

Plotting FLR objects with ggplotFL and ggplot2

Iago Mosqueira, EC JRC¹ - FLR Project

August 2013

Using ggplot2 with FLR objects

The ggplot2 package provides a powerful alternative paradigm for creating simple and complex plots in R using the ideas of Wilkinson's *Grammar of Graphics*²

To facilitate the use of ggplot2 methods in FLR, the ggplotFL package has been created. The main resources on offer in this package are overloaded versions of the ggplot() method that take directly certain FLR classes, a new set of basic plots for some FLR classes, based on ggplot2 instead of lattice, and some examples and documentation on how best make use of ggplot2's powerful paradigm and implementation to obtain high quality plots for even fairly complex data structures.

The overloaded 'ggplot' method

The standard ggplot functions expects a data.frame for its first argument, data. If ggplot is called on an FLR object, a conversion to data.frame takes place before the result, plus any other arguments provided, get passed to the original ggplot(). Conversion makes use of as.data.frame³ methods defined in FLCore, with the cohort argument set to TRUE.

FLQuant

Passing an FLQuant object to ggplot, we can specify the names of the dimensions as variables in the plot, where data refers to the column storing the actual numeric values. For example, to plot data (the catch slot from ple4 in this case) against year, we could use

```
ggplot(data = catch(ple4), aes(year, data)) + geom_point() +  
  geom_line() + ylab("Catch (t)") + xlab("")
```

FLQuants

Similarly, we can pass on to ggplot an object of class FLQuants, and the conversion to data.frame will make use of the corresponding method⁴. A new column gives the name of each FLQuant in the list, called qname. When can then use it to, for example, define a call to facet_wrap() to obtain a separate subplot per element.

¹ European Commission
Joint Research Center
IPSC - Maritime Affairs Unit
Ispra, Italy
<https://fishreg.jrc.ec.europa.eu/>

² Wilkinson, L. 1999. *The Grammar of Graphics*, Springer. ISBN 0-387-98774-6.

³ method?as.data.frame('FLQuant')

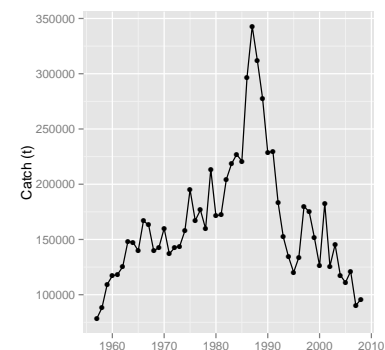


Figure 1: Combined line and point plot of a time series from an FLQuant object.

⁴ method?as.data.frame('FLQuants')

```
ggplot(data = FLQuants(Yield = catch(ple4), SSB = ssb(ple4),
  F = fbar(ple4)), aes(year, data)) + geom_line() +
  facet_wrap(~qname, scales = "free", nrow = 3) +
  labs(x = "", y = "")
```

This procedure is particularly useful when plotting information from objects with multiple FLQuant slots, as a subset of slots can be selected for plotting, and even transformations or computations can be carried out in the call to the FLQuants() creator.

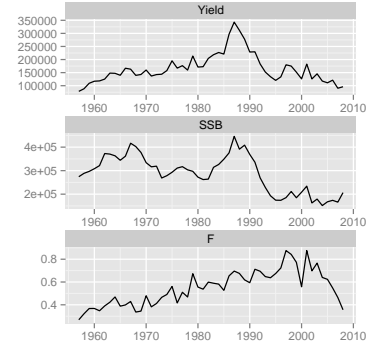


Figure 2: Facet wrap line plot of time series from an FLQuants object.

FLStock

A whole FLStock object can also be used as argument to ggplot(), even if the heterogeneity in scale of the data contained makes the plot slightly confusing. For example, we can plot time series of every FLQuant slot in ple4, with color applied to different age dimensions, as follows:

```
ggplot(data = ple4, aes(year, data)) + geom_line(aes(group = age,
  colour = factor(age))) + facet_wrap(~slot, scales = "free",
  nrow = 3) + labs(x = "", y = "") + theme(legend.position = "none")
```

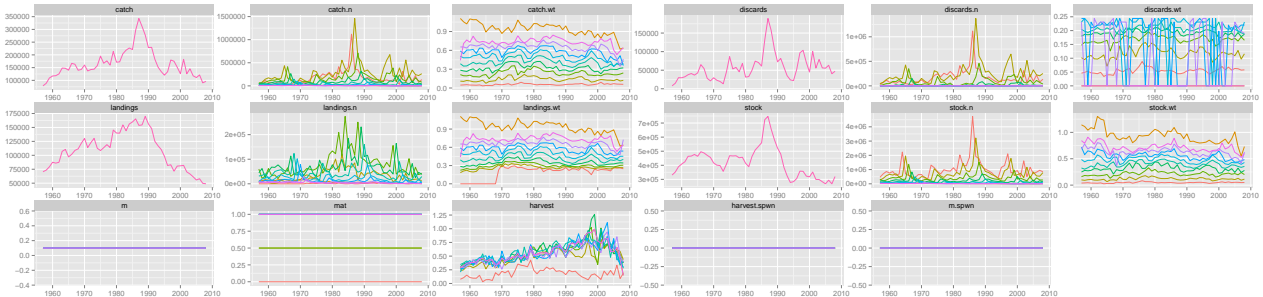


Figure 3: Overall ggplot of an FLStock object, faceted by slot.

New plot() methods for FLR classes

The ggplotFL package also provides new versions of the plot method for a number of FLR classes. Each S4 class defined in any FLR package has a plot() method defined that provides a quick visual summary of the contents of the object.

FLStock

The ggplotFL version of the standard plot for the FLStock class, contains the time series of recruitment (by calling rec()), SSB (ssb()),

catch (`catch()`), and fishing mortality or harvest (`fbar()`). The four panels are now arranged in a 4-row matrix to better display the trends in the time series.

Using ggplot2 by converting to data.frame

The methods shown above simply depend on conversion of *FLR* objects into `data.frame`, which can then be passed to `ggplot()`. Calling `ggplot` on an *FLR* object takes care of this conversion behind the scenes, but to obtain certain plots, it is best to directly convert the *FLR* objects into a `data.frame`.

Some examples

Example: plot quantiles of a simulation

To have full control over a plot of the median (or mean) and the confidence or probability intervals of a simulated or randomized time series, i.e. an *FLQuant* object with `iters`, we need to arrange the different values computed from the object in separate columns of a `data.frame`.

If we start with some random *FLQuant* object, such as

```
f1a <- rlnorm(100, FLQuant(exp(cumsum(rnorm(25, 0,
0.1)))), 0.1)
ggplot(f1a, aes(factor(year), data)) + geom_boxplot() +
  xlab("")
```

we can first compute the necessary statistics on the object itself, as these operations are very efficient on an array. `quantile()` on an *FLQuant* will return the specified quantiles along the `iter` dimension. Let's extract the 10th, 25th, 50th, 75th and 90th quantiles.

```
flq <- quantile(f1a, c(0.1, 0.25, 0.5, 0.75, 0.9))
```

The object can now be coerced to a `data.frame`

```
fdf <- as.data.frame(flq)
```

and inspected to see how the 100 iters have been now turned into the five requested quantiles in the `iter` column

```
##  quant year  unit season  area iter  data
## 1   all    1 unique   all unique 10% 2.307
## 2   all    2 unique   all unique 10% 2.462
## 3   all    3 unique   all unique 10% 2.159
```

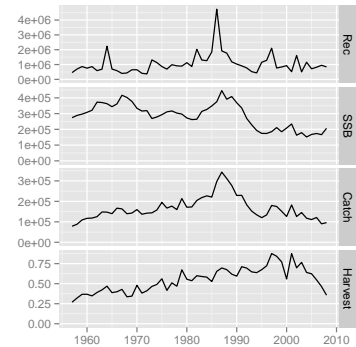


Figure 4: ggplot2 version of the standard `plot()` for *FLStock*, as applied to `ple4`

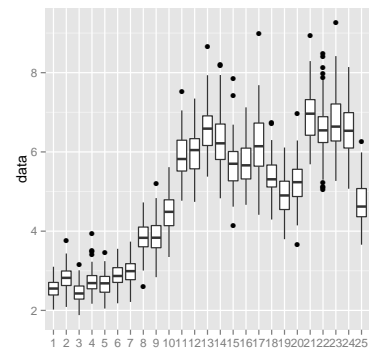


Figure 5: Distribution of values of a simulated time series plotted using `geom_boxplot()`

The long format data.frame can be reshaped into a wide format one so that we can instruct ggplot to use the quantiles, now in separate columns, to provide limits for the shaded areas in geom_ribbon. To do this we can use cast, as follows

```
fdw <- cast(fdf, quant + year + unit + season + area ~
  iter, value = "data")
```

This creates a wide data.frame in which the iter column is spread into five columns named as the levels of its conversion into factor

```
levels(fdw[, "iter"])
```

```
## [1] "10%" "25%" "50%" "75%" "90%"
```

We can now use those five quantile columns when plotting shaded areas using geom_ribbon. Please note that the column names returned by quantile() need to be quoted using backticks.

```
p <- ggplot(data = fdw, aes(x = year, y = '50%')) +
  geom_line() + geom_ribbon(aes(x = year, ymin = '10%',
    ymax = '90%'), fill = "red", alpha = 0.15) + geom_ribbon(aes(x =
    year, ymin = '25%', ymax = '75%'), fill = "red", alpha = 0.25) +
  ylab("data")
print(p)
```

Assigning the result of the call to ggplot() to a variable, as done above, will allow us to reuse the plot later on by modifying or adding components.

Example: Simulation trajectories plot

If the result of an stochastic simulation is summarised by showing credibility intervals, it is very informative to plot as well some of the individual iterations as a way of showing the fact that individual trajectories are generally not as smooth as, for example, the median shown in the figure above.

```
fds <- as.data.frame(iter(fla, c(1, 4, 23)))
```

```
p + geom_line(data = fds, aes(year, data, colour = iter),
  size = 1) + theme(legend.position = "none")
```

This is easy to do in ggplot2 by adding an extra element on top of the previous plot, stored in the p object from the code above.

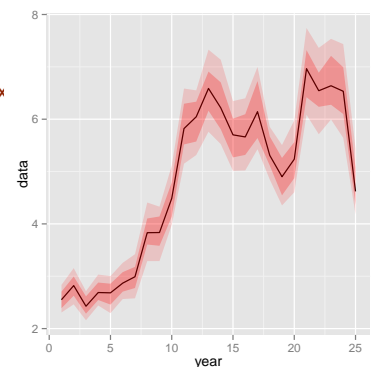


Figure 6: Time series with 75% and 90% credibility intervals plotted using geom_ribbon.

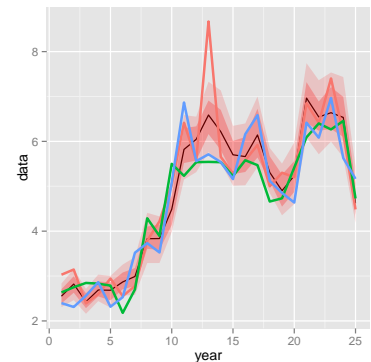


Figure 7: Spaghetti plot of an stochastic simulation, by calling geom_line on top of the stored ribbon plot.

Example: Using FLQuants

Coercion using `as.data.frame`, combined with the use of `cast` and `melt` (from the `reshape` package), provides the *FLR* user with most tools required to create a large range of `ggplots` out of any *FLR* object.

TODO: ADD text

Example: Bubble plots

TODO: ADD text

```
ggplot(catch.n(ple4), aes(year, as.factor(age), size = data)) +
  geom_point(shape = 21) + scale_size(range = c(1,
    20)) + ylab("age") + theme(legend.position = "none")
```

Example: Residual plots

TODO: ADD text

```
dat <- as.data.frame(catch.n(ple4))
dat$resid <- dat$data - mean(dat$data)
```

```
ggplot(dat, aes(year, as.factor(age), size = resid)) +
  geom_point(shape = 21, aes(colour = factor(sign(resid)),
    fill = factor(sign(resid)))) + scale_size(range = c(1,
    20)) + scale_colour_manual(values = c("black",
    "white")) + scale_fill_manual(values = c("lightgray",
    "black")) + theme(legend.position = "none") + ylab("age")
```

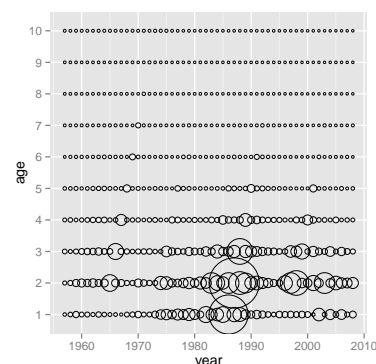


Figure 8:

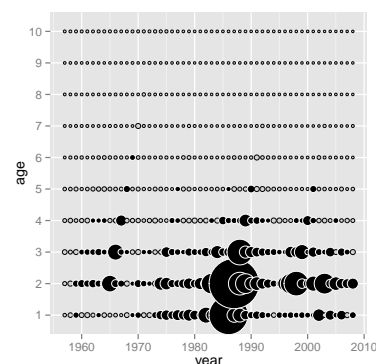


Figure 9:

More information

- You can submit bug reports, questions or suggestions on `ggplotFL` at the `ggplotFL` issue page ⁵, or on the *FLR* mailing list.
- Or send a pull request to <https://github.com/flr/ggplotFL/>
- For more information on the *FLR* Project for Quantitative Fisheries Science in R, visit the *FLR* webpage ⁶.
- To learn more about `ggplot2`, visit the `ggplot2` website ⁷, or look at the `ggplot2` book.⁸

⁵ <https://github.com/flr/ggplotFL/issues>

⁶ <http://flr-project.org>

⁷ <http://ggplot2.org/>

⁸ Wickham, H. 2009. *ggplot2: Elegant Graphics for Data Analysis*. Springer, Use R! Series. ISBN 978-0-387-98140-6

- The latest version of ggplotFL can always be installed using the devtools package, by calling

```
library(devtools)  
install_github("ggplotFL", "flr")
```

Package versions

- R: R version 3.0.1 (2013-05-16)
- ggplot2: 0.9.3.1
- ggplotFL: 2.15.20130823
- FLCore: 2.5.20130902

Compiled: Tue Sep 10 12:24:46 2013