Examples of using the msy package

Einar Hjörleifsson and Colin Millar

January 19, 2014

- R version 3.0.1 (2013-05-16), `x86_64-redhat-linux-gnu`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils
- Other packages: data.table 1.8.8, FLCore 2.5.20131114, ggplot2 0.9.3.1, gridExtra 0.9.1, knitr 1.2, lattice 0.20-15, lubridate 1.3.0, MASS 7.3-27, mgcv 1.7-27, msy 0.1.3, nlme 3.1-110, plyr 1.8, RColorBrewer 1.0-5, reshape2 1.2.2, scales 0.2.3, scam 1.1-6, stringr 0.6.2, xtable 1.7-1
- Loaded via a namespace (and not attached): colorspace 1.2-2, dichromat 2.0-0, digest 0.6.3, evaluate 0.4.4, formatR 0.8, gtable 0.1.2, labeling 0.2, Matrix 1.0-12, munsell 0.4.2, proto 0.3-10, stats4 3.0.1, tools 3.0.1

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

1 Installation

The developmental page for the msy package is located on github. To install the msy package the best way is to install Hadley Wickams devtools and use the function install_github. If you are using windows you will also need to install Rtools.exe which is a collection of software which enables you to compile R packages from source code. 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.

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

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:

```r
library(msy)
```

Besides functions the package comes with the following data:

- codNS: FLStock object of the North Sea cod
- codEB: FLStock object of the Eastern Baltic cod
- codWB: FLStock object of the Western Baltic cod
- ...

The current version of the msy - package implements two methods that go under the working names EqSim and plotMSY. These two approaches are described in the following sections (plotMSY not yet implemented).

2 EqSim

EqSim is a stochastic equilibrium reference point software that provides MSY reference points based on the equilibrium distribution of stochastic projections. Productivity parameters (i.e. year vectors for natural mortality, weights-at-age, maturities, and selectivity) are re-sampled at random from the e.g. last 3-5 years of the assessment (although there may be no variability in these values). 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 are emulated by applying a user-specified error (CV and autocorrelation) to the intended target fishing mortality.

2.1 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. A convenient way to store this set up information is to contain it in a list:

```r
codsetup <- list(
  data = codNS,
  wt.years = c(2008, 2012),
  sel.years = c(2008, 2012),
  Fscan = seq(0, 1.2, len = 40),
  flgsel = 1,
  flgmatwt = 1,
  Fcv = 0.25,
  Fphi = 0.30,
  Bpa = 150000,
  Blim = 70000,
  Btrigger = 150000,
  verbose = FALSE)
```

The eqsim approach consists of three components:

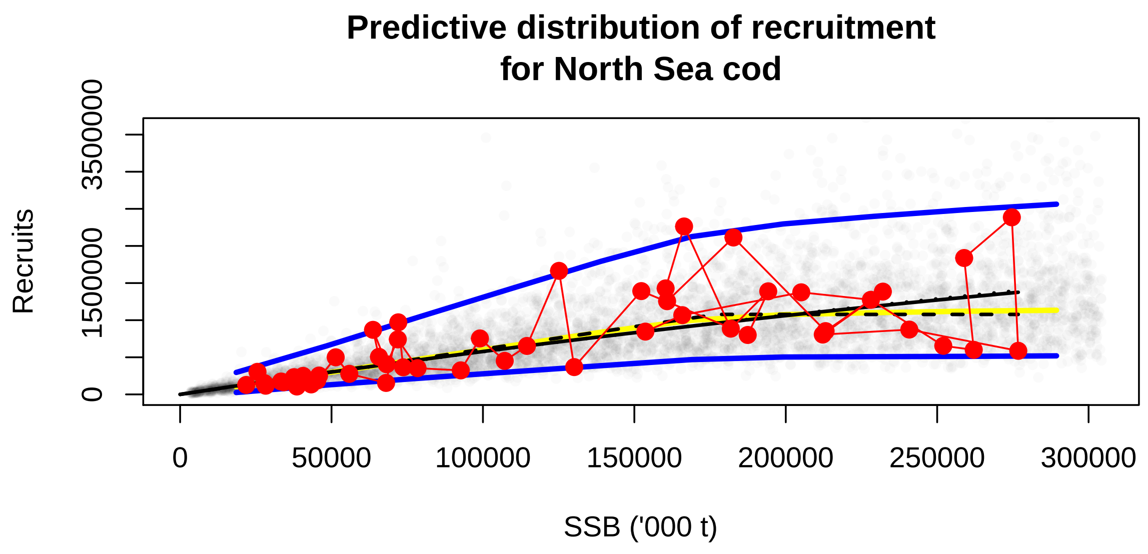
1. Estimate the stock recruitment relationship
2. Simulate a stock to equilibrium and continue simulating for some years
3. Calculate reference points from the simulated stock at equilibrium

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

```
codsg <-
  within(codsetup,
{
  fit <- eqsr_fit(data, nsamp = 2000, models = c("ricker", "segreg", "bevholt"))
  sim <- eqsim_run(fit, wt.years = wt.years, sel.years = sel.years, Fscan = Fscan,
               flgsel=flgsel,flgmatwt=flgmatwt,Fcv=Fcv,Fphi=Fphi,verbose = verbose)
  refLandings <- eqsim_plot0(sim, Blim = Blim, Bpa = Bpa, yield="landings",plot = FALSE)
  refCatch <- eqsim_plot0(sim, Blim = Blim, Bpa = Bpa, yield="catch",plot = FALSE)
})
```

The stock recruitment function can be plotted by:

```
with(codsg,eqsr_plot(fit))
```



Results based on catch

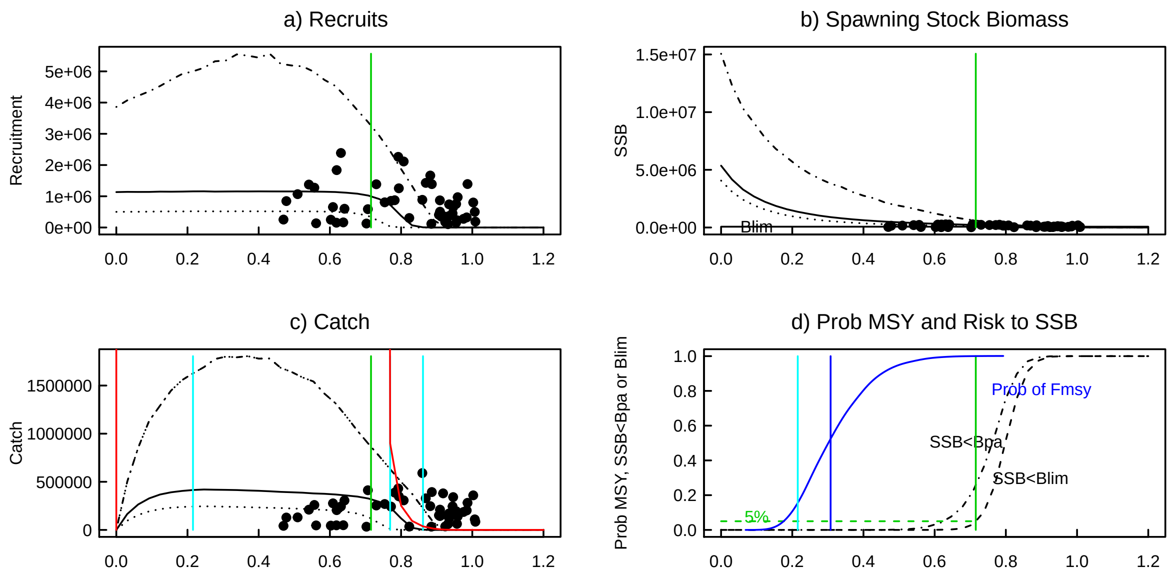
The reference points:

```
xtable(t(codsg$refCatch$Refs))
```

	F	SSB	catch
Flim	0.72	212938.84	318104.21
Flim10	0.74	186625.70	294702.74
Flim50	0.80	70017.95	127231.96
MSY:median	0.31	900173.52	416137.19
Maxmeanland	0.22	1362728.93	414837.17
FCrash5	0.77	137176.22	236182.24
FCrash50	0.86	989.64	2009.61

And summary plots for catch are got by calling the Eqplot function again:

```
with(codsg, eqsim_plot0(sim, Blim = Blim, Bpa = Bpa, yield="catch", plot = TRUE))
```



Results based on landings

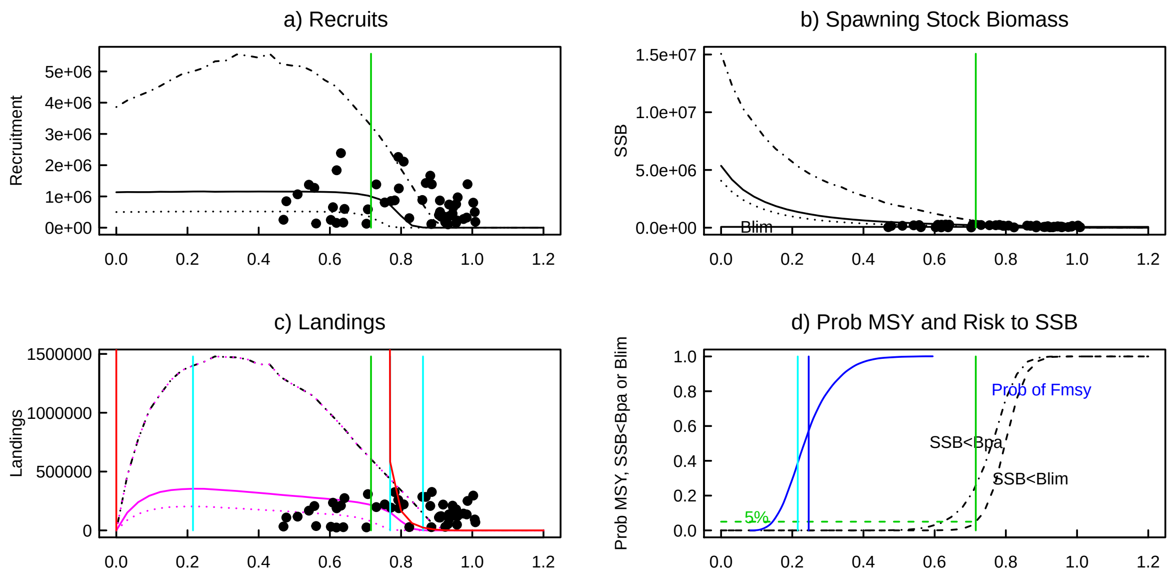
The reference points:

```
xtable(t(codsg$refLandings$Refs))
```

	F	SSB	landings
Flim	0.72	212938.84	214981.83
Flim10	0.74	186625.70	196484.21
Flim50	0.80	70017.95	82416.36
MSY:median	0.25	1184059.80	353577.45
Maxmeanland	0.22	1362728.93	353890.22
FCrash5	0.77	137176.22	153247.70
FCrash50	0.86	989.64	1287.21

And summary plots for landings are got by calling the Eqplot function again:

```
with(codsg, eqsim_plot0(sim, Blim = Blim, Bpa = Bpa, yield="landings",plot = TRUE))
```



These are the main functions of the EqSim approach. The following sections will cover each step in more detail.

2.2 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 all happens in 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`. 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", runid = NULL, remove.years = NULL, delta = 1.3,
##     nburn = 10000)
## NULL
```

Where `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), `remove.years` is used to remove years from the fit, `delta` and `nburn` are related to an MCMC based fitting procedure that is not complete, and `runid` is an opportunity for the user to name the fit.
The fit:

```
FIT <- eqsr_fit(codNS,nsamp=2000)
```

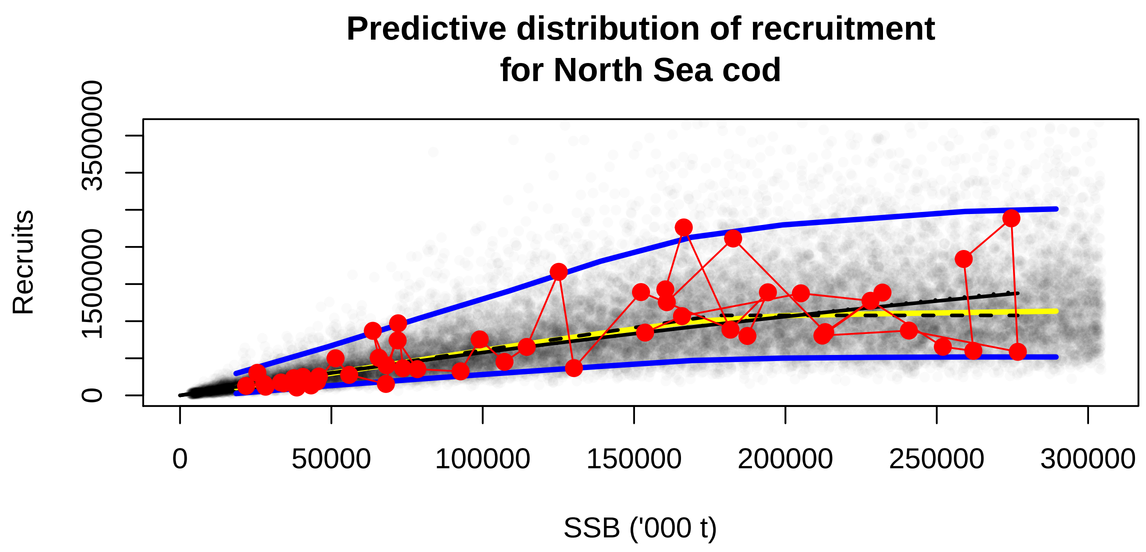
The results from the fitting process are returned to the user as a list:

```
str(FIT, 2)

## List of 6
##  $ fit   :'data.frame': 2000 obs. of  4 variables:
##   ..$ a    : num [1:2000] 6.71 6.27 5.82 5.78 5.71 ...
##   ..$ b    : num [1:2000] 160817 182729 187491 197343 182729 ...
##   ..$ cv   : num [1:2000] 0.494 0.427 0.425 0.442 0.45 ...
##   ..$ model: chr [1:2000] "segreg" "segreg" "segreg" "segreg" ...
##  $ pred  : num [1:2000, 1:49] 146637 136883 127120 126238 124652 ...
##  $ fits  :'data.frame': 3 obs. of  4 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"
##  $ data  :'data.frame': 49 obs. of  3 variables:
##   ..$ rec : num [1:49] 845768 1067681 1375049 1274418 654744 ...
##   ..$ ssb : num [1:49] 153638 165830 205112 228117 252019 ...
##   ..$ year: num [1:49] 1964 1965 1966 1967 1968 ...
##  $ stknam: chr "North Sea cod"
##  $ stk   :Formal class 'FLStock' [package "FLCore"] with 20 slots
```

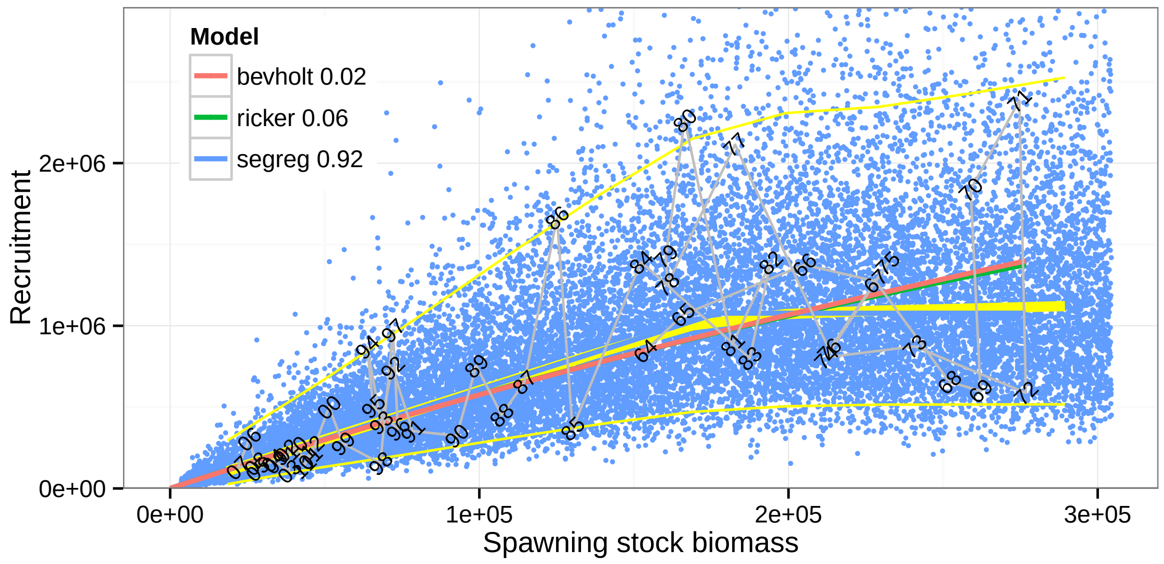
where `fit` is a sample from the joint distribution of the estimated model and parameters, `pred` is an associated sample from the predictive distribution of recruitment based on the model and parameters in fit and the SSB values used , `fits` contains the fitted parameters to the observed data, `data` contains the data used, `stknam` holds the name of the stock for plotting later on, and `stk` retains the original `FLStock` object.
To obtain a plot one simply calls:

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



The n stands here for the number of stochastic recruitment points desired on the plot. The various black dashed lines represent the best fit of the three 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 three 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 three 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. The numerical values can be accessed via:

```r
xtable(table(FIT$fit$model)/length(FIT$fit$model))
```

	V1
bevholt	0.02
ricker	0.06
segreg	0.92

2.3 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 200 samples from the stock-recruit relationship then 2000 simulations would be run in total. These simulations run for 200 years say, 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 (including CV).
Error is introduced within the simulations by generating process error about the constant stock-recruit fit, and by using historical variation in Maturity, natural mortality, weight at age etc. 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 arguments to the simulation function are

```r
args(eqsim_run)

## function (fit, Nrun = 200, wt.years = c(2008, 2012), sel.years = c(2008,
##     2012), Fscan = seq(0, 1, len = 20), process.error = TRUE,
##     verbose = TRUE, Btrigger = 0, Fphi = 0, Fcv = 0, flgsel = 0,
##     flgmatwt = 0)
## NULL
```

where fit is the output from fitModels, Nrun is the number of years to simulate forward, fixed for now is that the last 50 are used for summarising erqualibrium conditions, wt.years is the start and end year from which to generate the noise in weight at age etc., Fscan is the range of Fbar values to scan over, process.error allows the simulations to be run using the predictive distribition of recruitment or the mean recruitment, verbose = TRUE shows a progress bar to the user, and Btrigger is the location of a modifier of a HCR upon which $F_{advisory}$ becomes linearliy reduced. If Btrigger is 0 (default) this is equivalent to a constant F-rule.

```r
SIM <- eqsim_run(fit=FIT,wt.year=c(2008,2012),sel.years=c(2008,2012),verbose=FALSE)
```

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

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

## List of 7
##  $ Percentiles:List of 4
##   ..$ ssbs: num [1:7, 1:20] 3770206 4018761 4747157 5345496 6171548 ...
##   ..$ cats: num [1:7, 1:20] 0 0 0 0 0 ...
##   ..$ lans: num [1:7, 1:20] 0 0 0 0 0 ...
##   ..$ recs: num [1:7, 1:20] 416379 495981 804034 1123921 1595660 ...
##  $ rby        :List of 5
##   ..$ ferr : num [1:20, 1:50, 1:2000] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ ssbsa: num [1:20, 1:50, 1:2000] 5668837 3105987 2162530 1855039 1108691 ...
##   ..$ catsa: num [1:20, 1:50, 1:2000] 0 217100 306119 378947 338362 ...
##   ..$ lansa: num [1:20, 1:50, 1:2000] 0 190276 272750 343553 274076 ...
##   ..$ recsa: num [1:20, 1:50, 1:2000] 1028384 949785 1113022 2723539 1510053 ...
##  $ 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] 1.038 0.697 0.489 0.233 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] 0.322 0.918 2.292 4.108 6.065 ...
##   ..$ weca : num [1:7, 1:5] 0.322 0.918 2.292 4.108 6.065 ...
##   ..$ sel  : num [1:7, 1:5] 0.248 0.794 1.076 1.13 1.154 ...
##  $ land.cat   : num [1:7, 1:5] 0.0781 0.3471 0.6234 0.895 0.9759 ...
##  $ Fscan      : num [1:20] 0 0.0526 0.1053 0.1579 0.2105 ...
##  $ stk        :Formal class 'FLStock' [package "FLCore"] with 20 slots
##  $ data       :'data.frame': 49 obs. of  3 variables:
##   ..$ rec : num [1:49] 845768 1067681 1375049 1274418 654744 ...
##   ..$ ssb : num [1:49] 153638 165830 205112 228117 252019 ...
##   ..$ year: num [1:49] 1964 1965 1966 1967 1968 ...
```

where

- `Pecentile` 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.
- `rby` (results by year) contains the assessment error (ferr), SSB (ssbsa), catch (catsa), landings (lansa) and recruitment (recsa) for each Fscan value the final 50 years for each simulation.
- `ibya` (input by year and age) contains the biological and fisheries input data.
- `land.cat` the ratio of landings over catch (TODO: move to ibya)
- `Fscan` The $F_{advisory}$ that was tested
- `stk` The original `FLStock` object
- `data` The original stock and recruitment data

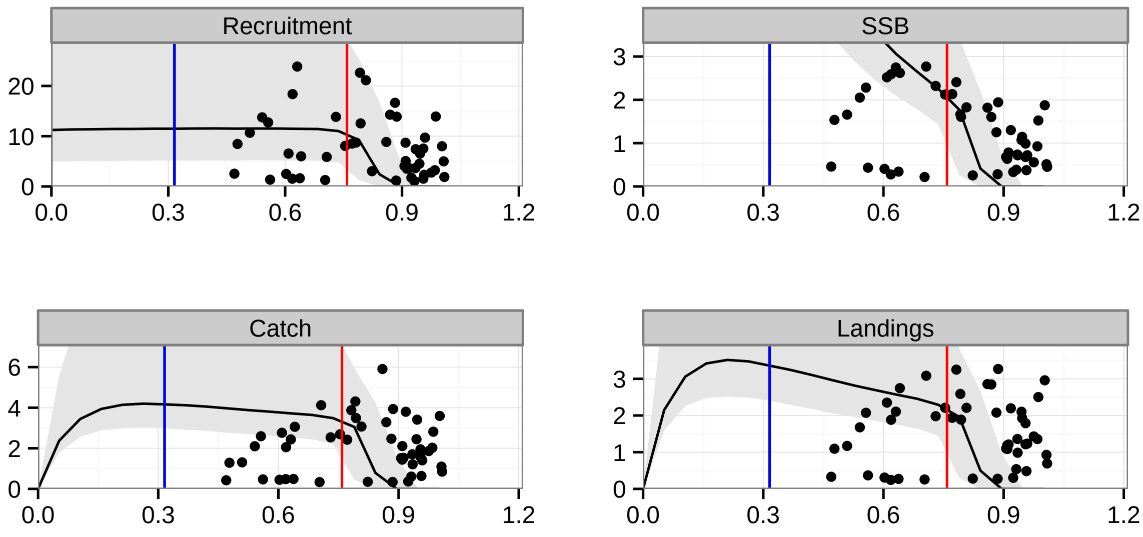
Summarise simulations:

```
catchSummary <- eqsim_summary(SIM, Blim=70000, Bpa=150000, yield="catch")
xtable(t(catchSummary$Refs))
```

	F	SSB	catch
Flim	0.76	205377.08	330484.25
Flim10	0.78	184312.03	311645.32
Flim50	0.83	67416.93	122434.39
MSY:median	0.32	885141.39	416335.34
Maxmeanland	0.32	885141.39	416335.34
FCrash05	0.84	41294.37	78899.41
FCrash50	0.89	344.47	726.49

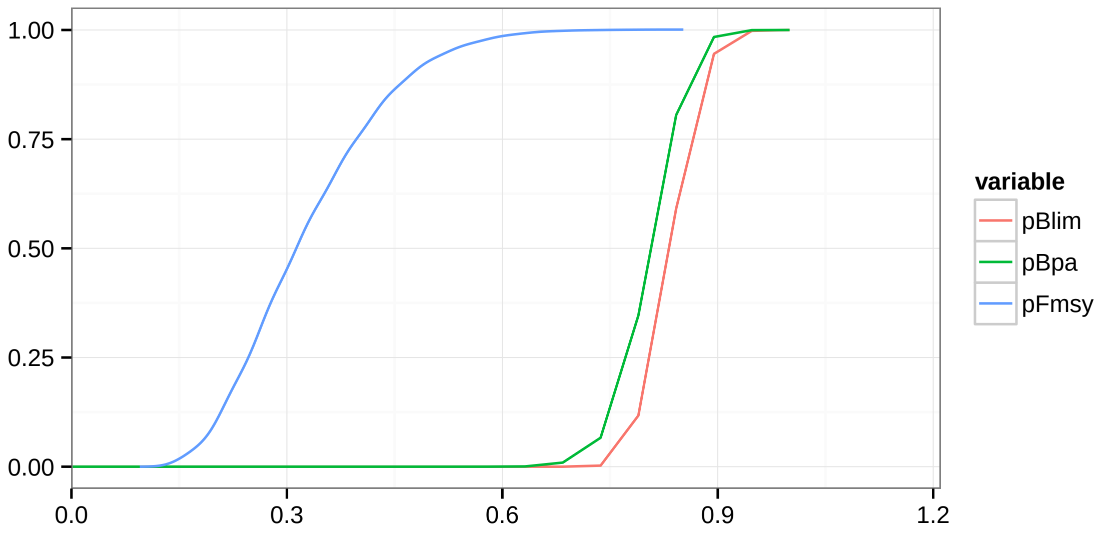
Plot simulations:

```
p <- eqsim_plot(catchSummary , Scale=1e5, plotit=TRUE)
```



Probability profile (for catch):

```
p$plotProbs
```


variable
pBlim
pBpa
pFmsy