What is new in FLR 2.6.15

true

17 May, 2020

A release of FLR packages is now on the FLR repository, with binary packages being made available for R version 4.0.0. What is new in the FLR packages being released? This is a brief introduction, including example code, of the main developments and changes that can be found on a number of the FLR packages.

FLCore 2.6.15

divide() separates object along iters into list.

An object with *iters* can be turned into a list, where each element comes from one of the object iters.

```
# An FLQuant with 5 iterations
flq <- FLQuant(rep(seq(1, 5), each=40),
  dimnames=list(age=1:4, year=1:10, iter=1:5))

# Check iter 1
iter(flq, 1)
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##    year
## age 1 2 3 4 5 6 7 8 9 10
##   1 1 1 1 1 1 1 1 1 1 1
##   2 1 1 1 1 1 1 1 1 1 1
##   3 1 1 1 1 1 1 1 1 1 1
##   4 1 1 1 1 1 1 1 1 1 1
##
## units:  NA
```

```
# divide()
dflq <- divide(flq)

# Check element 1
dflq[[1]]
```

```
## An object of class "FLQuant"
```

```
## , , unit = unique, season = all, area = unique
##
##    year
## age 1 2 3 4 5 6 7 8 9 10
##   1 1 1 1 1 1 1 1 1 1 1
##   2 1 1 1 1 1 1 1 1 1 1
##   3 1 1 1 1 1 1 1 1 1 1
##   4 1 1 1 1 1 1 1 1 1 1
##
## units:  NA
```

```
# Result is an FLlst, in this case
is(dflq)
```

```
## [1] "FLQuants" "FLlst"    "list"     "vector"
```

fwd(FLQuant) and fwd(FLStock) to move one year forward.

Survivors can be simply calculated and projected forward one year, either for a single population vector (as an `FLQuant`) or a whole stock (`FLStock`). In the first case, abundances are move forward one year and age, without any mortality. This is still useful when those abundances are already at the end of year. In the second case natural and fishing mortality is applied. In both cases the last age is treated as a plusgroup, and a value for recruitment can provided using argument `rec`.

```
# Get a single population abundance vector
naa <- stock.n(ple4)[,"2000"]
fwd(naa, rec=6.34e5)
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##     year
## age  2001
##   1  634000.0
##   2  857525.0
##   3  531403.0
##   4  381345.0
##   5  510105.0
##   6   64650.9
##   7   28114.9
##   8    9398.3
##   9    5445.6
##   10  14237.6
```

```
##
## units:  1000
```

```
# Extract a single year of ple4
test <- ple4[, "2000"]

# Bring it forward
ftest <- fwd(test, rec=rec(ple4)[,"2001"])

# stock.n from ple4 and fwd are almost identical
all.equal(stock.n(ftest), stock.n(ple4)[, "2001"])
```

```
## [1] "Mean relative difference: 4.598e-07"
```

`mohnMatrix()` to construct a table of metrics to compute Mohn's rho

The `icesAdvice`[1] package already provides a function to calculate Mohn's rho retrospective metric, from a matrix of years and retro peels. This matrix can be generated from the results of a retrospective analaysis, stored as an `FLstocks` object, by using `mohnMatrix`. A metric to compute the matrix for must be selected, as a function or function name, and defaults to F (`fbar`).

[1] https://github.com/ices-tools-prod/icesAdvice

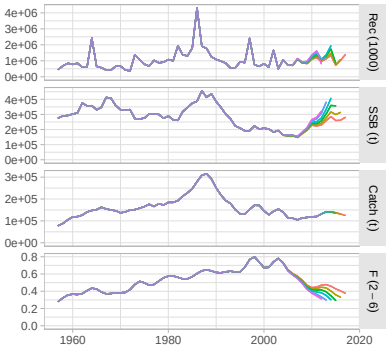

Figure 1: 'retro' is an FLStocks of FLXSA runs, 5 peels of one year less at a time.

```
# mohnMatrix returns the years * peel matrix
mohnMatrix(retro, metric=ssb)
```

```
##         base     -1     -2     -3     -4     -5
## 2010 228416 233236 239118 246959 257466 266074
## 2011 223386 230425 239626 251385 269576 280724
## 2012 230046 241472 257090 276252 307096 321010
## 2013 258728 277516 302933 333093 381062     NA
## 2014 285755 316605 359700 406433     NA     NA
## 2015 259635 299183 356650     NA     NA     NA
## 2016 261399 313386     NA     NA     NA     NA
## 2017 280662     NA     NA     NA     NA     NA
```

```
# to use with icesAdvice::mohn
icesAdvice::mohn(mohnMatrix(retro, metric="ssb"))
```

```
## [1] 0.3726
```

`standardUnits` provides a set of standard units of measurement

All slots in an `FLStock` can be set to have a set of standard units of measurement by setting, using `units<-`, the result of this method. The standard

for asn `FLStock` is to use `1000` for number in thousands, `kg` for weights, `t` for biomass, `m` and `f` for nruital and fishing mortality, and an empty string for proportions, such as `mat`, `m.spwn`, and `harvest.spwn`.

```
# Create an FLStock, no units in slots
fls <- FLStock(m=FLQuant(0.2,
  dimnames=list(age=1:4, year=2010:2018)))

# Get standard units for this class, and assign
units(fls) <- standardUnits(fls)
```

`window()` accepts a negative value as `end`

When shortening an object using `window`, a number of years can now be dropped uisng a negavive number of the `end` argument value.

```
# Drop last 3 years
window(FLQuant(seq(1:8)), end=-3)
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## quant 1 2 3 4 5
##   all 1 2 3 4 5
##
## units:  NA
```

`append()` to add an object after another along the `year` dimension

A single `FLQuant` or a whole `FLStock` can be extended in time with another object using `append()`. `year` dimnames are used to match both objects so no `year` is repeated. Shared `year` dimensions will be taken from the appended object, its second argument. If a gap exists, `NA` will appear.

```
# Append two objects with contiguous years
append(stock(ple4)[,ac(2005:2006)],
  stock(ple4)[,ac(2007:2008)])
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age    2005    2006    2007    2008
##   all 372721 422631 434641 528377
```

```
##
## units:  t
```

```
# Years clash, so values taken from second
append(stock(ple4)[,ac(2005:2006)],
  stock(ple4)[,ac(2006:2007)] / 1000)
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   2005      2006      2007
##   all 372720.65    422.63    434.64
##
## units:  t
```

```
# Gap in years, so left empty
append(stock(ple4)[,ac(2005:2006)],
  stock(ple4)[,ac(2008:2009)])
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##      year
## age   2005   2006   2007   2008   2009
##   all 372721 422631     NA 528377 629539
##
## units:  t
```

residuals methods

Calculation of residuals can now be carried out using different methods but under a common interface. The alternative residuals currently available are:

- Log-standardized, using method `rlogstandard()`
- Pearson, by calling `rstandard()`
- Studentized, using `rstudent()`

All of them requyire the observation ads first argument and the model fit as the second. In addition, standardized residuals need the standard deviation. If not supplied, it is calculated along the `year` dimension.

A call to `residuals()` will give you access to those methods, selected through the `type` argument.

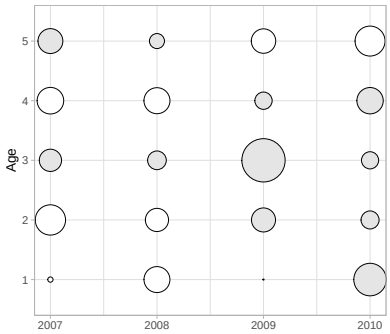

Figure 2: Example of log standardized residuals.

```
residuals(obs, fit, type="log")
```

```
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##    year
## age 2007     2008     2009     2010
##   1 -0.12320 -0.94970  0.09977  1.46022
##   2 -1.27190 -0.76932  0.82324  0.49378
##   3  0.71572  0.52504  2.55281  0.45434
##   4 -0.99305 -0.95430  0.46024  0.97784
##   5  0.87831  0.36645 -0.84863 -1.24705
##
## units:
```

`metrics()` can now process formulas combining functions and `FLPar`

The `metrics()` method computes metrics from complex objects and simplifies extracting values from, for example, an `FLStock`.

```
mets <- metrics(ple4, list(SB=ssb, REC=rec))
plot(mets)
```

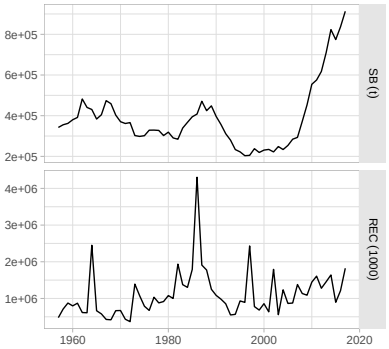

Figure 3: ssb() and rec() extracted using metrics()

Those metrics can now be defined to include a named numeric vector or an `FLPar`, for example a reference point. Names in the formula and the object need to match.

```
mets <- metrics(ple4, list(SB=~ssb / Bmsy,
  F=~fbar / Fmsy), FLPar(Bmsy=3.5e5, Fmsy=0.32))
plot(mets) + geom_hline(yintercept=1, linetype=2)
```

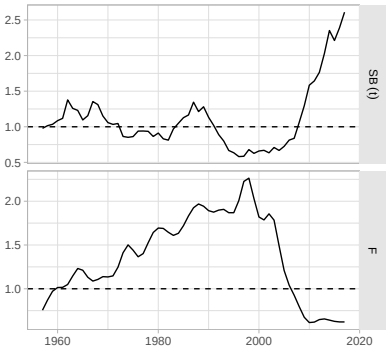

Figure 4: SB/B_{MSY} and F/F_{MSY} as computed from a call to metrics. as shown.

`intersect()` method called on two FLQuants subsets to common dimension names.

The common dimensions of two `FLQuant` objects can now be used to subset them at the same time by calling `intercept`. The method works on dimension names to find the matching sections on the objects. We can then operate on them in m, nay ays, as their dimensions match.

```
# Objcts with mismatching aged and years
fq1 <- FLQuant(2, dimnames=list(age=0:3,
  year=2015:2019))
fq2 <- FLQuant(5, dimnames=list(age=2:6,
```

```
  year=2016:2019))

# An FLQuants object with the subset inputs
intersect(fq1, fq2)
```

```
## $ NA
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##    year
## age 2016 2017 2018 2019
##   2 2    2    2    2
##   3 2    2    2    2
##
## units:  NA
##
## $ NA
## An object of class "FLQuant"
## , , unit = unique, season = all, area = unique
##
##    year
## age 2016 2017 2018 2019
##   2 5    5    5    5
##   3 5    5    5    5
##
## units:  NA
```

Other changes in FLCore 2.6.15

- `readVPAInterCatch()` creates an FLQuant from a VPA file in the format exported by ICES Intercatch system.

ggplotFL

FLa4a