xts Plots

Michael Weylandt

January 7, 2014

Contents

1 Introduction

As part of the Google Summer of Code 2012, the plotting functionalities availble for `xts` objects has been greatly extended. This new plotitng functionality is currently available in a package **xtsExtra** available off R-forge and is under active development. Care has been taken not to break back compatability with the published `xts::plot.xts` available from the `CRAN`˜version of **xts** while providing a new and powerful set of plotting routines for time-oriented data.

While care has been taken to make sure the new `plot.xts` behaves intuitively, flexibility does come at a price of some API complexity and this document is provided as a guide for both developers and advanced users who may wish to make use of the extended capabilities. Note that this document may lag development from time to time and is subject to non-back-compatible change.

The `xts` time series class was designed to provide users and developers an intuitive and transparent platform for time series analysis in `R`˜˜[3]. While the functions described in this document will only be called automatically when used on `xts` objects, the `reclass` paradigm described in the **xts** vignette [5] is used internally, so that all time series classes may make use of advanced plotting capabilities.

2 plot.xts

3 barplot.xts

The second of the graphical primitives provided in **xtsExtra** is a `barplot` method, adapted from Peter Carl's code in **PerformanceAnalytics**˜[6]. Implemented as an exported and registered method of the `S3` generic `barplot`, `barplot.xts` has the following arguments:

```
> names(formals(barplot.xts))

 [1] "height"       "stacked"      "scale"        "auto.legend"
 [5] "major.format" "ylim"         "space"        "cex.axis"
 [9] "cex.legend"   "cex.lab"      "cex.labels"   "cex.main"
[13] "xaxis"        "box.color"    "xlab"         "ylab"
[17] "major.ticks"  "minor.ticks"  "xaxis.labels" "col"
[21] "..."
```

Let us examine these arguments in order.

`height` So called for compatability with `graphics::barplot`, this should be an `xts`-ible object as it will be converted by `xts::try.xts` internally.

`stacked` Defaulting to `TRUE`, this defines whether a *stacked* barplot should be produced. In the author's opinion, stacked barplots are to be preferred for time-oriented barplots as they align observations into a single vertical unit.

`scale` Defaulting to `FALSE`, this applies the transform `x <- x / rowSums(x)` to data; this transform is useful for seeing how the relative makeup of `height` changes over time. Currently, if `any(height < 0)`, this option throws an error. It is also likely to cause problems if `any(rowSums(height) == 0)`.

`auto.legend` Defaulting to `TRUE`, this places a legend underneath the barplot. Column names are used as the legend labels. Attractive defaults have been chosen, but for more detailed control, the user is encouraged to add the legend himself using the unexported `xtsExtra::do_barplot.legend`, as described below.

`major.format` Control the format of the time axis labels; see the details `?strptime` for formatting codes. If left as the default (`TRUE`), `axTicksByTime` attempts to automatically pick an appropriate labelling convention.

`ylim` Control the y-axis limits of the resulting plot. If default (`NULL`), the limits will be chosen automatically. Expect handling to be moved to `...` in future development.

`space` Specifies the width of the interbar spacing ; not currently supported for plots with `stacked = FALSE`. Possibly will be reworked to provide more

robust support in light of the x-axis spacing given by the index of the `xts` class.

`cex.*` These arguments control the size of various labels. Expect their handling to be moved to use `...` and `par` arguments in future development.

`xaxis` Defaulting to `TRUE`, should an x-axis and labels be drawn?

`box.color` Defaulting to `"black"`, gives the color of less important plot elements, such as the outside boundaries and the legend box. Some authors prefer `box.color = "darkgray"` for a softer appearance.

`xlab, ylab` Labels for the x- and y-axes. Expect their handling to be moved to `...` in future development.

`major.ticks, minor.ticks` See the fuller description of these arguments given for `plot.xts` (Section 2).

`col` Color of the bars. If missing, defaults to `col = seq_len(NCOL(height))` chosen according to the somewhat unattractive `R`defaults provided by `palette`. See more on color palettes below.

3.1 Time Oriented Barplots

We begin by creating an example of an `xts` barplot and then we discuss its construction in more detail.

```
> x <- xts(matrix(abs(rnorm(72)), ncol = 6), Sys.Date() + 1:12)
> colnames(x) <- LETTERS[1:6]
> barplot(x)
```

producing the plot in figure 1.

We note immediately that, by default, the produced barplot is a so-called "stacked" barplot, corresponding to `beside = FALSE` in the default `barplot` method. An advantage of this display is that observations for each time period are aligned vertically; a current limitation of the barplot code is that x-axis spacing does not accurately reflect irregularities in the underlying data, as in figure 2. We see in both the preceeding plots that the default axis labels accurately reflect the underlying daily periodicity of our data set, as with `plot.xts` more control could be had by passing a format string to `major.format`. E.g., to remove year labelling, we would pass `major.format = "%b %d"` to print names of the format "Jul 25."

Note that negative values are stacked underneath the x-axis following the exampleof`barchart` in the **lattice** package.

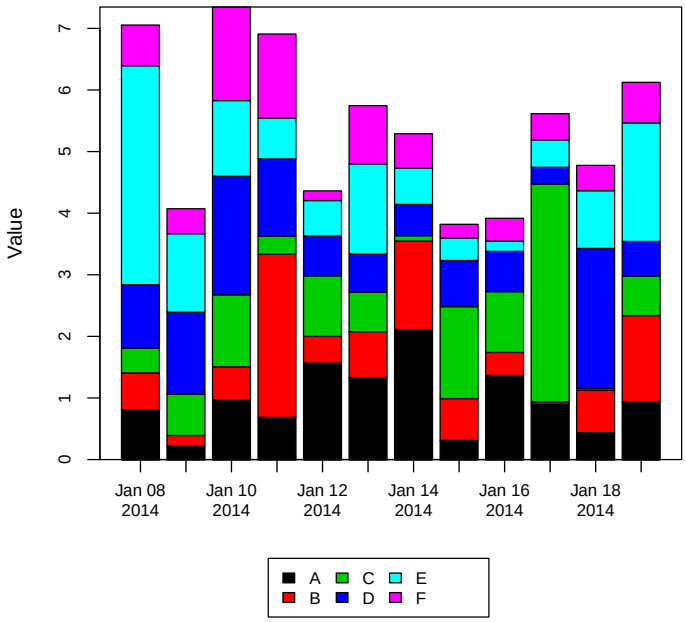


Figure 1: `barplot(x)`

3.2 Scaled and Unstacked Plots

A limitation of stacked barplots is that the eye is naturally drawn to the size of the bar, rather than the widths of the bands which comprise it. Further, while it is possible to compare the size of the bottom stripe, comparing the higher stripes is more difficult as they do not generally share a common baseline. Graphics experts like Cleveland˜[7] and Tufte˜[8] use of the trellis or small multiples paradigm to avoid these problems.[1] Over and against their better judgement, we provide two possible customizations to address these concerns.

In certain applications, e.g., asset class weights in finance, shifting population dynamics in ecology, or server load balance in IT, it is sometimes of greater interest to see how the relative make-up of quantities change over time rather than the scale of those quantities. For those cases, `barplot.xts` can be used with the option `scale = TRUE` which applies the transform `x <- x / rowSums(x)` before plotting.

For example, the data created above can be interpreted as asset classes and we

[1] See, e.g., `barchart` in the recommended **lattice** package.

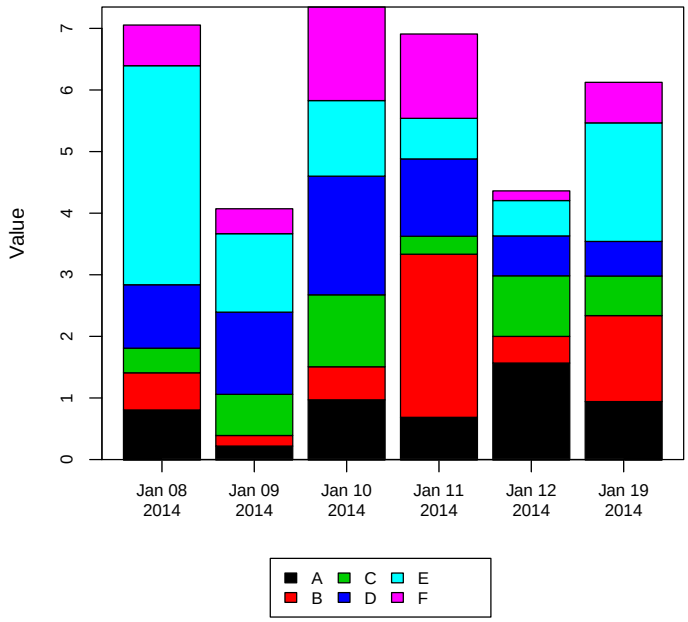


Figure 2: `barplot(x[c(1:5,12),])`

can show the effect of the scaled plots.

```
> colnames(x) <- c("Equity", "Fixed Income", "Commodities",
+                  "FX","Convertibles","Alternatives")
```

producing 3.

Currently, quantity scaling is only supported for non-negative data. For data of alternating sign, the scale transform is not uniquely defined and is, as such, left to the user. Two common choices are given here, but not implemented within the package.

```
> scale1 <- function(x){x/rowSums(abs(x))}
> scale2 <- function(x){
+     ## Can this be vectorized?
+     for(j in seq_len(NROW(x))){
+         x[j, x[j,] > 0] <- x[j, x[j,] > 0] / sum(x[j, x[j,] > 0])
+         x[j, x[j,] < 0] <- -1 * x[j, x[j,] < 0] / sum(x[j, x[j,] < 0])
+     }
+     x
+ }
```

`scale1()` transforms `x` such that the height of each bar is `1`, as with the non-negative scale transform. Since bars may now contain negative quantities, the entirety of the bar will shift up and down as a linear transform of its sum. This rescaling is helpful in seeing how the total value transforms over time.

`scale2()` scales both the positive and negative row elements to sum to `1` independently. This is useful in more specialized circumstances, such as examing exposures of a long-short portfolio.

If absolute quantities are of interest, it is sometimes desirable to create a time-oriented barplot without stacked bars, that is, with each data point being anchored on the x-axis. To do so, we simply use `stacked = FALSE`, which corresponds to `beside = TRUE` in the default method of `barplot`. This method has the slight disadvantage of no longer aligning simultaneous observations along the time axis, but can be helpful if properly interpreted, as shown in figure 4.

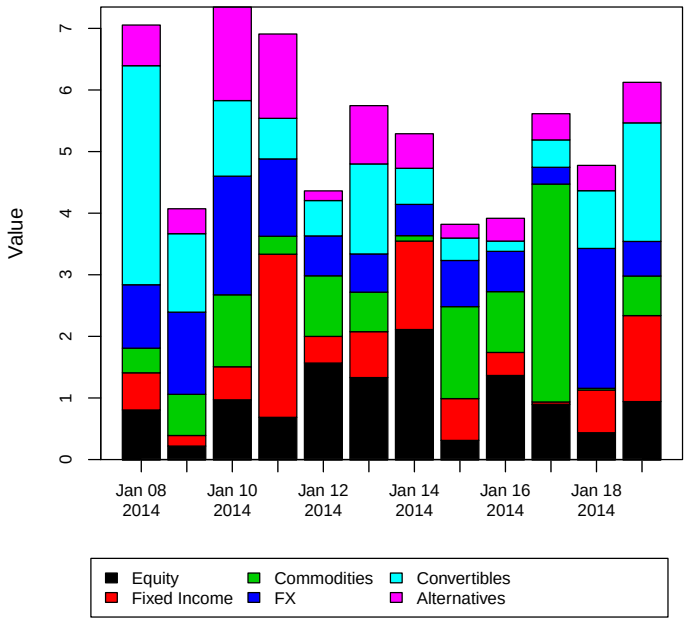
Note that the form of the unstacked barplot is subject to change as the author is not entirely happy with it.

3.3 Color Pallettes

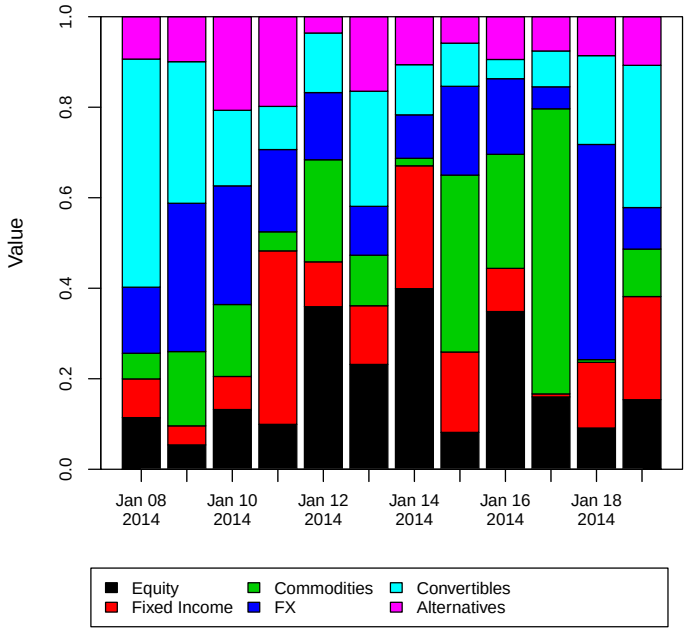
The color pallete defaulted to by `barplot.xts` is, in the eyes of many, somewhat garish; in the particular case of barplots, the eye is too strongly drawn to the brighter elements, particularly the green and purple, causing misinterpretation. **PerformanceAnalytics** provides four chosen color pallettes to mitigate this effect and we document them here. Their use is highly recommended:

```
> rainbow6equal <- c("#BF4D4D", "#BFBF4D", "#4DBF4D", "#4DBFBF",
+                    "#4D4DBF", "#BF4DBF")
> rainbow8equal <- c("#BF4D4D", "#BFA34D", "#86BF4D", "#4DBF69",
+                    "#4DBFBF", "#4D69BF", "#864DBF", "#BF4DA3")
> rainbow10equal <- c("#BF4D4D", "#BF914D", "#A8BF4D", "#63BF4D",
+                     "#4DBF7A", "#4DBFBF", "#4D7ABF", "#634DBF",
+                     "#A84DBF", "#BF4D91")
> rainbow12equal <- c("#BF4D4D", "#BF864D", "#BFBF4D", "#86BF4D",
+                     "#4DBF4D", "#4DBF86", "#4DBFBF", "#4D86BF",
+                     "#4D4DBF", "#864DBF", "#BF4DBF", "#BF4D86")
```

For more advanced pallete construction, see the `CRAN˜`packages **RColorBrewer** and **colorspace**.



(a) Unscaled Plot



(b) Scaled Plot

Figure 3: Scaled Barplots

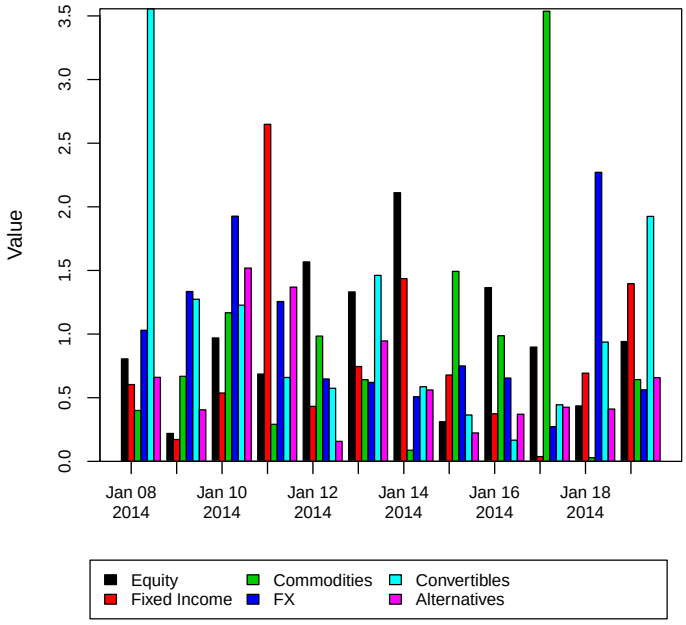


Figure 4: `barplot(x, stacked = FALSE)`

References

[1] Achim Zeileis and Gabor Grothendieck (2005):
zoo: S3 Infrastructure for Regular and Irregular Time Series.
Journal of Statistical Software, 14(6), 1-27.
http://www.jstatsoft.org/v14/i06/

[2] International Organization for Standardization (2004):
ISO 8601: Data elements and interchage formats — Information interchange — Representation of dates and time
http://www.iso.org

[3] R Development Core Team:
R: A Language and Environment for Statistical Computing,
R Foundation for Statistical Computing, Vienna, Austria.
ISBN 3-900051-07-0
http://www.R-project.org

[4] Jeffrey A. Ryan (2008): *quantmod: Quantitative Financial Modelling Framework.*
R package version 0.3-5.
http://www.quantmod.com
http://r-forge.r-project.org/projects/quantmod

[5] Jeffrey A. Ryan & Joshua M. Ulrich (2008):
xts: Extensible Time Series
R package version 0.8-7.
http://r-forge.r-project.org/projects/xts/

[6] Peter Carl and Brian G. Peterson (2012):
PerformanceAnalytics: Econometric tools for performance and risk analysis.,
R package version 1.0.4.5
http://r-forge.r-project.org/projects/returnanalytics/

[7] Cleveland, W.S. (1994):
The Elements of Graphing Data
Summit, NJ: Hobart Press.

[8] Tufte, Edward R. (2001):
The Visual Display of Quantitative Information, 2nd. ed.
Chesire, CN: The Graphics Press.
See also http://www.edwardtufte.com.