FLash:::fwd for stock projection

Laurence Kell

August 13th, 2014

Introduction

The precautionary approach[^books_be] requires harvest control rules (HCRs) to trigger pre-agreed conservation and management action. This requires limit reference points to set boundaries that constrain harvesting within safe biological limits within which stocks can produce the maximum sustainable yield (MSY) and targets to ensure that management objectives are met.

The performance of HCRs, i.e. how well they meet management objectives should be evaluated, ideally using Management Strategy Evaluation (MSE) where the HCRs is tested as part of a Management Procedure (MP). Where an MP is the combination of the data collection regime stock assessment procedure and the setting of managment regulations. HCRs can be modelled using the fwd method of FLR; see the MSE document for examples of simulation testing.

Simulating the evolution of a stock or population (i.e. a projection) may be required after an assessment for a range of catches to allow managers to decide upon a TAC or within an MSE for a management measure set by an MP.

fwd takes objects descibing historical stock status and assumptions about future dynamics(e.g. growth, maturity, natural mortality and recruitment dynamics), then performs a projection for future options e.g. for catches, fishing mortality.

Introduction

```
library(FLCore)
```

```
## Loading required package: grid
## Loading required package: lattice
## Loading required package: MASS
## FLCore (Version 2.5.20140919, packaged: 2014-09-19 13:22:54 UTC)
##
## Attaching package: 'FLCore'
##
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(FLash)
library(FLBRP)
```

```
## Loading required package: ggplotFL
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
##
## The following object is masked from 'package:FLCore':
##
##     %+%
##
## Loading required package: gridExtra
## Loading required package: reshape2
## Loading required package: plyr
##
## Attaching package: 'plyr'
##
## The following object is masked from 'package:FLCore':
##
##     desc
```

```
library(ggplotFL)
```

fwd is used to make future projections, e.g. to evaluate different management options such as Total Allowable Catches (TACs) once a stock assessment has been conducted or for simulating a Harvest Control Rule (HCR) as part of a Management Strategy Evaluation (MSE).

In this examples we use the ple4 FLSock object

```
data(ple4)
```

The future

To perform a projection requires assumptions about future processes such as growth and recruitment and the effect of management on selectivity.

Recruit is based on a stock recruitment relationship, which can be fitted to the historic time series

```
#### SRR
sr = as.FLSR(ple4, model = "bevholt")
sr = fmle(sr, control = list(silent = TRUE))
```

```
## Warning: unknown names in control: silent
```

While future growth and selectivity is often assummed to be an average of recent values. In which case these can be estimated using FLBRP. An advantage of using FLBRP is then the projections and reference points will be consistent.

```
#### BRPs
eql = FLBRP(ple4, sr = sr)
computeRefpts(eql)
```

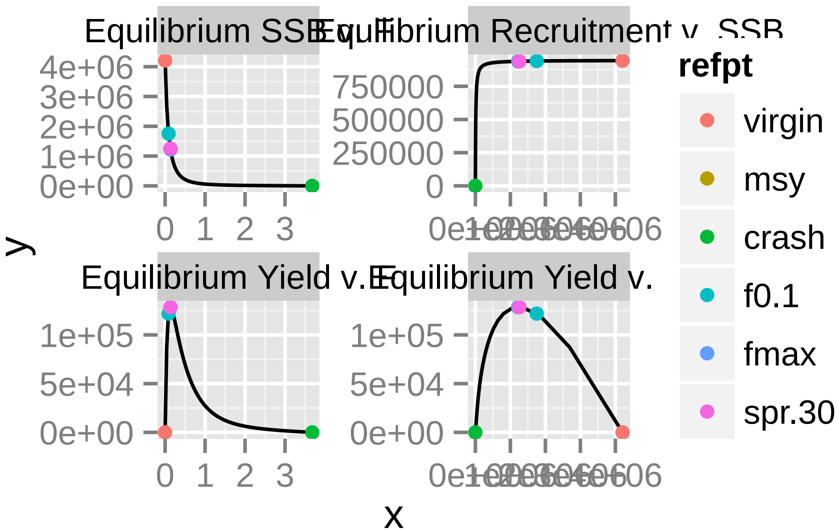
```
## An object of class "FLPar"
##         quantity
## refpt    harvest    yield      rec
##   virgin 0.0000e+00 0.0000e+00 9.4337e+05
##   msy    1.3378e-01 1.2850e+05 9.3821e+05
##   crash  3.6812e+00 1.7711e-06 3.7194e-04
##   f0.1   8.7602e-02 1.2185e+05 9.4036e+05
##   fmax   1.3538e-01 1.2849e+05 9.3813e+05
##   spr.30 1.3157e-01 1.2848e+05 9.3832e+05
##   mey            NA         NA         NA
##         quantity
## refpt    ssb        biomass    revenue
##   virgin 4.2043e+06 4.3812e+06         NA
##   msy    1.2351e+06 1.3923e+06         NA
##   crash  3.7903e-06 2.6298e-05         NA
##   f0.1   1.7536e+06 1.9172e+06         NA
##   fmax   1.2213e+06 1.3782e+06         NA
##   spr.30 1.2546e+06 1.4120e+06         NA
##   mey            NA         NA         NA
##         quantity
## refpt    cost       profit
##   virgin         NA         NA
##   msy            NA         NA
##   crash          NA         NA
##   f0.1           NA         NA
##   fmax           NA         NA
##   spr.30         NA         NA
##   mey            NA         NA
## units:  NA
```

```
eql = brp(eql)
```

```
plot(eql)
```

```
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
```

```
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
## Warning: using a local copy of '[[' which will be removed in later versions of FLCore
```

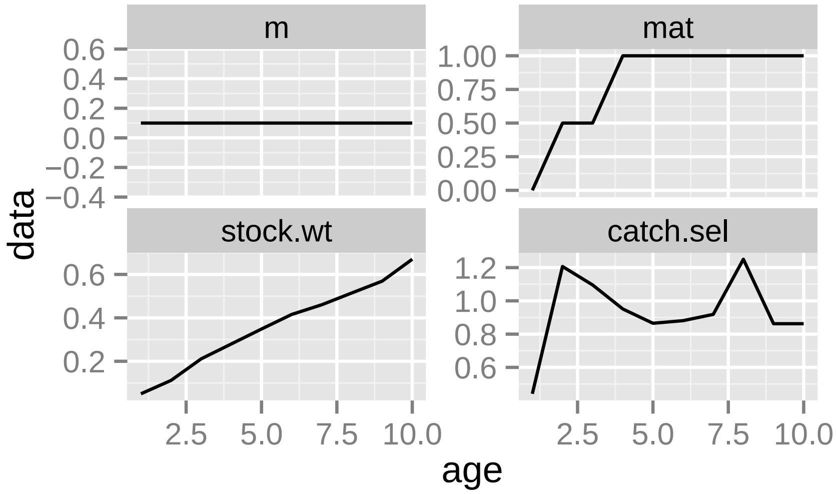


Expected parameters in the future

```
ggplot(FLQuants(eql, "m", "mat", "stock.wt", "catch.sel")) +
    geom_line(aes(age, data)) + facet_wrap(~qname,
    scale = "free_y")
```

Setting up the projection years can then be done by extending an FLStock uisng fwdWindow and passing the FLBRP object

```
stk = fwdWindow(ple4, end = 2020, eql)

## Warning: using a local copy of '[[<-' which
## will be removed in later versions of FLCore

unlist(dims(stk))

##     quant       age       min       max
##     "age"      "10"       "1"      "10"
##      year   minyear   maxyear plusgroup
##      "64"    "1957"    "2020"      "10"
##      unit    season      area      iter
##       "1"       "1"       "1"       "1"
```



Projections

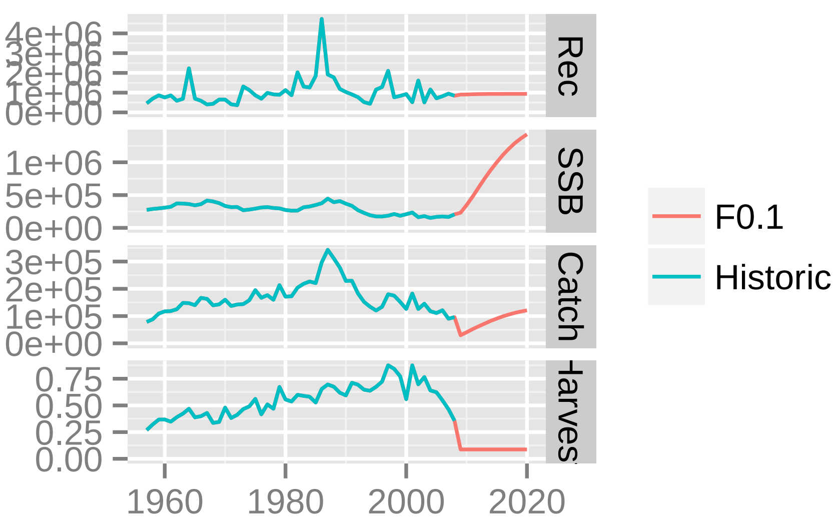
We first show how simple projections (e.g. for F and catch) can be performed. Later we show how a variety of HCRs can be simulated.

Simulate fishing at $F_{0.1}$, first create an FLQuant with the target Fs

```
F0.1 = FLQuant(refpts(eql)["f0.1", "harvest",
    drop = T], dimnames = list(year = 2009:2020))
```

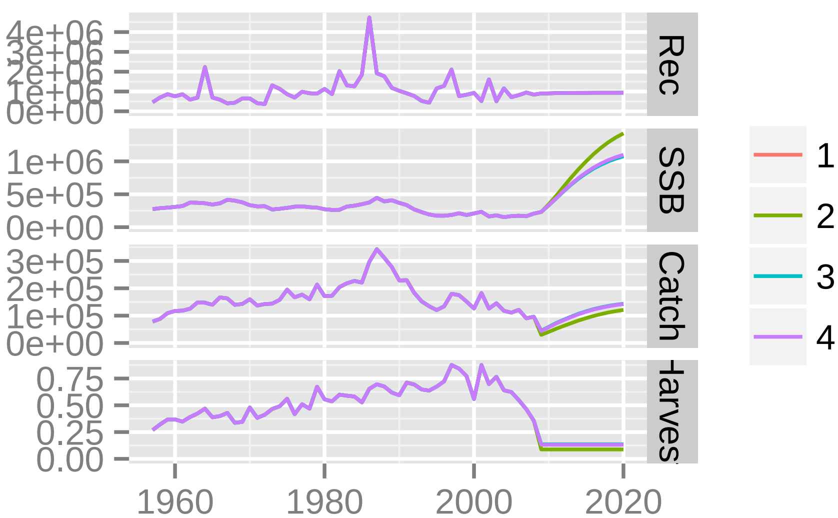
Then project forward, note that sr is also required

```
stk = fwd(stk, f = F0.1, sr = sr)

plot(FLStocks(Historic = ple4, F0.1 = stk))
```



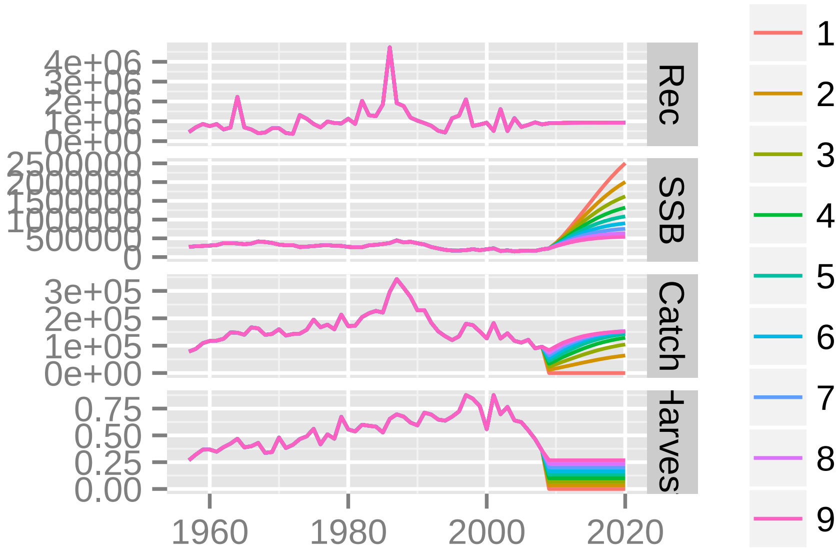
It is possible to project for different Fs i.e. alternative reference points

```
library(plyr)

dimnames(refpts(eql))$refpt

## [1] "virgin" "msy"    "crash"  "f0.1"
## [5] "fmax"   "spr.30" "mey"

refs = refpts(eql)[c("msy", "f0.1", "fmax", "spr.30"),
    "harvest", drop = T]

targetF = FLQuants(mlply(data.frame(refs), function(refs) FLQuant(refs,
    dimnames = list(year = 2009:2020))))

names(targetF) = names(refs)

names(targetF)[] = "f"
stks = fwd(stk, targetF, sr = sr)

plot(stks)
```



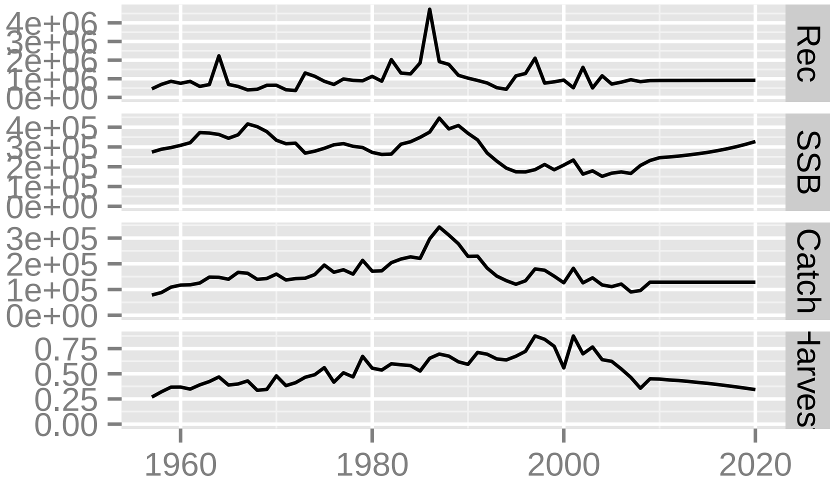
or different multipliers of F_{MSY}

```
msyTargets = FLQuants(mlply(seq(0, 2, 0.25), function(x) FLQuant(x *
    refs["msy"], dimnames = list(year = 2009:2020))))

names(msyTargets)[] = "f"

stks = fwd(stk, msyTargets, sr = sr)

plot(stks)
```



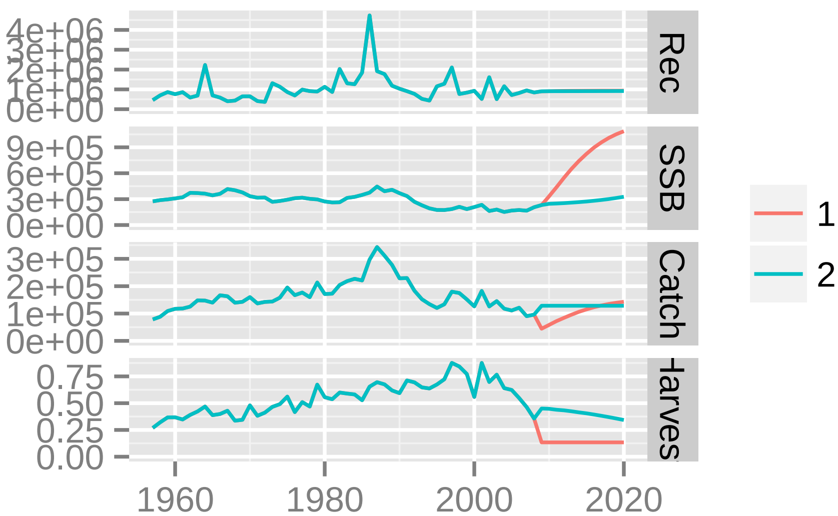
Catch projections are done in a similar way e.g. for *MSY*

```
refpts(eql)["msy"]

## An object of class "FLPar"
##      quantity
## refpt harvest    yield      rec
##   msy 1.3378e-01 1.2850e+05 9.3821e+05
##      quantity
## refpt ssb        biomass    revenue
##   msy 1.2351e+06 1.3923e+06         NA
##      quantity
## refpt cost       profit
##   msy         NA         NA
## units:  NA

refpts(eql)["msy", c("harvest", "yield")]

## An object of class "FLPar"
##      quantity
## refpt harvest    yield
##   msy 1.3378e-01 1.2850e+05
## units:  NA

msy = FLQuant(c(refpts(eql)["msy", "yield"]),
    dimnames = list(year = 2009:2020))

stks = fwd(stk, catch = msy, sr = sr)

plot(stks)
```



Compare F and Catch projections, e.g. for MSY and F_{MSY}

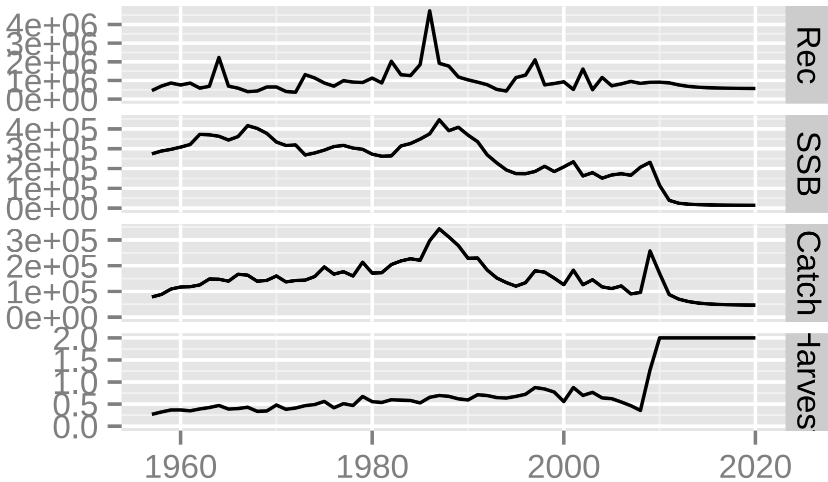
```
msys = FLQuants(f = targetF[[1]], catch = msy)
stks = fwd(stk, msys, sr = sr)

plot(stks)
```



If the projected catch is high you could simulate high Fs, however, there will be a cap of effort and capacity so in practice such high Fs may not be realised. Therefore there is a constraint on F.

```
catch = FLQuant(c(refpts(eql)["msy", "yield"]) *
    2, dimnames = list(year = 2009:2020))

stk = fwd(stk, catch = catch, sr = sr)
```

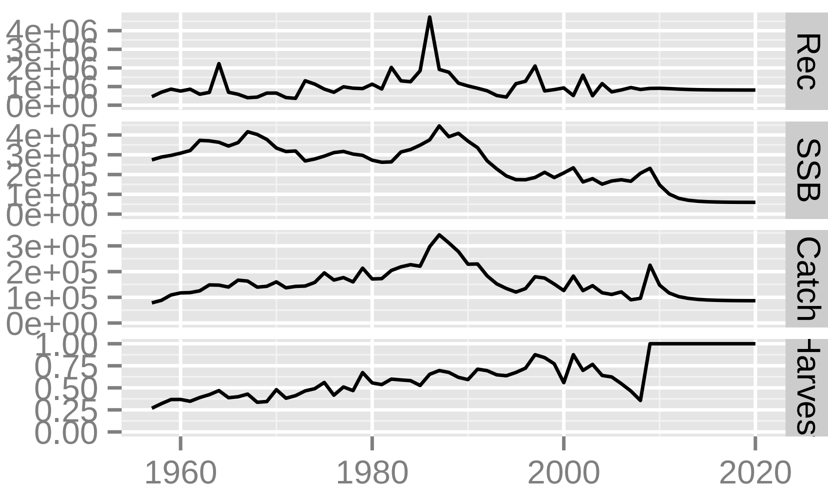
```
plot(stk)
```



i.e. maxF, this allows an upper limit to be set on F

```
stk = fwd(stk, catch = catch, sr = sr, maxF = 1)
```

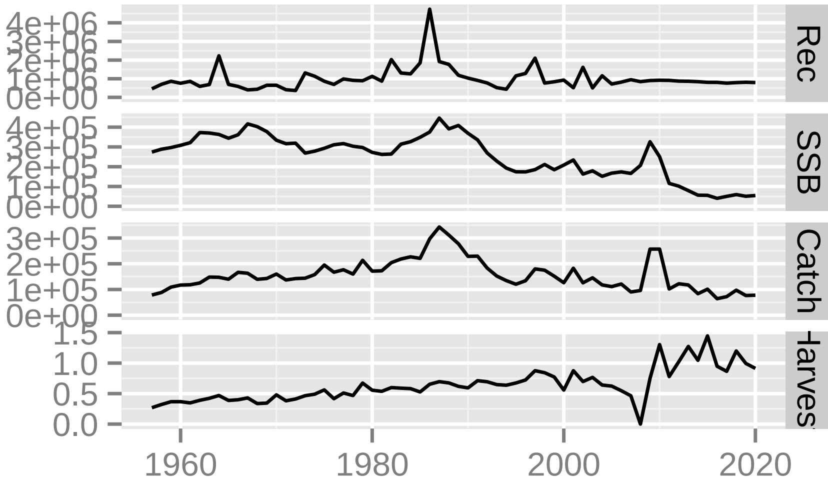
```
plot(stk)
```



This can also be used to model capacity

```
capacity = FLQuant(1, dimnames = list(year = 2009:2020))
q = rlnorm(1, FLQuant(0, dimnames = list(year = 2009:2020)),
    0.2)
maxF = q * capacity
```

```
stk = fwd(stk, catch = catch, sr = sr, maxF = maxF)

plot(stk)
```

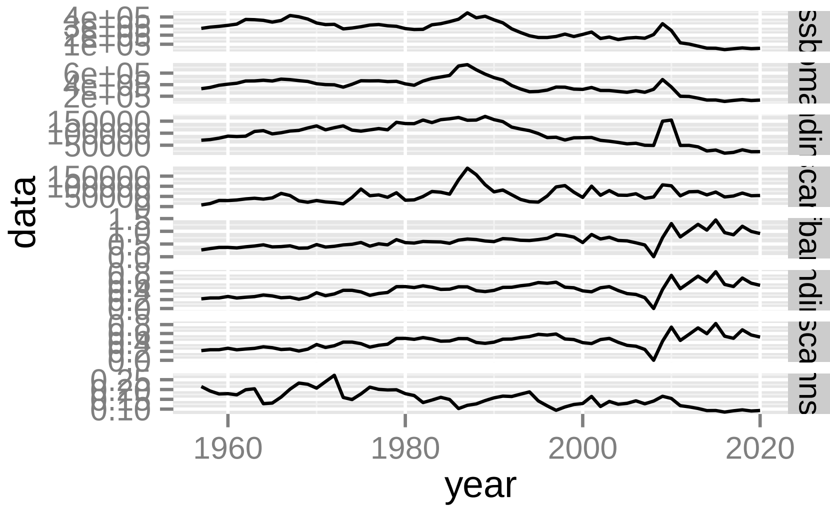


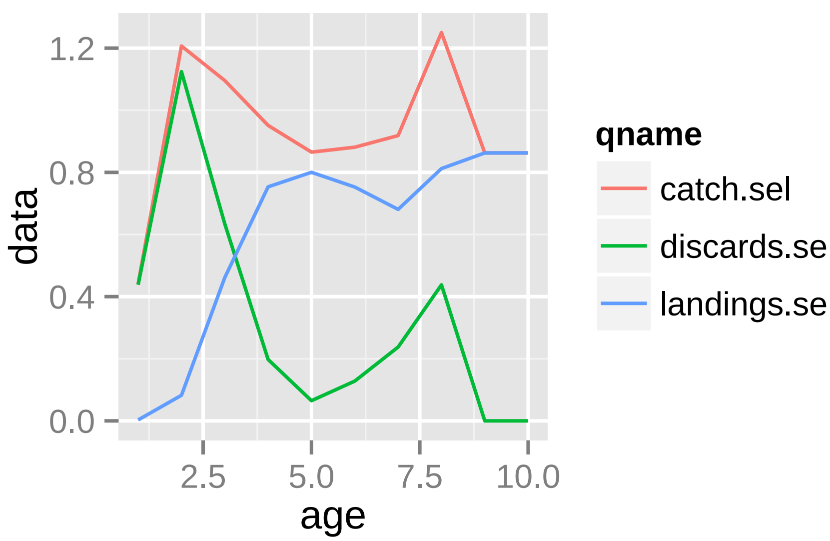
Other quantities can be considered in projections as well as catch and F, i.e. ssb, biomass, landings, discards, f, f.catch, f.landings, f.discards, effort, costs, revenue, profit, mnsz.

```
f.landings = function(x) apply((harvest(x) * landings.n(x)/catch.n(x))[ac(range(stk)["minfbar"]:range(stk)
    2, mean)
f.discards = function(x) apply((harvest(x) * landings.n(x)/catch.n(x))[ac(range(stk)["minfbar"]:range(stk)
    2, mean)
mnsz = function(x) apply(stock.n(x) * stock.wt(x),
    2, sum)/apply(stock.n(x), 2, sum)

flqs = FLQuants(stk, "ssb", biomass = stock, "landings",
    "discards", "fbar", "f.landings", "f.discards",
    "mnsz")

# effort, costs, revenue, profit, .

ggplot(flqs) + geom_line(aes(year, data)) + facet_grid(qname ~
    ., scale = "free_y")
```

Management has two main options, i.e. setting effort (as in the examples above) or relative F-at-age by changing the selection pattern. The selection pattern-at-age of landings is that of the catch less discards e.g.

```
ggplot(FLQuants(eql, "catch.sel", "discards.sel",
    "landings.sel")) + geom_line(aes(age, data,
    col = qname))
```





In the FLStock object there are therefore 3 selection pattern components, and unfortunate three ways of calculating each. fwd uses computeCatch to re-estimate the catch.n, landings.n and discards.n before calculating future selection patterns.

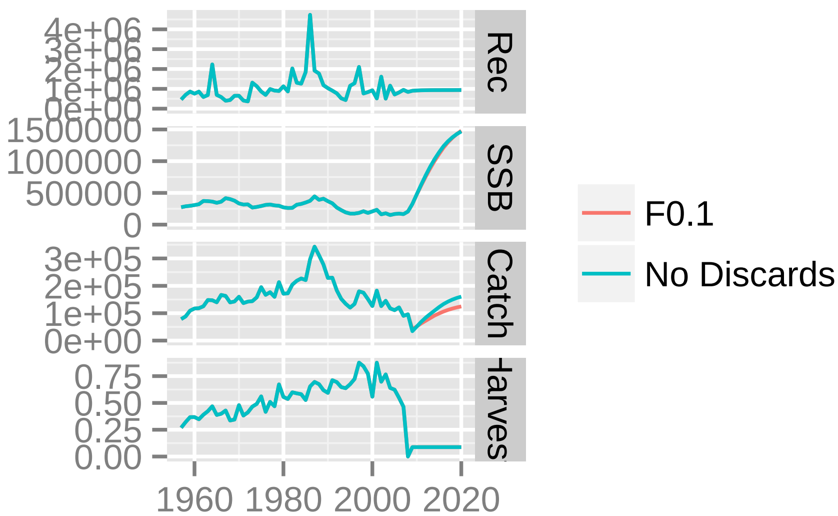
```
catch(stk) <- computeCatch(stk)
```

In fwd the selection patterns are then calculated as harvest*discards.n/catch.n, harvest*landings.n/catch.n and discards.sel+landings.sel

Simulation of gears that get rid of discarding can be done by

```
noDiscards = stk
discards.n(noDiscards)[, ac(2009:2020)] = 0
catch.n(noDiscards)[, ac(2009:2020)] <- landings.n(noDiscards)[,
```

```
    ac(2009:2020)]
catch(noDiscards) <- computeCatch(noDiscards)

## Note adjustment of harvest
harvest(noDiscards)[, ac(2009:2020)] = harvest(stk)[,
    ac(2009:2020)] * landings.n(stk)[, ac(2009:2020)]/catch.n(stk)[,
    ac(2009:2020)]

noDiscards = fwd(noDiscards, f = F0.1, sr = sr)
stk = fwd(stk, f = F0.1, sr = sr)

plot(FLStocks(‘No Discards‘ = noDiscards, F0.1 = stk))
```

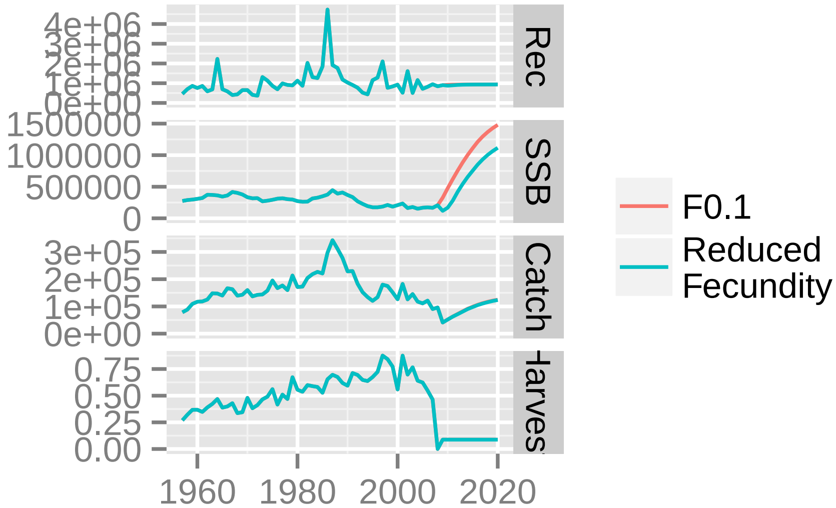


Non stationarity is seen in many biological processes, what happens if future fecundity decreases?

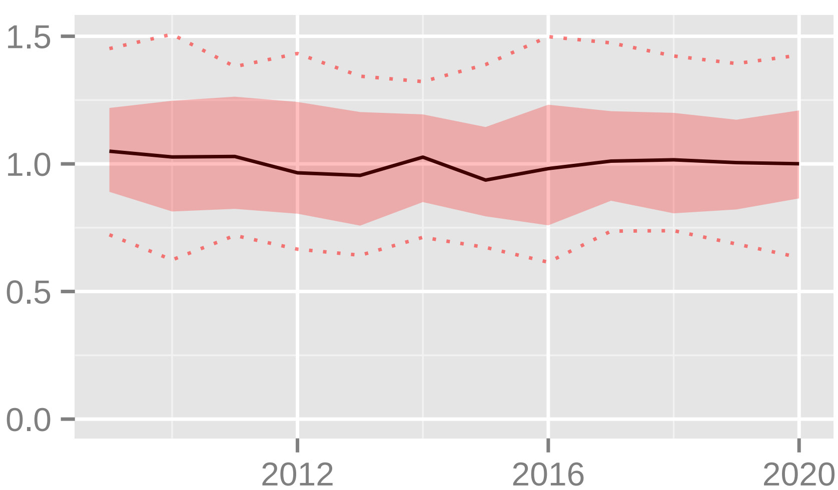
```
poorFec = stk
mat(poorFec)[1:5, ac(2009:2020)] = c(0, 0, 0,
    0, 0.5)

poorFec = fwd(poorFec, f = F0.1, sr = sr)

plot(FLStocks(‘Reduced \nFecundity‘ = poorFec,
    F0.1 = stk))
```

Stochasticity

Monte Carlo simulations based on future recruitment




```
srDev = rlnorm(100, FLQuant(0, dimnames = list(year = 2009:2020)),
    0.3)
plot(srDev)
```




```
stk = fwd(stk, f = F0.1, sr = sr, sr.residuals = srDev)
```

fwdControl

fwdControl is a more flexible but fiddly way of setting up projections. For example to replicate the $F_{0.1}$ projection above requires setting up a fwdControl object.

This can be done using a constructor and a data.frame

```
ctrl = fwdControl(data.frame(year = 2009:2018,
    val = c(refpts(eql)["f0.1", "harvest"]), quantity = "f"))
```

fwdControl is a class with 5 slots

```
slotNames(ctrl)
```

```
## [1] "target"    "effort"    "trgtArray"
## [4] "effArray"  "block"
```

For now we will concerntrate on just the target and trgtArray slots.

```
slotNames(ctrl)
```

```
## [1] "target"    "effort"    "trgtArray"
## [4] "effArray"  "block"
```

```
ctrl
```

```
##
## Target
##    year quantity min    val max
## 1  2009        f  NA 0.0876  NA
## 2  2010        f  NA 0.0876  NA
## 3  2011        f  NA 0.0876  NA
## 4  2012        f  NA 0.0876  NA
## 5  2013        f  NA 0.0876  NA
## 6  2014        f  NA 0.0876  NA
## 7  2015        f  NA 0.0876  NA
## 8  2016        f  NA 0.0876  NA
## 9  2017        f  NA 0.0876  NA
## 10 2018        f  NA 0.0876  NA
##
##
##      min      val      max
##   1        NA 0.087602       NA
##   2        NA 0.087602       NA
##   3        NA 0.087602       NA
##   4        NA 0.087602       NA
##   5        NA 0.087602       NA
##   6        NA 0.087602       NA
##   7        NA 0.087602       NA
##   8        NA 0.087602       NA
##   9        NA 0.087602       NA
##   10       NA 0.087602       NA
```

target specifies the quantity for the projection (e.g. “f”, “catch”, “ssb”, ...) and the projection year. The projection can be a target by specifying it in val. While min and max specify bounds. For example if you want to project for a target F but also to check that SSB does not fall below an SSB limit.

An example with high F that decreases SSB a lot

```
target = fwdControl(data.frame(year = 2009, val = 0.8,
    quantity = "f"))
stk = fwdWindow(ple4, end = 2010, eql)
```

```
## Warning: using a local copy of '[[<-' which
## will be removed in later versions of FLCore
```

```
stk = fwd(stk, ctrl = target, sr = eql)
```

```
fbar(stk)[, "2009"]
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   2009
##   all 0.8
##
## units:  f
```

```
ssb(stk)[, "2010"]
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   2010
##   all 177289
##
## units:  NA
```

Note that it is the end of year biomass that is constrained as in this case spawning is at Jan 1st and so fishing only has an effect of SSB next year

Constrain SSB so that it doesnt fall below 250000

```
target <- fwdControl(data.frame(year = c(2009,
    2009), val = c(0.8, NA), min = c(NA, 230000),
    quantity = c("f", "ssb")))
```

```
stk = fwd(stk, ctrl = target, sr = sr)
```

```
fbar(stk)[, "2009"]
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   2009
##   all 0.52058
##
## units:  f
```

```
ssb(stk)[, "2010"]
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   2010
##   all 230000
##
## units:  NA
```

If a stock spawns mid year so the adult population is affected by fishing then the SSB constraint is within year, e.g.

```
harvest.spwn(stk)[] = 0.5

stk = fwd(stk, ctrl = target, sr = sr)

fbar(stk)[, "2009"]
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   2009
##   all 0.01302
##
## units:  f
```

```
ssb(stk)[, c("2009", "2010")]
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   2009   2010
##   all 230000 228391
##
## units:  NA
```

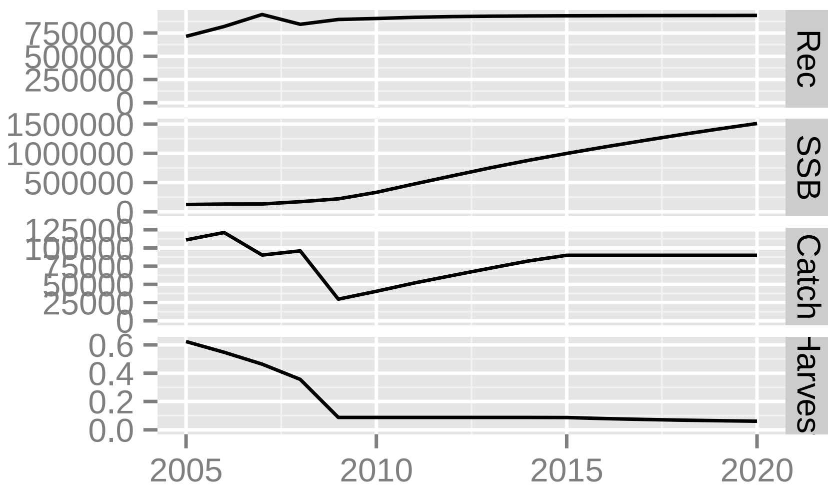
Harvest Control Rules

```
msy = refpts(eql)["msy", "yield"]
bmsy = refpts(eql)["msy", "ssb"]
f0.1 = refpts(eql)["f0.1", "harvest"]
stk = fwdWindow(stk, end = 2020, eql)
```

```
## Warning: using a local copy of '[[<-' which
## will be removed in later versions of FLCore
```

```
#### constant catch with an upper F bound
ctrl = fwdControl(data.frame(year = rep(2009:2020,
    each = 2), val = rep(c(msy * 0.7, NA), 12),
    max = rep(c(NA, f0.1), 12), quantity = rep(c("catch",
        "f"), 12)))
stk = fwd(stk, ctrl = ctrl, sr = sr)
```

```
plot(stk[, ac(2005:2020)])
```

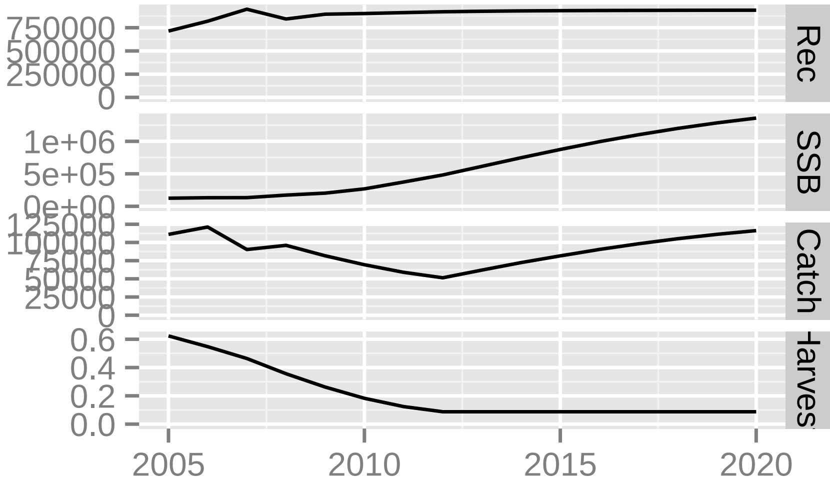


Reduce F to F0.1 but only let catch change by 15% a year

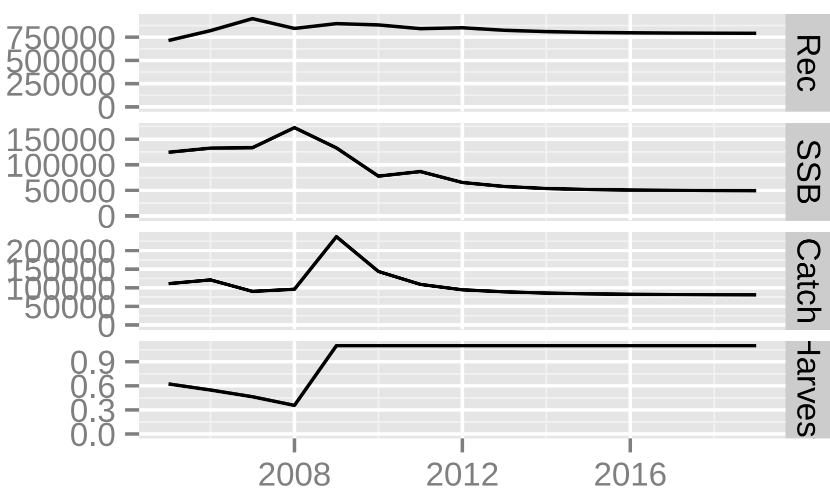
```
ctrl = fwdControl(data.frame(year = rep(2009:2020,
    each = 2), rel.year = c(t(array(c(rep(NA,
    12), 2008:2019), c(12, 2)))), val = rep(c(f0.1,
    NA), 12), min = rep(c(NA, 0.85), 12), quantity = rep(c("f",
    "catch"), 12)))
stk = fwd(stk, ctrl = ctrl, sr = sr)
```

```
plot(stk[, ac(2005:2020)])
```

10% SSB increase




```
ctrl = fwdControl(data.frame(year = rep(2009:2020,
    each = 2), rel.year = c(t(array(c(2008:2019,
    rep(NA, 12)), c(12, 2)))), max = rep(c(f0.1,
    NA), 12), val = rep(c(NA, 1.1), 12), quantity = rep(c("ssb",
    "f"), 12)))
stk = fwd(stk, ctrl = ctrl, sr = sr)

plot(stk[, ac(2005:2019)])
```




```
hcrF = function(iYr, SSB, Bpa, Blim, Fmin, Fmax) {
    val = pmin(Fmax, Fmax - (Fmax - Fmin) * (Bpa -
        SSB)/(Bpa - Blim))
    trgt = fwdTarget(year = iYr + 1, quantity = "f",
        valueval)
```

```
    return(trgt)
}
```

Recover stock to target SSB level corresponding to the 1980s in 2020 with a constant F strategy

```
load("/tmp/flash.RData")
stk = fwdWindow(stk, end = 2020, eql)

## Warning: using a local copy of '[[<-' which
## will be removed in later versions of FLCore

ssbTarget = mean(ssb(stk)[, ac(1970:1989)])

## function to minimise
f <- function(x, stk, ssbTarget, ctrl, sr) {

    # set target F for all years
    ctrl@target[, "val"] = x
    ctrl@trgtArray[, "val", ] = x

    # project
    stk = fwd(stk, ctrl = ctrl, sr = sr)

    # Squared Difference
    return((ssb(stk)[, ac(range(stk)["maxyear"])] -
        ssbTarget)^2)
}

## control object
ctrl = fwdControl(data.frame(year = 2009:2020,
    val = 0.5, rel = 2008, quantity = "f"))

xmin = optimize(f, c(0.1, 1), tol = 1e-07, stk = stk,
    ssbTarget = ssbTarget, ctrl = ctrl, sr = eql)
ctrl = fwdControl(data.frame(year = 2009:2020,
    val = xmin$minimum, rel = 2008, quantity = "f"))

stk = fwd(stk, ctrl = ctrl, sr = eql)

# update catch slot
catch(stk) = computeCatch(stk)

# Have we reached the target?
ssbTarget
```

```
## [1] 322339
```

```
ssb(stk)
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   1957   1958   1959   1960   1961
##   all 274205 288540 296825 308164 321354
##      year
## age   1962   1963   1964   1965   1966
##   all 372863 370373 363077 344013 361549
##      year
## age   1967   1968   1969   1970   1971
##   all 416563 402521 377432 333933 316343
##      year
## age   1972   1973   1974   1975   1976
##   all 319062 268714 278648 293136 310954
##      year
## age   1977   1978   1979   1980   1981
##   all 316929 303433 297122 272416 262061
##      year
## age   1982   1983   1984   1985   1986
##   all 263998 314021 326341 348675 375392
##      year
## age   1987   1988   1989   1990   1991
##   all 445855 391254 408489 368969 335747
##      year
## age   1992   1993   1994   1995   1996
##   all 269528 228668 193093 174408 173903
##      year
## age   1997   1998   1999   2000   2001
##   all 185308 211327 184733 208393 234078
##      year
## age   2002   2003   2004   2005   2006
##   all 162725 179158 151508 167531 173783
##      year
## age   2007   2008   2009   2010   2011
##   all 166061 206480 231522 260641 277268
##      year
## age   2012   2013   2014   2015   2016
##   all 290401 300924 308029 312489 316464
##      year
## age   2017   2018   2019   2020
```

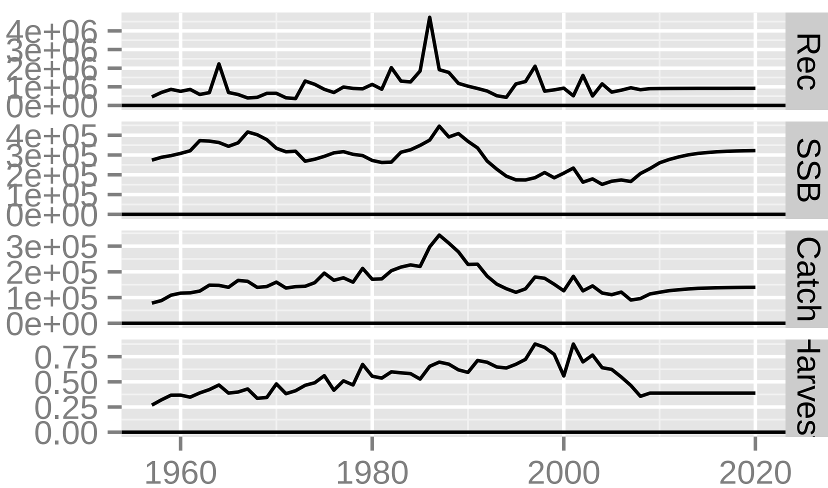
```
##   all 318754 320424 321560 322339
##
## units:  NA
```

```
# At what level of constant F
fbar(stk)
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   1957    1958    1959    1960
##   all 0.26857 0.32106 0.36734 0.36796
##      year
## age   1961    1962    1963    1964
##   all 0.34756 0.39012 0.42276 0.46878
##      year
## age   1965    1966    1967    1968
##   all 0.38796 0.39896 0.42923 0.33621
##      year
## age   1969    1970    1971    1972
##   all 0.34457 0.47965 0.38206 0.41158
##      year
## age   1973    1974    1975    1976
##   all 0.46551 0.49072 0.56113 0.41641
##      year
## age   1977    1978    1979    1980
##   all 0.51007 0.46862 0.67312 0.55555
##      year
## age   1981    1982    1983    1984
##   all 0.53705 0.59912 0.58934 0.58159
##      year
## age   1985    1986    1987    1988
##   all 0.52695 0.65386 0.69596 0.67530
##      year
## age   1989    1990    1991    1992
##   all 0.61895 0.59361 0.71195 0.69443
##      year
## age   1993    1994    1995    1996
##   all 0.64752 0.63741 0.67444 0.72301
##      year
## age   1997    1998    1999    2000
##   all 0.87588 0.84233 0.77264 0.55795
##      year
## age   2001    2002    2003    2004
```

```
##   all 0.87567 0.69763 0.76597 0.64015
##      year
## age   2005    2006    2007    2008
##   all 0.62343 0.54764 0.46392 0.35631
##      year
## age   2009    2010    2011    2012
##   all 0.38813 0.38813 0.38813 0.38813
##      year
## age   2013    2014    2015    2016
##   all 0.38813 0.38813 0.38813 0.38813
##      year
## age   2017    2018    2019    2020
##   all 0.38813 0.38813 0.38813 0.38813
##
## units:  f
```

```
plot(stk) + geom_hline(aes())
```



Recover stock to the desired SSB in 2006 with a constant Catch strategy Here val can be anything in the ctrl because it is overwritten in the optimisation loop

```
ctrl = fwdControl(data.frame(year = 2009:2020,
    val = c(catch(stk)[, "2001"]), quantity = "catch"))

xmin = optimize(f, c(100, 1e+05), tol = 1e-07,
    stk = stk, ssbTarget = ssbTarget, ctrl = ctrl,
    sr = sr)
ctrl = fwdControl(data.frame(year = 2009:2020,
    val = xmin$minimum, quantity = "catch"))
```

```
stkC = fwd(stk, ctrl = ctrl, sr = sr)

# Have we reached the target?
ssbTarget
ssb(stkC)[, ac(2002:2020)]
# At what level of constant catch
computeCatch(stkC)[, ac(2002:2020)]
# And at what level of F
fbar(stkC)[, ac(2002:2006)]
# Update the catch slot
catch(stkC) = computeCatch(stkC)
# 'ave a butchers
plot(stkC[, ac(1957:2006)])

# Assessment up to and including 2001
data(ple4)
black.bird = stf(stk, nyrs = 2)

# set courtship and egg laying in Autumn
black.bird@m.spwn[] = 0.66
black.bird@harvest.spwn[] = 0.66

# assessment is in year 2002, set catch
# constraint in 2002 and a first guess for F
# in 2003
ctrl = fwdControl(data.frame(year = 2002:2003,
    val = c(85000, 0.5), quantity = c("catch",
        "f")))
black.bird = fwd(black.bird, ctrl = ctrl, sr = list(model = "mean",
    params = FLPar(25000)))

# HCR specifies F=0.1 if ssb<100000, F=0.5 if
# ssb>300000 otherwise linear increase as SSB
# increases
min.ssb = 1e+05
max.ssb = 3e+05
min.f = 0.1
max.f = 0.5

# slope of HCR
a. = (max.f - min.f)/(max.ssb - min.ssb)
b. = min.f - a. * min.ssb

# plot of HCR
```

```
plot(c(0, min.ssb, max.ssb, max.ssb * 2), c(min.f,
    min.f, max.f, max.f), type = "l", ylim = c(0,
    max.f * 1.25), xlim = c(0, max.ssb * 2))

## find F through iteration
t. = 999
i = 0
while (abs(ctrl@target[2, "val"] - t.) > 1e-05 &
    i < 50) {
    t. = ctrl@target[2, "val"]  ## save last val of F

    # calculate new F based on SSB last iter
    ctrl@target[2, "val"] = a. * c(ssb(black.bird)[,
        "2003"]) + b.
    ctrl@trgtArray[2, "val", ] = a. * c(ssb(black.bird)[,
        "2003"]) + b.
    black.bird = fwd(black.bird, ctrl = ctrl,
        sr = list(model = "mean", params = FLPar(25000)))

    # 'av a gander
    points(c(ssb(black.bird)[, "2003"]), c(ctrl@target[2,
        "val"]), cex = 1.25, pch = 19, col = i)
    print(c(ssb(black.bird)[, "2003"]))
    print(c(ctrl@target[2, "val"]))
    i = i + 1
}

# F bounds
black.bird = fwd(black.bird, ctrl = ctrl, sr = list(model = "mean",
    params = FLPar(25000)))
plot(FLStocks(black.bird))
```

```
#### Create a random variable for M
albM = stk
m(albM) = propagate(m(albM), 100)

mDev = rlnorm(prod(dim(m(albM))), 0, 0.3)
mean(mDev)
var(mDev)^0.5

m(albM) = m(albM) * FLQuant(mDev, dimnames = dimnames(m(albM)))
plot(m(albM))

harvest(albM) = computeHarvest(albM)
```

```
catch(albM) = computeCatch(albM, "all")

plot(FLStocks(albM, stk28))

ctrl = fwdControl(data.frame(year = 2009:2020,
    val = ctch, quantity = "catch"))
albM = fwd(albM, ctrl = ctrl, sr = sr)

plot(albM)

#### Create a random variable for M
albM1 = albM
m(albM1)[1:3, ] = m(albM)[1:3, ] * 2

harvest(albM1) = computeHarvest(albM1)
catch(albM1) = computeCatch(albM1, "all")
albM1 = fwd(albM1, ctrl = ctrl, sr = sr)

albM2 = albM
m(albM2)[, ac(2000:2020)] = m(albM)[, ac(2000:2020)] *
    2

harvest(albM2) = computeHarvest(albM2)
catch(albM2) = computeCatch(albM2, "all")
albM2 = fwd(albM2, ctrl = ctrl, sr = sr)

plot(FLStocks(albM, albM1, albM2))

#### process error in recruitment
srDev = FLQuant(rlnorm(20 * 100, 0, 0.3), dimnames = list(year = 2008:2020,
    iter = 1:100))
sr = fwd(albM, ctrl = ctrl, sr = sr, sr.residuals = srDev)
plot(sr)

#### SRR regime shifts
albSV = as.FLSR(albNEA)
model(sr) = bevholtSV()
albSV = fmle(albSV, fixed = list(spr0(albNEA)))
albSV1 = fmle(albSV, fixed = list(spr0 = spr0(albNEA),
    s = 0.75))
albSV2 = fmle(albSV, fixed = list(spr0 = spr0(albNEA),
    v = 0.75 * params(albSV)["v"]))

#### Prior for steepness
albSV3 = fmle(albSV, fixed = list(s = qnorm(seq(0.01,
```

```
    0.99, length.out = 101), 0.75, 0.1)))
albSV3 = fwd(albSV3, ctrl = ctrl, sr = sr, sr.residuals = srDev)

plot(albSV1, albSV2, albSV3)

#### SRR regime shifts
albBRP = brp(FLBRP(albM))
refpts(albBRP)

albSV3 = fmle(albSV, fixed = list(s = qnorm(seq(0.01,
    0.99, length.out = 101), 0.75, 0.1)))
albSV3 = fwd(albSV3, ctrl = ctrl, sr = sr, sr.residuals = srDev)

plot(albSV1, albSV2, albSV3)

# F bounds
black.bird = fwd(black.bird, ctrl = ctrl, sr = list(model = "mean",
    params = FLPar(25000)))
plot(FLStocks(black.bird))

library(FLash)
library(FLAssess)

#### Set up a short term forecast for an FLStock
#### object by adding extra years The default
#### forecast is 3 years,
alb3 = stf(alb)

## Check what?s happened
summary(alb)
summary(alb3)

## by default future F is the mean of last 3
## years
mean(fbar(alb)[, ac(2007 - (0:2))])
fbar(alb3)[, ac(2007 + (1:3))]

## by default future F is the mean of last 3
## years
mean(fbar(alb)[, ac(2007 - (0:2))])
fbar(alb3)[, ac(2007 + (1:3))]

## Constant F Projection for a 20 year
## projection
stk = stf(alb, nyear = 20)
```

```
#### SRR
sr = as.FLSR(alb)
model(sr) = bevholt()
sr = fmle(sr)

#### BRPs
albBRP = FLBRP(alb, sr = sr)
computeRefpts(albBRP)


albBRP = brp(albBRP)


# Use F0.1 as fishing mortality target
F0.1 = refpts(albBRP)["f0.1", "harvest", drop = T]
#### bug
ctrl = fwdControl(data.frame(year = 2008:2027,
    val = F0.1, quantity = "f"))

albF1 = fwd(stk, ctrl = ctrl, sr = sr)

plot(albF1)
ctrl = fwdControl(data.frame(year = 2008:2027,
    val = F0.1 * 0.5, quantity = "f"))
albF2 = fwd(stk, ctrl = ctrl, sr = sr)

ctrl = fwdControl(data.frame(year = 2008:2027,
    val = F0.1 * 2, quantity = "f"))
albF3 = fwd(stk, ctrl = ctrl, sr = sr)


## Create an FlStock object
albF0.1 = FLStocks(F0.1 = albF1, half = albF2,
    double = albF3)
plot(albF0.1)

## Cut the plots
plot(lapply(albF0.1, window, start = 1990))

## Compare alternatives
lapply(lapply(albF0.1, window, start = 2008),
    computeCatch)

#### Total catch
```

```
lapply(lapply(lapply(albF0.1, window, start = 2008),
    computeCatch), sum)

#### Short-term
unlist(lapply(lapply(lapply(albF0.1, window, start = 2008,
    end = 2013), computeCatch), sum))
#### Medium-term
unlist(lapply(lapply(lapply(albF0.1, window, start = 2016,
    end = 2020), computeCatch), sum))
#### Long-term
unlist(lapply(lapply(lapply(albF0.1, window, start = 2023,
    end = 2027), computeCatch), sum))

#### constant catch startegies
ctch = mean(computeCatch(alb)[, ac(2003:2007)])

albC = FLStocks()
ctrl = fwdControl(data.frame(year = 2008:2027,
    val = ctch, quantity = "catch"))
albC[["1.0"]] = fwd(stk, ctrl = ctrl, sr = sr)

ctrl = fwdControl(data.frame(year = 2008:2027,
    val = 0.5 * ctch, quantity = "catch"))
albC[["0.5"]] = fwd(stk, ctrl = ctrl, sr = sr)

ctrl = fwdControl(data.frame(year = 2008:2027,
    val = 1.5 * ctch, quantity = "catch"))
albC[["1.5"]] = fwd(stk, ctrl = ctrl, sr = sr)
plot(albC)

#### compare startegies
plot(FLStocks(albC[[1]], albF0.1[[1]]))

#### constant catch with an upper F bound
ctrl = fwdControl(data.frame(year = rep(2008:2027,
    each = 20), val = rep(c(ctch * 1.5, NA), 20),
    max = rep(c(NA, F0.1), 20), quantity = rep(c("catch",
        "f"), 20)))
albFC = fwd(stk, ctrl = ctrl, sr = sr)
plot(albFC)

#### 5% F reduction
ctrl = fwdControl(data.frame(year = rep(2008:2027,
    each = 2), rel.year = c(t(array(c(2007:2026,
    rep(NA, 20)), c(20, 2)))), val = rep(c(0.95,
```

```
    NA), 20), min = rep(c(NA, F0.1 * 0.5), 20),
    quantity = rep(c("catch", "f"), 20)))
albFC = fwd(stk, ctrl = ctrl, sr = sr)
plot(albFC)

#### 10% SSB increase
ctrl = fwdControl(data.frame(year = 2008:2027,
    rel.year = 2007:2026, min = 1.1, quantity = "ssb"))
albSSB = fwd(stk, ctrl = ctrl, sr = sr)
plot(albSSB)

hcrF = function(iYr, SSB, Bpa, Blim, Fmin, Fmax) {
    val = pmin(Fmax, Fmax - (Fmax - Fmin) * (Bpa -
        SSB)/(Bpa - Blim))
    trgt = fwdTarget(year = iYr + 1, quantity = "f",
        valueval)

    return(trgt)
}

## Ogives
dnormal = function(x, a, sL, sR) {
    pow = function(a, b) a^b

    func = function(x, a, sL, sR) {
        if (x < a)
            return(pow(2, -((x - a)/sL * (x -
                a)/sL))) else return(pow(2, -((x - a)/sR * (x -
            a)/sR)))
    }

    sapply(x, func, a, sL, sR)
}

logistic = function(x, a50, ato95) {
    pow = function(a, b) a^b

    func = function(x, a50, ato95) {
        if ((a50 - x)/ato95 > 5)
            return(0)
        if ((a50 - x)/ato95 < -5)
            return(1)

        return(1/(1 + pow(19, (a50 - x)/ato95)))
    }
```

```
    sapply(x, func, a50, ato95)
}

prices = data.frame(rbind(cbind(Age = 1:10, Price = dnormal(1:10,
    3, 10, 20), Type = "Peaking"), cbind(age = 1:10,
    Price = logistic(1:10, 2, 3), Type = "Increasing")))
prices$Age = as.numeric(ac(prices$Age))

p = ggplot(prices, aes(x = Age, y = Price, group = Type))
p = p + geom_line(aes(colour = Type))
p

refIPrice = brp(FLBRP(alb, fbar = seq(0, 1, length.out = 101)))
refPPrice = refIPrice

price(refIPrice) = logistic(1:15, 4, 3)
price(refPPrice) = dnormal(1:15, 5, 1, 5)

refIPrice = brp(refIPrice)
refPPrice = brp(refPPrice)

breakEven = refIPrice
#### bug why not no recycling
refpts(breakEven) = refpts(as.numeric(c(refpts(refIPrice)["fmax",
    "revenue"] * 2, rep(NA, 7))), refpt = c("breakEven"))
computeRefpts(breakEven)[, "revenue"]

vcost(refIPrice) = c(computeRefpts(breakEven)[,
    "revenue"] * 0.2)
fcost(refIPrice) = vcost(refIPrice) * 4

vcost(refPPrice) = vcost(refIPrice)
fcost(refPPrice) = fcost(refIPrice)

refIPrice = brp(refIPrice)
refPPrice = brp(refPPrice)

price(refIPrice) = price(refIPrice)/c(refpts(refIPrice)["mey",
    "profit"])
price(refPPrice) = price(refPPrice)/c(refpts(refPPrice)["mey",
    "profit"])

refIPrice = brp(refIPrice)
```

```
refPPrice = brp(refPPrice)

plot(refPPrice)
plot(refIPrice)

data(ple4)

# Set up the stock for the next 6 years
stk = stf(ple4, 6)

# Set a constant recruitment based on the
# geometric mean of last 10 years
mnRec = FLPar(exp(mean(log(rec(ple4)[, ac(1992:2001)]))))
# Set ssb target to level 19 years ago
ssbTarget = ssb(ple4)[, "1992"]

## function to minimise
f = function(x, stk, ssbTarget, ctrl, sr) {
    ctrl@target[, "val"] = x
    ctrl@trgtArray[, "val", ] = x

    ssb. = c(ssb(fwd(stk, ctrl = ctrl, sr = sr))[,
        "2006"])

    return((ssb. - ssbTarget)^2)
}

## Recover stock to BMY in 2006 with a constant
## F strategy
ctrl = fwdControl(data.frame(year = 2002:2006,
    val = 0.5, rel = 2001, quantity = "f"))

xmin = optimize(f, c(0.1, 1), tol = 1e-07, stk = stk,
    ssbTarget = ssbTarget, ctrl = ctrl, sr = list(model = "mean",
        params = mnRec))
ctrl = fwdControl(data.frame(year = 2002:2006,
    val = xmin$minimum, rel = 2001, quantity = "f"))
stkF = fwd(stk, ctrl = ctrl, sr = list(model = "mean",
    params = mnRec))

# update catch slot
catch(stkF) = computeCatch(stkF)

# Have we reached the target?
```

```
ssbTarget
ssb(stkF)[, ac(2002:2006)]
# At what level of constant F
fbar(stkF)[, ac(2002:2006)]
# 'ave a butchers
plot(stkF[, ac(1957:2006)])

plot(albSSB)

data(ple4)
stk = stf(ple4, 6)

## Recover stock to the desired SSB in 2006
## with a constant Catch strategy Here val can
## be anything in the ctrl because it is
## overwritten in the optimisation loop
ctrl = fwdControl(data.frame(year = 2002:2006,
    val = c(catch(stk)[, "2001"]), quantity = "catch"))

xmin = optimize(f, c(100, 1e+05), tol = 1e-07,
    stk = stk, ssbTarget = ssbTarget, ctrl = ctrl,
    sr = list(model = "mean", params = mnRec))
ctrl = fwdControl(data.frame(year = 2002:2006,
    val = xmin$minimum, quantity = "catch"))
stkC = fwd(stk, ctrl = ctrl, sr = list(model = "mean",
    params = mnRec))

# Have we reached the target?
ssbTarget
ssb(stkC)[, ac(2002:2006)]
# At what level of constant catch
computeCatch(stkC)[, ac(2002:2006)]
# And at what level of F
fbar(stkC)[, ac(2002:2006)]
# Update the catch slot
catch(stkC) = computeCatch(stkC)

plot(stkC[, ac(1957:2006)])

# Assessment upto and including 2001
data(ple4)
black.bird = stf(stk, nyear = 2)

# set courtship and egg laying in Autumn
black.bird@m.spwn[] = 0.66
```

```
black.bird@harvest.spwn[] = 0.66

# assessment is in year 2002, set catch
# constraint in 2002 and a first guess for F
# in 2003
ctrl = fwdControl(data.frame(year = 2002:2003,
    val = c(85000, 0.5), quantity = c("catch",
        "f")))
black.bird = fwd(black.bird, ctrl = ctrl, sr = list(model = "mean",
    params = FLPar(25000)))

# HCR specifies F=0.1 if ssb<100000, F=0.5 if
# ssb>300000 otherwise linear increase as SSB
# increases
min.ssb = 1e+05
max.ssb = 3e+05
min.f = 0.1
max.f = 0.5

# slope of HCR
a. = (max.f - min.f)/(max.ssb - min.ssb)
b. = min.f - a. * min.ssb

# plot of HCR
plot(c(0, min.ssb, max.ssb, max.ssb * 2), c(min.f,
    min.f, max.f, max.f), type = "l", ylim = c(0,
    max.f * 1.25), xlim = c(0, max.ssb * 2))

## find F through iteration
t. = 999
i = 0
while (abs(ctrl@target[2, "val"] - t.) > 1e-05 &
    i < 50) {
    t. = ctrl@target[2, "val"]  ## save last val of F

    # calculate new F based on SSB last iter
    ctrl@target[2, "val"] = a. * c(ssb(black.bird)[,
        "2003"]) + b.
    ctrl@trgtArray[2, "val", ] = a. * c(ssb(black.bird)[,
        "2003"]) + b.
    black.bird = fwd(black.bird, ctrl = ctrl,
        sr = list(model = "mean", params = FLPar(25000)))

    # 'av a gander
```

```
    points(c(ssb(black.bird)[, "2003"]), c(ctrl@target[2,
        "val"]), cex = 1.25, pch = 19, col = i)
    print(c(ssb(black.bird)[, "2003"]))
    print(c(ctrl@target[2, "val"]))
    i = i + 1
}

# F bounds
black.bird = fwd(black.bird, ctrl = ctrl, sr = list(model = "mean",
    params = FLPar(25000)))
plot(FLStocks(black.bird))

#### Create a random variable for M
albM = albF1
m(albM) = propagate(m(albM), 100)

mDev = rlnorm(prod(dim(m(albM))), 0, 0.3)
mean(mDev)
var(mDev)^0.5

m(albM) = m(albM) * FLQuant(mDev, dimnames = dimnames(m(albM)))
plot(m(albM))

harvest(albM) = computeHarvest(albM)
catch(albM) = computeCatch(albM, "all")

ctrl = fwdControl(data.frame(year = 2008:2027,
    val = c(fbar(albF1)[, ac(2008:2027)]), quantity = "f"))
albM = fwd(albM, ctrl = ctrl, sr = sr)

plot(FLStocks(albM, albF1))

#### Create a random variable for M
albM1 = albM
m(albM1)[1:3, ] = m(albM)[1:3, ] * 2

harvest(albM1) = computeHarvest(albM1)
catch(albM1) = computeCatch(albM1, "all")
albM1 = fwd(albM1, ctrl = ctrl, sr = sr)

albM2 = albM
m(albM2)[, ac(2000:2027)] = m(albM)[, ac(2000:2027)] *
    2

harvest(albM2) = computeHarvest(albM2)
```

```
catch(albM2) = computeCatch(albM2, "all")
albM2 = fwd(albM2, ctrl = ctrl, sr = sr)

plot(FLStocks(albM, albM1, albM2))

#### process error in recruitment
srDev = FLQuant(rlnorm(20 * 100, 0, 0.3), dimnames = list(year = 2008:2027,
    iter = 1:100))
sr = fwd(albM, ctrl = ctrl, sr = sr, sr.residuals = srDev)
plot(sr)

sr = as.FLSR(alb, model = "bevholtSV")
sr1 = fmle(sr, fixed = list(spr0 = spr0(alb)))

#### SRR regime shifts
sr2 = fmle(sr, fixed = list(spr0 = spr0(alb),
    v = 0.75 * params(sr)["v"]))

alb2 = fwd(sr3, ctrl = ctrl, sr = sr2, sr.residuals = srDev)

plot(FLStocks(sr, sr2))
```

Syntax

Biology

Selectivity

Monte Carlo simulation

Targets

Catch and F strategies

Limits

Relative targets and limits

Harvest Control Rules

Multi-annual management

Recovery Plans

Long-term plans

Technical measures

fwdControl

target

trgtArray

effort

effArray

blocks

Management Objectives

Performance statistics

Examples

Albacore

Herring

Cod

North Sea flatfish

Bluefin

Mediterannean Swordfish

References

fwd