Using the msy package

John Simmond, Colin Millar and Einar Hjörleifsson

September 26, 2014

- R version 3.1.1 (2014-07-10), `x86_64-redhat-linux-gnu`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils
- Other packages: dplyr 0.2.0.99, FLCore 2.5.20140919, ggplot2 1.0.0, gridExtra 0.9.1, knitr 1.6, lattice 0.20-15, lubridate 1.3.3, MASS 7.3-27, mgcv 1.8-3, msy 0.1.12, nlme 3.1-110, plyr 1.8.1, R2admb 0.7.10, RColorBrewer 1.0-5, reshape2 1.4, scales 0.2.3, scam 1.1-6, stringr 0.6.2, xtable 1.7-3
- Loaded via a namespace (and not attached): assertthat 0.1, colorspace 1.2-2, dichromat 2.0-0, digest 0.6.4, evaluate 0.5.3, formatR 0.10, gtable 0.1.2, labeling 0.2, Matrix 1.1-4, memoise 0.1, munsell 0.4.2, parallel 3.1.1, proto 0.3-10, Rcpp 0.11.2, stats4 3.1.1, tools 3.1.1

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

1 Preamble

This document is as much as the msy-package itself still in developement.
The origin of this package is from an intial coding by John Simmonds which was restructured by Colin Millar into an R-package with additional development, including coding the Buckland method. Einar Hjörleifsson compartmentalized the structure of the code as well as providing the output of the analysis in a more structured format.
At this moment there is no person responsible for the maintainance of the package, including the above mentioned names.

2 Installation

The developmental repository for the `msy` package is located on github at two locations:

- github.com/wgmg/msy: This version reflects to a large degree what was available and used at the WKMSYREF3 in January 2014.
- github.com/einarhjorleifsson/msy: This was forked from the wgmg site in September 2014. .... ST

The easiest way to install the `msy` package is to use the function `install_github` in the `devtools` package. The `Rtools.exe` software is needed for building packages under Microsoft Windows. Run the following lines to install the latest version of `msy`, any other packages that you require will automatically be downloaded from `CRAN`, the R package repository. All except for `FLCore` package, which is also installed from `github`.

```
library(devtools)
install_github("FLCore", "flr")
install_github("msy", "einarhjorleifsson", ref = "master")
```

The above is equivalent to `install.packages` and hence need only to be performed once. However, since the `msy` package is currently under development (including bug-fixing) one may expect more frequent code updating in the package than what one may be familiar with for packages on `CRAN`. Once the packages have been installed the library is simply loaded via the familiar:

```
library(msy)
```

Besides functions the package comes with the following data:

- codCS: `FLStock` object of the Celtic Sea cod
- codEB: `FLStock` object of the Eastern Baltic cod
- codIS: `FLStock` object of the Icelandic cod
- codNS: `FLStock` object of the North Sea cod
- codWB: `FLStock` object of the Western Baltic cod
- codWS: `FLStock` object of the West of Scotland cod
- saiFO: `FLStock` object of the Faroe saithe
- saiIS: `FLStock` object of the Icelandic saithe
- solKA: `FLStock` object of the Kattegat sole

These are all stored in the icesStocks list object.
The current version of the `msy` implements two methods that go under the working names EqSim and plotMSY. Only usage of functions for the EqSim approaches are described in the following sections.

3 EqSim

EqSim is a stochastic equilibrium software that may be used to explore MSY reference points. Productivity parameters (i.e. year vectors for natural mortality, weights-at-age and maturities) as well as selection are re-sampled at random from user specified range of years from the assessment. Fixing these parameters to an average over specified years can also be set by the user. Recruitments are re-sampled from their predictive distribution. Uncertainty in the stock-recruitment model is taken into account by applying model averaging using smooth AIC weights (Buckland et al. 1997). In addition assessment errors can be emulated by applying a user-specified error (CV and autocorrelation) to the intended target fishing mortality.
The current version of EqSim only takes FLStock objects as inputs.

3.1 A quick start

In the following subsections we will simulate the north sea cod stock into the future under some basic assumptions. For the simulations we need to choose which years we will use to generate noise in the quantities: weight at age, maturity at age, natural mortality at age, and selection pattern. We also need to choose a set of Fbar values to simulate over in order estimate F reference points.
The eqsim approach consists of three components:

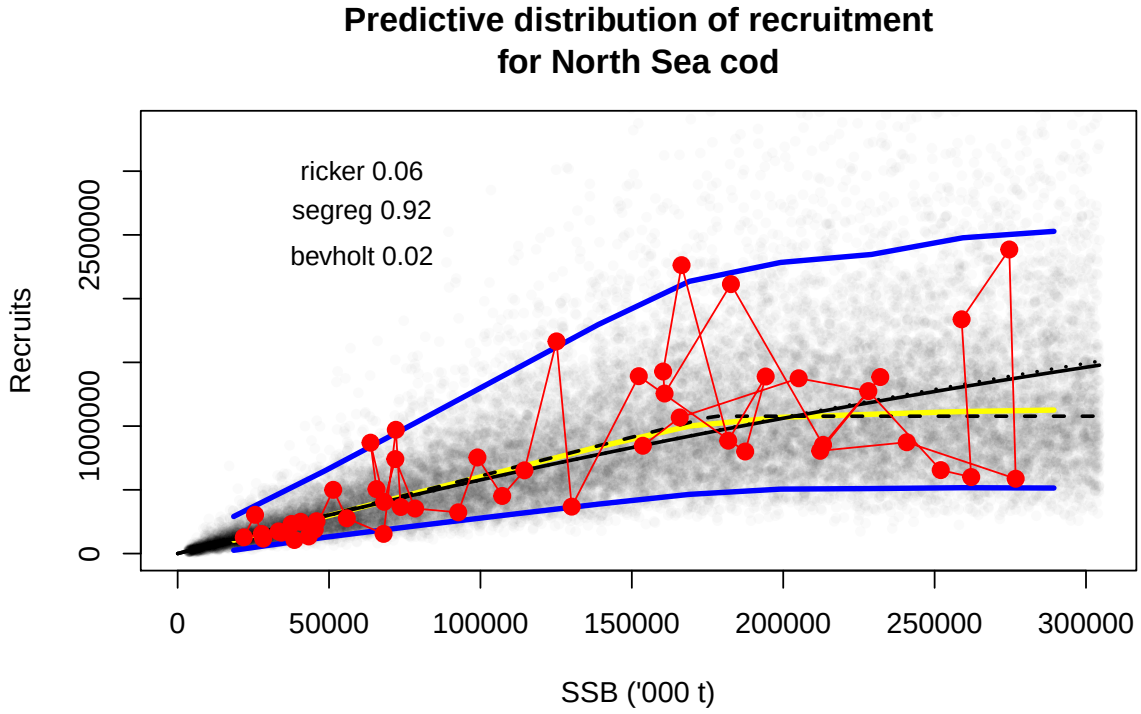
1. Estimate the stock recruitment relationship
2. Simulate a stock to equilibrium and continue simulating for some years
3. Calculate reference points from the simulated stock at equilibrium (last 50 years of the runs are used)

This can be done in one go with the following code:

```
FIT <- eqsr_fit(icesStocks$codNS,nsamp=1000)
SIM <- eqsim_run(FIT, Fcv=0.25, Fphi=0.30,
                 Blim=70000,Bpa=150000,
                 Fscan = seq(0,1.2,len=40),
                 verbose=FALSE,extreme.trim=c(0.05,0.95))
```

The stock recruitment function can be plotted by:

```
eqsr_plot(FIT,n=2e4)
```



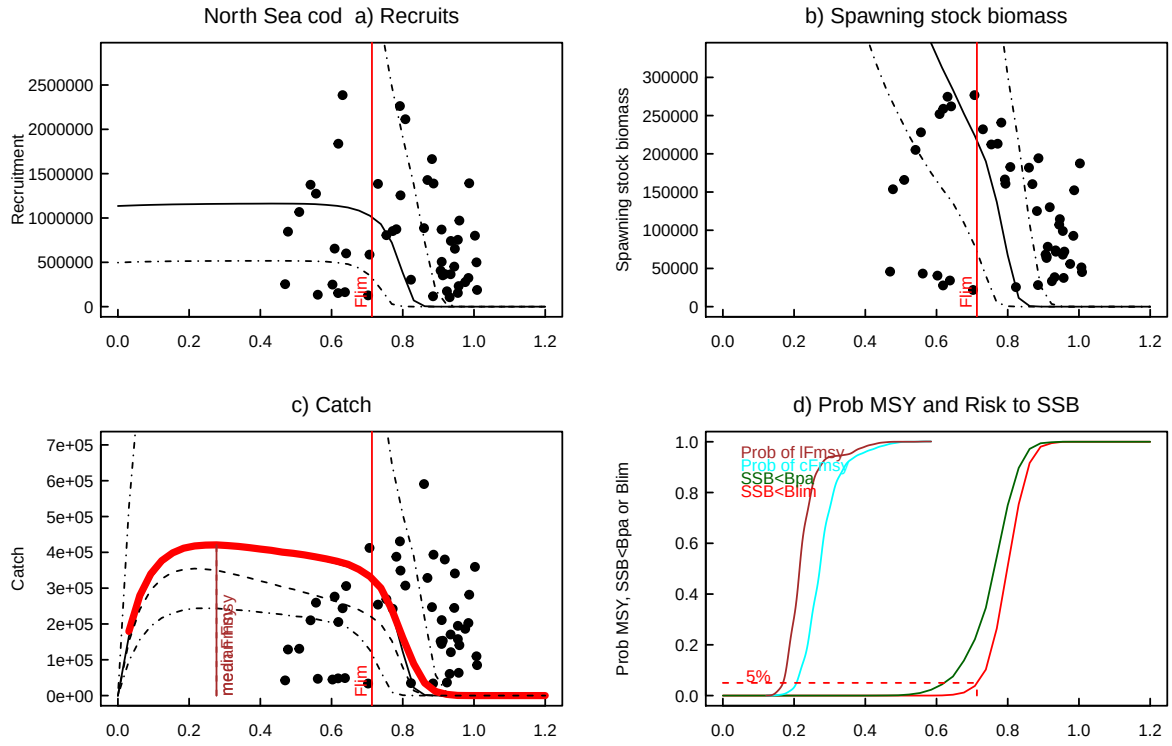
A summary of the key results can be obtained by:

```
xtable(SIM$Refs)
```

	Flim	Flim10	Flim50	medianMSY	meanMSY	FCrash05	FCrash50
catF	0.71	0.74	0.80	0.28	0.28	0.77	0.86
lanF				0.22	0.22		
catch	322840.20	300902.00	127006.37	420750.12	420750.12	235248.73	1589.94
landings				354603.71	354603.71		
catB	217195.90	191398.01	70137.42	1035951.11	1035951.11	136913.27	779.78
lanB				1364156.27	1364156.27		

A summary plots conditioned on maximizing **catch** are obtained by:

```
eqsim_plot(SIM,catch=TRUE)
```



3.2 The recruitment model

Model fitting is done by maximum likelihood using the `nlminb` optimiser in R. By refitting to non-parametric bootstrap resamples of the stock and recruit pairs, samples from the approximate joint distribution of the model parameters can be made. This is done by invoking the `eqrs_fit` function. The function first sets up the stock and recruit pairs based on the information in the `FLStock` object and removes any incomplete pairs, before dispatching on the model fitting / averaging algorithm chosen. Currently only a bootstrap based model averaging method called smooth AIC is implemented fully. The details can be found in `eqrs_Buckland` function. The algorithm implemented is:

1. Take a resample with replacement from the stock and recruit pairs
2. Fit every stock-recruit model under consideration and store the AIC of each
3. Retain the parameter estimates from the best model
4. Repeat

This process provides a robust way to average over several models, as long as the bootstrap resampling procedure provides an adequate approximation to the empirical distribution of the stock and recruit pairs. The arguments to the fitting function are

```
args(eqsr_fit)

function (stk, nsamp = 5000, models = c("ricker", "segreg", "bevholt"),
    method = "Buckland", id.sr = NULL, remove.years = NULL, delta = 1.3,
    nburn = 10000)
NULL
```

Here:

- `stk` is an `FLStock` object

- `nsamp` is the number of simulations to run (often referred to as iterations)
- `models` is the models to average over (any of the combination of these can be supplied, including only a single model)
- `method` the method used (only Buckland as of now)
- `id.sr` placeholder if one wants to name the fit
- `remove.years` is used to remove years from the fit
- `delta` and `nburn` are related to an MCMC based fitting procedure (not implemented yet)

The results from the fitting process are returned to the user as a list:

```
str(FIT, 2, give.attr=FALSE)

## List of 6
##  $ sr.sto:'data.frame': 1000 obs. of  4 variables:
##   ..$ a    : num [1:1000] 6.58 5.71 5.17 5.92 6.06 ...
##   ..$ b    : num [1:1000] 1.79e+05 1.66e+05 1.79e+05 2.76e+05 4.44e-07 ...
##   ..$ cv   : num [1:1000] 0.48 0.449 0.443 0.411 0.434 ...
##   ..$ model: chr [1:1000] "segreg" "segreg" "segreg" "segreg" ...
##  $ sr.det:'data.frame': 3 obs. of  6 variables:
##   ..$ a    : num [1:3] 6.3 6.09 6.25
##   ..$ b    : num [1:3] 8.54e-07 1.77e+05 8.57e-07
##   ..$ cv   : num [1:3] 0.482 0.463 0.482
##   ..$ model: chr [1:3] "ricker" "segreg" "bevholt"
##   ..$ n    : int [1:3] 58 918 24
##   ..$ prop : num [1:3] 0.058 0.918 0.024
##  $ pRec  : num [1:1000, 1:49] 143712 124630 113032 129273 131144 ...
##  $ stk   :Formal class 'FLStock' [package "FLCore"] with 20 slots
##  $ rby   :'data.frame': 49 obs. of  6 variables:
##   ..$ year    : int [1:49] 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 ...
##   ..$ rec     : num [1:49] 845768 1067681 1375049 1274418 654744 ...
##   ..$ ssb     : num [1:49] 153638 165830 205112 228117 252019 ...
##   ..$ fbar    : num [1:49] 0.478 0.509 0.541 0.556 0.609 ...
##   ..$ landings: num [1:49] 109409 117035 167733 207369 235070 ...
##   ..$ catch   : num [1:49] 128686 130740 210237 259416 276387 ...
##  $ id.sr : chr "North Sea cod"
```

where

- `sr.sto` is the the (joint) stochastic distribution of the estimated model and parameters. The number of rows of the data frame is equivalent to the value supplied to `nsamp` in the `eqsr_fit` function.
- `sr.det` is the conventional determinimstic predictive estimate. The `n` indicates the number of times a particular function is drawn in the stochastic sample and the `prop` the proportion, given `nsamp`.
- `pRec` contains the fitted parameters to the observed data
- `stk` retains the original `FLStock` object passed to the function.
- `rby` (results by year) contains a summary of the ssb and rec data used in the fitting as well as other stock summary information used later down the line
- `id.rs` is the user specified id

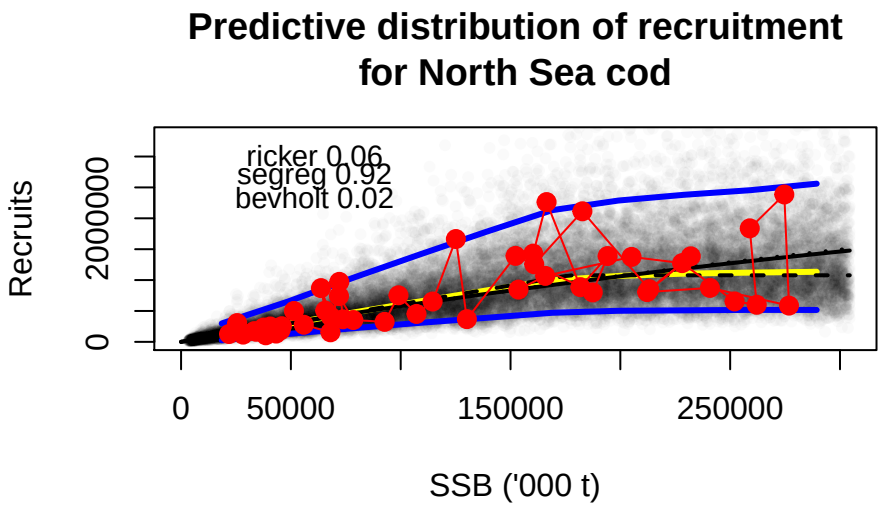
The contribution of each of the models can be obtained by printing out the `sr.det`:

```
xtable(FIT$sr.det,digits=c(0,2,-2,2,0,0,3))
```

	a	b	cv	model	n	prop
1	6.30	8.54E-07	0.48	ricker	58	0.058
2	6.09	1.77E+05	0.46	segreg	918	0.918
3	6.25	8.57E-07	0.48	bevholt	24	0.024

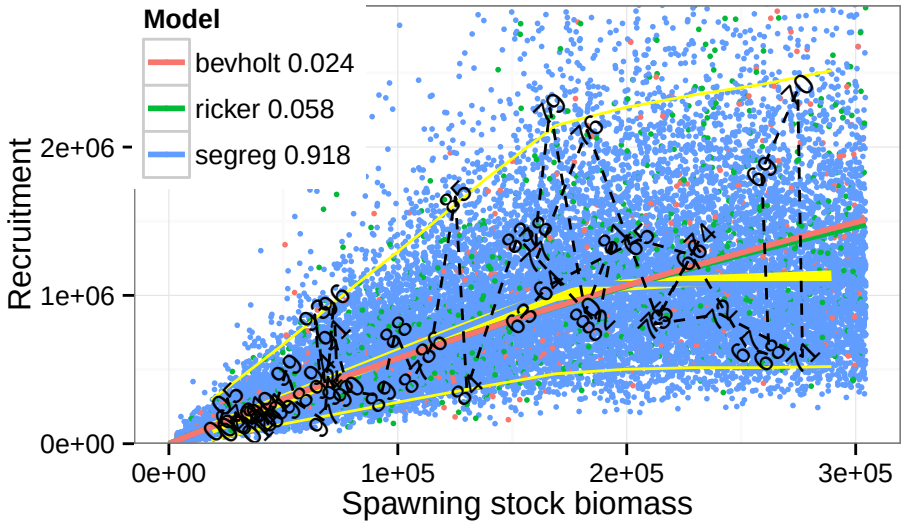
Here the a, b and cv are the estimated parameters from the deterministic fit for each model. The `n` and `prop` is a summary of the number and proportion that each model contributes to the final fit.
Again to obtain a plot one simply calls:

```
eqsr_plot(FIT,n=2e4)
```



The `n` supplied to the `eqsr_plot` stands here for the number of stochastic recruitment points desired to include in the plot. The various black dashed lines represent the best fit of the different recruitment models and the yellow and blue lines the median and 5% and 95% percentiles of the distributions of the stochastic recruits drawn from the models. The input data are represented by red points.
An alternative to the `base` plot is a `ggplot2` version (with too fancy colours :-):

```
eqsr_plot(FIT,n=2e4,ggPlot=TRUE)
```



Here the model fits are represented in different colours with the yellow lines indicating the 5%, 50% and 95% percentiles of the stochastic recruitment distribution. The input data are represented by text

indicating year class. The weight of each model in the final stochastic recruitment draw is indicated as a proportion in the legends and by different colours for the stochastic draws.

3.3 The simulation

Simulating forward is done using the `eqsim_run` function. The function takes as input the output from the `eqsr_fit` function. Simulations are run independently for each sample from the distribution of model and parameters. This is done for a range of $F_{advisory}$ values. For example if we scanned over 10 values of $F_{advisory}$ and had taken 2000 samples from the stock-recruit relationship then 20000 simulations would be run in total. These simulations are run for 200 years (default, specified with `Nrun`), and the last 50 years are retained to calculate summaries, like the proportion of times the stock crashes at a given $F_{advisory}$. It is important to note that each simulation is conditioned on a single stock recruit relationship with fixed parameters and cv.
Error is introduced within the simulations by generating process error about the constant stock-recruit fit, and by using variation in maturity, natural mortality, weight at age and selection estimates. Note that if there is no variability in these quantities in the stock object then no variability will be taken in to the simulations. The user can also specify using average values for these parameters.
The arguments to the simulation function are:

```
args(eqsim_run)

## function (fit, bio.years = c(2008, 2012), bio.const = FALSE,
##     sel.years = c(2008, 2012), sel.const = FALSE, Fscan = seq(0,
##         1, len = 20), Fcv = 0, Fphi = 0, Blim, Bpa, recruitment.trim = c(3,
##         -3), Btrigger = 0, Nrun = 200, process.error = TRUE,
##     verbose = TRUE, extreme.trim)
## NULL
```

where:

- `fit` is the output list from `eqsr_fit`
- `bio.years` is the start and end year from which to generate noise in maturity, M and weights.
- `bio.const` is a flag indicating if the average maturity, M and weights over the specified years should be used (`TRUE`) or not (`FALSE`).
- `sel.years` is the start and end year from which to generated noise in the selection at age
- `sel.const` is a flag indicating if the average selection over the specified years should be used (`TRUE`) or not (`FALSE`).
- `Fscan` is the range of $F_{advisory}$ values to scan over
- `Btrigger` is the location of a modifier of a HCR upon which $F_{advisory}$ becomes linearily reduced. If `Btrigger` is 0 (default) this is equivalent to a constant F-rule.
- `Fcv` The assessment error in the advisory year.
- `Fphi` The autocorrelation in assessment error
- `Blim` B_{lim}
- `Bpa` B_{pa}
- `Nrun` is the number of years to simulate forward (fixed for now is that the last 50 years from those are used for summarising equilibrium conditions)
- `process.error` allows the simulations to be run using the predictive distribition of recruitment or the mean recruitment
- `verbose` controls if progress bar is displayed during the simulation
- `extreme.trim` A numerical vector of length 2 containing the lower and upper percentiles. If specified, recruitement values outside this range are trimmed (ignored).

The results from the simulation process are returned to the user as a list

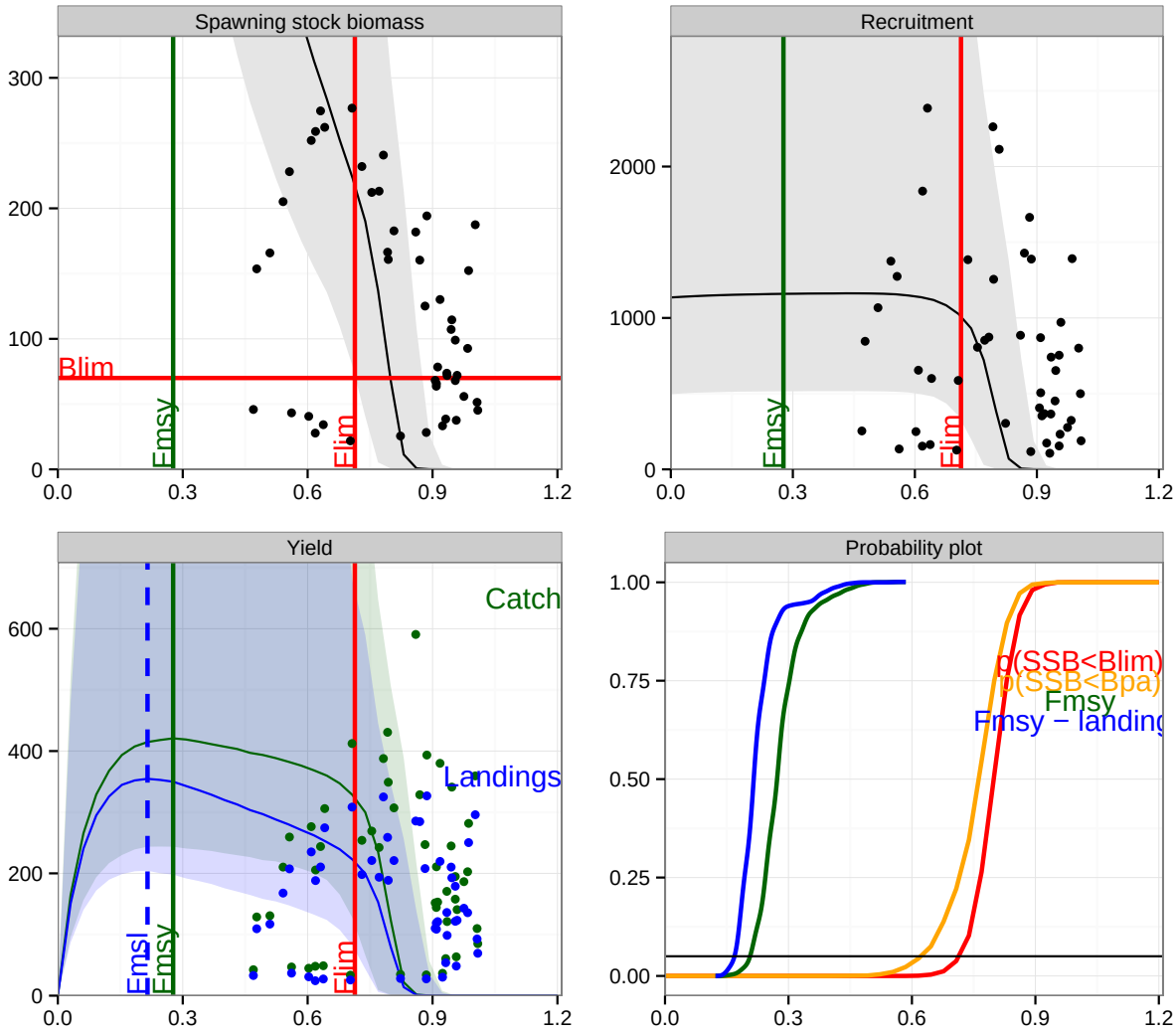
```
str(SIM, 2, give.attr = FALSE)
```

```
## List of 8
##  $ ibya    :List of 7
##   ..$ Mat  : num [1:7, 1:5] 0.01 0.05 0.23 0.62 0.86 1 1 0.01 0.05 0.23 ...
##   ..$ M    : num [1:7, 1:5] 1.039 0.696 0.487 0.232 0.2 ...
##   ..$ Fprop: Named num [1:7] 0 0 0 0 0 0 0
##   ..$ Mprop: Named num [1:7] 0 0 0 0 0 0 0
##   ..$ west : num [1:7, 1:5] 0.33 0.904 1.971 3.834 5.692 ...
##   ..$ weca : num [1:7, 1:5] 0.33 0.904 1.971 3.834 5.692 ...
##   ..$ sel  : num [1:7, 1:5] 0.246 0.795 1.087 1.118 1.142 ...
##  $ rby     :'data.frame': 49 obs. of  6 variables:
##   ..$ year    : int [1:49] 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 ...
##   ..$ rec     : num [1:49] 845768 1067681 1375049 1274418 654744 ...
##   ..$ ssb     : num [1:49] 153638 165830 205112 228117 252019 ...
##   ..$ fbar    : num [1:49] 0.478 0.509 0.541 0.556 0.609 ...
##   ..$ landings: num [1:49] 109409 117035 167733 207369 235070 ...
##   ..$ catch   : num [1:49] 128686 130740 210237 259416 276387 ...
##  $ rbp     :'data.frame': 160 obs. of  10 variables:
##   ..$ Ftarget : num [1:160] 0 0.0308 0.0615 0.0923 0.1231 ...
##   ..$ variable: chr [1:160] "Recruitment" "Recruitment" "Recruitment" "Recruitment" ...
##   ..$ p025    : num [1:160] 419839 428184 432069 435451 437914 ...
##   ..$ p05     : num [1:160] 496458 501939 506344 508486 510504 ...
##   ..$ p25     : num [1:160] 809012 812153 815043 817365 819201 ...
##   ..$ p50     : num [1:160] 1135311 1139997 1144264 1147753 1150500 ...
##   ..$ p75     : num [1:160] 1631315 1640186 1646286 1653743 1659540 ...
##   ..$ p95     : num [1:160] 3891780 4070696 4228494 4480689 4669759 ...
##   ..$ p975    : num [1:160] 25579487 26321607 26655710 28084605 29198515 ...
##   ..$ Mean    : num [1:160] NA NA NA NA NA NA NA NA NA NA ...
##  $ Blim    : num 70000
##  $ Bpa     : num 150000
##  $ Refs    : num [1:6, 1:7] 7.13e-01 NA 3.23e+05 NA 2.17e+05 ...
##  $ pProfile:'data.frame': 1104 obs. of  3 variables:
##   ..$ Ftarget : num [1:1104] 0.123 0.124 0.125 0.125 0.126 ...
##   ..$ value   : num [1:1104] 1.57e-06 3.60e-06 6.22e-06 9.55e-06 1.38e-05 ...
##   ..$ variable: chr [1:1104] "pFmsyCatch" "pFmsyCatch" "pFmsyCatch" "pFmsyCatch" ...
##  $ id.sim  : chr "North Sea cod"
```

where

- `ibya` (input by year and age) contains the biological and fisheries input data.
- `rby` (results by year) contains the stock summary data.
- `rbp` contains the 0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975 percentiles of the simulations of SSB, catch, landings and recruitment for each Fscan value.
- `Blim` B_{lim} input value
- `Bpa` B_{pa} input value
- `Refs` Calculated reference points
- `pProfile` The probability profiles for a given target F for B_{lim}, B_{pa} and F_{msy} (both for catch and landings).

```
eqsim_ggplot(SIM,1000)
```



4 plotMSY

Documentation is pending, further coding needed ...