Plotting 'timeSeries' Objects

Diethelm Würtz and Tobias Setz ETH Zurich and Rmetrics Association Zurich

May 12, 2014

Contents

1	Int	roduction	4
2	Sta	ndard Time Series Plots	8
	2.1	Univariate Single Plots	9
	2.2	Multivariate Single Plots	11
	2.3	Multiple Plots	13
	2.4	Combining Single Plots	16
	2.5	Layout of Single Plots	17
	2.6	Bivariate Scatter Plots	19
3	Tin	ne Axis Layout	22
	3.1	Time Axis: "pretty" or "chic"?	23
	3.2	Time Axis - Selecting Minor Tick Marks	25
	3.3	Time Axis - One Column Multiple Plot Layout	26
	3.4	Time Axis - Two Column Multiple Plot Layout	27
	3.5	Tick and Format Layout: The <tailored> axis style</tailored>	28
4	An	notations	30
	4.1	Discarding all Annotations	31
	4.2	Adding Title and Labels	32
	4.3	Changing Axis Font Size	33
	4.4	Flipping Value Axes	34
5	Dec	corations	36
	5.1	Modifying Types	37
	5.2	Changing Colors by Names	38
	5.3	Changing Colors by Color Palettes	39
	5.4	Changing Line Styles	41
	5.5	Modifying Line Widths	42
	5.6	Changing Plot Symbols	43
	5.7	Modifying Plot Symbol Sizes	44
	5.8	Discarding Grid Lines	45
	5.9	Drawing a Box	46

6	The	Panel Function	48
	6.1	Adding a Horizental Zero Line	49
	6.2	Adding an Rug to Multiple Return Plots	50
	6.3	Adding an EMA to Multiple Index Plots	51
7	Con	clusions	54
8	App	pendix	56
	8.1	Margins: mar and oma	57
	8.2	Character Table	59
	8.3	Color Table	60
	8.4	Color Palettes I	61
	8.5	Color Palettes II	62
	8.6	Symbol Table	63
	8.7	Axis Style "pretty"	64
	8.8	Axis Style "chic"	65

1 Introduction

The Rmetrics timeDate and timeSeries packages are workhorses to deal with chronological objects. Since their inception 2009 under their original names fCalendar and fSeries they have been only slightly modified. With version R 3.1. we have essentially improved the plot function, but we also took care that the functionality is almost upward compatible.

In this vignette we show how to work with the recently updated S4 generic plot function plot. The function is written to display Rmetrics S4 timeSeries objects. The basic functionality of the plot function is to display single and multiple views on univariae and multivariate timeSeries objects. The function plot.ts from R's base environment, which displays basic ts time series objects, served as a model for our design of the generic S4 plot function for timeSeries objects. Similarly, plot.ts can be considered as the prototype for the S3 plot.zoo method. The xts plot function was build to display univariate xts time series objects which inherit from zoo's objects for ordered time series objects.

The generic S4 time series plotting function can dispay univariate and multivariate time series in single and multiple frames. The plots can be tailored with respect to several viewing components: colors (col), line types (lty), plot symbols (pch), line widths (lwd), symbol sizes (cex), axis layout (pretty, chic, tailored), minor tick mark appearence, font styles and font sizes, frame positioning (mar, oma), as well as tailored panel functions (panel).

General Plot Settings and Design Apects:

Plot Type: Univariate time series are displayed by default in plot.type="single" frames, multivariate time series are displayed by default in plot.type="multiple" frames. The default line style for a plot is type ="1" is drawn with "lines".

Time Axis Layout: For the time axis layout the function pretty determines in an automative way the at="pretty" positions of the ticks. The format="auto" is extracted from the time stamps of the time series object or can be overwritten by the user with a POSIX format string. Alternatively, one can select "chic" to generate time axis styles. We called this method "chic" to give reference to the underlying function axTicksByTime from the Chicago xts package which generates tick positions and axis labels. Furthermore, a "tailored" method can be applied which allows for fully arbitrary user defined positions and formatted labels. Minor ticks can be added in several fashions.

Annotations: The annotations of the plots are reduced to the y-label. These are taken by default from the column names of the time series object. This gives

the user the freedom to have full control about his views how the plot should be look like. Note, multivariate time series in single plots show the string "Values" as label on the y-axis. Main title, sub title, and the x-label on the time axis are not shown by default. We prefer and recommend to add these decorations calling the function title. This allows also much more flexibility compared to passing the arguments through the plot functions. All default annotations (including the y-label) can be suppressed setting the plot function argument to ann=FALSE. The argument axes=FALSE suppresses to draw both axes on the plot frame.

Decorations: There are several options to decorate the plot: These include colors (col), plotting symbols (pch), scaling factor of plotting characters and symbols (cex), line types (lty), and lindwidths (lwd). Note, all these parameters may be vectors of the same length as the number of time series, so that each series can be addressed to its own individual style, color, and size. A grid and the plot frame (box) can be added or suppresse specifying the arguments grid and frame.plot in the argument list of the plot function.

Panel Function: In the case of multiple plots the plot frames, are also called panels. By default in each panel the appropriate curve is drawn calling R's lines function panel=lines. This function can be replaced by a user defined function. This offers a wide range of new views on your time series. So for example yo can show zero or any other reference lines on the panels, or you can add rugs to (return) charts, or you can add for an example an EMA indicator (or any other kind of indicator) to curves shown in individual panels.

Example "timeSeries" Objects: To demonstrate the wide range of options to dispaly S4 timeSeries objects, we use the the daily index values from the Swiss Pension Fund Benchmark LPP2005. The time series is part of the timeSeries package. For this we have introduced some abbreviations:

> Sys.setlocale("LC_ALL", "C")

```
[1] "LC_CTYPE=C;LC_NUMERIC=C;LC_TIME=C;LC_COLLATE=C;LC_MONETARY=C;LC_MESSAGES=en_GB.UTF-8;
```

```
> require(timeSeries)
> require(xts)
> require(PerformanceAnalytics)
> require(fTrading)
> tS1 <- 100 * cumulated(LPP2005REC[, 1])  # SBI (univariate)
> tS2 <- 100 * cumulated(LPP2005REC[, 1:2])  # SBI & SPI (bivariate)
> tS3 <- 100 * cumulated(LPP2005REC[, 1:3])  # SBI, SPI, SWIIT (Swiss Market)
> tS6 <- 100 * cumulated(LPP2005REC[, 1:6])  # Swiss and Foreign Market Indexes</pre>
```

2 Standard Time Series Plots

The plot function from the timeSeries package allows for five different views on standard plot layouts. These include

- Univeriate single plots
- Multivariate single plots
- One column multiple plots
- Two column multiple plots
- Scatter plots

The only argument we have to set is the plot.type parameter to determine the layout of the plot. The default value is "multiple", and the alternative value is "single". The arguments can be abbreviated as "m" or "s", respectively.

Univariate Single Plots were designed to plot univariate timeSeries objects in one single graph frame. Nothing then the timeSeries object has to be specified, the plot.type is forced to "s".

Multivariate Single Plots will be used when a set of multivariate timeSeries objects should be drawn in one common data frame. For this argument the vlue plot.type="s" has to be specified.

One Column Multiple Plots display multivariate timeSeries objects, such that each series is plotted in its own frame. Up to four series, the frames are displayed in one column, for more series the frames are arranged in a two column display.

Two Column Multiple Plots handel the case of more than four timeSeries objects. Then the series are displayed in two colums. In total, the number of rows is not restricted.

2.1 Univariate Single Plots

The most simple time series plot shows an univariate curve in a single plot. The axis is designed from "pretty" positions calculated from R's base function pretty. The labels are printed in the ISO 8601 standard date/time format.

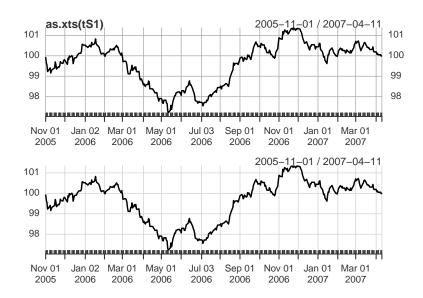
```
> par(mfrow=c(1, 1))
> plot(tS1)
```



Figure 1: The chart shows an univariate time series (here the Swiss Bond Index) in a single frame. For all plot options default values have been chosen. You can decorate the plot, making it more conveniant to your needs, e.g.: change the color (col), add a main title and x-label calling the function title, or remove the grid lines setting the argument grid=FALSE. You can also design the minor tick marks, setting instead of the value "auto" oe of the following spreads: "day", the default, "week", or "month".

Two other plot function implementations for xts time series objects can be found in the contributesd R packages xts and PerformanceAnalytics. Let us compare how they generate plot positions and time label formats.

```
> require(PerformanceAnalytics)
> par(mfrow=c(3, 1))
> xts::plot.xts(as.xts(tS1))
> PerformanceAnalytics::chart.TimeSeries(as.xts(tS1))
> plot(tS1)
```



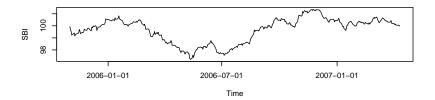


Figure 2: The group of the three charts shows an univariate time series in a single frame for the plot functions as implemented in the packages xts, PerformanceAnalytics, and timeSeries. For example in the case of daily time series records xts uses U.S. style labels whereas PerformanceAnalytics and timeSeries use ISO standard date labels YYYY-mm-dd. The plot decorations are those from default settings.

2.2 Multivariate Single Plots

Multivariate time series plots in a single panel are constructed by default in the way that the first curve is plotted calling the function plot and the remaining curves by calling the function lines.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="s")
```

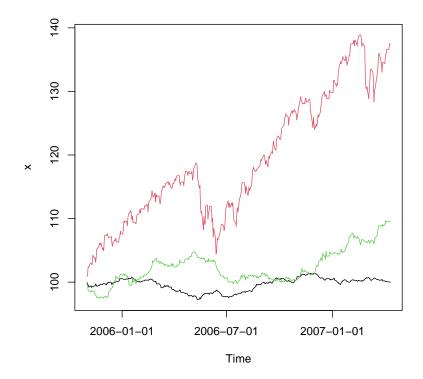


Figure 3: This chart shows a multivariate time series in a single frame. Note, we have to set the argument plot.type="s". Again, for all plot options the default settings have been used. You can decorate the plot, making it more conveniant to your needs, e.g.: change the color vector (col), add a main title and x-label calling the function title, or remove grid lines setting the argument grid=FALSE. Note, to change the color settings you can set the argument col=1:3 which would result in "black", "red", "green" for the three curves, or you can just set the colors by name, or selecting them from a color palette.

Now let us compare the plot function from the timeSeries package with the chart.TimeSeries plotting function from the PerformanceAnalytics function. (Note, the (xts) has no multivariate plot function implemented.)

```
> par(mfrow=c(2, 1))
> require(PerformanceAnalytics)
> PerformanceAnalytics::chart.TimeSeries(as.xts(tS3))
> plot(tS3, plot.type="s")
```



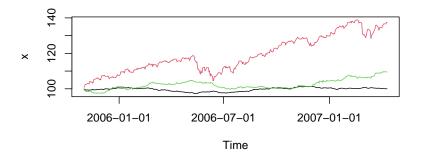


Figure 4: The two charts show a multivariate time series plotted in a single frame. We use for the plot the functions as implemented in the packages PerformanceAnalytics, and timeSeries.

2.3 Multiple Plots

Multiple plots enormously simplify the display of different curves in multiple panels. These are the ideal plots when it comes to the task to create a quick overview over several time series. Multiple plotting is exclusive to timeSeries objects, (xts) and PerformanceAnalytics offer no multiple plotting tool.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="m")
```

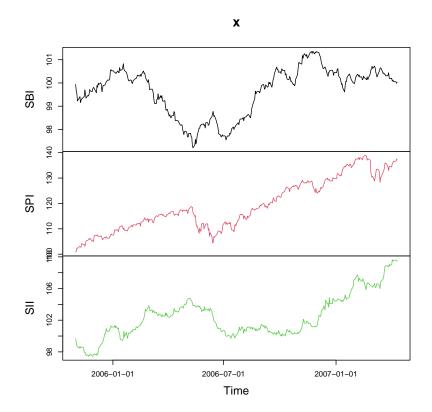


Figure 5: As long as we plot less than 4 time series in a multivariate frame, we get a one column layout. Annotations show by default only the y-labels which are taken from the colmun names of the time series to be drawn. Feel free to add main title, sub title, and x-label calling the function title

For more than four curves the frames of the plot design are arranged in two columns.

```
> par(mfrow=c(1, 1))
> plot(tS6, plot.type="m")
```

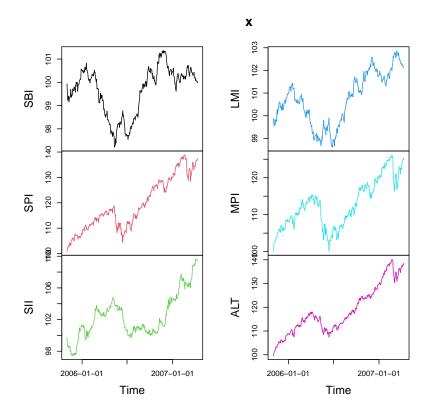


Figure 6: The graph shows the layout how it is created for six curves. There are two columns with three panels to the left and also three panels to the right. Note, it is easily possible to adapt the margin sizes and the gap between the two columns of plots calling the function mar and oma.

If you like a design with a small gap between the panel rows, you can modify the mar parameter to introduce a small gap, here with a width of 0.3. Feel free to modify it.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="m", mar=c(gap=0.3, 5.1, gap=0.3, 2.1))
```

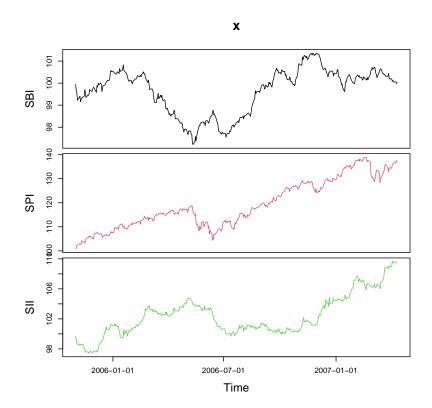


Figure 7: One can use in a multiple plot the mar parameter setting to create a small gap between the rows of the individual charts. This lets a plot look more elegant.

2.4 Combining Single Plots

You can also create your own multiple panel plots. Just combine single panels in an array of rows and columns using the parameter settings for mfrow, mfcol, and mar.

```
> par(mfrow=c(2, 1))
> par(mar = c(bottom=1.5, 5.1, top=4, 2.1))
> plot(tS2[, 1])
> par(mar = c(bottom=4, 5.1, top=1.5, 2.1))
> plot(tS2[, 2])
```

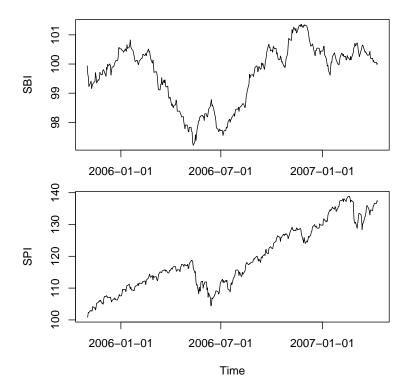


Figure 8: One can use in single plots the mfrow and mar parameter settings to place single plots either row by row or column by column. Here, mfrow and mfcol to the job. In this case a vector of the form c(nr, nc) draws subsequent figures in an nr-by-nc array on the device by columns (mfcol) or rows (mfrow), respectively.

2.5 Layout of Single Plots

There is another option in R to create panel layouts, not necessarilly in an rectangular array. Have a look to the help page of the function layout, her comes a simple example.

```
> nf <- layout(mat=matrix(c(1, 1, 2, 3), byrow = TRUE, nrow=2))
> par(mar = c(bottom=2, 5.1, top=3, 2.1))
> plot(tS3[, 1])
> par(mar = c(bottom=3, 5.1, top=2, 1.1))
> plot(tS3[, 2])
> par(mar = c(bottom=3, 4.1, top=2, 2.1))
> plot(tS3[, 3])
```

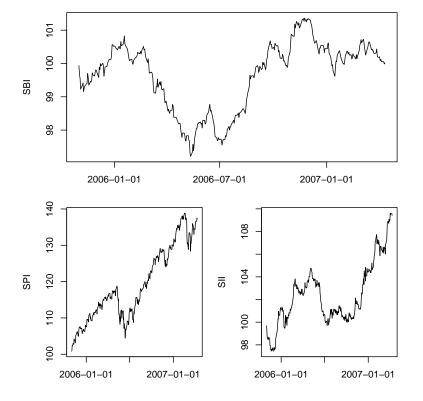


Figure 9: With the function layout you can devide the plot device in rows and columns expressed in matrix form defined by the argument mat.

In addition widths and heights of the layout can be different from row to row, and/or from column to column. The sizes are expressed by the arguments widths and heights of the function layout.

```
> nf <- layout(mat=matrix(c(1, 1, 2, 3), byrow=TRUE, nrow=2), heights=c(2.5,1))
> par(mar = c(bottom=2, 5.1, top=3, 2.1))
> plot(tS3[, 1])
> par(mar = c(bottom=3, 5.1, top=1.5, 1.1))
> plot(tS3[, 2])
> par(mar = c(bottom=3, 4.1, top=1.5, 2.1))
> plot(tS3[, 3])
```

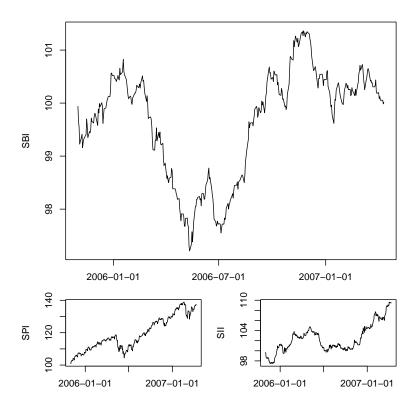


Figure 10: With the function layout one can also define the widths and heights of the columns and rows.

2.6 Bivariate Scatter Plots

For historical reasons, like in the function plot.ts, there is also the option to create an scatter plot from two univariaye time series. Since this is not a "true" time series plot, we will not go in further detail for this display.

```
> par(mfrow=c(1,1))
> plot(tS2[, 1], tS2[, 2])
```

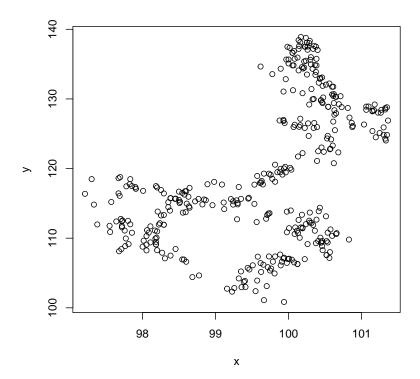


Figure 11: If (x) and (y) are univariate time series, then the plot function will display a scatter plot.

3 Time Axis Layout

The function plot comes with three options to design the time axis layout: "pretty", "chic", and <tailored> (note this not a string argument. <tailored> should just abbreviate that we have to input character strings of fully arbitray at positions. For the first two options the style of the axis annotation is generated in a fully automated way, whereas in the tailored case each tick on the axis to be user defined.

The "pretty" time axis layout is the default setting for the argument at. Internally the function pretty is used to compute a sequence of about n+1 equally spaced round values which cover the range of the values in the time stamps time(x) of the series x. The values are chosen so that they are 1, 2 or 5 times a power of 10.

The "chic" time axis layout is the alternative setting for the argument at. Internally the function axTicksByTime from the package xts is used to compute the sequence of axis positions and the format labels.

The *<tailored>* time axis layout leaves it to the user to specify by himself the positions (at), the time label formatting (format), and the minor tick marks (minor.ticks).

3.1 Time Axis: "pretty" or "chic"?

Our plotting function comes with two axis-styles. The first is called "pretty", which is the default style, and calculates positions from R's base function pretty. The other is called "chic" to remember its origin, arising from the "Chicago" xts package.

```
> par(mfcol = c(2, 1))
> plot(tS1, at = "pretty")
> plot(tS1, at = "chic")
```





Figure 12: The graph shows the two flavours from the at="pretty" and the "chic" axis designs. The first style uses the function pretty from R's base environment to compute the positions for the major ticks. The second style uses the function axTicksByTime from the xts package to compute x-axis tick mark locations by time. In the upper graph the minor ticks are calendar days, whereas in the lower graph weekdays are drawn (therefore the small gaps between the minor ticks become visible). Note, the time series is in both cases an object of class timeSeries.

Now let us plot a multivariate 3-column time series in a single panel. Again we compare the outcome of the "pretty" and the "chic" axis style.

```
> par(mfcol=c(2, 1))
> plot(tS3, plot.type="s", at="pretty")
> plot(tS3, plot.type="s", at="chic")
```



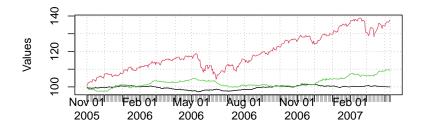
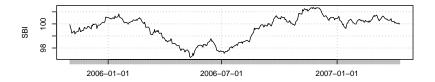


Figure 13: The only difference of this graph compared to the previous is the fact that we consider here a multivariate time series. Three curves are shown in a common plot. Note, when using the "chic" style, then the vertical gridlines are narrower compared to the "pretty" style.

3.2 Time Axis - Selecting Minor Tick Marks

The "pretty" style allows to draw the minor tick marks on different time scales. These are: "day", "week", and "month".

```
> par(mfrow=c(3, 1))
> plot(tS1, minor.ticks="day", at="pretty")
> plot(tS1, minor.ticks="week", at="pretty")
> plot(tS1, minor.ticks="month", at="pretty")
```





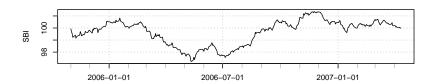


Figure 14: It is worth to note that a good selection of minor tick marks makes a plot much better readable.

3.3 Time Axis - One Column Multiple Plot Layout

In the multiple plot layout the axis are drawn along the same principles as they are drawn in the case of the single plot layout.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="m", at="pretty")
```

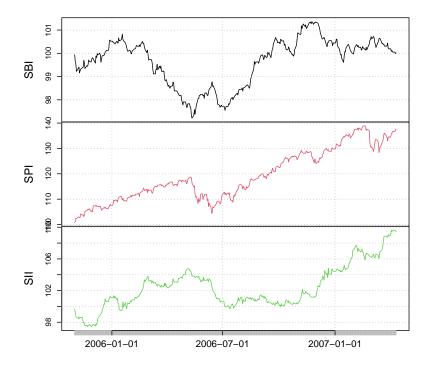


Figure 15: This graph shows a one column multiple plot layout. The one column layout is generated for up to four time series. When the multivariate time series has more then four time series then a two column layout is displayed. It is up to you which axis style you prefer, at="pretty" or at="chic".

3.4 Time Axis - Two Column Multiple Plot Layout

Concerning the style of the axis, there is now difference between the one and two column plot designs.

```
> par(mfrow=c(1, 1))
> plot(tS6, plot.type="m", at="chic")
```

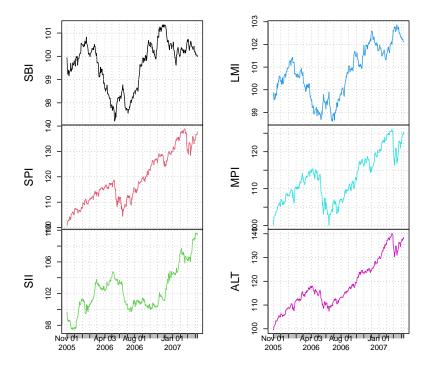
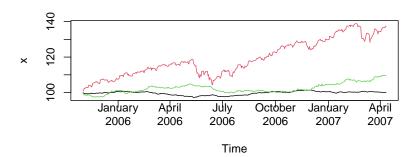


Figure 16: When we have more then four time series, then the display will be generated in two columns. Note, it is possible to modify the width of the gap between the two columns.

3.5 Tick and Format Layout: The <tailored> axis style

The third alternative to style the axis offers the users to define format positions according to his preferences. Here comes an example:

```
> par(mfrow=c(2, 1))
> at <- paste0("200", c("6-01", "6-04", "6-07", "6-10", "7-01", "7-04"), "-01")
> plot(tS3, plot.type="s", format="%B\n%Y", at=at)
> plot(tS3, plot.type="s", format="%b/%y", at=at)
```



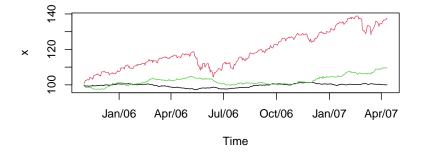


Figure 17: This graph shows plots with user tailored positions and formatted axis labels.

4 Annotations

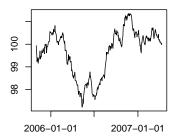
Plot annotations are elelents which can be added to plots or completely discarded. To discard all annotations you have to set ann=FALSE in the argument list of the timSeries plot function. To display annotation you can call the function title. This allows to add the main title, the sub title, and the x- and y-labels to a plot. Together with the appropriate character strings, you can also specify the placement of these annotations by the arguments line and outer.

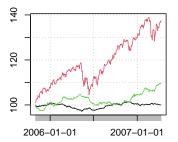
There are additional functions to add annotations to a plot. These are text and mtext.

4.1 Discarding all Annotations

In a default plot we display only the value-label(s) which are taken from the units or column names of the time time series object. Title, sub title, and time-label are not shown. To discard the appearance of all annotations on a plot you have to set the plot argument ann=FALSE.

```
> par(mfrow=c(2, 2))
> plot(tS1, ann=FALSE)
> plot(tS3, plot.type="s", ann=FALSE, at="pretty")
> plot(tS6, plot.type="s", ann=FALSE, at="pretty")
```





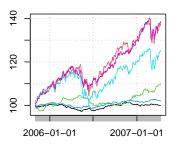


Figure 18: This graph shows a plot where all annotations have been discarded. Now feel free to add your own annotations.

4.2 Adding Title and Labels

To add a main title, a sub, title, and x- and y-labels you can call the function title.

```
> par(mfrow=c(2, 2))
> plot(tS1); title(main = "Index")
> plot(tS3, plot.type="s"); title(main = "Index")
> plot(tS3, plot.type="s"); title(main = "Index", xlab = "Date")
> plot(tS6, plot.type="s"); title(main = "Index", xlab = "Date")
```

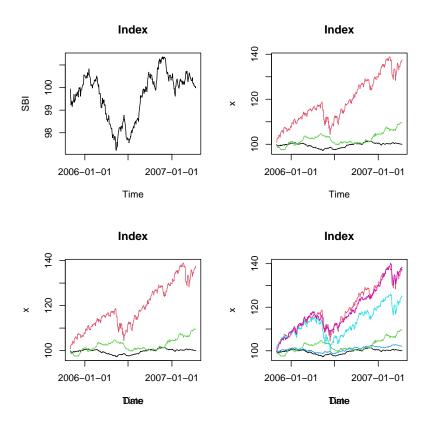
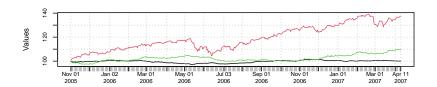


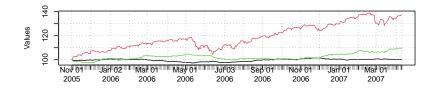
Figure 19: This graph displays in a two by two array four single plots. We have added title and x-lable annotations.

4.3 Changing Axis Font Size

Sometimes the axis font size may be considered as too small or too large. Then you can use the plot argument cex.axis to upsize or downsize the font.

```
> par(mfrow=c(3, 1))
> plot(tS3, at="chic", plot.type="s", cex.axis=0.75)
> plot(tS3, at="chic", plot.type="s", cex.axis=1.00)
> plot(tS3, at="chic", plot.type="s", cex.axis=1.25)
```





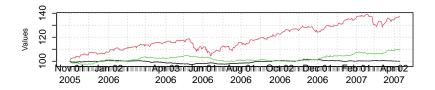


Figure 20: This is an example how to change the size of the axis labels relatively to its default value. The upper graph shows a font size decreased by 20%, the lower graph a font size increased by 25%. You can proceed in the same way when using the "pretty" axis style.

4.4 Flipping Value Axes

Flipping every second axis label in a multiple plot from left to rigth might be meaningful in the case when axis labels overwrite themselves.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="m", yax.flip = TRUE)
```

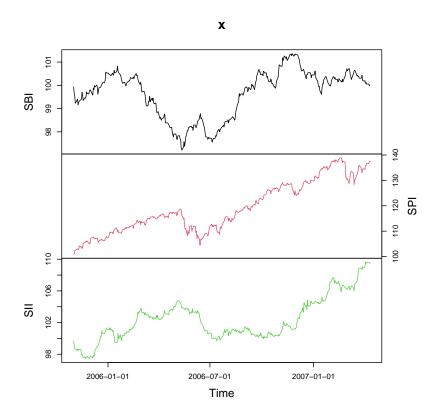


Figure 21: The graph shows an one column multiple plot, where the axis of the middle panel is flipped from the left to the right. Note, the same procedure can also be applied two two column multiple plots.

5 Decorations

There exist several options to decorate plots in different ways. Plot types (lines, points, horizontal bars, etc.) can be modified, colors can be changed, lines can be modified by style and seize, points can be selected by symbol and size.

In the following we will give some examples

- Modifying Types
- Changing Colors by Names
- Changing Colors by Color Palettes
- Changing Line Styles
- $\bullet\,$ Modifying Line Widths
- Changing Plot Symbols
- Modifying Plot Symbol Sizes
- Discarding Grid Lines
- Drawing a Box

to show a few of the many types of cdecorations. Play around to achieve your perfect layout.

5.1 Modifying Types

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="m", type=c("l", "p", "h"), at="pretty")
```

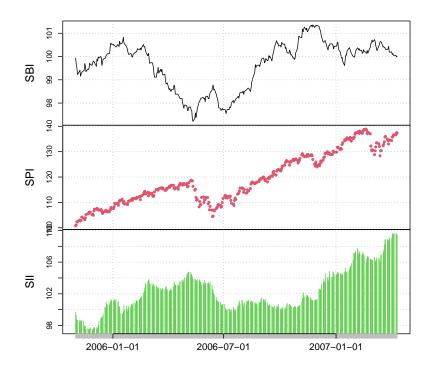


Figure 22: If we like to consider in a multiple plot for each panel its own plot style then we can set the parameter type.

5.2 Changing Colors by Names

Colors can be changed in several ways. Just by their numbers, e.g. 1 (black), 2 (red), 3 (green) etc., or by name, e.g. "black", "red", "green", etc. or by using well designed color palettes.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="m", col=c("blue", "orange", "darkgreen"))
```

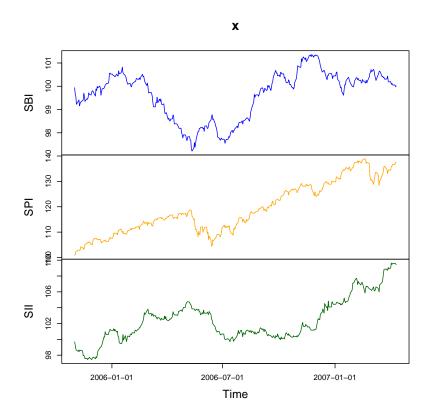


Figure 23: This graph shows how to assign colors by name in the case of a multiple plot. You can do it in the same way setting plot.type="s" if you like to display all three curves in a common single plot.

5.3 Changing Colors by Color Palettes

When the number of curves increases, then it can become quite difficult to find a set of nice colors. In such cases it is convenient to select the colors from color palettes.

```
> par(mfrow=c(1, 1))
> plot(tS6, plot.type="s", col=heat.colors(n=6, alpha = 1),
+ at="chic", format = "%B\n%Y")
```

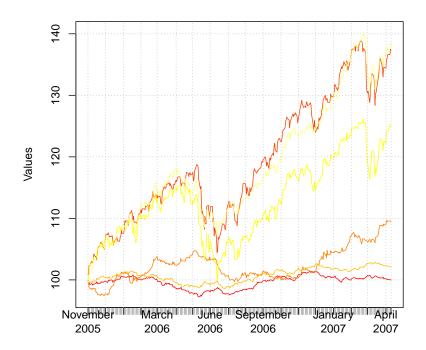


Figure 24: This graph shows an example of six curves diplayed in a sequence of reds. For this we called the function seqPalette.

FUNCTION: COLOUR PALETTE

rainbowPaletteContiguous rainbow colour palette heatPalette Contiguous heat colour palette terrainPalette Contiguous terrain colour palette Contiguous topo colour palette topoPalette Contiguous cm colour palette cmPalette R's gamma-corrected gray palette greyPalette timPalette Tim's MATLAB-like colour palette rampPalette Colour ramp palettes seqPalette Sequential colour brewer palettes ${\tt divPalette}$ Diverging colour brewer palettes Qualified colour brewer palettes qualiPalette ${\tt focusPalette}$ Red, green and blue focus palettes monoPalette Red, green and blue mono palettes

5.4 Changing Line Styles

In multiple plot to each curve an own line style 1ty can be assigned: 0 "blank", 1 "solid", 2 "dashed", 3 "dotted", 4 "dotdash", 5 "longdash", or 6 "twodash".

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="m", col=1, lty=1:3, at="chic")
```

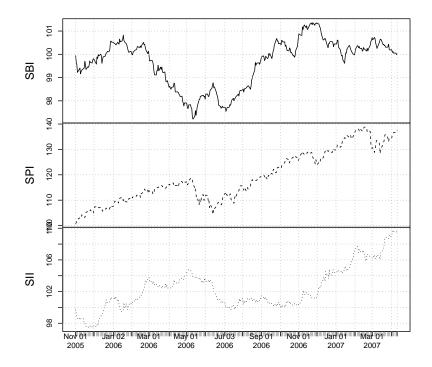


Figure 25: When we like to print plots in black and white, then its makes much sense to use different line types so that we can distinguish the curves one from each other.

5.5 Modifying Line Widths

Not only the line type, but also the line width can be modified for each curve in an individual kind.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="m", col=1, lwd=3:1, at="chic")
```

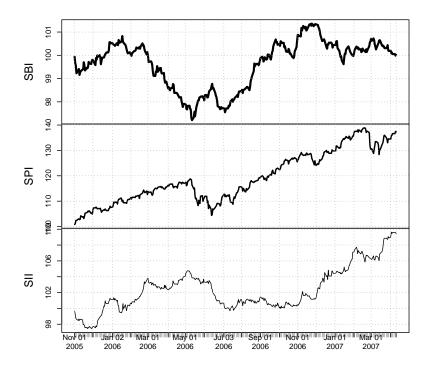


Figure 26: The graph shows three line widths, the upper's curve width is thick, the middle's curve width is medium, and the lowest's curve width is the thinnest one.

5.6 Changing Plot Symbols

To use different plot symbols we can assign them by the parameter pch. Don't forget also to set type="p".

5.7 Modifying Plot Symbol Sizes

The argument cex.pch allows to increase or decrease plot symbol sizes with respect to the current plot symbol size.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="s", type="p",
+ col=1:3, pch=21:23, cex.pch=c(0.2, 0.2, 0.2), at="pretty")
```

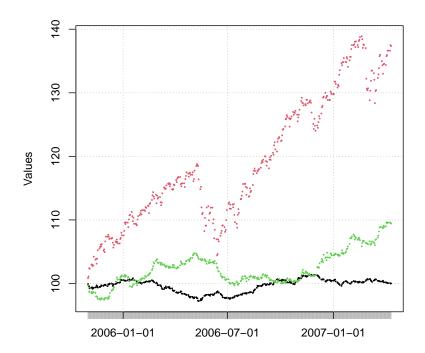


Figure 27: This plot show how to change the size of plot symbols in a single plot setting the argument cex.pch. Note, for each curve its own size can be set. The same approach can be used also for multiple plots.

5.8 Discarding Grid Lines

By default grid lines are displayed. To discard the grid lines from the plot set the arguments $\tt grid=FALSE$.

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="s", grid=FALSE)
```

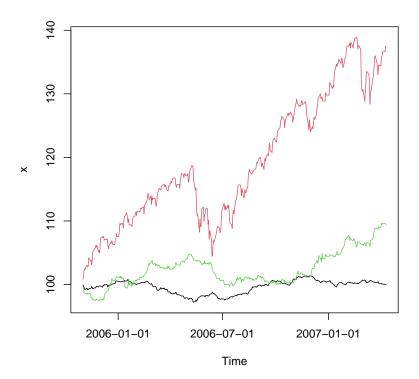


Figure 28: By default a grid is drawn on top of the plot. You can remove it by setting the argument $\tt grid=FALSE$

5.9 Drawing a Box

```
> par(mfrow=c(1, 1))
> plot(tS3, plot.type="s", frame.plot=FALSE, grid=FALSE)
> box()
> box(bty = "7", col = "white") # boxL
> grid(NA, NULL, col = "darkgrey") # hgrid
```

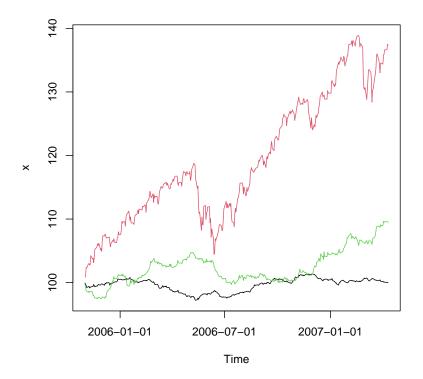


Figure 29: By default the plots are displayed as frame plots. This means that the graphs are surrounded by a box. This box can be discarded setting the plot argument frame.plot=FALSE.

6 The Panel Function

Multiple plots are very powerful plotting designs. Each panel in a graph can individually tailored by the user. By default each curve in a panel is generated by the function lines. You can define your own panel function(s) by setting the plot argument panel to a user dfined functions.

In the following we will show three examples.

6.1 Adding a Horizental Zero Line

In this example we show how to write a panel function which allows to add a horizontal zero line to each plot panel.

```
> par(mfrow=c(1, 1))
> lines2 <- function(X, Y, type, xlab, ylab, col, pch, lty, lwd, cex) {
+ lines(x=X, y=Y, col=col)
+ abline(h=0, col = "brown", lwd=2)}
> plot(returns(tS3), plot.type="m", col = .colorwheelPalette(3),
+ panel=lines2, at="pretty")
```

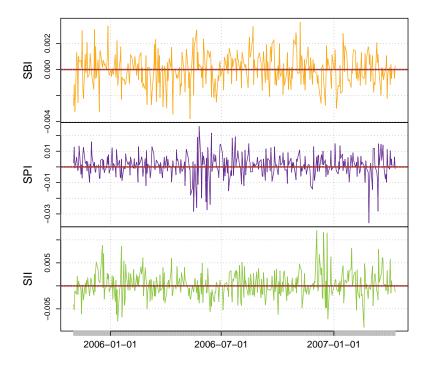


Figure 30: This multiple plot shows panels with curves having a horizontal zero reference line.

6.2 Adding an Rug to Multiple Return Plots

This example shows how to add in each panel rugs to the righ Y-axis.

```
> par(mfrow=c(1, 1))
> lines2 <- function(X, Y, type, xlab, ylab, col, pch, lty, lwd, cex) {
+ lines(x=X, y=Y, type="h", col=col)
+ rug(Y, side=4, col="steelblue") }
> plot(returns(tS6), plot.type="m", col = .colorwheelPalette(6),
+ panel=lines2, at="pretty")
```

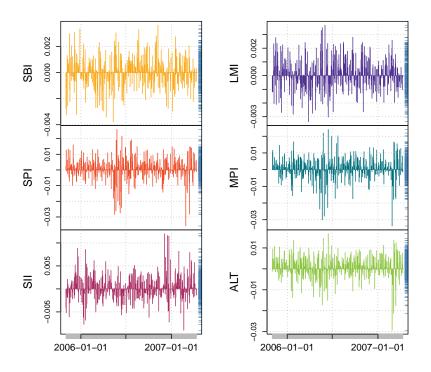


Figure 31: This multiple plot shows panels with rugs on the right Y-axis.

6.3 Adding an EMA to Multiple Index Plots

This example shows how to add an EMA indicator to each plot panel. The emaTA() function is provided by the fTrading package.

```
> par(mfrow=c(1, 1))
> lines2 <- function(X, Y, type, xlab, ylab, col, pch, lty, lwd, cex) {
+ lines(x=X, y=Y, type="l", col=col)
+ lines(x=X, y=emaTA(Y), col="black") }
> plot(tS3, plot.type="m", col = .colorwheelPalette(3), panel=lines2,
+ grid=TRUE, at="pretty")
```

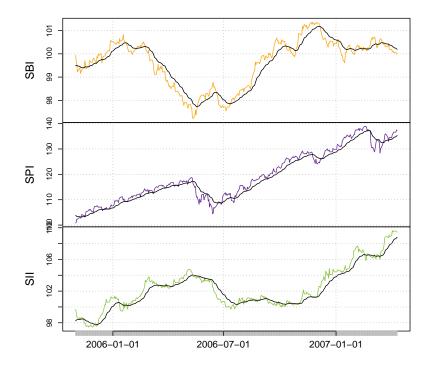


Figure 32: A multiple graph with EMA indicators in each panel.

7 Conclusions

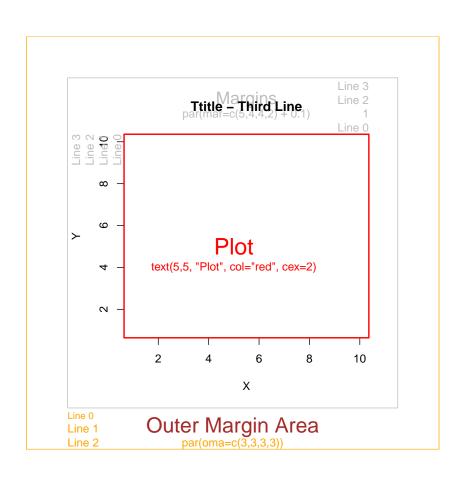
The plot function in the timeSeries package is a very powerful tool to create plots from time series objects. This includes to display univariate and multivariate time series in single and multiple panels, to select from two styles for the time-axis or even to tailor positions and formats according to his own needs, and to modify annotations and decorations of plots.

8 Appendix

In the appendix we have summarized graphs and tables which are useful tools to create plots. We have reprinted the default color table from R, we have summarized several color palettes as available in the fBasics package and other contributed R packages, and two tables with font characters and plot symbols.

8.1 Margins: mar and oma

```
> # Plot:
> # - oma stands for 'Outer Margin Area'
> # - mar represents the 'figure Margins'
> # - The default size is c(5,4,4,2) + 0.1
> # - The axes tick marks will go in the first lines
> par(mfrow=c(1, 1))
> par(oma=c(3,3,3,3)) # all sides have 3 lines of space
> par(mar=c(5,4,4,2) + 0.1)
> plot(x=1:10, y=1:10, type="n", xlab="X", ylab="Y")
> # Add Text tot the Plot Part - red
> text(5,5, "Plot", col="red", cex=2)
> text(5,4, "text(5,5, \"Plot\", col=\"red\", cex=2)", col="red", cex=1)
> box("plot", col="red", lwd=2)
> # Add text to thebThe Figure Part - grey
> mtext("Margins", side=3, line=2, cex=1.5, col="grey")
> mtext("par(mar=c(5,4,4,2) + 0.1)", side=3, line=1, cex=1, col="grey")
> mtext("Line 0", side=3, line=0, adj=1.0, cex=1, col="grey")
> mtext("
             1", side=3, line=1, adj=1.0, cex=1, col="grey")
> mtext("Line 2", side=3, line=2, adj=1.0, cex=1, col="grey")
> mtext("Line 3", side=3, line=3, adj=1.0, cex=1, col="grey")
> mtext("Line 0", side=2, line=0, adj=1.0, cex=1, col="grey")
> mtext("Line 1", side=2, line=1, adj=1.0, cex=1, col="grey")
> mtext("Line 2", side=2, line=2, adj=1.0, cex=1, col="grey")
> mtext("Line 3", side=2, line=3, adj=1.0, cex=1, col="grey")
> box("figure", col="grey")
> # The title will fit in the third line on the top of the graph.
> title("Ttitle - Third Line")
> # Note 'outer=TRUE' moves us from the figure to the outer margins.
> mtext("Outer Margin Area", side=1, line=1, cex=1.8, col="brown", outer=TRUE)
> mtext("par(oma=c(3,3,3,3))", side=1, line=2, cex=1, col="orange", outer=TRUE)
> mtext("Line 0", side=1, line=0, adj=0.0, cex=0.8, col="orange", outer=TRUE)
> mtext("Line 1", side=1, line=1, adj=0.0, cex=1, col="orange", outer=TRUE)
> mtext("Line 2", side=1, line=2, adj=0.0, cex=1, col="orange", outer=TRUE)
> box("outer", col="orange")
```



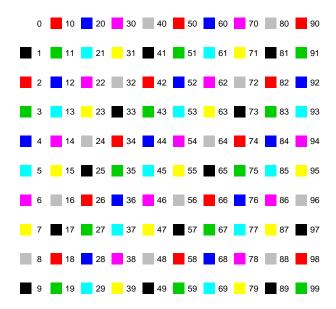
8.2 Character Table

Table of Characters

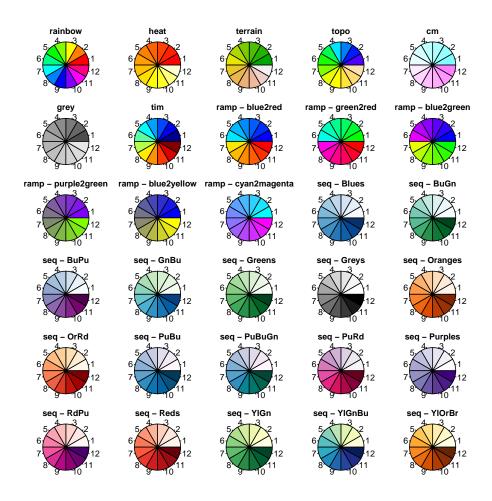
	0	1	2	3	4	5	6	7
4		!	A	#	3	%	&	Э
5	()	*	+	,	_		/
6	0	1	2	3	4	5	6	7
7	8	9	:	;	<	=	>	?
10	≅	Α	В	X	Δ	E	Φ	Γ
11	H	I	θ	K	Λ	M	N	О
12	П	Θ	P	Σ	T	Y	ς	Ω
13	Ξ	Ψ	Z	[]	1	_
14	_	α	β	χ	δ	ε	ф	γ
15	η	ι	φ	κ	λ	μ	ν	0
16	π	θ	ρ ζ	σ	τ	υ	σ	ω
17	ξ	Ψ	ζ	{	- 1	}	~	
20								
21								
22								
23								
24	€	Υ	,	≤	/	∞	f	*
25	•	•	^	\leftrightarrow	\leftarrow	↑	\rightarrow	1
26	۰	±	"	≥	×	∞	9	•
27	÷	≠	=	≈	•••		_	4
30	×	3	R	P	8	⊕	Ø	\cap
31	U	\supset	⊇	⊄	C	\subseteq	€	∉
32	_	∇	®	©	TM	П	√	
33	_	^	V	\Leftrightarrow	←	1	\Rightarrow	↓
34	◊	<	®	©	TM	Σ {	(
35	(Γ	- 1	L	ſ	{	l	
36		>	ſ	ſ		J	ĺ	- 1
37)	1		J]	}	Ĵ	

8.3 Color Table

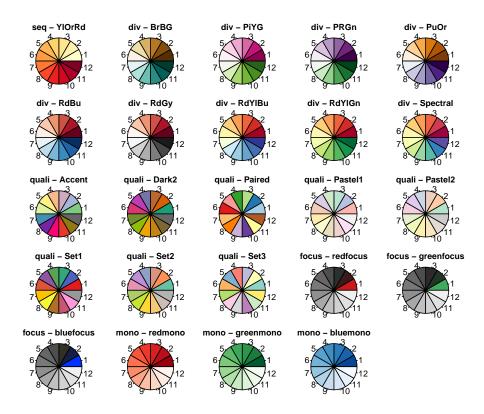
Table of Color Codes



8.4 Color Palettes I



8.5 Color Palettes II



8.6 Symbol Table

Table of Plot Characters

```
▼ 25 2 50 K 75 d 100 } 125
26 3 51 L 76 e 101 ~ 126
0
                                    150
                                          175
                                               200
                                                     225
                                                           250
0
                                               201
                                                     226
                                                           251
                                    151
                                          176
Δ 2
        27 4 52 M 77 f 102
                              127
                                    152
                                          177
                                               202
                                                     227
                                                           252
+
        28 5 53 N 78 9 103
                              128
                                    153
                                          178
                                               203
                                                     228
                                                           253
        29 6 54 O 79 h 104
                              129
                                    154
                                          179
                                               204
                                                     229
                                                           254
        30 7 55 P 80
                      i 105
                              130
                                    155
                                          180
                                               205
                                                     230
                                                           255
        31 8 56 Q 81
                                    156
                                          181
                                                206
                                                     231
        32 9 57 R 82 k 107
                              132
                                    157
                                          182
                                               207
                                                     232
  8 ! 33 : 58 S 83 | 108
                                    158
                                          183
                                               208
                                                     233
4 9
        34 ; 59 T 84 m 109
                              134
                                    159
                                          184
                                               209
                                                     234
● 10 # 35 < 60 U 85 n 110
                                    160
                                          185
                                               210
                                                     235
☆ 11 $ 36 = 61 ∨ 86 • 111
                              136
                                    161
                                          186
                                               211
                                                     236
⊞ 12 % 37 > 62 W 87 P 112
                                    162
                                          187
                                               212
                                                     237
🛚 13 & 38 ? 63 X 88 9 113
                              138
                                    163
                                          188
                                               213
                                                     238
      · 39 @ 64 Y 89 r 114
                              139
                                    164
                                          189
                                               214
                                                     239
15
     ( 40 A 65 Z 90 s 115
                              140
                                    165
                                          190
                                               215
                                                     240
• 16 ) 41 B 66 [ 91 t 116
                              141
                                    166
                                          191
                                               216
                                                     241
17
      * 42 C 67
                 \ 92 u 117
                              142
                                    167
                                          192
                                               217
                                                     242
• 18 + 43 D 68 ] 93 v 118
                              143
                                    168
                                          193
                                               218
                                                     243
• 19
     , 44 E 69 ^ 94 w 119
                              144
                                    169
                                          194
                                               219
                                                     244
• 20 - 45 F 70 - 95 x 120
                              145
                                    170
                                          195
                                               220
                                                     245
o 21 · 46 G 71 · 96 y 121
                              146
                                    171
                                          196
                                               221
                                                     246
□ 22 / 47 H 72 a 97 z 122
                              147
                                    172
                                          197
                                               222
                                                     247
♦ 23 0 48 I 73 b 98 { 123
                              148
                                          198
                                               223
                                                     248
                                    173
△ 24 1 49 J 74 C 99 | 124
                              149
                                    174
                                          199
                                                     249
```

8.7 Axis Style "pretty"

```
> FORMAT <- tS10format

> FORMAT

[1] "%Y-%m-%d"

> POSITIONS <- pretty(tS1)
> POSITIONS

GMT
[1] [2005-07-01] [2006-01-01] [2006-07-01] [2007-01-01] [2007-07-01]

> LABELS <- pretty(tS1)
> LABELS

GMT
[1] [2005-07-01] [2006-01-01] [2006-07-01] [2007-01-01] [2007-07-01]
```

8.8 Axis Style "chic"

```
> axTicksByTime <-
+ function (x, ticks.on = "auto", k = 1, labels = TRUE, format.labels = TRUE,
      ends = TRUE, gt = 2, 1t = 30)
+ {
      if (timeBased(x)) x \leftarrow xts(rep(1, length(x)), x)
      tick.opts <- c("years", "months", "weeks", "days", "hours", "minutes", "seconds")
+
      tick.k.opts <- c(10, 5, 2, 1, 6, 1, 1, 1, 4, 2, 1, 30, 15, 1, 1)
+
      if (ticks.on %in% tick.opts) {
          cl <- ticks.on[1]</pre>
          ck <- k
+
      } else {
          tick.opts \leftarrow paste(rep(tick.opts, c(4, 2, 1, 1, 3, 3, 1)), tick.k.opts)
+
+
          is <- structure(rep(0, length(tick.opts)), .Names = tick.opts)</pre>
+
          for (i in 1:length(tick.opts))
+
          {
               y <- strsplit(tick.opts[i], " ")[[1]]</pre>
               ep <- endpoints(x, y[1], as.numeric(y[2]))</pre>
               is[i] <- length(ep) - 1</pre>
               if (is[i] > lt) break
          }
+
          nms <- rev(names(is)[which(is > gt & is < lt)])[1]
          cl <- strsplit(nms, " ")[[1]][1]</pre>
+
+
          ck <- as.numeric(strsplit(nms, " ")[[1]][2])</pre>
      if (is.null(cl)) ep <- NULL else ep <- endpoints(x, cl, ck)</pre>
      if (ends) ep \leftarrow ep + c(rep(1, length(ep) - 1), 0)
+
      if (labels)
+
+
      {
+
          if (is.logical(format.labels) || is.character(format.labels))
+
          {
               unix <- ifelse(.Platform$OS.type == "unix", TRUE, FALSE)</pre>
               time.scale <- periodicity(x)$scale</pre>
               fmt <- ifelse(unix, "%n%b%n%Y", "%b %Y")</pre>
               if (time.scale == "weekly" | time.scale == "daily")
                   fmt <- ifelse(unix, "%b %d%n%Y", "%b %d %Y")
               if (time.scale == "minute" | time.scale == "hourly")
                   fmt <- ifelse(unix, "%b %d%n%H:%M", "%b %d %H:%M")</pre>
               if (time.scale == "seconds")
                   fmt <- ifelse(unix, "%b %d%n%H:%M:%S", "%b %d %H:%M:%S")</pre>
               if (is.character(format.labels))
+
                   fmt <- format.labels</pre>
               names(ep) <- format(index(x)[ep], fmt)</pre>
              names(ep) <- as.character(index(x)[ep])</pre>
          }
        ер
```

```
> ticks <- axTicksByTime(as.xts(tS1))</pre>
> ticks
Nov 01\n2005 Dec 01\n2005 Jan 02\n2006 Feb 01\n2006 Mar 01\n2006 Apr 03\n2006
        1 23
                                45 67
                                                        87
May 01\n2006 Jun 01\n2006 Jul 03\n2006 Aug 01\n2006 Sep 01\n2006 Oct 02\n2006
                    153
                                175
                                            196
                                                        219
Nov 01\n2006 Dec 01\n2006 Jan 01\n2007 Feb 01\n2007 Mar 01\n2007 Apr 02\n2007
                                305
                                            328
                                                        348
Apr 11\n2007
        377
```

About the Authors

Diethelm Würtz is professor at the Institute for Theoretical Physics, ITP, and for the Curriculum Computational Science and Engineering, CSE, at the Swiss Federal Institute of Technology in Zurich. He teaches Econophysics at ITP and supervises seminars in Financial Engineering. Diethelm is senior partner of Finance Online, an ETH spin-off company in Zurich, and co-founder of the Rmetrics Association in Zurich.

Tobias Setz has a Bachelor and Master in Computational Science from ETH in Zurich and has contributed with his Thesis projects on wavelet analytics and Bayesian change point analytics to this handbook. He is now a PhD student in the Econophysics group at ETH Zurich at the Institute for Theoretical Physics.

About Rmetrics

Rmetrics Open Source Project

With hundreds of functions built on modern methods, the Rmetrics open source software combines exploratory data analysis, statistical modelling and rapid model prototyping. The R/Rmetrics packages are embedded in R, building an environment which creates a first class system for applications in teaching statistics and finance. Rmetrics covers Time Series Econometrics, Hypothesis Testing, GARCH Modelling and Volatility Forecasting, Extreme Value Theory and Copulae, Pricing of Derivatives, Portfolio Analysis, Design and Optimization, and much more.

The Rmetrics Association

is a non-profit taking association working in the public interest. The Rmetrics Association provides support for innovations in financial computing. We believe that the Rmetrics Open Source software has become a valuable educational tool and that it is worth ensuring its continued development and the development of future innovations in software for statistical and computational research in finance. Rmetrics provides a reference point for individuals and institutions that want to support or interact with the Rmetrics development community. Rmetrics encourages students to participate in Rmetrics' activities in the context of Student Internships.

Rmetrics Software Evalution

If you like to get an impression of the size and quality of the Open Source Rmetrics Program have a look on the Ohloh Rmetrics Software Evaluation. The Evalutions gives an overview about the Software Development (Code Analysis, Estimated Cost), the people behind it, and its community.

$Contributions \ to \ Rmetrics$

are coming from several educuational institutions world wide. These include the Rmetrics web site and documentation project supported by ITP/CSE ETH Zurich, the organization of Summerschools and Workshops supported by ITP/CSE ETH Zurich, the R-sig-Finance Help and Mailing List, supported by SfS ETH Zurich, the R-forge development server, supported by University of Economics in Vienna, CRAN Test and Distribution Server for R software, supported by University of Economics Vienna, the Debian Linux integration supported by the Debian Association. Many thanks to all behind these projects who gave us continuous support over the last years.

Rmetrics Association www.rmetrics.org

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. URL http://www.jstatsoft.org/v14/i06/
- [2] Adrian Trapletti and Kurt Hornik (2007): tseries: Time Series Analysis and Computational Finance. R package version 0.10-11.
- [3] Diethelm Würtz et al. (2007): Rmetrics: Rmetrics Financial Engineering and Computational Finance. R package version 260.72. http://www.rmetrics.org
- [4] International Organization for Standardization (2004): ISO 8601: Data elements and interchage formats Information interchange Representation of dates and time URL http://www.iso.org
- [5] R Development Core Team: R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org
- [6] Jeffrey A. Ryan (2008): quantmod: Quantitative Financial Modelling Framework. R package version 0.3-5. URL http://www.quantmod.com URL http://r-forge.r-project.org/projects/quantmod