# Getting Started with FLSAM

Mark R. Payne <mpa@aqua.dtud.dk>
Niels Hinzten <niels.hintzen@wur.nl>

August 23, 2012

## 1 Introduction

In this document we will demonstrate the basic functionality of FLSAM, the FLR wrapper around the ADMB "SAM" stock assessment package developed by Anders Nielsen (DTU-Aqua). This package allows you to combine the flexibility and functionality of the FLR library with the power of the SAM assessment tool in a convenient and easy to use manner, allowing you to both have your cake and eat it too!

## 2 Installation

Firstly, and most importantly, FLSAM should be considered as a method under-development - whilst it is relatively stable and approaching the point where it is suitable for routine use, bugs, rough edges and occasional lacking documentation are to be expected. If you find a bug, or think that something could be done better, please contact us and let us know!

The most reliable way to install FLSAM is to download the release from the HAWG googlecode repository, http://hawg.googlecode.com, and install it manually - click on the "Downloads" tab above and select the version for your operating system. This is where you will find the latest official release. In the future, this package will also be directly available from the FLR repository.

Once you have downloaded the file, you can install it in Windows by selecting Packages/"Install from Local Zip file" from the menus in R and selecting the appropriate zip file.

Alternatively, for both Windows and Linux, you can also install from the R command prompt using:

```
> install.packages("<pathToFLSAM>/<FLSAM file>")
```

or using the following from the command line:

```
R CMD INSTALL <FLSAM file>
```

### 2.1 FLCore dependency

Please note that FLSAM will only run when FLCore (the main "core" package of the FLR library) is installed too! The current FLSAM release requires FLCore 2.4 or greater. If you don't already have it, you can install the latest FLCore release by issuing the following command in R.

```
> install.packages("FLCore", repos="http://flr-project.org/R")
```

# 3 Setup the Assessment Objects

The following assumes some degree of familarity with the concepts and basics of both R and FLR. If you're not familar with FLR, have a look at the Teach Yourself FLR tutorials on the FLR homepage, http://flr-project.org/.

Firstly, load the FLCore package

```
> library(FLCore)
```

This package provides the basic classes and methods that we are going to use in our assessment. The next step is obviously to get some data to work on! There are two types of objects that you need to create.

- an FLStock object, which stores all of the data relating to the stock (e.g. natural mortality, catches, weights etc).
- an FLIndices object, which contains all of the information relating to the tuning indices.

The easiest way to create these objects is to read the input data directly from a set of Lowestoft VPA files using the built-in functionality in the FLCore package. As an example,

```
> stck <- readFLStock(file.path(".","data","index.txt"),no.discards=TRUE)
```

will create an FLStock object called `stck` from the Lowestoft VPA files stored in `./data/` (with discards set to zero). Similarly,

```
> tun <- readFLIndices(file.path(".","data","fleet.txt"))
```

will create an FLIndices object from the same location. For more information, see the appropriate help files by issuing the commands `?readFLstock` and `?readFLIndices` or refer to the FLR homepage. You can view a summary of the objects that you have created using the `summary` command:

```
> summary(stck)
> summary(tun)
```

Alternatively, and what we will do for the remainder of this tutorial, is to use the sample objects stored in the FLSAM package.

```
> library(FLSAM)
> data(NSH)
> stck <- NSH
> tun  <- NSH.tun
```

This loads two objects called `NSH` and `NSH.tun` containing the stock and tuning indices respectively for the North Sea herring stock, and then copies them into new objects called `stck` and `tun` (just for consistency with everything else). These datasets contain (shortented versions) of the complete FLR stock object and FLR tuning indices objects used to perform the assessment of this stock in 2012.

# 4 Configure the FLSAM assessment

The next step is to set the various options for the FLSAM assessment. These are controlled through an object of type "FLSAM.control". Generally the structure of this object is complex, and highly specific to the assessment at hand. We can, however, use the automatic configuration built into FLSAM to get us started - this builds up an FLSAM.control template based on the FLStock and FLIndices objects suppplied. It works as follows:

```
> ctrl <- FLSAM.control(stck,tun)
```

The resulting FLSAM.control object, `ctrl`, contains a structure that can be used to configure the assessment - in the default mode, however, all parameters are "unset". The initial configuration contained in this object will almost certainly not converge - it is up to the user to configure the assessment in the appropriate manner. Configuration options can be set by the user by directly accessing and modifying the contents of `FLSAM.control` object e.g

```
> ctrl@catchabilities["HERAS","8"] <- 12
```

The choice of the "correct" configuration is up to the user, and is not something that we're going to go into in detail here. However, for the sake of this tutorial, here's one we prepared earlier: the control object used in the NSH assessment is also contained and in FLSAM package, and can be loaded as follows:

```
>  data(NSH.sam)
>  ctrl <- NSH.ctrl
```

More information regarding the configuration can be found in the helpfiles for `FLSAM.control`, accessible with the command `?FLSAM.control`

# 5 Running the assessment

You now have the three elements necessary to perform an assessment: an FLStock object `stck`, an FLIndices object `tun` and an FLSAM.control object `ctrl`, so you're ready to go! Run the assessment as follows:

```
>  sam <- FLSAM(stck,tun,ctrl)
```

You will immediately start to see a series of outputs from the model, as it runs through the optimisation process. The model fitting procedure can take some time, depending on how powerful your machine is - the full NSH assessment takes 5 minutes on a powerful workstation, about 10 minutes on a relatively new laptop, and 20 minutes on a 2007-vintage laptop. Patience, young jedi.

Once the assessment is complete, you will have an object called `sam`, which is of the type `FLSAM`. You can copy the results of this assessment into an FLStock object as follows

```
> new_stck <- stck + sam
```

`new_stck` can then be used by other methods in the FLR library e.g. making short-term forecasts. However, for the remainder of this tutorial, we will focus on the FLSAM object, `sam` and some of the tools that can be used to analyse it.

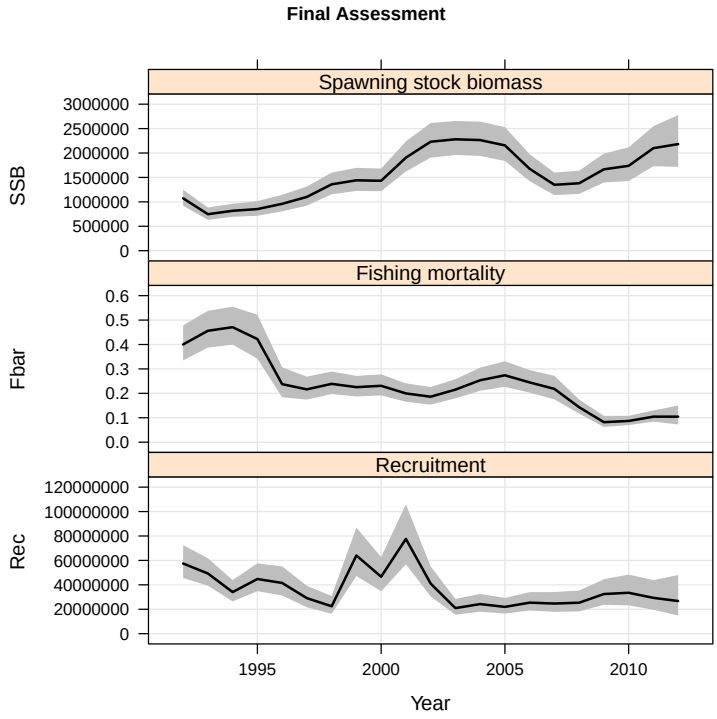


Figure 1: Stock summary plot for the North Sea herring object with associated 95% uncertainty (grey areas) showing SSB (top panel), F ages 2-6 (middle panel) and recruitment (bottom panel).

# 6 Plotting

The next step is to look at your results! Firstly, lets have a look at the results.

```
> plot(sam)
```

This returns a "stock-summary" plot, showing the ssb, fbar and recruitment estimated by FLSAM, including the 95% confidence intervals (Figure 1).

But the key question is, how well does the model fit the observations? We should never base our judgement of the assessment on its results alone! Lets have a look at the fit diagnostics instead.

# 7 Diagnostics

Firstly, lets look at the residuals. We can generate a series of residual plots with the `residual.diagnostics` functionality (Figure 2).

```
> residual.diagnostics(sam)
```

This returns a series of "ICA-style" diagnostic plots. The example shown here doesn't look too bad, although the Tukey-Anscombe plot is not-ideal.

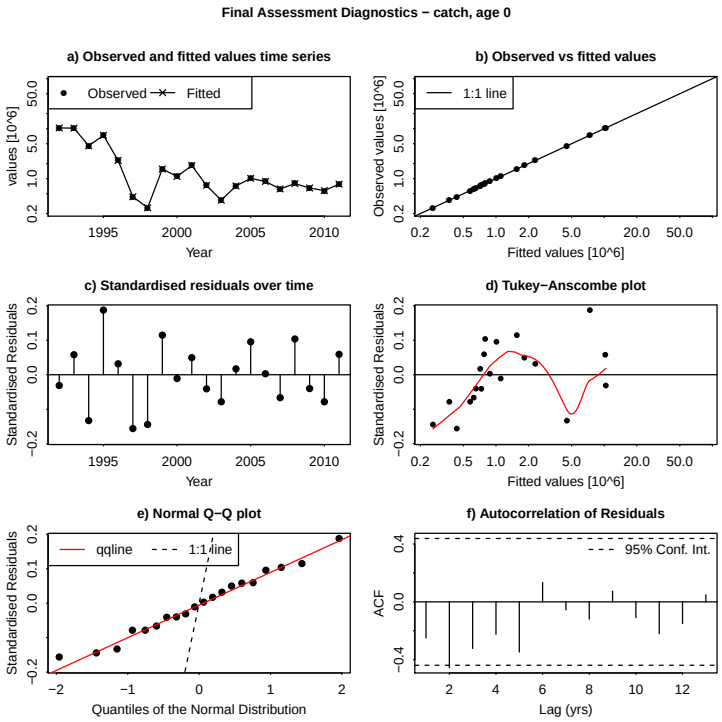


Figure 2: Example of a residual diagostics plot for the catch at age 0 wr time series. Top left: Estimates of numbers at 0 wr (line) and numbers predicted from index abundance at 0 wr. Top right: scatterplot of index observations versus assessment model estimates of numbers at 0 wr with the best-fit catchability model (linear function). Middle right: index observation versus standardized residuals at 0 wr. Middle left: Time series of standardized residuals of the index at 0 wr. Bottom left: normal Q-Q plot of standardized residuals. Bottom right: Autocorrelation plot.

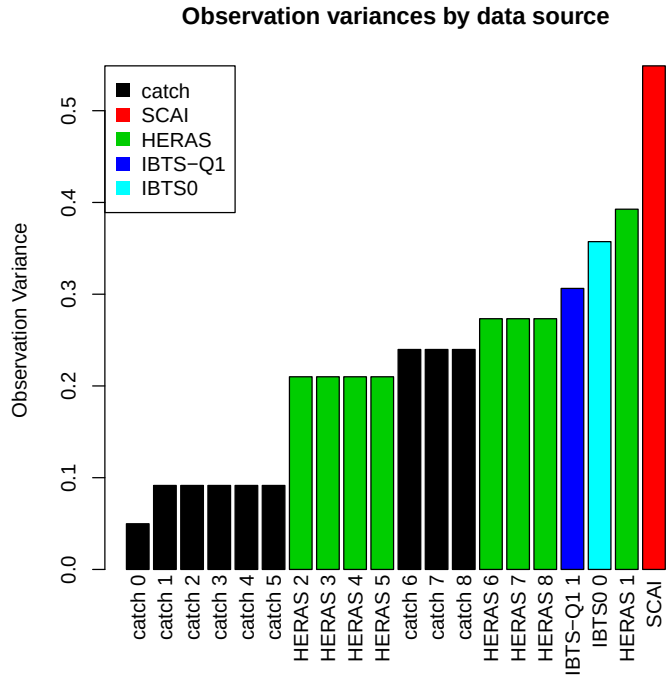


Figure 3: Observation variance by data source as estimated by the assessment model. Observation variance is ordered from least (left) to most (right). Colours indicate the different data sources. Observation variance is not individually estimated for each data source individually thereby reducing the parameters needed to be estimated in the assessment model. In these cases of parameter bindings, observation variances have equal values.

However, the plots usually go past too fast to keep track of them. A more convenient way to view them is by setting the `par(ask=TRUE)` argument first i.e.

```
> par(ask=TRUE)
> residual.diagnostics(sam)
```

and then pressing enter to move between them. Alternatively, you can also capture these and other outputs as a PDF e.g

```
> pdf("Diagnostics.pdf")
> residual.diagnostics(sam)
> dev.off()
```

We'd also like to know about how the model weights the individual data sets. The observation variance plot can be used to visualise this (Figure 3).

```
> obsvar.plot(sam)
```

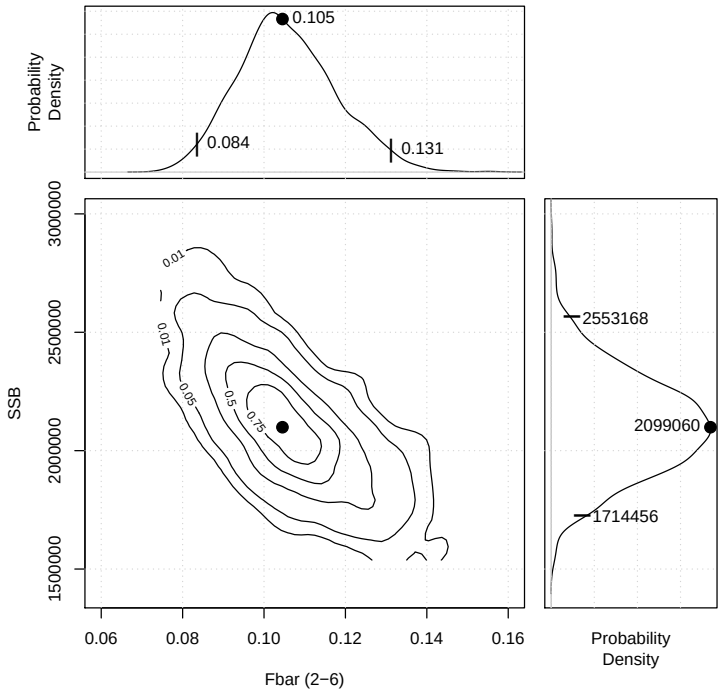


Figure 4: Model uncertainty; distribution and quantiles of estimated SSB and F2-6 in the terminal year of the assessment. Estimates of precision are based on a sampling from the FLSAM estimated variance / covariance estimates from the model.

The model is clearly giving the most weight to the catch data. The HERAS survey is the "best" of the tuning fleets, with the SCAI survey being the most noisy (according to the SAM model).

The correlation between the terminal SSB and F values can be visualised with an otolith plot (Figure 4).

```
> otolith(sam,year=2011,plot=T,n=1000)
```

The somewhat uneven and noisy correlation between these values arises from the process of sampling from the variance-covariance matrix and the fact that we haven't used enough samples. We can improve this plot by increasing the number of samples (the `n` argument)

```
> otolith(sam,year=2011,plot=T,n=1e5)
```

but this will obviously increase computation time.

Correlation between parameters can be visualised and detected with the correlation plot (Figure 5).

```
> cor.plot(sam)
```

In this case, some correlation between the parameters is apparent, especially in the bottom left-corner.

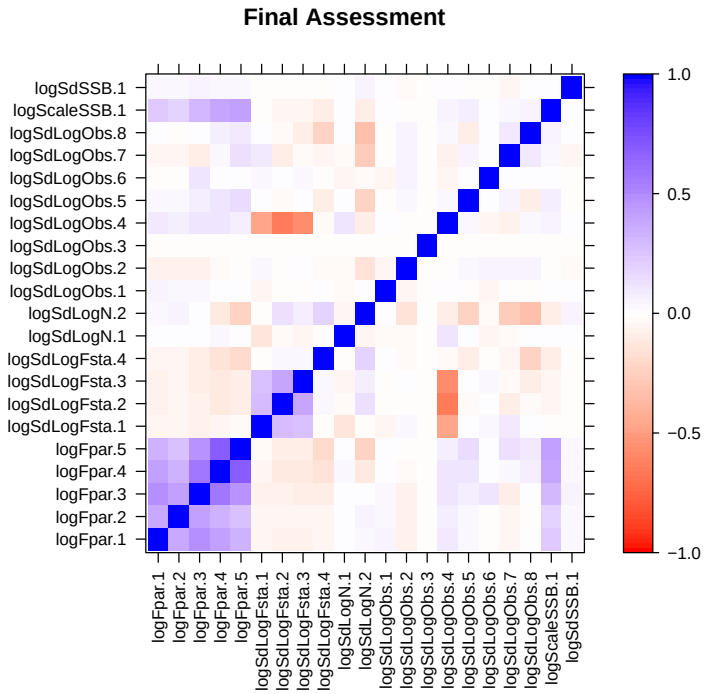


Figure 5: Correlation plot of the parameters in the assessment model. The diagonal represents the correlation with the data source itself.

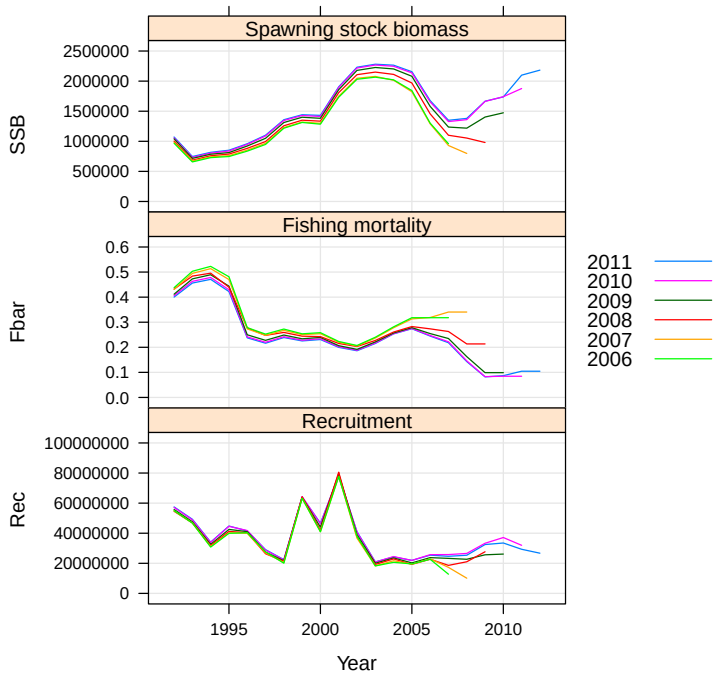


Figure 6: Retrospective pattern of SSB (top panel) F (middle panel) and recruitment (bottom panel) for the assessments with respectively terminal years from 2011 to 2008.

We can perform an analytical retrospective analysis using the `retro` function and plot the results using the built in methods (Figure 6).

```
> retro.sams <- retro(stck,tun,ctrl,retro=5)
> plot(retro.sams)
```

The model appears to struggle to capture the change in the SSB trend in this case. This is relatively unusual for the SAM method, and may be related to the relatively short time series employed.

Functionality is also provided to perform permutation analyses of the surveys, where the assessment is repeated using only a subset of the survey information that is availabe. The `looi()` function performs all such combinations, the `loi()` function performs a leave-one-in (LOI) analysis (where only one survey is used) and the `loo()` function performs a leave-one-out (LOO) analysis (where only a single survey is dropped). For example, a LOO analysis can be performed as follows, with the results shown in Figure 7

```
> loo.sams <- loo(stck,tun,ctrl)
> plot(retro.sams)
```

The HERAS survey clearly appears to have the strongest influence on the perception of the stock, which also agrees with the results in Figure 3.

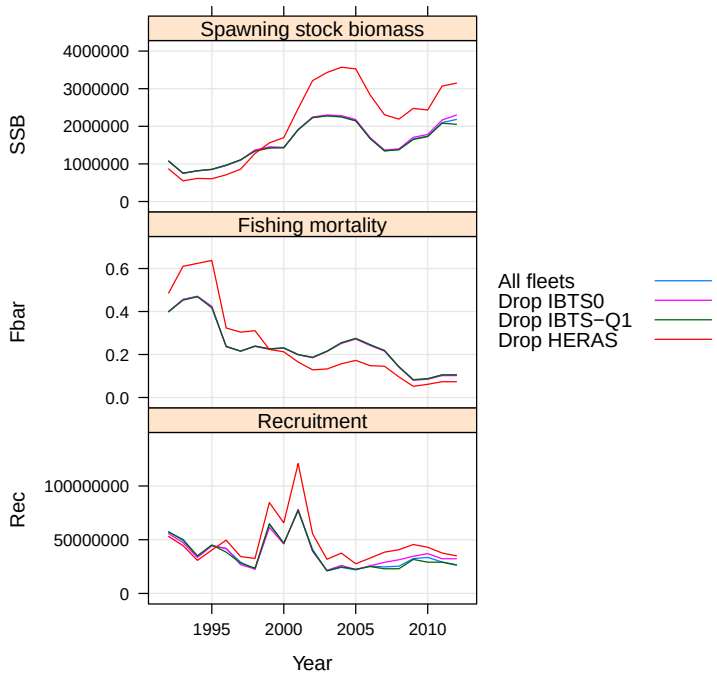


Figure 7: Leave-one-out (LOO) analysis for the North-Sea herring analysis

# 8 Documentation

Once the user is fully satisifed with the model fit, they will want to write a report and document their results. Fortunately, FLSAM provides a quick and easy function to generate the necessary tables:

```
> sam.out <- FLSAM.out(stck,tun,sam,
+                 format="Table 1.%i North Sea herring assessment")
```

This function produces a set of formatted character strings that can be used directly as documentation. e.g.

```
> sam.out[1:10]
```

```
 [1] "Table 1.1 North Sea herring assessment CATCH IN NUMBER"
 [2] ""
 [3] "Units  :  thousands "
 [4] "   year"
 [5] "age     1992     1993    1994    1995    1996   1997    1998    1999    2000"
 [6] "  0 10331300 10265400 4498900 7438469 2311226 431175  259526 1566349 1105085"
 [7] "  1  2303100  3826800 1785200 1664874 1606393 479702  977680  303520 1171677"
 [8] "  2  1284900  1176300 1783200 1444061  642084 687920 1220105  616354  622853"
 [9] "  3   442700   609000  489100  816703  525601 446909  537932 1058716  463170"
[10] "  4   361500   305500  347600  231794  172099 284920  276333  294066  646814"
```

The document can be written to a file as follows.

```
> options("width"=80,"scipen"=1000)
> write(sam.out,file="sam.out")
```

and from there directly incorporated into a report.

# 9 Advanced features

## 9.1 Analysing the fit

FLSAM provides a series of accessor functions allowing quick and easy extraction of key variables from an `FLSAM` object. These include

- `ssb` - Spawning stock biomass
- `fbar` - Meaning fishing mortality
- `tsb` - Total stock biomass
- `rec` - Recruitment
- `f` - Estimated fishing mortality at age
- `n` - Estimated numbers at age

A number of accessor functions also exist to extract key model parameters from the FLSAM class.

- `catchabilities` - Extracts the survey catchabilities

- `power.law.exps` - Extracts the power-law exponents of the catchability models, if applied
- `obs.var` - Extracts the estimated observation variance associated with each time series
- `params`, `coef` and `coeffficients` can all be used to extract the full set of fitted parameters
- `residuals` extracts the residuals to the fit

These functions can be used to develop more detailed, customized, analyses of the fit - the possiblities are only limited by the users imagination. To give one such example, we extract the catchabilties estimated by the model for the various age groups in the HERAS survey, and plot them using the `xyplot` function from the lattice package (Figure 8).

```
> plot.dat <- catchabilities(sam)
> library(lattice)
> print(xyplot(value+ubnd+lbnd ~ age | fleet,
+           data=plot.dat,subset=fleet %in% c("HERAS"),
+           scale=list(alternating=FALSE,y=list(relation="free")),as.table=TRUE,
+           type="l",lwd=c(2,1,1),col=c("black","grey","grey"),
+           main="HERAS catchability parameters",
+           ylab="Catchability",xlab="Age",ylim=c(0,2)))
```

### 9.2 Other features

Other advanced features include

- `AIC` - the Aikaike Information Criteria for an FLSAM object
- `lr.test` - a likelihood ratio test for comparing two objects
- `update` - Repeat an assessment using the results of a previous assessment as initial guesses (thereby speeding up the assessment dramatically
- `monteCarloStock` - Redraw a (large) set of F and N estimates from FLSAM result and place them into an FLStock object. Of particular use in management strategy evaluations
- `FLR2SAM`, `runSAM`, and `SAM2FLR` provide the interface wrapper to SAM. In some cases, it can be useful to work with them directly.

## 10 Next Steps

This document provides a brief overview of the functions available in the FLSAM package. More details on all these functions are availabe through the online documentation i.e the R help system. A pdf compiling all of these help files is also available: use `vignette("FLSAM-manual")` to open it.

There are many things that can go wrong along the way - SAM is a complex assessment tool. If you encounter problems with your assessment, or perhaps

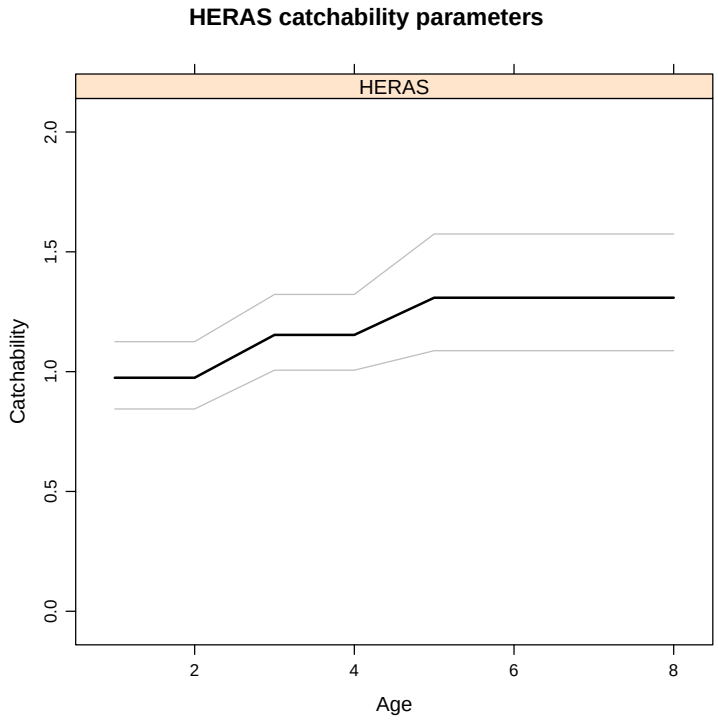


Figure 8: Catchabilities as a function of age for each of the HERAS time-series. Grey lines indicate the 95% confidence intervals. The step-like nature of the plot arises from the binding of the parameters

uncover a bug, please report it to the maintainers and the FLR mailing list. Similarly, if there are enhancements that you would like to see, or things that you think are missing, please contact us - we can't promise a solution, but we will promise to help!

Good luck!