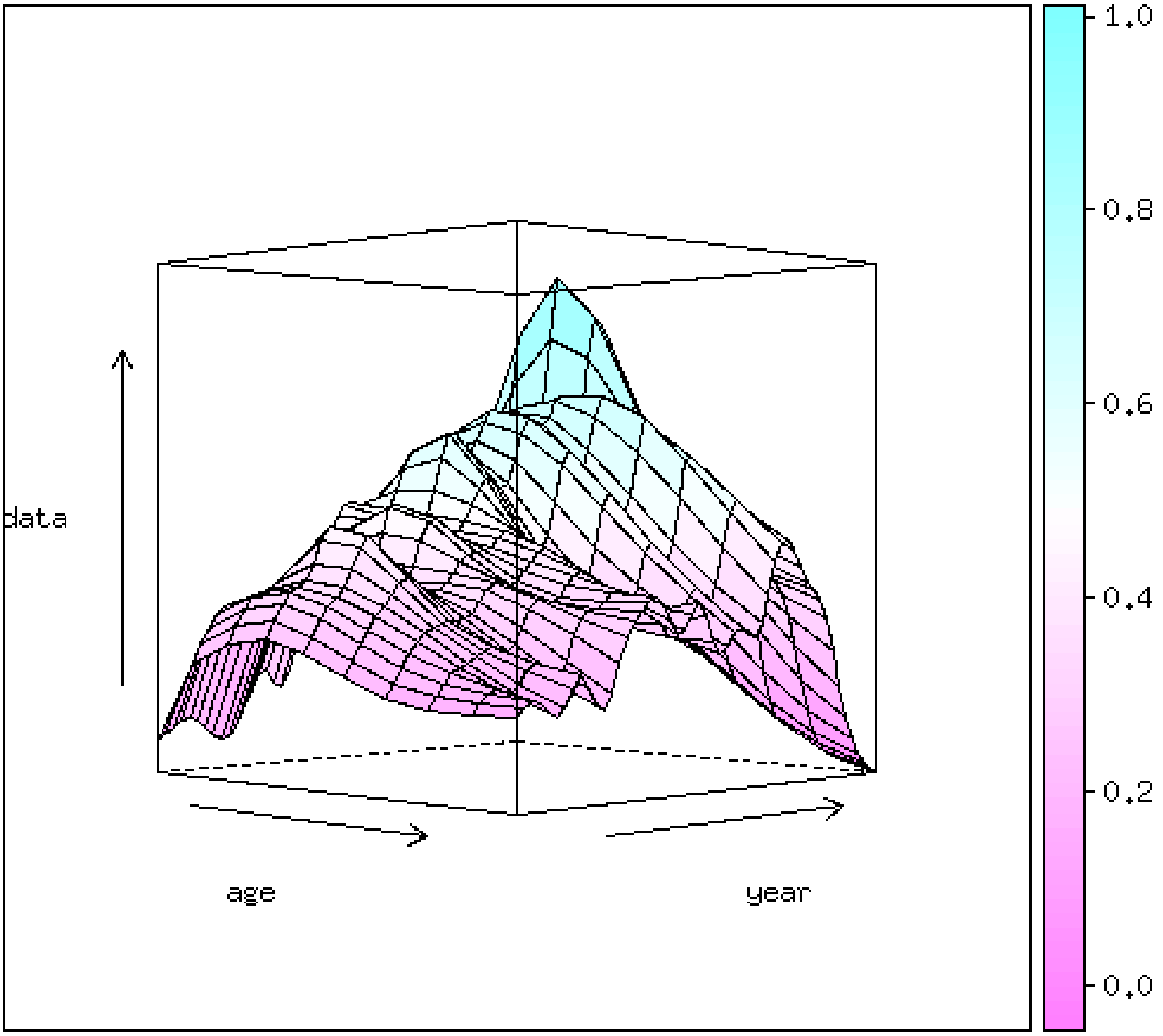

```
fmodel <- ~s(age, k = 4) + s(year, k = 20) + te(age, year, k = c(3, 3))
fit2 <- sca(ple4, ple4.indices[1], fmodel, qmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

wireframe(data ~ age + year, data = as.data.frame(harvest(fit2)), drape = TRUE,
    screen = list(x = -90, y = -45))
```




```r
fmodel <- ~te(age, year, k = c(4, 20))
fit3 <- sca(ple4, ple4.indices[1], fmodel, qmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

wireframe(data ~ age + year, data = as.data.frame(harvest(fit3)), drape = TRUE,
    screen = list(x = -90, y = -45))
```



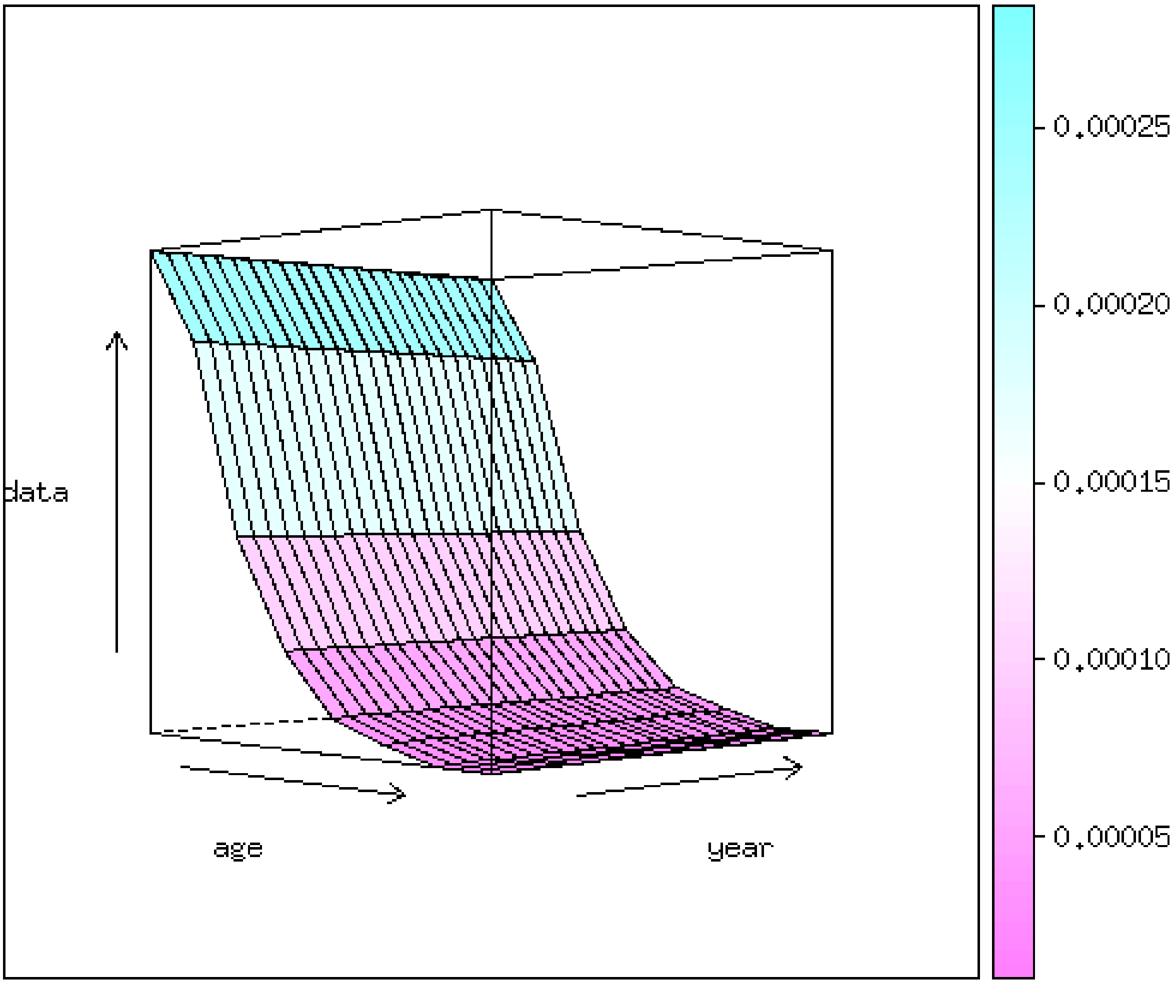

```
fmodel <- ~te(age, year, k = c(4, 20)) + s(year, k = 5, by = as.numeric(age ==
    1))
fit4 <- sca(ple4, ple4.indices[1], fmodel, qmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

wireframe(data ~ age + year, data = as.data.frame(harvest(fit4)), drape = TRUE,
    screen = list(x = -90, y = -45))
```



5.3.2 Catchability submodel

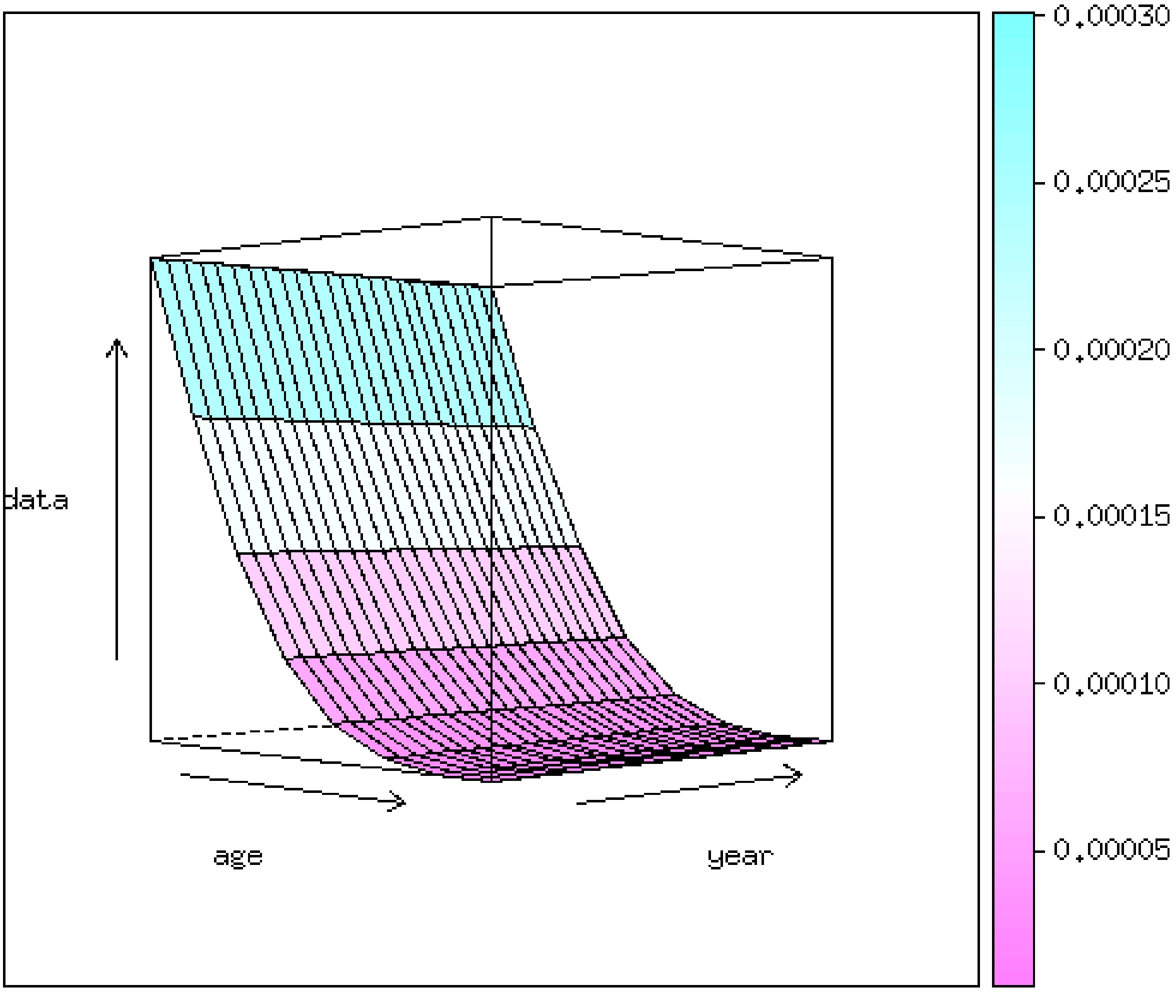
```
sfrac <- mean(range(ple4.indices[[1]])[c("startf", "endf")])
fmodel <- ~factor(age) + factor(year)


qmodel <- list(~factor(age))
fit <- sca(ple4, ple4.indices[1], fmodel, qmodel)

## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##      Predictions will be made for missing observations.

Z <- (m(ple4) + harvest(fit)) * sfrac
lst <- dimnames(fit@index[[1]])
lst$x <- stock.n(fit) * exp(-Z)
stkn <- do.call("trim", lst)
```

```
wireframe(data ~ age + year, data = as.data.frame(index(fit)[[1]]/stkn), drape = TRUE,
    screen = list(x = -90, y = -45))
```



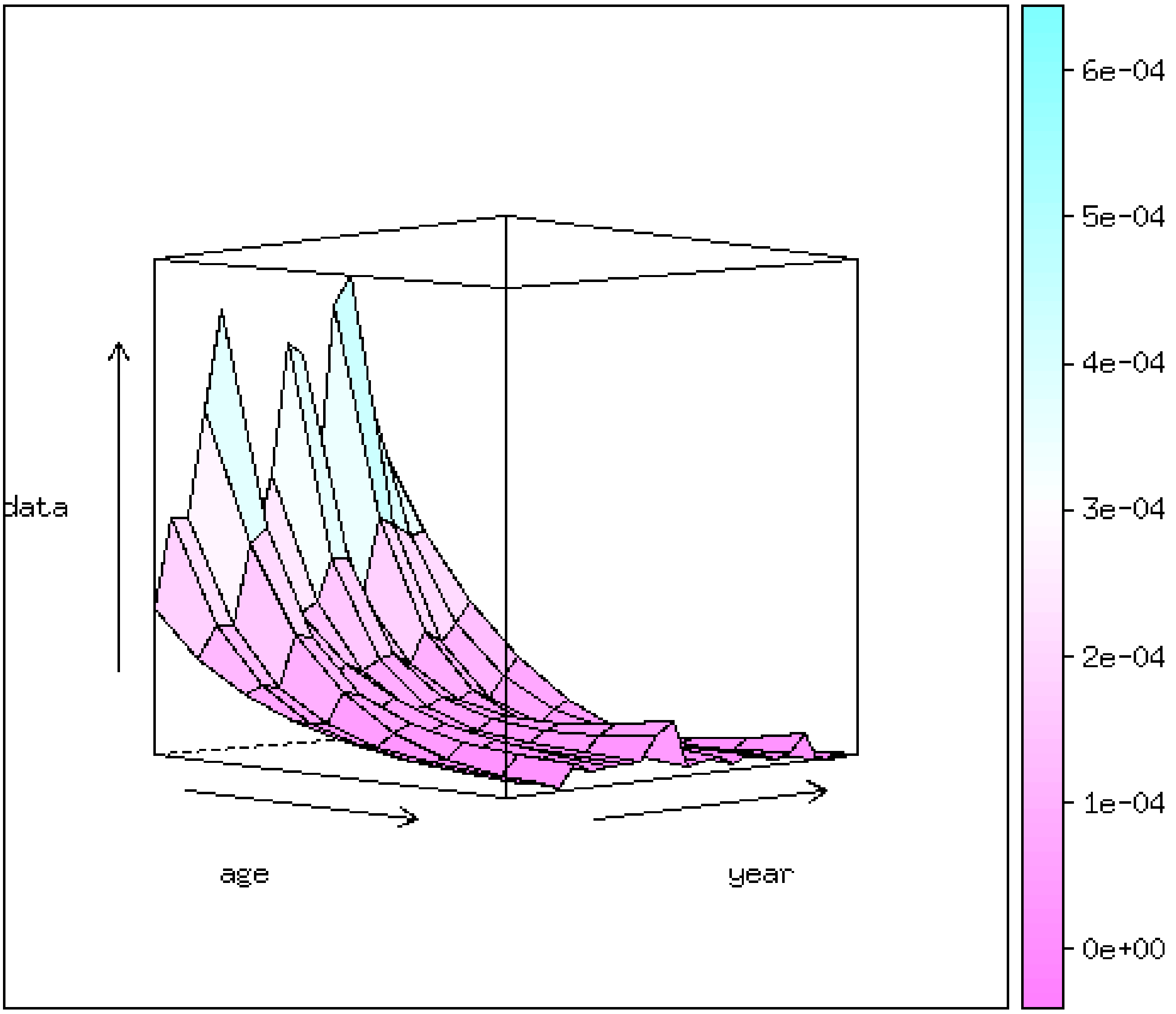

```
qmodel <- list(~s(age, k = 4))
fit1 <- sca(ple4, ple4.indices[1], fmodel, qmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

Z <- (m(ple4) + harvest(fit1)) * sfrac
lst <- dimnames(fit1@index[[1]])
lst$x <- stock.n(fit1) * exp(-Z)
stkn <- do.call("trim", lst)
```

```
wireframe(data ~ age + year, data = as.data.frame(index(fit1)[[1]]/stkn), drape = TRUE,
    screen = list(x = -90, y = -45))
```



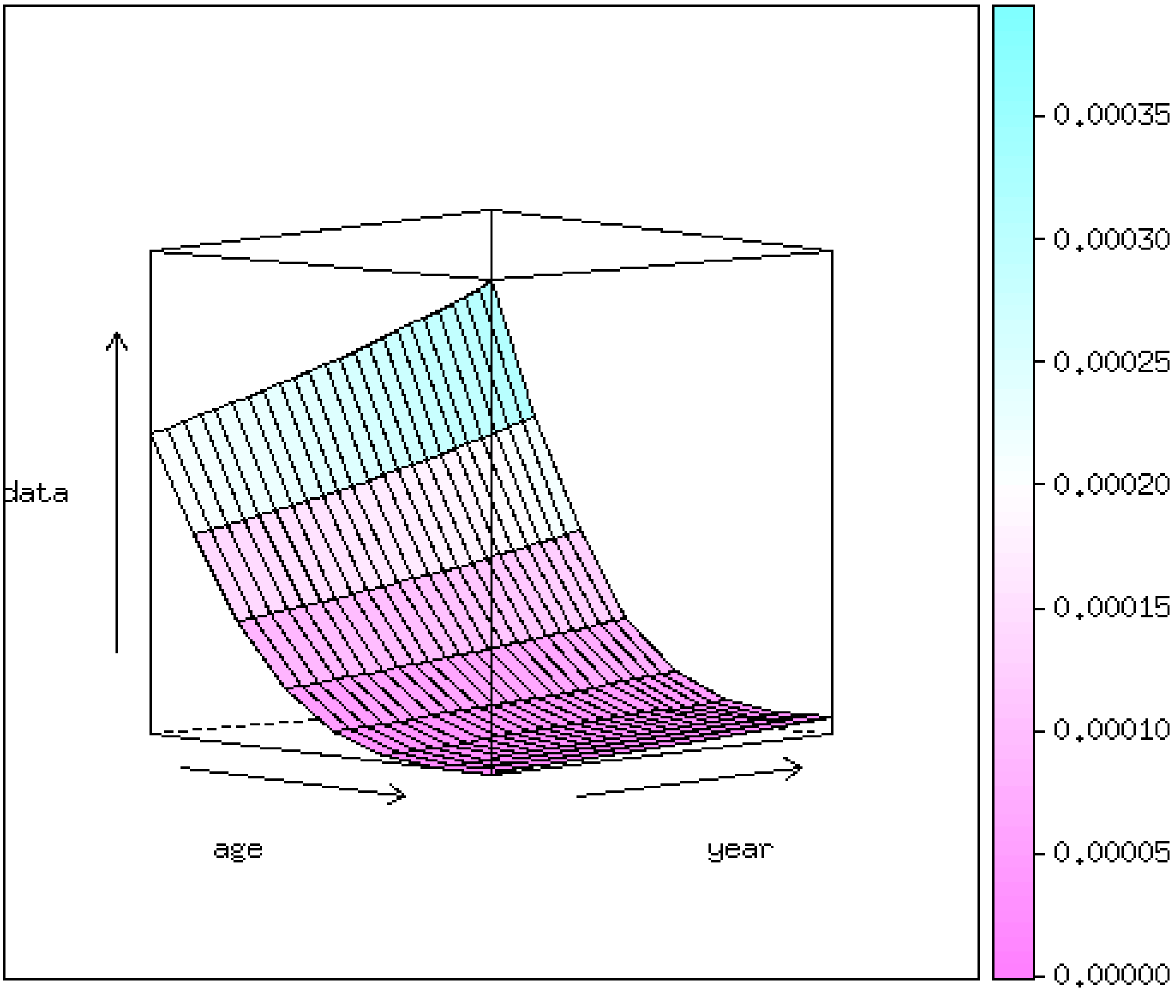

```
qmodel <- list(~te(age, year, k = c(3, 40)))
fit2 <- sca(ple4, ple4.indices[1], fmodel, qmodel)

## Note:  The following observations are treated as being missing at random:
##       fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##        Predictions will be made for missing observations.

Z <- (m(ple4) + harvest(fit2)) * sfrac
lst <- dimnames(fit2@index[[1]])
lst$x <- stock.n(fit2) * exp(-Z)
stkn <- do.call("trim", lst)
```

```
wireframe(data ~ age + year, data = as.data.frame(index(fit2)[[1]]/stkn), drape = TRUE,
    screen = list(x = -90, y = -45))
```




```
qmodel <- list(~s(age, k = 4) + year)
fit3 <- sca(ple4, ple4.indices[1], fmodel, qmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

Z <- (m(ple4) + harvest(fit3)) * sfrac
lst <- dimnames(fit3@index[[1]])
lst$x <- stock.n(fit3) * exp(-Z)
stkn <- do.call("trim", lst)
```

```
wireframe(data ~ age + year, data = as.data.frame(index(fit3)[[1]]/stkn), drape = TRUE,
    screen = list(x = -90, y = -45))
```

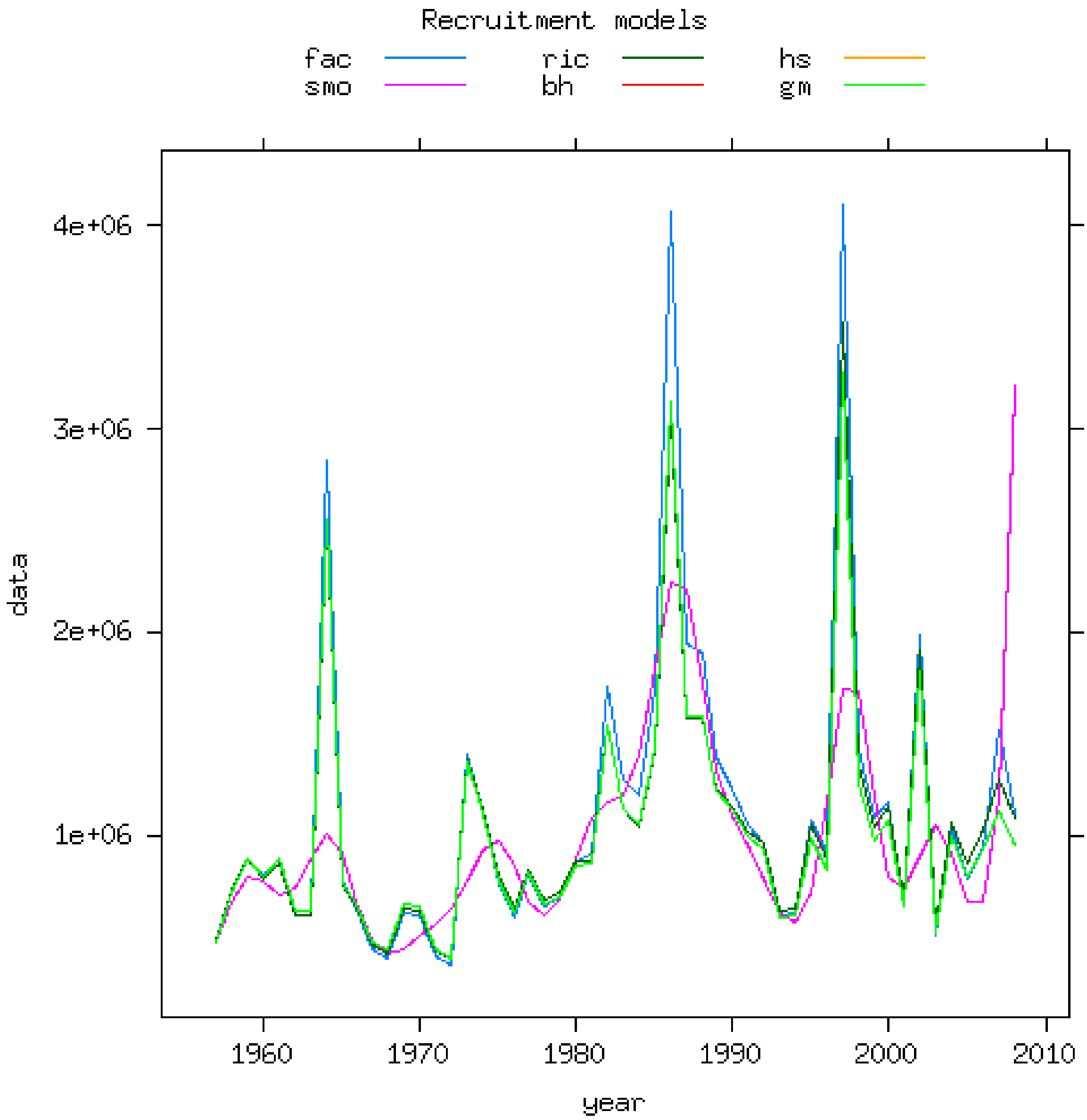


5.3.3 Stock-recruitment submodel

```
fmodel <- ~s(age, k = 4) + s(year, k = 20)
qmodel <- list(~s(age, k = 4))
```

```
srmodel <- ~factor(year)
fit <- sca(ple4, ple4.indices[1], fmodel = fmodel, qmodel = qmodel, srmodel = srmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.
```

```
srmodel <- ~s(year, k = 20)
fit1 <- sca(ple4, ple4.indices[1], fmodel, qmodel, srmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

srmodel <- ~ricker(CV = 0.05)
fit2 <- sca(ple4, ple4.indices[1], fmodel, qmodel, srmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

srmodel <- ~bevholt(CV = 0.05)
fit3 <- sca(ple4, ple4.indices[1], fmodel, qmodel, srmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

srmodel <- ~hockey(CV = 0.05)
fit4 <- sca(ple4, ple4.indices[1], fmodel, qmodel, srmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

srmodel <- ~geomean(CV = 0.05)
fit5 <- sca(ple4, ple4.indices[1], fmodel, qmodel, srmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

flqs <- FLQuants(fac = stock.n(fit)[1], smo = stock.n(fit1)[1], ric = stock.n(fit2)[1],
    bh = stock.n(fit3)[1], hs = stock.n(fit4)[1], gm = stock.n(fit5)[1])


xyplot(data ~ year, groups = qname, data = flqs, type = "l", main = "Recruitment models",
    auto.key = list(points = FALSE, lines = TRUE, columns = 3))
```



5.4 The a4aSCA method - advanced features

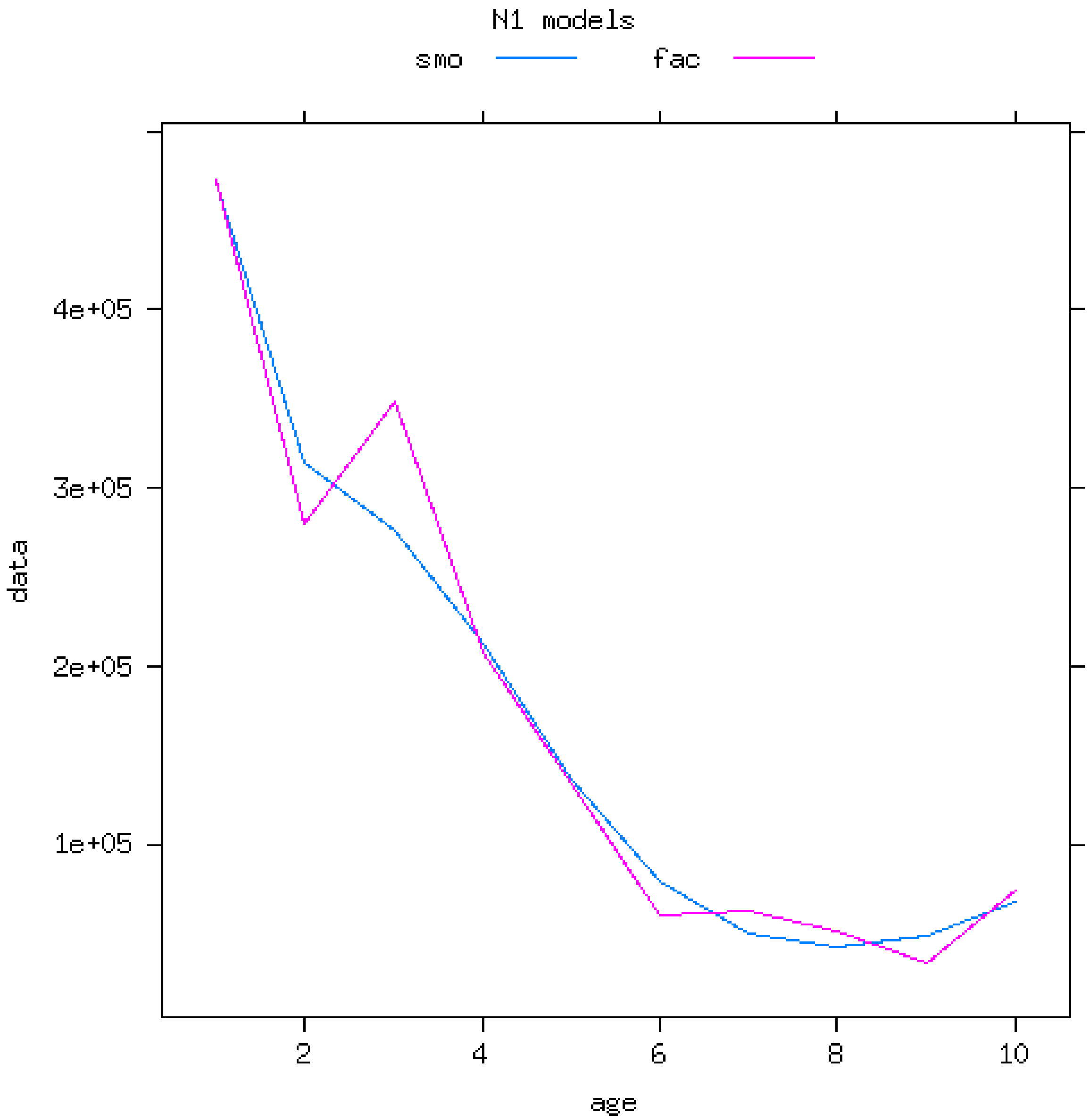
The default fit is "assessment", which means that the hessian is going to e computed by default.

```
# fmodel <- ~ s(age, k=4) + s(year, k = 20) qmodel <- list( ~ s(age, k=4) +
# year) srmodel <- ~s(year, k=20)
fit <- a4aSCA(ple4, ple4.indices[1])

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.

submodels(fit)

##   fmodel: ~s(age, k = 3) + factor(year)
##  srmodel: ~factor(year)
```

```
##  n1model: ~factor(age)
##   qmodel:
##     BTS-Isis: ~1
##   vmodel:
##     catch:    ~s(age, k = 3)
##     BTS-Isis: ~1
```

5.4.1 N1 model

```
n1model <- ~s(age, k = 4)
fit1 <- a4aSCA(ple4, ple4.indices[1], n1model = n1model)

## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##      Predictions will be made for missing observations.

flqs <- FLQuants(smo = stock.n(fit1)[, 1], fac = stock.n(fit)[, 1])


xyplot(data ~ age, groups = qname, data = flqs, type = "l", main = "N1 models",
    auto.key = list(points = FALSE, lines = TRUE, columns = 2))
```



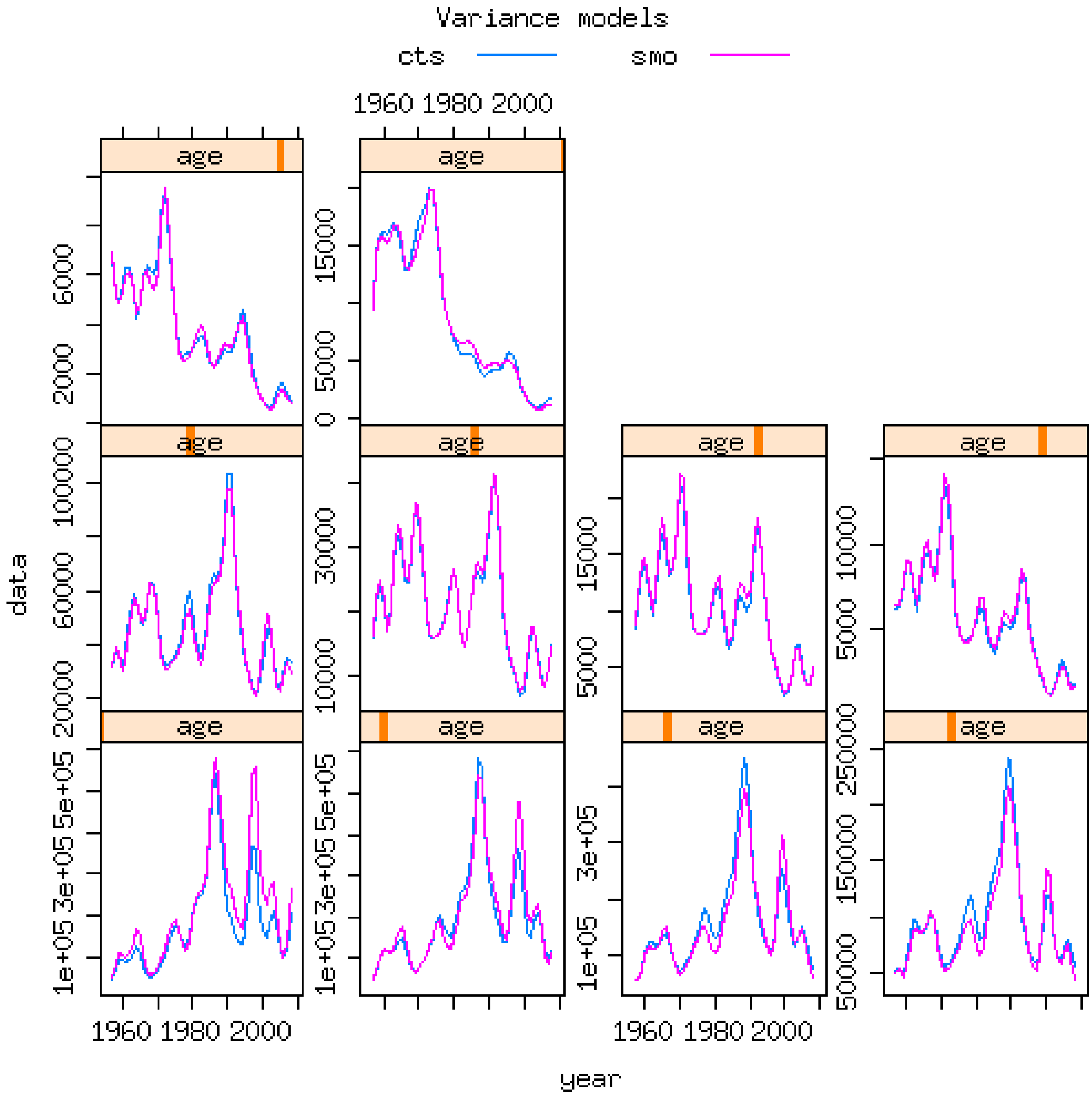
5.4.2 Variance model

One important subject related with fisheries data used for input to stock assessment models is the shape of the variance of the data. It's quite common to have more precision on the most represented ages and less precision on the less frequent ages. Due to the fact that the last do not show so often on the auction markets, on the fishing operations or on survey samples.

By default the model assumes constant variance over time and ages (1 model) but it can use other models specified by the user. This feature requires a call to the a4aInternal method, which gives more options than the a4a method, which in fact is a wrapper.

```
vmodel <- list(~1, ~1)
fit1 <- a4aSCA(ple4, ple4.indices[1], vmodel = vmodel)

## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##      Predictions will be made for missing observations.
```

```
vmodel <- list(~s(age, k = 4), ~1)
fit2 <- a4aSCA(ple4, ple4.indices[1], vmodel = vmodel)

## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##      Predictions will be made for missing observations.

flqs <- FLQuants(cts = catch.n(fit1), smo = catch.n(fit2))


xyplot(data ~ year | age, groups = qname, data = flqs, type = "l", main = "Variance models",
    scales = list(y = list(relation = "free")), auto.key = list(points = FALSE,
        lines = TRUE, columns = 2))
```

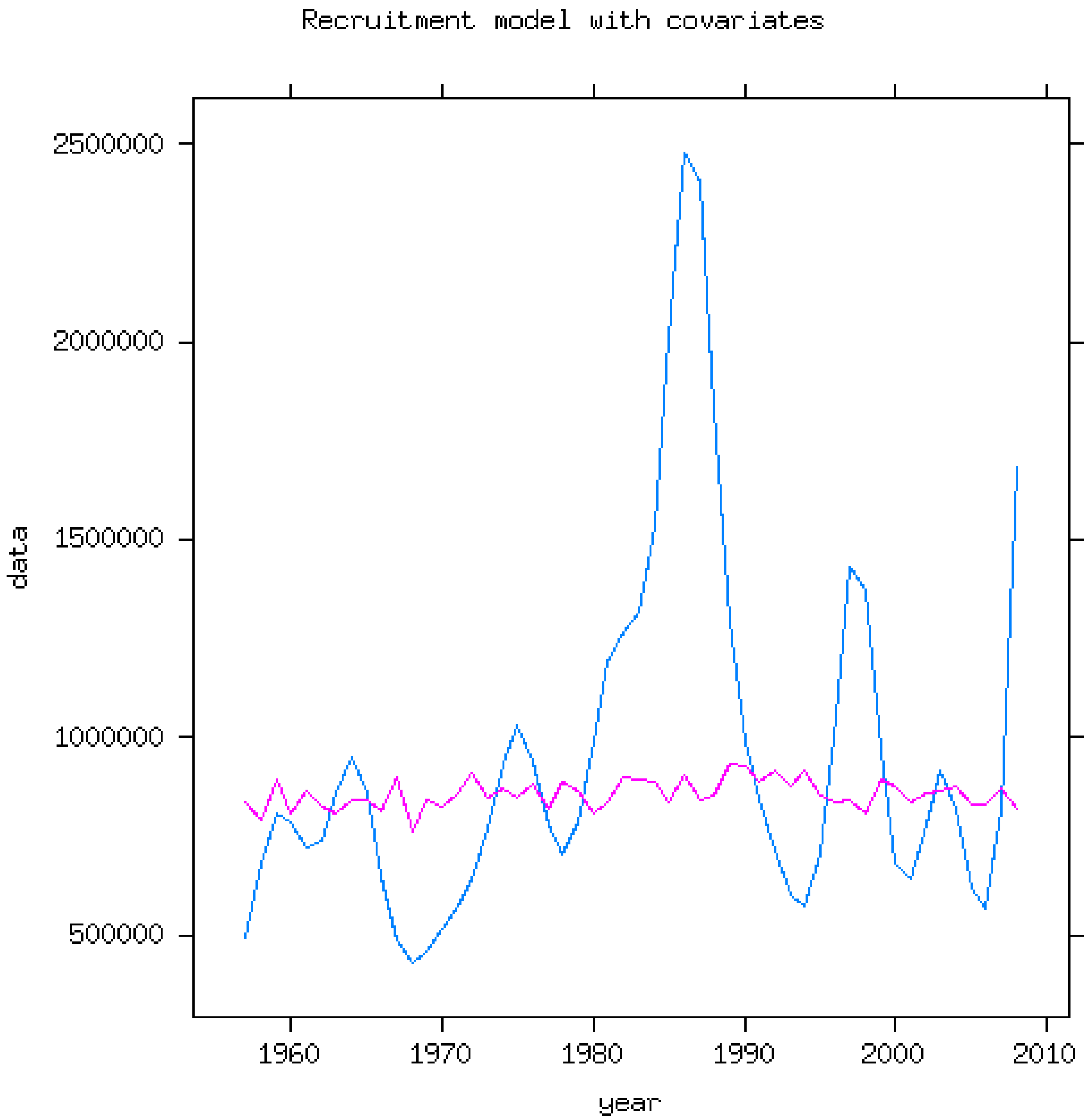


5.4.3 Working with covariates

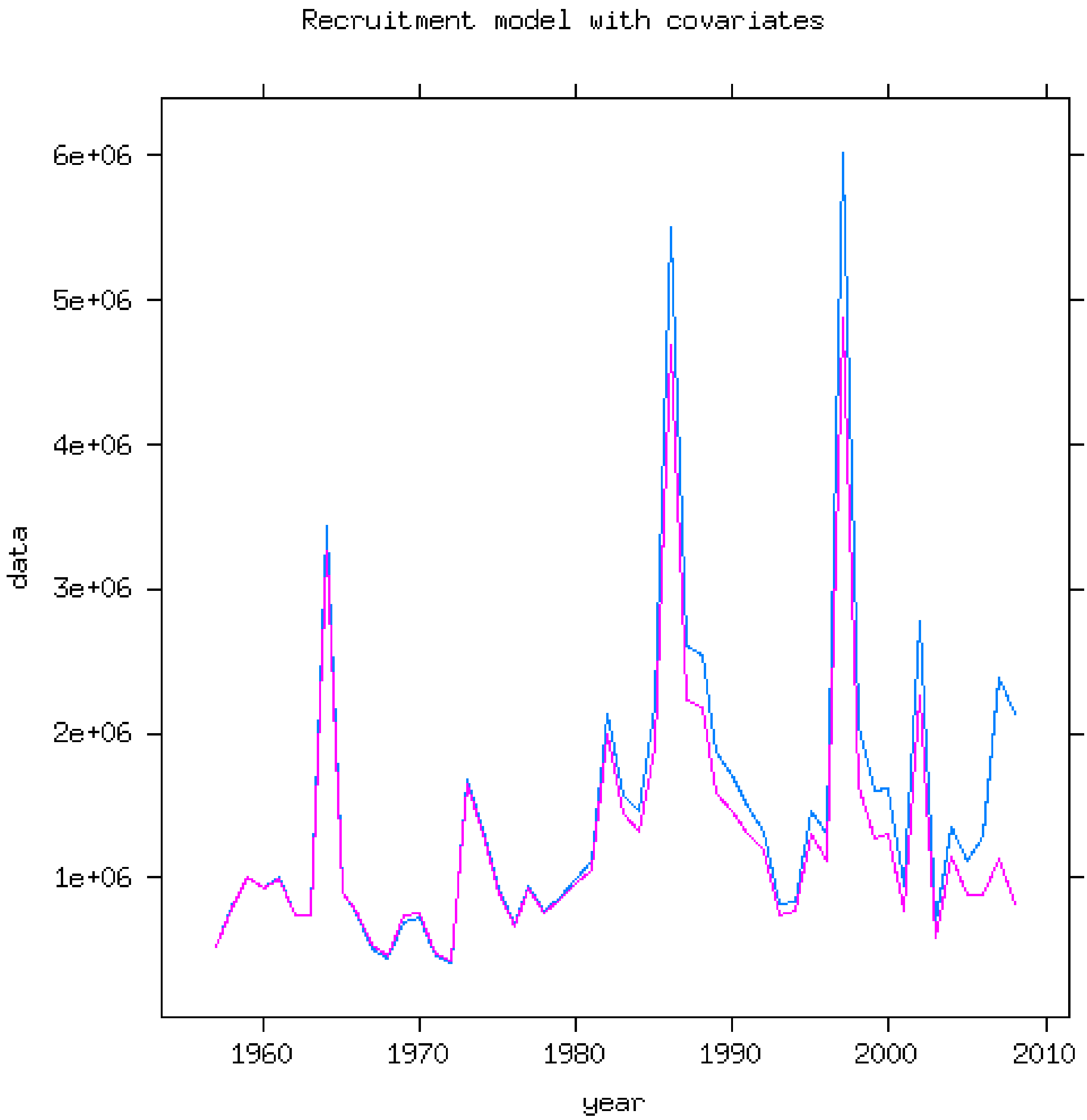
```r
nao <- read.table("http://www.cdc.noaa.gov/data/correlation/nao.data", skip = 1,
    nrow = 62, na.strings = "-99.90")
dnms <- list(quant = "nao", year = 1948:2009, unit = "unique", season = 1:12,
    area = "unique")
nao <- FLQuant(unlist(nao[, -1]), dimnames = dnms, units = "nao")
nao <- seasonMeans(trim(nao, year = dimnames(stock.n(ple4))$year))
nao <- as.numeric(nao)
```

```r
srmodel <- ~nao
fit2 <- a4aSCA(ple4, ple4.indices[1], qmodel = list(~s(age, k = 4)), srmodel = srmodel)

## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##      Predictions will be made for missing observations.

flqs <- FLQuants(fac = stock.n(fit)[1], cvar = stock.n(fit2)[1])
```

```r
xyplot(data ~ year, groups = qname, data = flqs, type = "l", main = "Recruitment model with covariates"
```




```
srmodel <- ~ricker(a = ~nao, CV = 0.1)
fit3 <- a4aSCA(ple4, ple4.indices[1], qmodel = list(~s(age, k = 4)), srmodel = srmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.

flqs <- FLQuants(fac = stock.n(fit)[1], cvar = stock.n(fit3)[1])


xyplot(data ~ year, groups = qname, data = flqs, type = "l", main = "Recruitment model with covariates"
```



5.4.4 Assessing ADMB files

To inspect the ADMB files the user must specify the working dir and all files will be left there.

```
fit1 <- a4aSCA(stk, idx, wkdir = "mytest")

## Error:  error in evaluating the argument 'indices' in selecting a method for function
'a4aSCA': Error:  object 'idx' not found
```

5.5 Predict and simulate

```
# fmodel <- ~ s(age, k=4) + s(year, k = 20) qmodel <- list( ~ s(age, k=4) +
# year) srmodel <- ~s(year, k=20)
fit <- a4aSCA(ple4, ple4.indices[1])

## Note:  The following observations are treated as being missing at random:
##     fleet year age
```

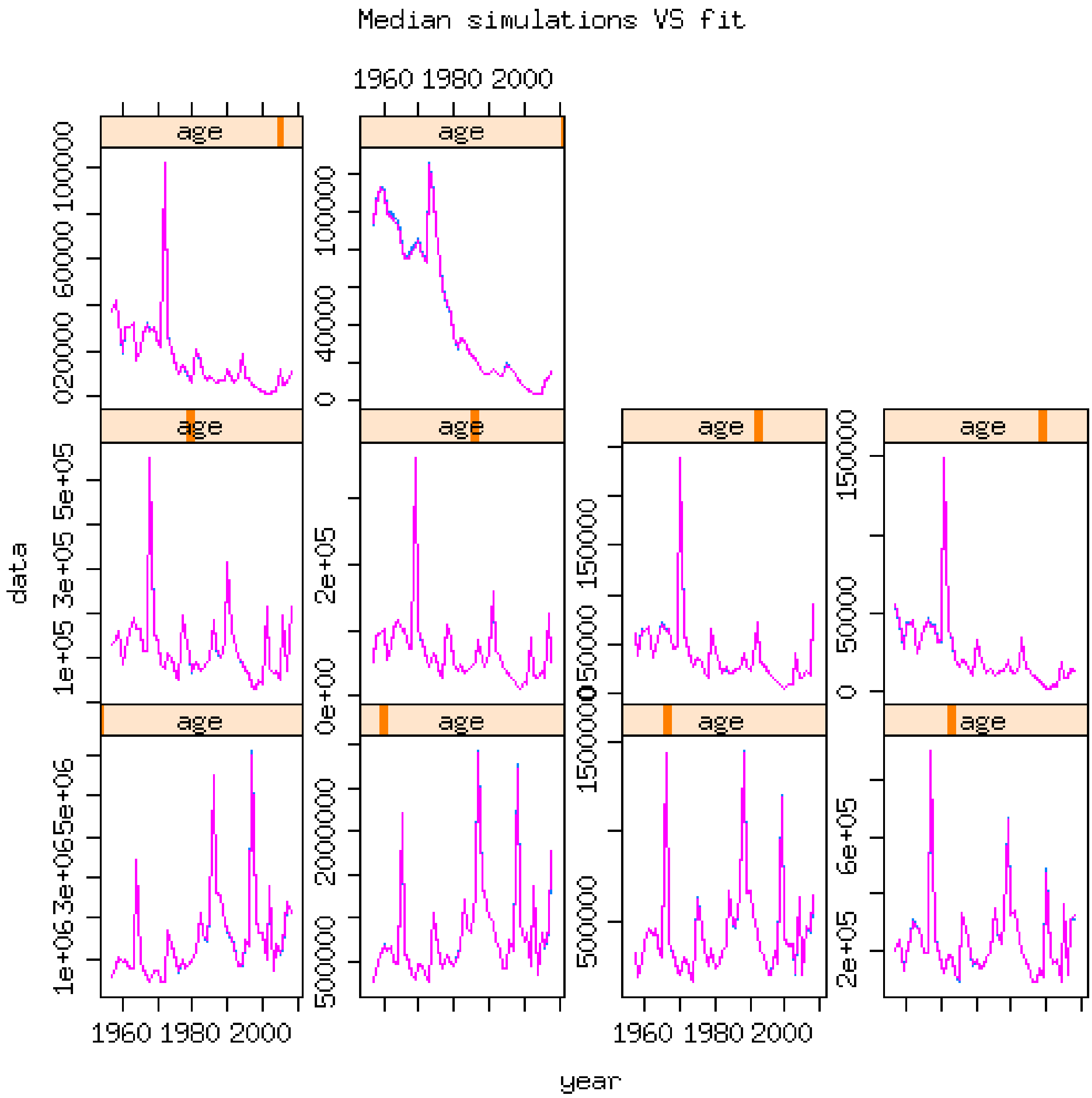
```
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##      Predictions will be made for missing observations.
```

5.5.1 Predict

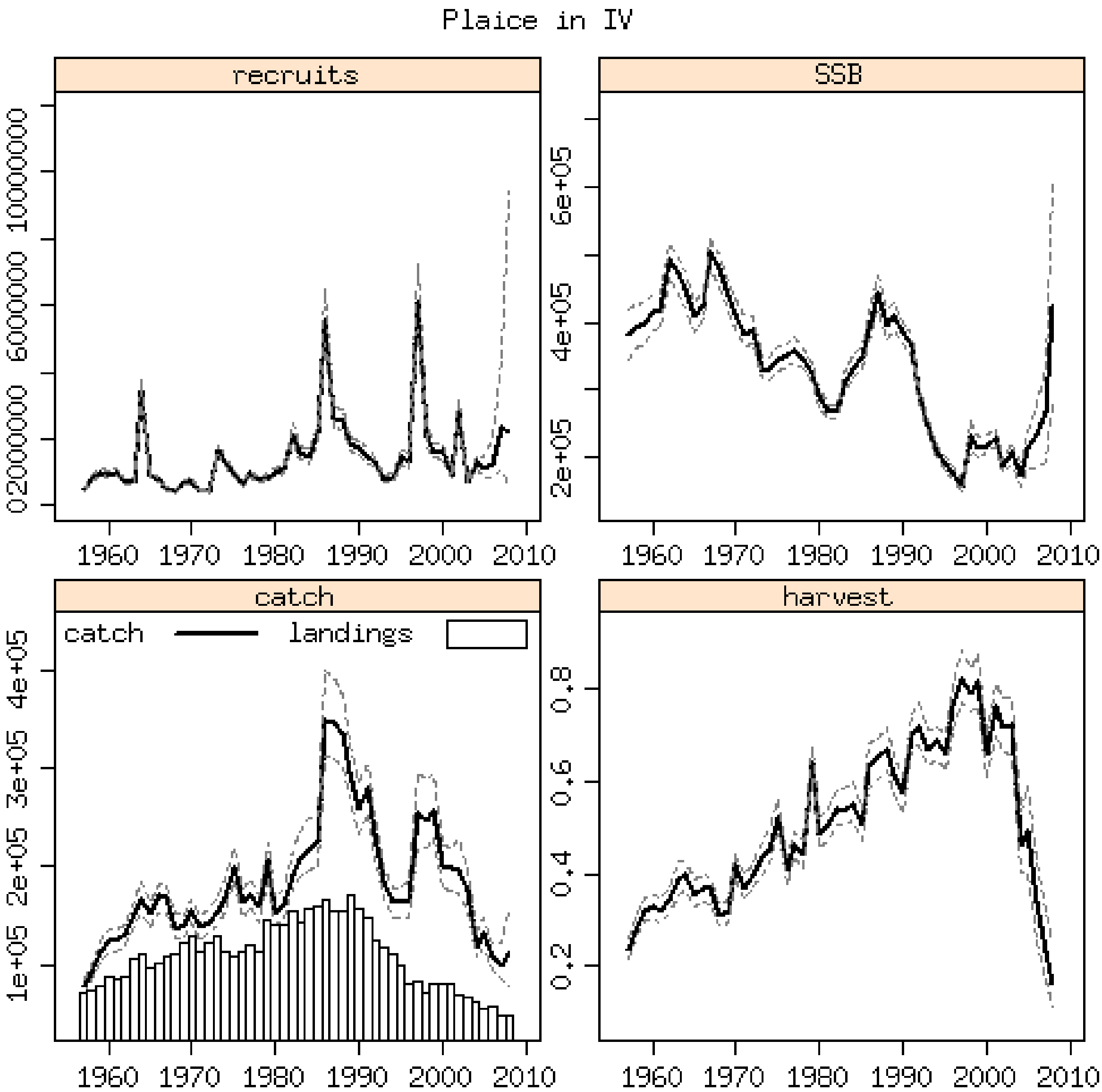
```
fit.pred <- predict(fit)
lapply(fit.pred, names)

## $stkmodel
## [1] "harvest" "rec"     "ny1"
##
## $qmodel
## [1] "BTS-Isis"
##
## $vmodel
## [1] "catch"    "BTS-Isis"
```

5.5.2 simulate

```
fits <- simulate(fit, 100)
flqs <- FLQuants(sim = iterMedians(stock.n(fits)), det = stock.n(fit))


xyplot(data ~ year | age, groups = qname, data = flqs, type = "l", main = "Median simulations VS fit",
    scales = list(y = list(relation = "free")))
```




```
stks <- ple4 + fits
plot(stks)

## Warning:  invalid factor level, NA generated
```



5.6 Geeky stuff

```
# fmodel <- ~ s(age, k=4) + s(year, k = 20) qmodel <- list( ~ s(age, k=4) +
# year) srmodel <- ~s(year, k=20)
vmodel <- list(~1, ~1)
fit <- a4aSCA(ple4, ple4.indices[1], vmodel = vmodel)

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.
```

5.6.1 External weigthing of likelihood components

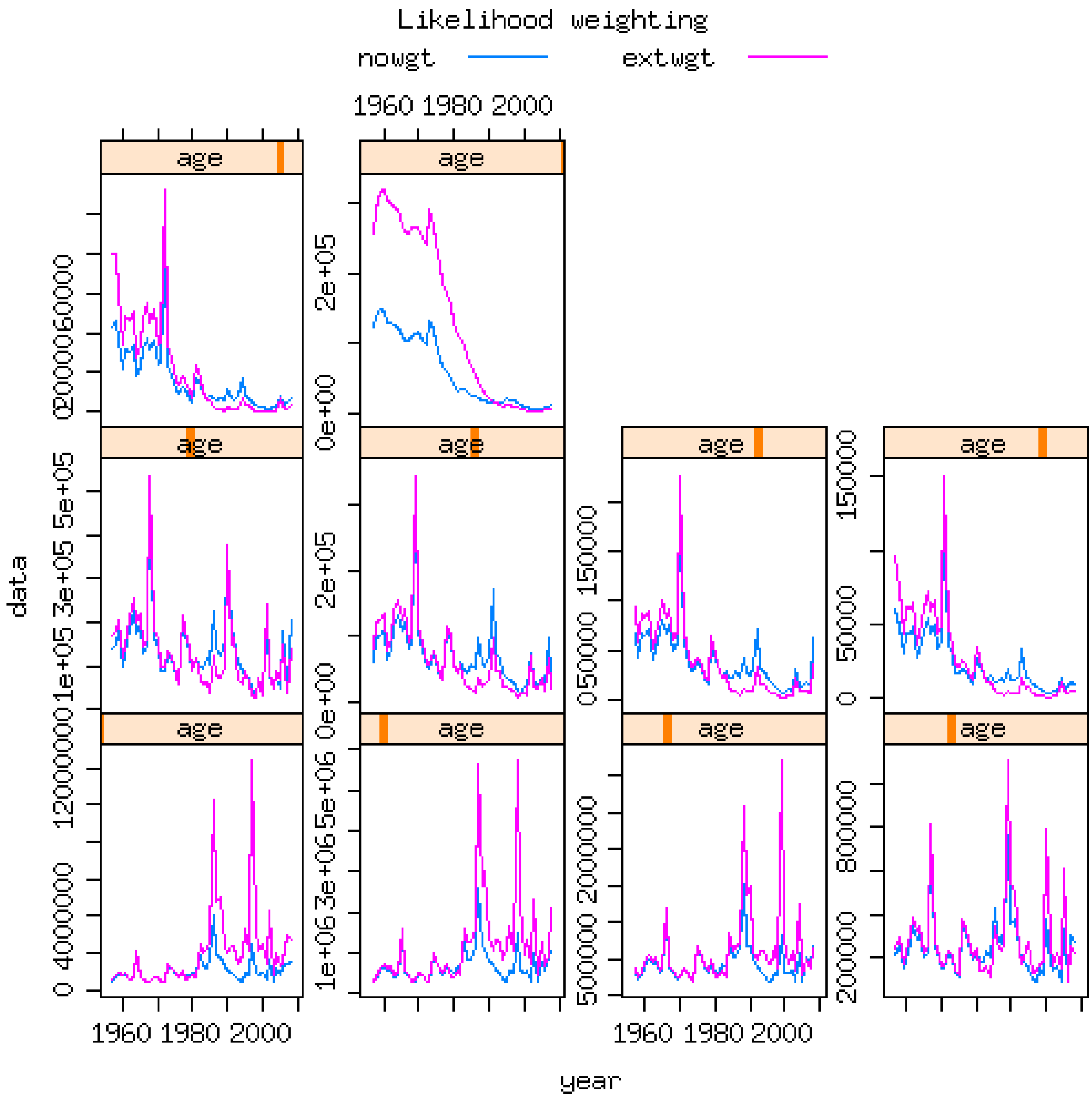
By default the likelihood components are weighted using inverse variance of the parameters estimates. However the user may change this weights by setting the variance of the input parameters, which is done by adding a variance matrix to the catch.n and index.n slots of the stock and index objects.

```
stk <- ple4
idx <- ple4.indices[1]
# variance of observed catches
varslt <- catch.n(stk)
varslt[] <- 0.4
catch.n(stk) <- FLQuantDistr(catch.n(stk), varslt)
# variance of observed indices
varslt <- index(idx[[1]])
varslt[] <- 0.1
index.var(idx[[1]]) <- varslt
# run
fit1 <- a4aSCA(stk, idx, vmodel = vmodel)
```

```
## Note:  Provided variances will be used to weight observations.
## Weighting assumes variances are on the log scale or equivalently log(CV^2 + 1).
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##      Predictions will be made for missing observations.
```

```
flqs <- FLQuants(nowgt = stock.n(fit), extwgt = stock.n(fit1))
```

```
xyplot(data ~ year | age, groups = qname, data = flqs, type = "l", main = "Likelihood weighting",
    scales = list(y = list(relation = "free")), auto.key = list(points = FALSE,
        lines = TRUE, columns = 2))
```



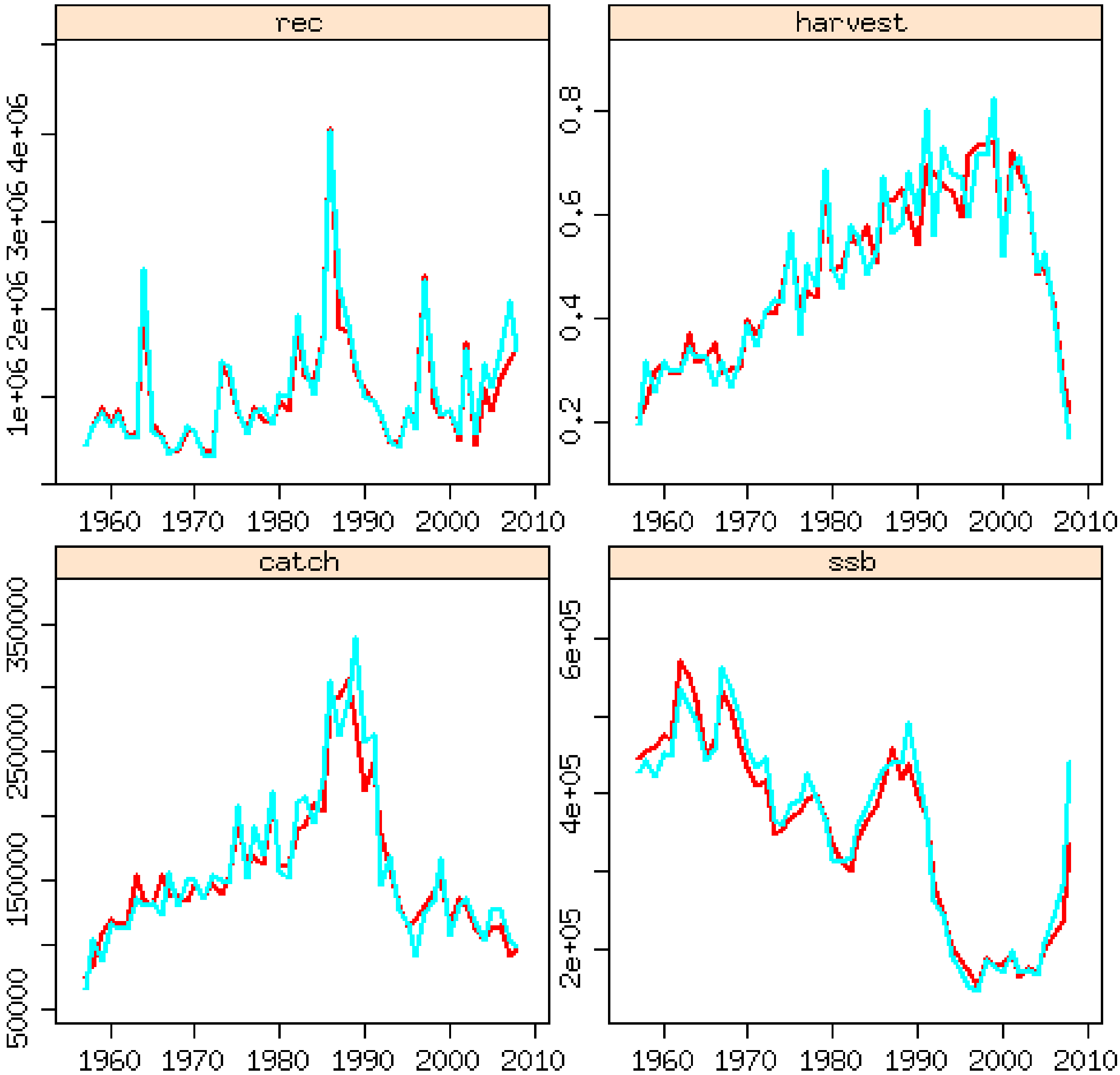
5.6.2 WCSAM exercise - replicating itself

```
fits <- simulate(fit, 25)
stks <- ple4 + fits
fit1 <- a4aSCA(stks, ple4.indices[1], fit = "MP")

## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997   1
## BTS-Isis 1997   2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
```

```
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##      fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
```

```
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
## Note:  The following observations are treated as being missing at random:
##     fleet year age
## BTS-Isis 1997  1
## BTS-Isis 1997  2
##       Predictions will be made for missing observations.
```

```
plot(FLStocks(sim = ple4 + fit1, orig = ple4 + fit))
```

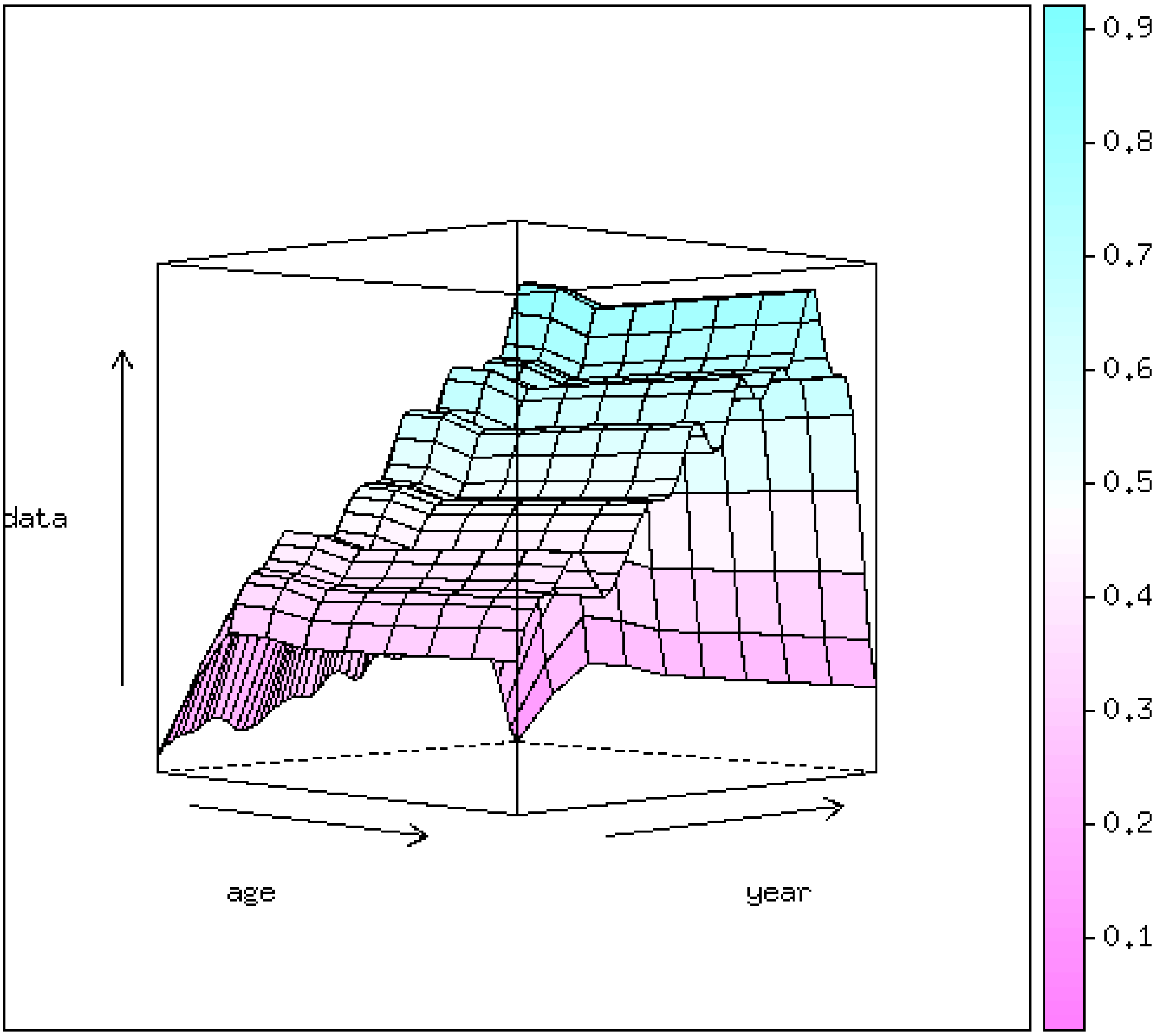


5.6.3 More models

```
# constant fishing mortality for ages older than 5
fmodel = ~s(replace(age, age > 5, 5), k = 4) + s(year, k = 20)
fit <- sca(ple4, ple4.indices, fmodel = fmodel)

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##          SNS 1997   1
##          SNS 1997   2
##          SNS 2003   1
```

```
##          SNS 2003   2
##          SNS 2003   3
##      Predictions will be made for missing observations.
```

```
wireframe(data ~ age + year, data = as.data.frame(harvest(fit)), drape = TRUE,
    screen = list(x = -90, y = -45))
```

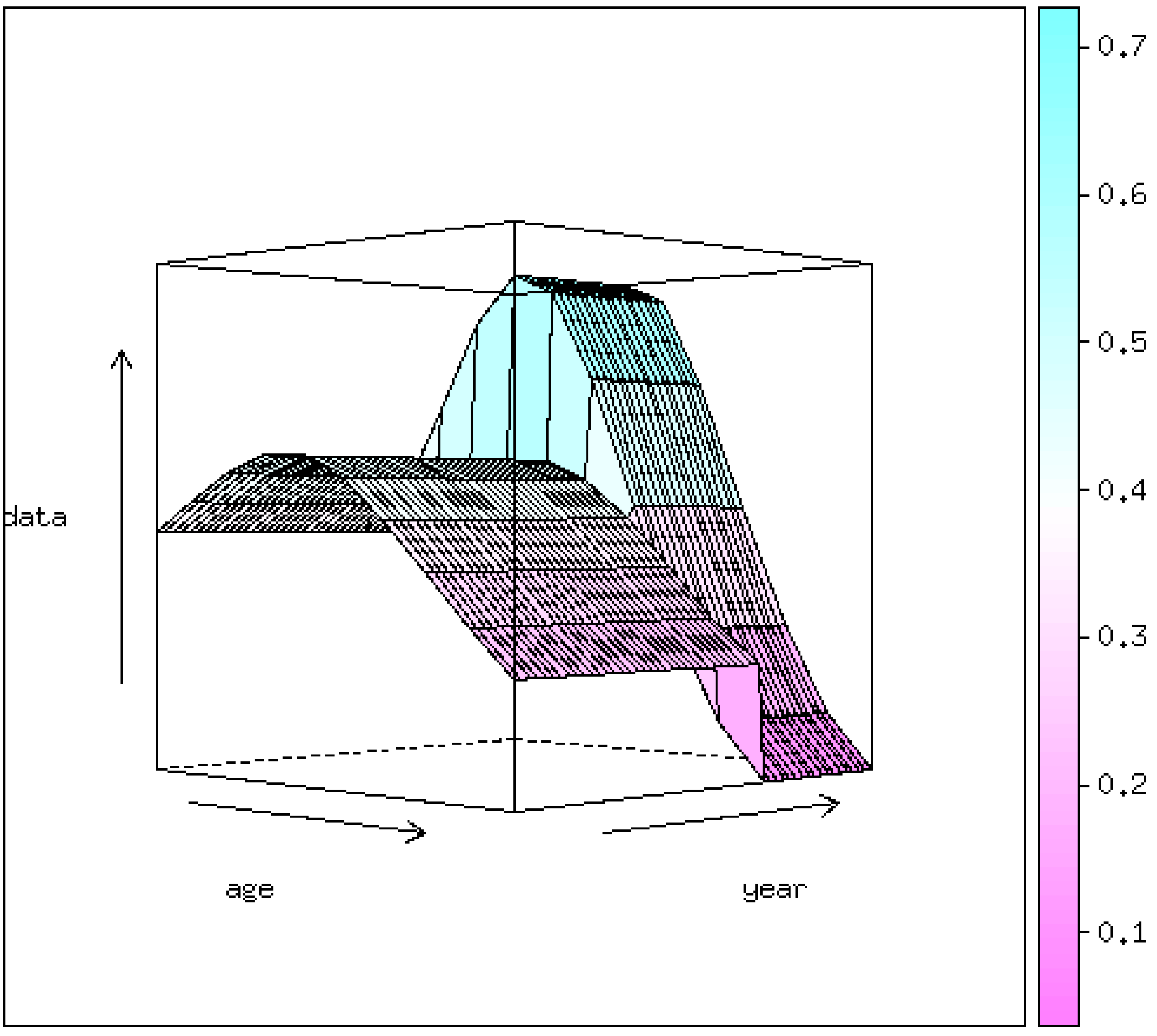



```
# the same model for two periods
fmodel = ~s(age, k = 3, by = breakpts(year, 1990))
fit <- sca(ple4, ple4.indices, fmodel = fmodel)
```

```
## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
```

```
##          SNS 1997   1
##          SNS 1997   2
##          SNS 2003   1
##          SNS 2003   2
##          SNS 2003   3
##       Predictions will be made for missing observations.
```

```
wireframe(data ~ age + year, data = as.data.frame(harvest(fit)), drape = TRUE,
    screen = list(x = -90, y = -45))
```

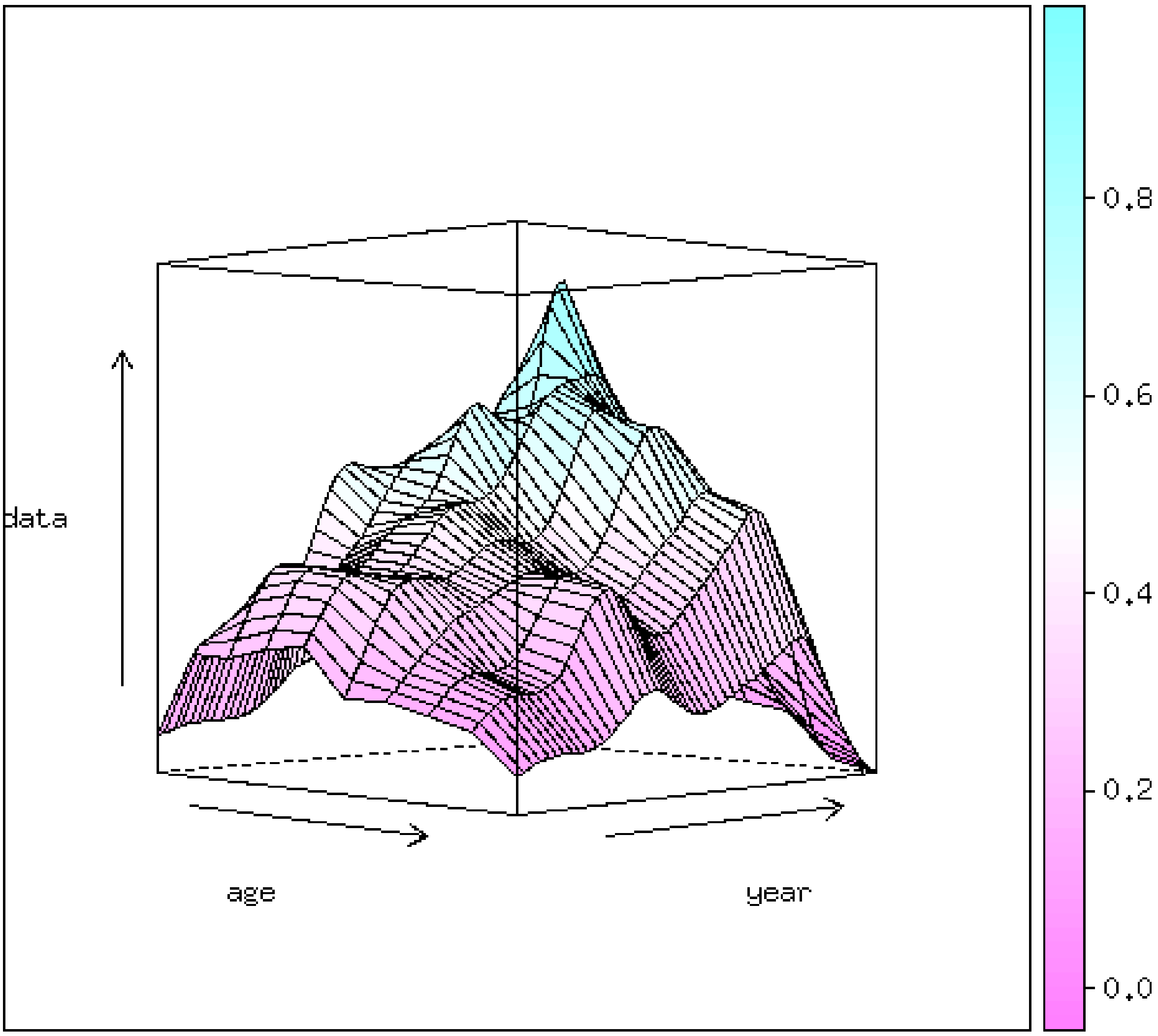



```
# smoother for each age
fmodel <- ~factor(age) + s(year, k = 10, by = breakpts(age, c(2:8)))
fit <- sca(ple4, ple4.indices, fmodel = fmodel)
```

```
## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997   1
```

```
##      BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##       Predictions will be made for missing observations.
```

```
wireframe(data ~ age + year, data = as.data.frame(harvest(fit)), drape = TRUE,
    screen = list(x = -90, y = -45))
```



5.6.4 Propagate M uncertainty

```
fit <- sca(ple4, ple4.indices)

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
##   BTS-Tridens 1997   1
##   BTS-Tridens 1997   2
##           SNS 1997   1
##           SNS 1997   2
##           SNS 2003   1
##           SNS 2003   2
##           SNS 2003   3
##       Predictions will be made for missing observations.

nits <- 25

shape2 <- FLModelSim(model = ~exp(-age - 0.5))
level4 <- FLModelSim(model = ~k^0.66 * t^0.57, params = FLPar(k = 0.4, t = 10),
    vcov = matrix(c(0.002, 0.01, 0.01, 1), ncol = 2))

trend4 <- FLModelSim(model = ~b, params = FLPar(b = 0.5), vcov = matrix(0.02))
m4 <- a4aM(shape = shape2, level = level4, trend = trend4)
m4 <- mvrnorm(nits, m4)
range(m4)[] <- range(ple4)[]
range(m4)[c("minmbar", "maxmbar")] <- c(1, 1)
flq <- m(m4)[]
quant(flq) <- "age"
stk <- propagate(ple4, nits)
m(stk) <- flq

fit1 <- sca(stk, ple4.indices)

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
##   BTS-Tridens 1997   1
##   BTS-Tridens 1997   2
##           SNS 1997   1
##           SNS 1997   2
##           SNS 2003   1
##           SNS 2003   2
##           SNS 2003   3
##       Predictions will be made for missing observations.

## Warning:  incompatible units of measurements in FLQuant objects:  f + NA

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
##   BTS-Tridens 1997   1
##   BTS-Tridens 1997   2
##           SNS 1997   1
##           SNS 1997   2
##           SNS 2003   1
##           SNS 2003   2
```

```
##         SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997   1
##     BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##         SNS 1997   1
##         SNS 1997   2
##         SNS 2003   1
##         SNS 2003   2
##         SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997   1
##     BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##         SNS 1997   1
##         SNS 1997   2
##         SNS 2003   1
##         SNS 2003   2
##         SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997   1
##     BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##         SNS 1997   1
##         SNS 1997   2
##         SNS 2003   1
##         SNS 2003   2
##         SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997   1
##     BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##         SNS 1997   1
##         SNS 1997   2
##         SNS 2003   1
##         SNS 2003   2
##         SNS 2003   3
##      Predictions will be made for missing observations.
```

```
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997  1
##     BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##         SNS 1997  1
##         SNS 1997  2
##         SNS 2003  1
##         SNS 2003  2
##         SNS 2003  3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997  1
##     BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##         SNS 1997  1
##         SNS 1997  2
##         SNS 2003  1
##         SNS 2003  2
##         SNS 2003  3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997  1
##     BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##         SNS 1997  1
##         SNS 1997  2
##         SNS 2003  1
##         SNS 2003  2
##         SNS 2003  3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997  1
##     BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##         SNS 1997  1
##         SNS 1997  2
##         SNS 2003  1
##         SNS 2003  2
##         SNS 2003  3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
```

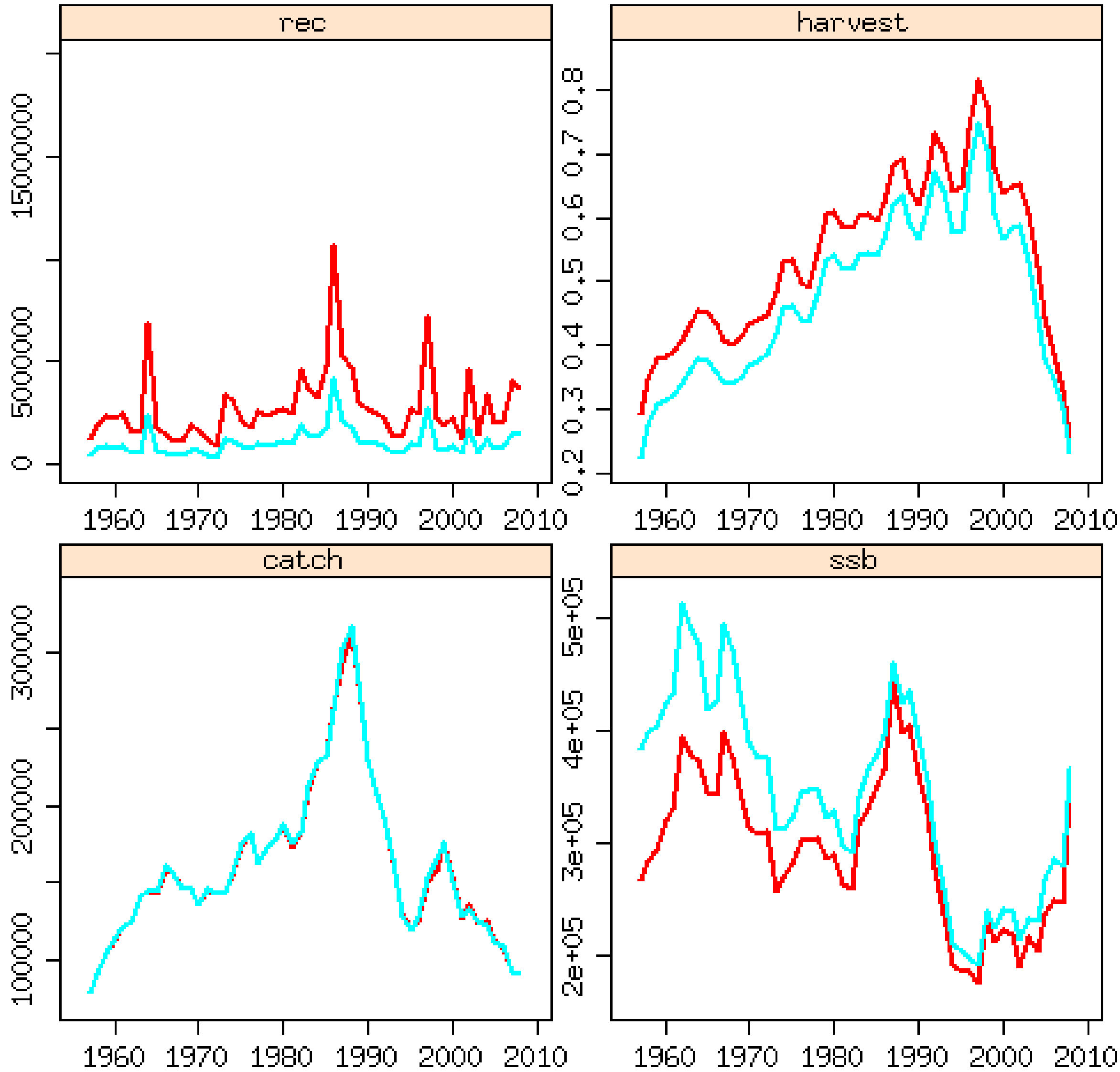
```
##      BTS-Isis 1997  1
##      BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##       Predictions will be made for missing observations.

## Warning:  incompatible units of measurements in FLQuant objects:  f + NA

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997  1
##      BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##       Predictions will be made for missing observations.

## Warning:  incompatible units of measurements in FLQuant objects:  f + NA

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997  1
##      BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##       Predictions will be made for missing observations.

## Warning:  incompatible units of measurements in FLQuant objects:  f + NA

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997  1
##      BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##       Predictions will be made for missing observations.

## Warning:  incompatible units of measurements in FLQuant objects:  f + NA

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997  1
##      BTS-Isis 1997  2
```

```
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##         SNS 1997   1
##         SNS 1997   2
##         SNS 2003   1
##         SNS 2003   2
##         SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##         SNS 1997   1
##         SNS 1997   2
##         SNS 2003   1
##         SNS 2003   2
##         SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##         SNS 1997   1
##         SNS 1997   2
##         SNS 2003   1
##         SNS 2003   2
##         SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##         SNS 1997   1
##         SNS 1997   2
##         SNS 2003   1
##         SNS 2003   2
##         SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
```

```
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997  1
##     BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997  1
##     BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997  1
##     BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
##          SNS 2003  1
##          SNS 2003  2
##          SNS 2003  3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997  1
##     BTS-Isis 1997  2
## BTS-Tridens 1997  1
## BTS-Tridens 1997  2
##          SNS 1997  1
##          SNS 1997  2
```

```
##          SNS 2003   1
##          SNS 2003   2
##          SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997   1
##     BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##          SNS 1997   1
##          SNS 1997   2
##          SNS 2003   1
##          SNS 2003   2
##          SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
## Note:  The following observations are treated as being missing at random:
##       fleet year age
##     BTS-Isis 1997   1
##     BTS-Isis 1997   2
## BTS-Tridens 1997   1
## BTS-Tridens 1997   2
##          SNS 1997   1
##          SNS 1997   2
##          SNS 2003   1
##          SNS 2003   2
##          SNS 2003   3
##      Predictions will be made for missing observations.
## Warning:  incompatible units of measurements in FLQuant objects:  f + NA
```

```
plot(FLStocks(munc = ple4 + fit1, m02 = ple4 + fit))
```



5.6.5 Likelihood profiling - ToDo

```
# ks <- seq(10,50,5) qmodel <- list( ~ s(age, k=4) + year) srmodel <-
# ~s(year, k=20) liks <- liks <- data.frame(k=ks, nlogl=NA) for(i in ks){
# fmodel <- as.formula(substitute(~s(age, k=4) + s(year, k=x), list(x=i)))
# liks[liks$k==i,2] <- as.numeric(BIC(a4aSCA(ple4, ple4.indices[1], fmodel,
# qmodel, srmodel))) }
```

5.6.6 Paralell computing (not working yet ...)

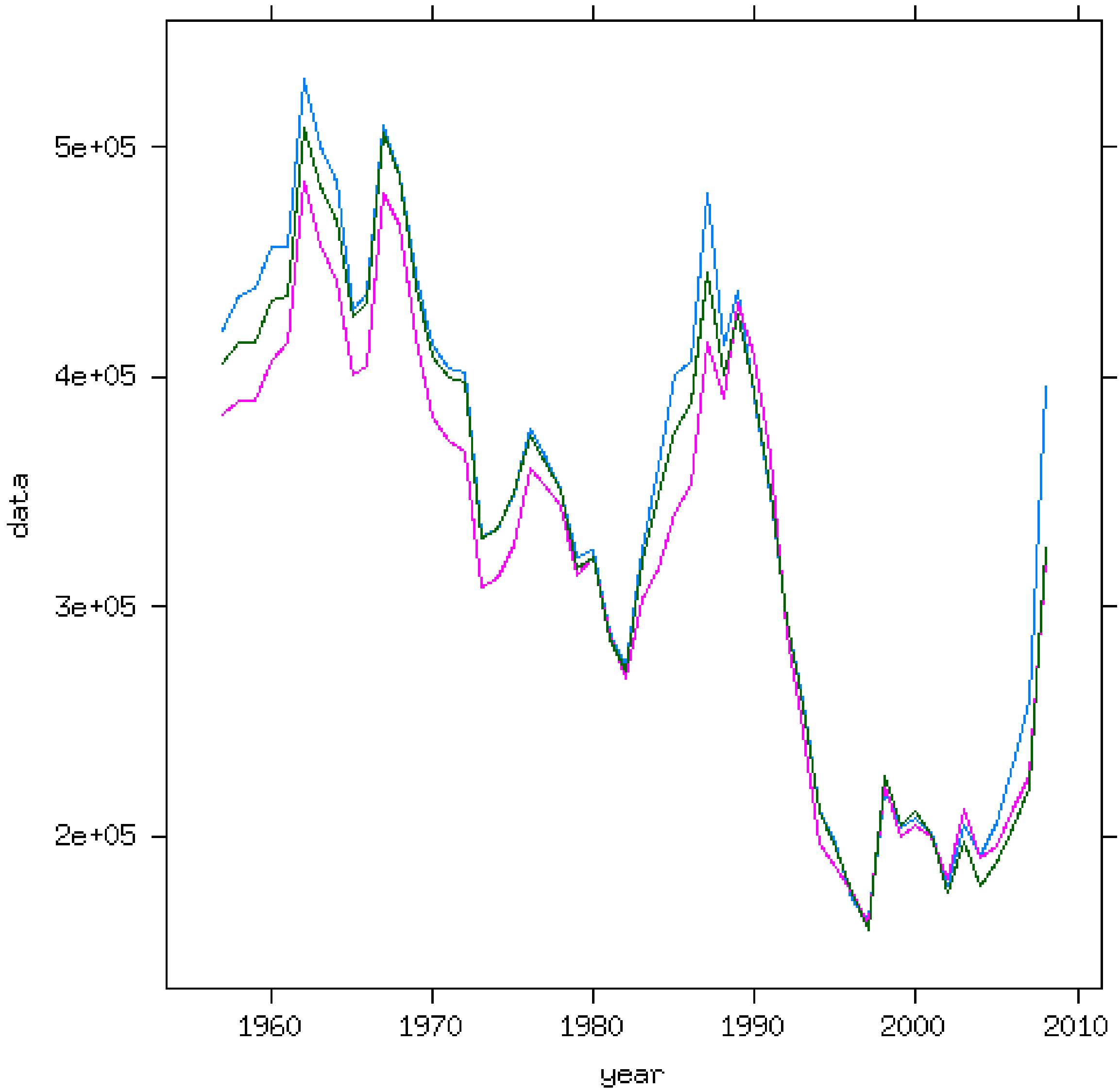
```
# fmodel <- ~ s(age, k=4) + s(year, k = 20) qmodel <- list( ~ s(age, k=4) +
# year) srmodel <- ~s(year, k=20) fit <- a4aSCA(ple4, ple4.indices[1],
# fmodel, qmodel, srmodel) fits <- simulate(fit, 25) stks <- ple4 + fits
# idxs <- ple4.indices[1] index(idxs[[1]]) <- index(fits)[[1]]
# library(parallel) options(mc.cores=1) lst <- mclapply(split(1:25, 1:25),
# function(x){ fit <- a4aSCA(stks[,,,,,x], FLIndices(idxs[[1]][,,,,,x]),
```

```
# fmodel, qmodel, srmodel, fit='MP') })
```

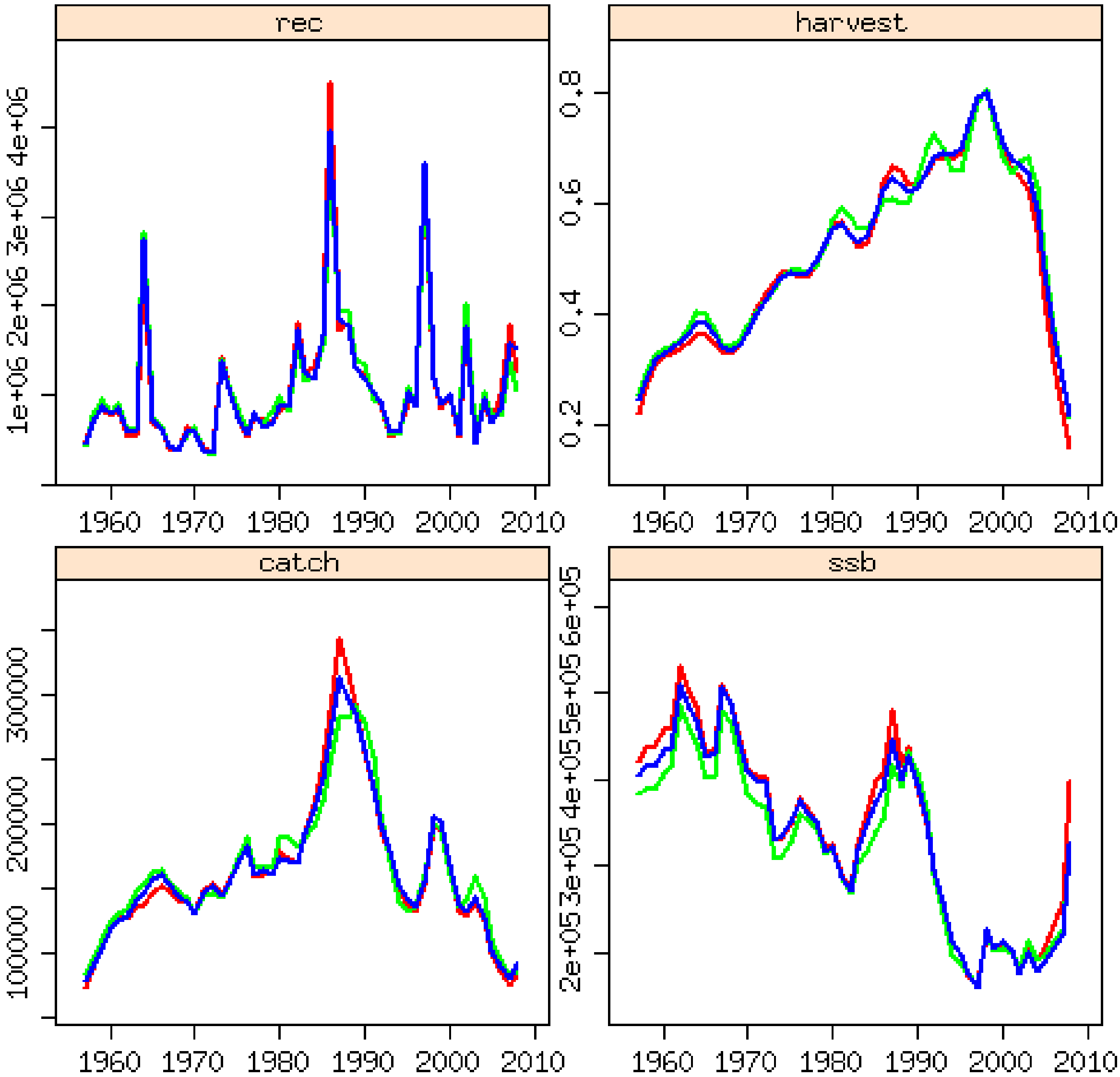
5.7 Model averaging

We follow the paper Colin et al. Only the AIC averaging is implemented for now.

```
data(ple4)
data(ple4.indices)
f1 <- sca(ple4, ple4.indices, fmodel = ~factor(age) + s(year, k = 20), qmodel = list(~s(age,
    k = 4), ~s(age, k = 4), ~s(age, k = 3)), fit = "assessment")

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
##  BTS-Tridens 1997   1
##  BTS-Tridens 1997   2
##          SNS 1997   1
##          SNS 1997   2
##          SNS 2003   1
##          SNS 2003   2
##          SNS 2003   3
##       Predictions will be made for missing observations.

f2 <- sca(ple4, ple4.indices, fmodel = ~factor(age) + s(year, k = 20), qmodel = list(~s(age,
    k = 4) + year, ~s(age, k = 4), ~s(age, k = 3)), fit = "assessment")

## Note:  The following observations are treated as being missing at random:
##        fleet year age
##      BTS-Isis 1997   1
##      BTS-Isis 1997   2
##  BTS-Tridens 1997   1
##  BTS-Tridens 1997   2
##          SNS 1997   1
##          SNS 1997   2
##          SNS 2003   1
##          SNS 2003   2
##          SNS 2003   3
##       Predictions will be made for missing observations.

stock.sim <- ma(a4aFitSAs(list(f1 = f1, f2 = f2)), ple4, AIC, nsim = 100)

## model weights are
##    weight..perc.
## f1         28.15
## f2         71.85

stks <- FLStocks(f1 = ple4 + f1, f2 = ple4 + f2, ma = stock.sim)
flqs <- lapply(stks, ssb)
flqs <- lapply(flqs, iterMedians)


xyplot(data ~ year, groups = qname, data = flqs, type = "l")
```




```
plot(stks)
```



rec
harvest
catch
ssb
1e+06 2e+06 3e+06 4e+06
0.2
0.4
0.6
0.8
100000
200000
300000
2e+05 3e+05 4e+05 5e+05 6e+05
1960 1970 1980 1990 2000 2010