Examples of using the msy package

Colin Millar

December 10, 2013

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, which is also installed from github.

```
library(devtools)
install_github("msy", "wgmg")
install_github("FLCore", "flr")
```

1 Exploring Maximum Sustainable Yeild of North Sea Cod using EqSim

First load the library and load in some data. The north sea cod data has been preloaded as an FLStock object. This is the EqSim method requires an FLStock to work.

```
library(msy)
```

```
load("data/codNS.rData")
```

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 chose 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 referece points. A convenient way to store this set up information is to contain it in a list

```
codsetup <- list(
           data = codNS,
           wt.years = c(2008, 2012),
           Fscan = seq(0, 1.5, len = 40),
           Bpa = 150000,
           Blim = 70000,
           Btrigger = 150000,
           verbose = FALSE) # set verbose to TRUE if you want to see simulation progress
```

1.1 Segmented regression worked example

Using the settings defined in the previous section we can run the three components of the simulation approach to estimating MSY reference points. The components are:

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 is done in one go with the code

```
codsg <- within(codsetup, {
    fit <- fitModels(data, nsamp = 2000, model = "segreg")
    sim <- EqSim(fit, wt.years = wt.years, Fscan = Fscan, verbose = verbose)
    ref <- Eqplot(sim, fit, Blim = Blim, Bpa = Bpa, plot = FALSE)
})
```

The reference points can be accessed by

```
t(codsg$ref$Refs)

##                   F      SSB  Catch
## Flim         0.7723 193985.6 322413
## Flim10       0.7817 181018.7 305683
## Flim50       0.8330  68887.8 125793
## MSY:median   0.3462 765573.4 407897
## Maxmeanland  0.3077 891186.0 410688
## FCrash5      0.8077 145367.5 259684
## FCrash50     0.8846    827.4   1714
```

And summary plots are got by calling the Eqplot function again

```
with(codsg, Eqplot(sim, fit, Blim = Blim, Bpa = Bpa))
```

The stock recruitment fit is returned by fitModels and is plotted using the SRplot function

```
SRplot(codsg$fit)
```

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

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 fitModels function. The fitModels 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 fitModelsBuck. The algorithm implented 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 resampleing procedure prevides an adequate approximation to the empirical distribution of the stock and recruit pairs. The arguments to the fitting function are

```
args(fitModels)

## 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, nsamp is the number of simulations to run a simulation for, models is the models to average over (one model may be supplied), remove.years is used to remove outlying years from the fit, delta and nburn are related to an MCMC based fitting procedure that is not complete, and runid is an oportunity for the user to name the fit.
The results from the fitting process are returned to the user as a list

```
str(codsg$fit, 2)

## List of 6
##  $ fit   :'data.frame': 2000 obs. of  4 variables:
##   ..$ a    : num [1:2000] 6.22 6.33 4.8 5.59 4.96 ...
##   ..$ b    : num [1:2000] 166604 161872 240825 190059 285356 ...
##   ..$ cv   : num [1:2000] 0.438 0.432 0.419 0.452 0.407 ...
##   ..$ model: chr [1:2000] "segreg" "segreg" "segreg" "segreg" ...
##  $ pred  : num [1:2000, 1:49] 135847 138191 104869 122097 108367 ...
##  $ fits  :'data.frame': 1 obs. of  4 variables:
##   ..$ a    : num 6.09
##   ..$ b    : num 177048
##   ..$ cv   : num 0.463
##   ..$ model: Factor w/ 1 level "segreg": 1
##  $ data  :'data.frame': 49 obs. of  3 variables:
##   ..$ rec : num [1:49] 845768 1067681 1375049 1274418 654744 ...
##   ..$ ssb : num [1:49] 153588 165474 204933 228076 252322 ...
##   ..$ year: num [1:49] 1964 1965 1966 1967 1968 ...
##  $ stknam: chr "North Sea/Skagerrak/Eastern Channel Cod, Tuning data. INCLUDES DISCARDS"
##  $ 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 accosiated 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 origional FLStock object.

Simulating forward is done using the EqSim function. The function takes as input the output from the fitModel function. Simulations are run independently for each sample from the distribution of model and parameters. This is done for a range of Fbar values. For example if we scanned over 10 values of Fbar 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 level of F. 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 randomly 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

```
args(EqSim)

## function (fit, Nrun = 200, wt.years = c(2007, 2011), Fscan = seq(0,
##     1, len = 20), process.error = TRUE, verbose = TRUE, Btrigger = 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 the ramp in a simple HCR, if Btrigger is 0 this is like having no HCR present.

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

```
str(codsg$sim, 2)

## List of 14
##  $ ssbs : num [1:7, 1:40] 3594589 3760648 4366111 4904481 5650432 ...
##  $ cats : num [1:7, 1:40] 0 0 0 0 0 ...
##  $ recs : num [1:7, 1:40] 444087 516098 819854 1131301 1569357 ...
##  $ ssbsa: num [1:40, 1:50, 1:2000] 4478184 3179649 2533362 2393653 1803408 ...
##  $ catsa: num [1:40, 1:50, 1:2000] 0 153849 256893 366760 383617 ...
##  $ recsa: num [1:40, 1:50, 1:2000] 905941 1319530 548875 885930 2070334 ...
##  $ Mat  : Named num [1:6] 0.01 0.05 0.23 0.62 0.86 1
##   ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
##  $ M    : Named num [1:6] 1.038 0.697 0.489 0.233 0.2 ...
##   ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
##  $ Fprop: Named num [1:6] 0 0 0 0 0 0
##   ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
##  $ Mprop: Named num [1:6] 0 0 0 0 0 0
##   ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
##  $ west : num [1:6, 1:5] 0.322 0.918 2.292 4.108 6.065 ...
##  $ weca : num [1:6, 1:5] 0.322 0.918 2.292 4.108 6.065 ...
##  $ sel  : num [1:6, 1:5] 0.248 0.794 1.076 1.13 1.154 ...
##  $ Fscan: num [1:40] 0 0.0385 0.0769 0.1154 0.1538 ...
```

where ssbs, cats and recs contains the 0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975 percentiles of the simulations of SSB, catch and recruitment for each Fscan value. ssbsa, catsa and recsa contain, for each Fscan value the final 50 years for each simulation of SSB, catch and recruitment. The remaining values show the range of variability included for maturity, natural mortality etc. and the Fbar values used.

Reference points are derived from the output of EqSim.

1.2 Beverton and Holt

```
codbh <- within(codsetup, {
    fit <- fitModels(data, nsamp = 2000, model = "bevholt")
    sim <- EqSim(fit, wt.years = wt.years, Fscan = Fscan, verbose = verbose)
    ref <- Eqplot(sim, fit, Blim = Blim, Bpa = Bpa, plot = FALSE)
})
```

1.3 Ricker

```
codrk <- within(codsetup, {
    fit <- fitModels(data, nsamp = 2000, model = "bevholt")
    sim <- EqSim(fit, wt.years = wt.years, Fscan = Fscan, verbose = verbose)
    ref <- Eqplot(sim, fit, Blim = Blim, Bpa = Bpa, plot = FALSE)
})
```

1.4 Model averaging

```
codall <- within(codsetup, {
    fit <- fitModels(data, nsamp = 2000, model = c("segreg", "bevholt", "ricker"))
    sim <- EqSim(fit, wt.years = wt.years, Fscan = Fscan, verbose = verbose)
    ref <- Eqplot(sim, fit, Blim = Blim, Bpa = Bpa, plot = FALSE)
})
```

1.5 Model averaging with extra bits

```
codall_variations <- within(codall, {
    # simulate without process error in the recruitment predictions
    sim2 <- EqSim(fit, wt.years = wt.years, Fscan = Fscan, process.error = FALSE,
        verbose = verbose)
    # simulate with process error (i.e. using predictive distrution of
    # recruitment) and include a simple HCR
    sim3 <- EqSim(fit, wt.years = wt.years, Fscan = Fscan, Btrigger = Btrigger,
        verbose = verbose)
    # simulate without process error (i.e. using model and parameter error only,
    # not including 'observation' error) and include a simple HCR
    sim4 <- EqSim(fit, wt.years = wt.years, Fscan = Fscan, process.error = FALSE,
        Btrigger = Btrigger, verbose = verbose)
    # now calculate the reference points for each simulation
    ref2 <- Eqplot(sim2, fit, Blim = Blim, Bpa = Bpa, plot = FALSE)
    ref3 <- Eqplot(sim3, fit, Blim = Blim, Bpa = Bpa, plot = FALSE)
    ref4 <- Eqplot(sim4, fit, Blim = Blim, Bpa = Bpa, plot = FALSE)
})
```

1.6 A summary of the simulations

```
# out <- data.frame(statistic = 'Med', method = rep(paste0('M3',1:4), 3),
# what = rep(c('FMSY','BMSY','MSY'), each = 4), value = c(refNS $
# Refs['F',4], refNS2 $ Refs['F',4], refNS3 $ Refs['F',4], refNS4 $
# Refs['F',4], refNS $ Refs['SSB',4], refNS2 $ Refs['SSB',4], refNS3 $
# Refs['SSB',4], refNS4 $ Refs['SSB',4], refNS $ Refs['Catch',4], refNS2 $
# Refs['Catch',4], refNS3 $ Refs['Catch',4], refNS4 $ Refs['Catch',4]))

# out $ value[-(1:4)] <- out $ value[-(1:4)] / 1000


# out <- do.call(rbind, modout) out $ model <- rep(c('S','B','R','all'),
# each = 12) write.table(out, file = 'colins.csv', sep = ',', row.names =
# FALSE)
```

A Code

The EqSim function takes a fitted model

```
fitModels

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

```
## {
##     dms <- dims(stk)
##     rage <- dms$min
##     if (rage == 0) {
##         data <- data.frame(rec = stock.n(stk)[1, drop = TRUE],
##             ssb = ssb(stk)[drop = TRUE], year = with(dms, 1:year +
##                 minyear - 1))
##     }
##     else {
##         data <- data.frame(rec = stock.n(stk)[1, -seq(rage),
##             drop = TRUE], ssb = ssb(stk)[1, seq(dms$year - rage),
##             drop = TRUE], year = with(dms, (rage + 1):year +
##             minyear - 1))
##     }
##     if (!is.null(remove.years)) {
##         data$ssb[data$year %in% remove.years] <- NA
##     }
##     data <- data[complete.cases(data), ]
##     if (is.null(runid))
##         runid <- name(stk)
##     method <- match.arg(method, c("Buckland", "Simmonds", "King",
##         "Cadigan"))
##     if (!is.character(models))
##         stop("models arg should be character vector giving names of stock recruit models")
##     if (method == "Buckland") {
##         c(fitModelsBuck(data, runid, nsamp, models), list(stk = stk))
##     }
##     else {
##         cat("The", method, "is not ready yet!  Working on it!\n")
##     }
## }
## <environment: namespace:msy>
```

```
fitModelsBuck

## function (data, runid, nsamp = 5000, models = c("ricker", "segreg",
##     "bevholt"), ...)
## {
##     nllik <- function(param, ...) -1 * llik(param, ...)
##     ndat <- nrow(data)
##     fit <- lapply(1:nsamp, function(i) {
##         sdat <- data[sample(1:ndat, replace = TRUE), ]
##         fits <- lapply(models, function(mod) nlminb(initial(mod,
##             sdat), nllik, data = sdat, model = mod, logpar = TRUE))
##         best <- which.min(sapply(fits, "[[", "objective"))
##         with(fits[[best]], c(a = exp(par[1]), b = exp(par[2]),
##             cv = exp(par[3]), model = best))
##     })
##     fit <- as.data.frame(do.call(rbind, fit))
##     fit$model <- models[fit$model]
##     pred <- t(sapply(seq(nsamp), function(j) exp(match.fun(fit$model[j])(fit[j,
##         ], sort(data$ssb)))))
##     fits <- do.call(rbind, lapply(models, function(mod) with(nlminb(initial(mod,
##         data), nllik, data = data, model = mod, logpar = TRUE),
##         data.frame(a = exp(par[1]), b = exp(par[2]), cv = exp(par[3]),
##             model = mod))))
```

```
##     list(fit = fit, pred = pred, fits = fits, data = data, stknam = runid)
## }
## <environment: namespace:msy>
```

```
SRplot

## function (fit)
## {
##     modset <- fit$fit
##     data <- fit$data
##     ssb <- data$ssb
##     rec <- data$rec
##     mn <- length(ssb)
##     minSSB <- min(ssb, max(ssb) * 0.05)
##     maxSSB <- max(ssb) * 1.1
##     maxrec <- max(rec * 1.5)
##     plot(ssb, rec, xlim = c(0, maxSSB), ylim = c(0, maxrec),
##         type = "n", xlab = "SSB ('000 t)", ylab = "Recruits",
##         main = paste("Predictive distribution of recruitment\nfor",
##             fit$stknam))
##     out <- do.call(rbind, lapply(sample(1:nrow(modset), 500),
##         function(i) {
##             fssb <- runif(500, minSSB, maxSSB)
##             FUN <- match.fun(modset$model[i])
##             frec <- exp(FUN(modset[i, ], fssb) + rnorm(500, sd = modset$cv[i]))
##             points(fssb, frec, pch = 20, col = paste0(grey(0),
##                 "05"), cex = 0.0625)
##             data.frame(ssb = fssb, rec = frec)
##         }))
##     out$grp <- with(out, floor(10 * (ssb - min(ssb))/(max(ssb) -
##         min(ssb) + 0.001)))
##     out$mid.grp <- with(out, (grp + 0.5)/10 * (max(ssb) - min(ssb)) +
##         min(ssb))
##     summ <- with(out, t(simplify2array(tapply(rec, grp, quantile,
##         c(0.5, 0.05, 0.95)))))
##     mid.grp <- sort(unique(out$mid.grp))
##     lines(mid.grp, summ[, 1], col = 7, lwd = 3)
##     lines(mid.grp, summ[, 2], col = 4, lwd = 3)
##     lines(mid.grp, summ[, 3], col = 4, lwd = 3)
##     x <- fit$fits
##     y <- seq(1, round(max(ssb)), length = 100)
##     sapply(1:nrow(x), function(i) lines(y, exp(match.fun(as.character(x$model[i]))(x[i,
##         ], y)), col = "black", lwd = 2, lty = i))
##     lines(ssb, rec, col = 10)
##     points(ssb, rec, pch = 19, col = 10, cex = 1.25)
## }
## <environment: namespace:msy>
```

```
EqSim

## function (fit, Nrun = 200, wt.years = c(2007, 2011), Fscan = seq(0,
##     1, len = 20), process.error = TRUE, verbose = TRUE, Btrigger = 0)
## {
##     btyr1 <- wt.years[1]
##     btyr2 <- wt.years[2]
```

```
##     flgsel <- 0
##     flgmatwt <- 0
##     keep <- min(Nrun, 50)
##     SR <- fit$fit
##     data <- fit$data
##     stk <- fit$stk
##     stk.win <- window(stk, start = btyr1, end = btyr2)
##     west <- matrix(stock.wt(stk.win), ncol = btyr2 - btyr1 +
##         1)
##     weca <- matrix(catch.wt(stk.win), ncol = btyr2 - btyr1 +
##         1)
##     sel <- matrix(harvest(stk.win), ncol = btyr2 - btyr1 + 1)
##     Fbar <- matrix(fbar(stk.win), ncol = btyr2 - btyr1 + 1)
##     sel <- sweep(sel, 2, Fbar, "/")
##     if (flgsel == 0) {
##         sel[] <- apply(sel, 1, mean)
##     }
##     if (flgmatwt == 0) {
##         west[] <- apply(west, 1, mean)
##         weca[] <- apply(weca, 1, mean)
##     }
##     Mat <- apply(mat(stk.win), 1, mean)[drop = TRUE]
##     M <- apply(m(stk.win), 1, mean)[drop = TRUE]
##     Fprop <- apply(harvest.spwn(stk.win), 1, mean)[drop = TRUE]
##     Mprop <- apply(m.spwn(stk.win), 1, mean)[drop = TRUE]
##     Nmod <- nrow(SR)
##     NF <- length(Fscan)
##     ages <- dims(stk)$age
##     ssby <- array(0, c(Nrun, Nmod))
##     Ny <- Fy <- WSy <- WCy <- Cy <- Wy <- array(0, c(ages, Nrun,
##         Nmod))
##     rsam <- array(sample(1:ncol(weca), Nrun * Nmod, TRUE), c(Nrun,
##         Nmod))
##     Wy[] <- c(weca[, c(rsam)])
##     R <- mean(data$rec)
##     ssbs <- cats <- recs <- array(0, c(7, NF))
##     pssb1 <- pssb2 <- array(0, NF)
##     ssbsa <- catsa <- recsa <- array(0, c(NF, keep, Nmod))
##     begin <- Nrun - keep + 1
##     if (verbose)
##         loader(0)
##     for (i in 1:NF) {
##         Fbar <- Fscan[i]
##         Zpre <- (Fbar * Fprop * sel[, rsam[1, ]] + M * Mprop)
##         Zpos <- (Fbar * (1 - Fprop) * sel[, rsam[1, ]] + M *
##             (1 - Mprop))
##         Zcum <- c(0, cumsum(Fbar * sel[c(1:ages, rep(ages, 49 -
##             ages)), rsam[1, ]] + M[c(1:ages, rep(ages, 49 - ages))]))
##         N1 <- R * exp(-unname(Zcum))
##         Ny[, 1, ] <- c(N1[1:(ages - 1)], sum(N1[ages:50]))
##         ssby[1, ] <- colSums(Mat * Ny[, 1, ] * west[, rsam[1,
##             ]]/exp(Zpre))
##         for (j in 2:Nrun) {
##             SSB <- ssby[j - 1, ]
##             if (process.error) {
##                 allrecs <- sapply(unique(SR$mod), function(mod) exp(match.fun(mod)(SR,
```

```
##                    SSB) + rnorm(Nmod, 0, SR$cv)))
##             }
##             else {
##                 allrecs <- sapply(unique(SR$mod), function(mod) exp(match.fun(mod)(SR,
##                   SSB)))
##             }
##             select <- cbind(seq(Nmod), as.numeric(factor(SR$mod,
##                 levels = unique(SR$mod))))
##             Ny[1, j, ] <- allrecs[select]
##             Fnext <- Fbar * pmin(1, SSB/Btrigger)
##             Zpre <- rep(Fnext, each = length(Fprop)) * Fprop *
##                 sel[, rsam[j, ]] + M * Mprop
##             Fy[, j - 1, ] <- rep(Fnext, each = ages) * sel[,
##                 rsam[j - 1, ]]
##             Ny[-1, j, ] <- Ny[1:(ages - 1), j - 1, ] * exp(-Fy[1:(ages -
##                 1), j - 1, ] - M[1:(ages - 1)])
##             Ny[ages, j, ] <- Ny[ages, j, ] + Ny[ages, j - 1,
##                 ] * exp(-Fy[ages, j - 1, ] - M[ages])
##             ssby[j, ] <- apply(array(Mat * Ny[, j, ] * west[,
##                 rsam[j, ]]/exp(Zpre), c(ages, Nmod)), 2, sum)
##             Cy[, j, ] <- Ny[, j - 1, ] * Fy[, j - 1, ]/(Fy[,
##                 j - 1, ] + M) * (1 - exp(-Fy[, j - 1, ] - M))
##         }
##         Cw <- Cy * Wy
##         Cat <- apply(Cw, 2:3, sum)
##         quants <- c(0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975)
##         ssbs[, i] <- quantile(ssby[begin:Nrun, ], quants)
##         cats[, i] <- quantile(Cat[begin:Nrun, ], quants)
##         recs[, i] <- quantile(Ny[1, begin:Nrun, ], quants)
##         ssbsa[i, , ] <- ssby[begin:Nrun, ]
##         catsa[i, , ] <- Cat[begin:Nrun, ]
##         recsa[i, , ] <- Ny[1, begin:Nrun, ]
##         if (verbose)
##             loader(i/NF)
##     }
##     list(ssbs = ssbs, cats = cats, recs = recs, ssbsa = ssbsa,
##         catsa = catsa, recsa = recsa, Mat = Mat, M = M, Fprop = Fprop,
##         Mprop = Mprop, west = west, weca = weca, sel = sel, Fscan = Fscan)
## }
## <environment: namespace:msy>
```

```
Eqplot
```

```
## function (sim, fit, Blim, Bpa = 1.4 * Blim, ymax = c(NA, NA,
##     NA), plot = TRUE)
## {
##     stk <- fit$stk
##     Nmod <- dim(sim$ssbsa)[3]
##     Nyrs <- dim(sim$ssbsa)[2]
##     Fscan <- sim$Fscan
##     catm <- apply(sim$catsa, 1, mean)
##     maxcatm <- which.max(catm)
##     catsam <- apply(sim$catsa, c(1, 3), mean)
##     maxpf <- apply(catsam, 2, which.max)
##     fmsy <- Fscan[maxpf]
##     msym <- mean(fmsy)
```

```
##     vcum <- median(fmsy)
##     fmsy.dens <- density(fmsy)
##     vmode <- fmsy.dens$x[which.max(fmsy.dens$y)]
##     pssb1 <- apply(sim$ssbsa > Blim, 1, mean)
##     pssb2 <- apply(sim$ssbsa > Bpa, 1, mean)
##     pp1 <- max(which(pssb1 > 0.95))
##     grad <- diff(Fscan[pp1 + 0:1])/diff(pssb1[pp1 + 0:1])
##     flim <- Fscan[pp1] + grad * (0.95 - pssb1[pp1])
##     maint <- fit$stknam
##     rec <- fit$data$rec
##     ssb <- fit$data$ssb
##     Catchs <- catch(stk)[, 1:length(ssb), drop = TRUE]
##     FbarO <- fbar(stk)[, 1:length(ssb), drop = TRUE]
##     recs <- sim$recs
##     ssbs <- sim$ssbs
##     cats <- sim$cats
##     ssbsa <- sim$ssbsa
##     NF <- length(Fscan)
##     pp1 <- max(which(pssb1 > 0.5))
##     grad <- diff(Fscan[pp1 + 0:1])/diff(pssb1[pp1 + 0:1])
##     flim50 <- Fscan[pp1] + grad * (0.5 - pssb1[pp1])
##     pp1 <- max(which(pssb1 > 0.9))
##     grad <- diff(Fscan[pp1 + 0:1])/diff(pssb1[pp1 + 0:1])
##     flim10 <- Fscan[pp1] + grad * (0.9 - pssb1[pp1])
##     maxcatm <- which.max(catm)
##     if (plot) {
##         op <- par(mfrow = c(2, 2), mar = c(2.5, 4, 1.5, 1), oma = c(0,
##             0, 0, 0), cex.axis = 0.75, tcl = 0.25, mgp = c(0,
##             0.25, 0), las = 1)
##         xmax <- max(Fscan)
##         y.max <- if (!is.na(ymax[1]))
##             ymax[1]
##         else max(recs[7, ], rec)
##         plot(Fscan, recs[7, ], type = "l", lty = 4, ylim = c(0,
##             y.max), xlim = c(0, xmax), ylab = "", xlab = "")
##         title(ylab = "Recruitment", xlab = "F bar", cex.lab = 0.75,
##             line = 2.5, cex.main = 0.75)
##         mtext(text = paste(maint, " a) Recruits"), cex = 0.75,
##             side = 3, line = 0.5)
##         lines(Fscan, recs[6, ], lty = 3)
##         lines(Fscan, recs[5, ], lty = 2)
##         lines(Fscan, recs[4, ], lty = 1)
##         lines(Fscan, recs[3, ], lty = 2)
##         lines(Fscan, recs[2, ], lty = 3)
##         lines(Fscan, recs[1, ], lty = 4)
##         points(FbarO, rec, pch = 21, cex = 0.75, bg = 1)
##         lines(c(flim, flim), c(0, y.max), col = 3)
##         y.max <- if (!is.na(ymax[2]))
##             ymax[2]
##         else max(ssbs[7, ])
##         plot(Fscan, ssbs[7, ], type = "l", lty = 4, ylim = c(0,
##             y.max), xlim = c(0, xmax), ylab = "", xlab = "")
##         title(ylab = "SSB", xlab = "F bar", cex.lab = 0.75, line = 2.5,
##             cex.main = 0.75)
##         mtext(text = "b) Spawning Stock Biomass", cex = 0.75,
##             side = 3, line = 0.5)
```

```
##         lines(Fscan, ssbs[6, ], lty = 3)
##         lines(Fscan, ssbs[5, ], lty = 2)
##         lines(Fscan, ssbs[4, ], lty = 1)
##         lines(Fscan, ssbs[3, ], lty = 2)
##         lines(Fscan, ssbs[2, ], lty = 3)
##         lines(Fscan, ssbs[1, ], lty = 4)
##         lines(c(0, xmax), c(Blim, Blim))
##         text(x = 0.1, y = Blim * 1.1, "Blim", cex = 0.7)
##         points(FbarO, ssb, pch = 21, cex = 0.75, bg = 1)
##         lines(c(flim, flim), c(0, y.max), col = 3)
##     }
##     FCrash5 <- Fscan[which.max(cats[2, ]):NF][which(cats[2, which.max(cats[2,
##         ]):NF] < 0.05 * max(cats[2, ]))[1]]
##     FCrash50 <- Fscan[which.max(cats[4, ]):NF][which(cats[4,
##         which.max(cats[4, ]):NF] < 0.05 * max(cats[4, ]))[1]]
##     if (plot) {
##         y.max <- if (!is.na(ymax[3]))
##             ymax[3]
##         else max(cats[7, ])
##         plot(Fscan, cats[7, ], type = "l", lty = 4, ylim = c(0,
##             y.max), xlim = c(0, max(Fscan)), ylab = "", xlab = "")
##         title(ylab = "Catch", xlab = "F bar", cex.lab = 0.75,
##             line = 2.5, cex.main = 0.75)
##         mtext(text = "c) Catch", cex = 0.75, side = 3, line = 0.5)
##         lines(Fscan, cats[7, ], lty = 4)
##         lines(Fscan, cats[6, ], lty = 3)
##         lines(Fscan, cats[5, ], lty = 2)
##         lines(Fscan, cats[4, ], lty = 1)
##         lines(Fscan, cats[3, ], lty = 2)
##         lines(Fscan, cats[2, ], lty = 3)
##         lines(Fscan, cats[1, ], lty = 4)
##         points(FbarO, Catchs, pch = 21, cex = 0.75, bg = 1)
##         lines(c(flim, flim), c(0, y.max), col = 3)
##         lines(c(FCrash5, FCrash5), c(0, y.max), col = 5)
##         lines(c(FCrash50, FCrash50), c(0, y.max), col = 5)
##         lines(Fscan, catm, lty = 1, col = 2)
##         lines(rep(Fscan[maxcatm], 2), c(0, y.max), lty = 1, col = 5)
##         plot(Fscan, 1 - pssb1, type = "l", lty = 2, ylim = c(0,
##             1), xlim = c(0, max(Fscan)), ylab = "", xlab = "")
##         title(ylab = "Prob MSY, SSB<Bpa or Blim", xlab = "F bar",
##             cex.lab = 0.75, line = 2.5, cex.main = 0.75)
##         mtext(text = "d) Prob MSY and Risk to SSB", cex = 0.75,
##             side = 3, line = 0.5)
##         lines(Fscan, 1 - pssb2, lty = 4)
##         text(x = max(Fscan[pssb2 > 0.5]) - 0.05, y = 0.5, "SSB<Bpa",
##             cex = 0.75)
##         text(x = max(Fscan[pssb1 > 0.7]) + 0.1, y = 0.3, "SSB<Blim",
##             cex = 0.75)
##         lines(c(flim, flim), c(0, 1), col = 3)
##         lines(c(0, flim), c(0.05, 0.05), lty = 2, col = 3)
##         text(x = 0.1, y = 0.075, "5%", cex = 0.75, col = 3)
##         lines(c(flim10, flim10), c(0, 1), col = "darkgreen")
##         lines(c(0, flim10), c(0.1, 0.1), lty = 2, col = "darkgreen")
##         text(x = 0.05, y = 0.125, "10%", cex = 0.75, col = "darkgreen")
##         lines(fmsy.dens$x, cumsum(fmsy.dens$y * diff(fmsy.dens$x)[1]),
##             col = 4)
```

```
##         text(x = 0.9, y = 0.8, "Prob of Fmsy", cex = 0.75, col = 4)
##         lines(rep(vcum, 2), c(0, 1), lty = 1, col = 4)
##         lines(c(Fscan[maxcatm], Fscan[maxcatm]), c(0, 1), col = 5)
##     }
##     out <- c(Blim, Bpa)
##     outF <- c(flim, flim10, flim50, vcum, Fscan[maxcatm], FCrash5,
##         FCrash50)
##     outB <- approx(Fscan, ssbs[4, ], xout = outF)$y
##     outC <- approx(Fscan, cats[4, ], xout = outF)$y
##     outTable <- rbind(outF, outB, outC)
##     rownames(outTable) <- c("F", "SSB", "Catch")
##     colnames(outTable) <- c("Flim", "Flim10", "Flim50", "MSY:median",
##         "Maxmeanland", "FCrash5", "FCrash50")
##     list(Blim = Blim, Bpa = Bpa, Refs = outTable)
## }
## <environment: namespace:msy>
```