a4a stock assessment framework
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

March 31, 2014

Contents

1 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.

```
> library(FLa4a)
> data(ple4)
> data(ple4.indices)
```

1.1 a4aFit* - The fit classes

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      index
Class:      call    numeric     array   FLQuant   FLQuant   FLQuant  FLQuants

Name:       name       desc      range
Class: character  character    numeric

Extends: "FLComp"

Known Subclasses: "a4aFitSA"
```

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   catch.n
Class:   SCAPars       call    numeric     array   FLQuant   FLQuant   FLQuant

Name:      index       name       desc      range
Class:  FLQuants  character  character    numeric

Extends:
Class "a4aFit", directly
Class "FLComp", by class "a4aFit", distance 2
```

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

> showClass("a4aStkParams")

Class "a4aStkParams" [package "FLa4a"]

Slots:

Name:       fMod     n1Mod     srMod    params      vcov centering     distr
Class:   formula   formula   formula     FLPar     array   numeric character

Name:          m     units      name      desc     range
Class:   FLQuant character character character   numeric

Extends: "FLComp"
```

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.

1.2 The submodels

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))
```

1.3 Run !!

running the model is done by

```
> fit <- sca(fmodel, qmodel, stock = ple4, indices = ple4.indices[1])
```

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.

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

```
> fitstk <- ple4 + fit
> plot(fbar(fitstk))
```

1.4 Some more examples

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.

```
> fmodel <- ~ s(age, k=4) + s(year, k = 20)
> qmodel <- list( ~ factor(age))
> fit1 <- sca(fmodel, qmodel, stock = ple4, indices = ple4.indices[1])
> wireframe(data ~ year + age, data = as.data.frame(harvest(fit1)), drape = TRUE)
```

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

```
> fmodel <- ~ s(age, k=4) + s(year, k = 20) + te(age, year, k = c(3,3))
> qmodel <- list( ~ factor(age))
> fit2 <- sca(fmodel, qmodel, stock = ple4, indices = ple4.indices[1])
> wireframe(data ~ year + age, data = as.data.frame(harvest(fit2)), drape = TRUE)
```

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

```
> fmodel <- ~ te(age, year, k = c(4,20))
> qmodel <- list( ~ factor(age))
> fit2 <- sca(fmodel, qmodel, stock = ple4, indices = ple4.indices[1])
> wireframe(data ~ year + age, data = as.data.frame(harvest(fit2)), drape = TRUE)
```

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

```
> fmodel <- ~ te(age, year, k = c(4,20)) + s(year, k = 5, by = as.numeric(age==1))
> qmodel <- list( ~ factor(age))
> fit3 <- sca(fmodel, qmodel, stock = ple4, indices = ple4.indices[1])
> wireframe(data ~ year + age, data = as.data.frame(harvest(fit3)), drape = TRUE)
```

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.

1.5 Inspecting ADMB files

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

```
> fit. <- a4aSCA(fmodel, qmodel, stock = ple4, indices = ple4.indices[1], wkdir="mydir")

Model and results are stored in working directory [mydir-1]
```

1.6 Variances of input data and likelihood weighting

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.

```
> # data
> stk <- ple4
> idx <- ple4.indices[1]
> # models
> fmodel <- ~ s(age, k=4) + s(year, k = 20)
> qmodel <- list( ~ s(age, k=4))
> # variance of observed catches
> varslt <- catch.n(stk)
> varslt[] <- 1
> catch.n(stk) <- FLQuantDistr(catch.n(stk), varslt)
> # variance of observed indices
> varslt <- index(idx[[1]])
> varslt[] <- 0.1
> index(idx[[1]]) <- FLQuantDistr(index(idx[[1]]), varslt)
> # run
> fit0 <- sca(fmodel, qmodel, stock = ple4, indices = ple4.indices[1])
> fit. <- sca(fmodel, qmodel, stock = stk, indices = idx)
```

1.7 The 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.

```
> # data
> stk <- ple4
> idx <- ple4.indices[1]
> # models
> fmodel <- ~ s(age, k=4) + s(year, k = 20)
> qmodel <- list( ~ s(age, k=4))
> vmodel <- list(~1, ~1)
> # run
> fit0 <- a4aSCA(fmodel, qmodel, stock = ple4, indices = ple4.indices[1])
> fit00 <- a4aSCA(fmodel, qmodel, vmodel=vmodel, stock = ple4, indices = ple4.indices[1])
> all.equal(fit0, fit00)
```

```
 [1] "Attributes: < Component 1: target, current do not match when deparsed >"
 [2] "Attributes: < Component 2: Mean relative difference: 0.1946614 >"
 [3] "Attributes: < Component 4: Mean relative difference: 0.08812653 >"
 [4] "Attributes: < Component 6: Mean relative difference: 0.7127212 >"
 [5] "Attributes: < Component 7: Mean relative difference: 0.08098264 >"
 [6] "Attributes: < Component 8: Component 1: Mean relative difference: 0.1074805 >"
 [7] "Attributes: < Component 10: Attributes: < Component 2: Component 1: Attributes: < Component 7: Me
 [8] "Attributes: < Component 10: Attributes: < Component 2: Component 1: Attributes: < Component 9: Me
 [9] "Attributes: < Component 10: Attributes: < Component 3: Attributes: < Component 9: Mean relative d
[10] "Attributes: < Component 10: Attributes: < Component 3: Attributes: < Component 13: Mean relative
```

```
[11] "Attributes: < Component 10: Attributes: < Component 4: Component 1: Attributes: < Component 5: fo
[12] "Attributes: < Component 10: Attributes: < Component 4: Component 1: Attributes: < Component 7: At
[13] "Attributes: < Component 10: Attributes: < Component 4: Component 1: Attributes: < Component 7: Nu
[14] "Attributes: < Component 10: Attributes: < Component 4: Component 1: Attributes: < Component 9: At
[15] "Attributes: < Component 10: Attributes: < Component 4: Component 1: Attributes: < Component 9: Nu
[16] "Attributes: < Component 10: Attributes: < Component 4: Component 2: Attributes: < Component 7: Me
[17] "Attributes: < Component 10: Attributes: < Component 4: Component 2: Attributes: < Component 9: Me
[18] "Attributes: < Component 12: Mean relative difference: 0.1137231 >"
```

```
> vmodel <- list(~(age)^2-1, ~1)
> fit. <- a4aSCA(fmodel, qmodel, stock = stk, indices = idx)
```