Plotting FLR objects with ggplotFL and ggplot2

Iago Mosqueira, EC JRC[1] *- FLR Project*

August 2014

[1] European Commission
Joint Research Center
IPSC - Maritime Affairs Unit G04
Ispra, Italy
`https://fishreg.jrc.ec.europa.eu/`

Using ggplot2 with FLR *objects*

The `ggplot2` package provides a powerful alternative paradigm for creating both 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 certaing *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.

[2] Wilkinson, L. 1999. *The Grammar of Graphics*, Springer. doi 10.1007/978-3-642-21551-3_13.

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.frame`[3] methods defined in `FLCore`, with the `cohort` argument set to `TRUE`.

[3] `method?as.data.frame('FLQuant')`

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("")
```

where we pass directly an `FLQuant` object for the `data` argument in `ggplot`, specify an aesthetic mapping (`aes(year, data)`), and add both points (`geom_point()`) and lines (`geom_line()`), together with the appropriate axis labels.

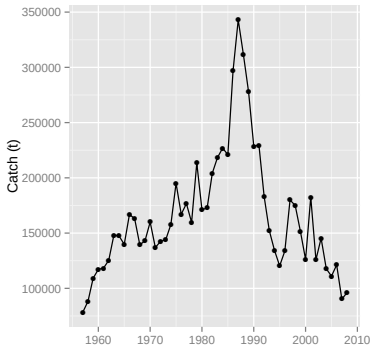


Figure 1: Combined line and point plot of a time series from an FLQuant object.

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 method [4]. 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.

```
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.

[4] `method?as.data.frame('FLQuants')`

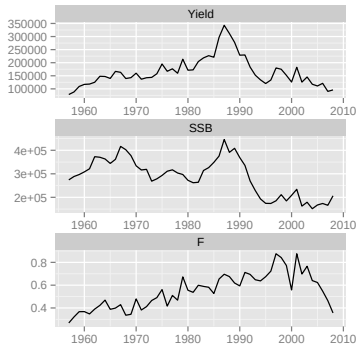


Figure 2: Facet wrap line plot of time series from an FLQuants object.

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, by calling

```
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* pack-

age has a `plot()` method available that provides a quick visual summary of the contents of the object.

FLQuant

The standard `plot()` method for `FLQuant` defined in `ggplotFL` uses the faceting capabilities of `ggplot` to better present some of the multiple dimensions of these objects. If any dimension, other than `year` and `iter`, has length greater than one, it will be added to the formula used by `facet_grid`. For example, an `FLQuant` with dimensions

```
dim(catch.n(ple4))
```

```
## [1] 10 52  1  1  1  1
```

will generate a plot with a time series by year of the data it contains, with horizontal facets for the only dimension, other than `year`, of length greater than 1, `age`.

```
plot(catch.n(ple4))
```

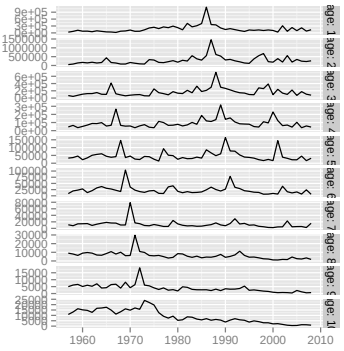


Figure 4: Standard ggplot2-based plot for an FLQuant object with multiple *year*s and *age*s.

For `FLQuant` objects with iterations, the `plot` method will calculate, by default, the 50% (median), 10%, 25%, 75% and 90% quantiles, to be plotted as a solid line (50%), a dotted line (90%) and a coloured ribbon (75%).

```
plot(rlnorm(200, fbar(ple4), 0.15))
```

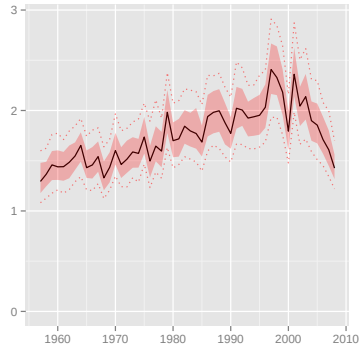


Figure 5: Standard ggplot2-based plot for an FLQuant object with multiple iterations

Different quantiles can be specified using the `probs` arguments, and the alpha transparency of the ribbon will be proportional to the probability value. A vector of odd length must be passed, and the central point should be 0.50 so the central tendency line represents the median. The most extreme quantiles will be plotted as dotted lines, while all other will show up as ribbons.

```
plot(rlnorm(200, fbar(ple4), 0.15), probs = c(0.05,
    0.1, 0.25, 0.5, 0.75, 0.9, 0.95))
```

FLQuants

The `plot` method for `FLQuants` will now by default show each object in a horizontal panel, with indpendent scales, by using `facet_grid`. Objects with iterations will have, as with `plot` for `FLQuant`, their median, 10%, 25%, 75% and 90% quantiles shown as a black line and red ribbons with different levels of transparency, respectively.

```
fqs <- FLQuants(F = rlnorm(200, fbar(ple4), 0.15),
    SSB = ssb(ple4), Rec = rec(ple4))
plot(fqs)
```

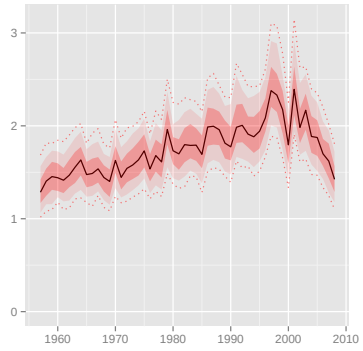
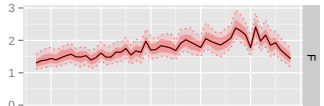


Figure 6: Standard ggplot2-based plot for an FLQuant object with multiple iterations and user-specified quantiles.



FLStock

The `ggplotFL` version of the standard plot for the `FLStock` class, contains the time series of recruitment (obtained by calling `rec()`), SSB (`ssb()`), catch (`catch()`), and fishing morality or harvest (`fbar()`). The four panels are now arranged in a 4-row matrix to better display the trends in the time series.

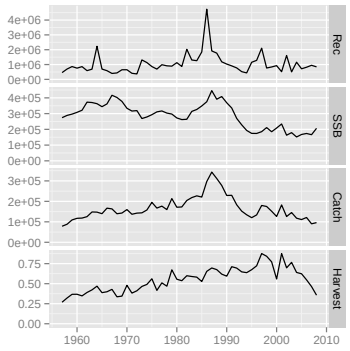


Figure 8: ggplot2 version of the standard plot() for FLStock, as applied to `ple4`

FLStocks

Similarly, the standard `plot()` method for the `FLStocks` class now relies on ggplot. For example, we can create an example `FLStocks` object by splitting the female and male units of `ple4sex` and adding them as separate elements in the list. A call to `plot()` would give us the corresponding plot. Remember the object returned by `ggplot` can always be assigned to a variable in the workspace and modified as required.

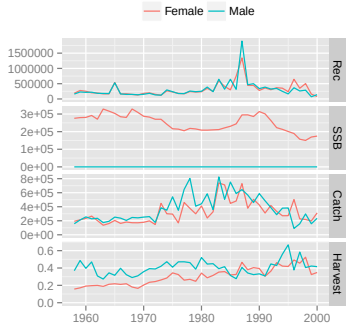


Figure 9: ggplot2 version of the standard plot() for FLStocks, as applied to the sex-separated FLStock object `ple4sex`

FLSR

The `ggplotFL` version of the class plot for `FLSR` contains the same six panels as before: (1) stock-recruit data, fitted model and lowess smoother, (2) residuals by year, (3) lag 1-correlated residuals, (4) residuals by SSB, (5) residuals qqplot and (6) residuals by fitted values. Blud lines are lo9wess smoothers, to better visualize trends in the data shown.

```
plot(nsher)
```

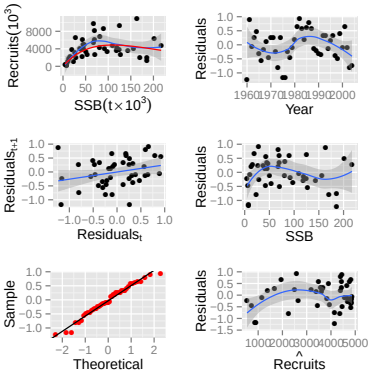


Figure 10: Standard ggplot2-based plot for an FLSR.

Using ggplot2 by converting to data.frame

The methods shown above 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 full control and develop certains plots, it is best to explicitely convert the *FLR* objects into a `data.frame`. Different conventions are used in the naming of the data.frame columns created from various *FLR* classes, which need to be used when the plot is specified. For further information, please see the help pages for each `data.frame()` method [5].

[5] For example `method?as.data.frame('FLQuants')`

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

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

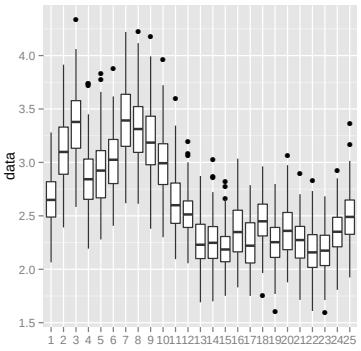


Figure 11: Distribution of values of a simulated time series plotted using geom_boxplot()

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(fla, 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% 2.38
## 2   all    2 unique    all unique  10% 2.69
## 3   all    3 unique    all unique  10% 2.87
```

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 `reshape2::dcast`, as follows

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

```
## Using data as value column: use value.var to override.
```

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(fdf[, "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_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) +
    geom_line() + 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.

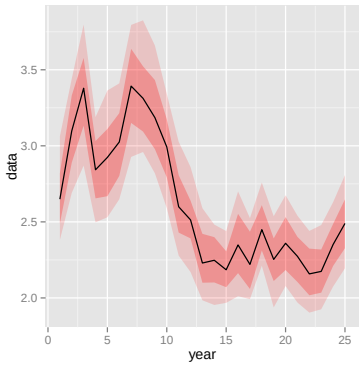


Figure 12: Time series with 75% and 90% credibility intervals plotted using geom_ribbon.

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.

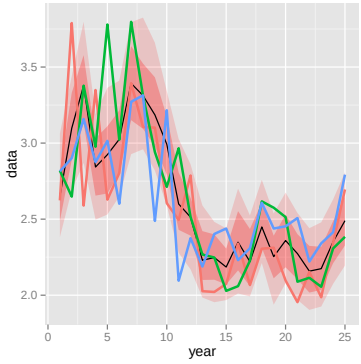


Figure 13: Spaghetti plot of an stochastic simulation, by calling geom_line on top of the stored ribbon plot.

Example: Using FLQuants

Coercion using `as.data.frame`, combined with the use of `dcast` and `melt` (from the `reshape2` package[6]), provides the *FLR* user with the tools required to create a large range of `ggplots` out of any *FLR* object.

TODO: ADD text & example

Example: Bubble plots

Bubble plots allow us to represent a third continuous dimension in a scatter plot by sizing points according the value of a variable. For example, catch in numbers by age and year can be visualized using

[6] `http://cran.r-project.org/web/packages/reshape2/index.html`

```
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")
```

where *data* is used to size the bubbles in the call to *aes()*. This single line of code replaces the functionality offered by the *lattice*-based *bubbles()* method available in *FLCore*.

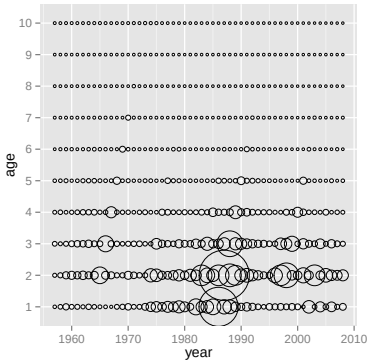


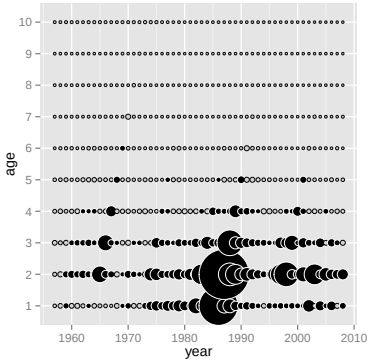
Figure 14: Bubble plot of catch by age in numbners for North Sea plaice.

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")
```



More information

- You can submit bug reports, questions or suggestions on `ggplotFL` at the `ggplotFL` issue page [7], 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 [8].
- To learn more about ggplot2, visit the ggplot2 website [9], or look at the ggplot2 book.[10]
- The latest version of `ggplotFL` can always be installed using the `devtools` package, by calling

[7] `https://github.com/flr/ggplotFL/issues`

[8] `http://flr-project.org`

[9] `http://ggplot2.org/`

[10] Wickham, H. 2009. *ggplot2: Elegant Graphics for Data Analysis*. Springer, Use R! Series. doi:10.1111/j.1467-985X.2010.00676_9.x

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

Software Versions

- R version 3.1.1 (2014-07-10)

- ggplotFL: 2.5.20140917
- FLCore: 2.5.20140919
- ggplot2: 1.0.0
- **Compiled**: Thu Oct 2 21:08:25 2014
- **Git Hash**: 1074019

`knitr` options

```
opts_chunk$set(dev = "pdf", cache = TRUE, fig.width = 4.5,
    fig.height = 4.5, tidy = TRUE)
```