zoo Quick Reference

Ajay Shah
Ministry of Finance, New Delhi

Achim Zeileis
Wirtschaftsuniversität Wien

Gabor Grothendieck
GKX Associates Inc.

Abstract

This vignette gives a brief overview of (some of) the functionality contained in **zoo** including several nifty code snippets when dealing with (daily) financial data. For a more complete overview of the package's functionality and extensibility see **?** (contained as vignette "zoo" in the package), the manual pages and the reference card.

Keywords: irregular time series, daily data, weekly data, returns.

Read a series from a text file

To read in data in a text file, `read.table()` and associated functions can be used as usual with `zoo()` being called subsequently. The convenience function `read.zoo` is a simple wrapper to these functions that assumes the index is in the first column of the file and the remaining columns are data.

Data in `demo1.txt`, where each row looks like

```
23 Feb 2005|43.72
```

can be read in via

```
R> inrusd <- read.zoo("demo1.txt", sep = "|", format = "%d %b %Y")
```

The `format` argument causes the first column to be transformed to an index of class `"Date"`.

The data in `demo2.txt` look like

```
Daily,24 Feb 2005,2055.30,4337.00
```

and requires more attention because of the format of the first column.

```
R> tmp <- read.table("demo2.txt", sep = ",")
R> z <- zoo(tmp[, 3:4], as.Date(as.character(tmp[, 2]), format = "%d %b %Y"))
R> colnames(z) <- c("Nifty", "Junior")
```

Query dates

To return all dates corresponding to a series `index(z)` or equivalently

```
R> time(z)
```

```
 [1] "2005-02-10" "2005-02-11" "2005-02-14" "2005-02-15" "2005-02-17"
 [6] "2005-02-18" "2005-02-21" "2005-02-22" "2005-02-23" "2005-02-24"
[11] "2005-02-25" "2005-02-28" "2005-03-01" "2005-03-02" "2005-03-03"
[16] "2005-03-04" "2005-03-07" "2005-03-08" "2005-03-09" "2005-03-10"
```

can be used. The first and last date can be obtained by

```
R> start(z)

[1] "2005-02-10"

R> end(inrusd)

[1] "2005-03-10"
```

Convert back into a plain matrix

To strip off the dates and just return a plain vector/matrix `coredata` can be used

```
R> plain <- coredata(z)
R> str(plain)

 num [1:20, 1:2] 2063 2082 2098 2090 2062 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:20] "1" "2" "3" "4" ...
  ..$ : chr [1:2] "Nifty" "Junior"
```

Union and intersection

Unions and intersections of series can be computed by `merge`. The intersection are those days where both series have time points:

```
R> m <- merge(inrusd, z, all = FALSE)
```

whereas the union uses all dates and fills the gaps where one series has a time point but the other does not with `NA`s (by default):

```
R> m <- merge(inrusd, z)
```

`cbind(inrusd, z)` is almost equivalent to the `merge` call, but may lead to inferior naming in some situations hence `merge` is preferred

To combine a series with its lag, use

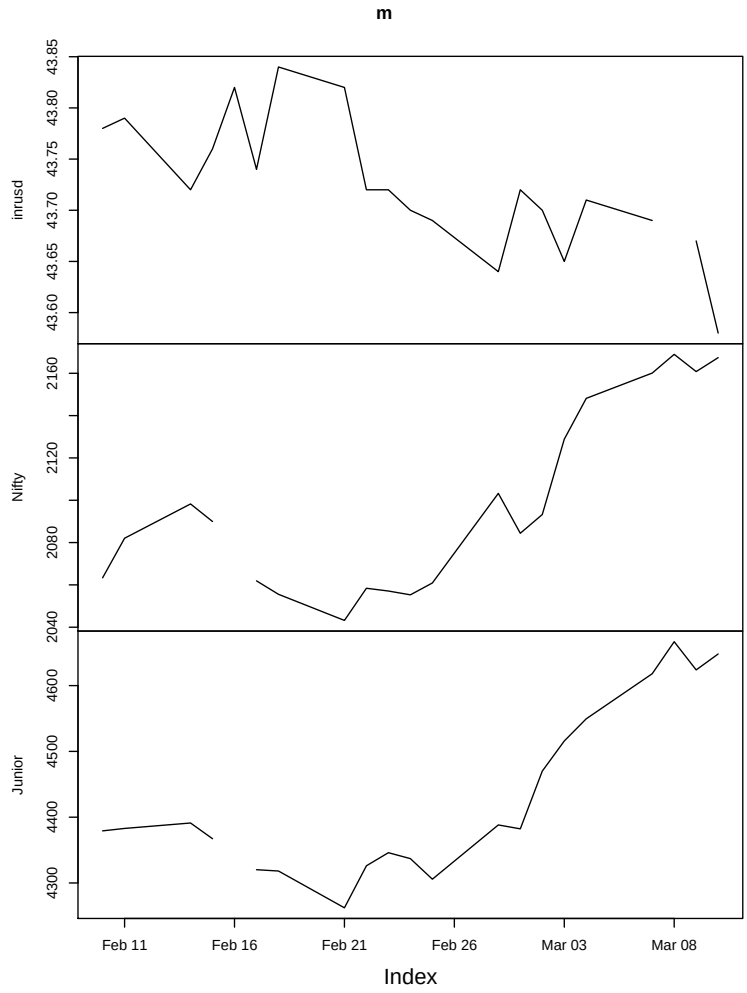
```
R> merge(inrusd, lag(inrusd, -1))

           inrusd lag(inrusd, -1)
2005-02-10  43.78              NA
2005-02-11  43.79           43.78
2005-02-14  43.72           43.79
2005-02-15  43.76           43.72
2005-02-16  43.82           43.76
2005-02-17  43.74           43.82
2005-02-18  43.84           43.74
2005-02-21  43.82           43.84
2005-02-22  43.72           43.82
2005-02-23  43.72           43.72
2005-02-24  43.70           43.72
2005-02-25  43.69           43.70
2005-02-28  43.64           43.69
2005-03-01  43.72           43.64
```

```
2005-03-02  43.70          43.72
2005-03-03  43.65          43.70
2005-03-04  43.71          43.65
2005-03-07  43.69          43.71
2005-03-09  43.67          43.69
2005-03-10  43.58          43.67
```

Visualization

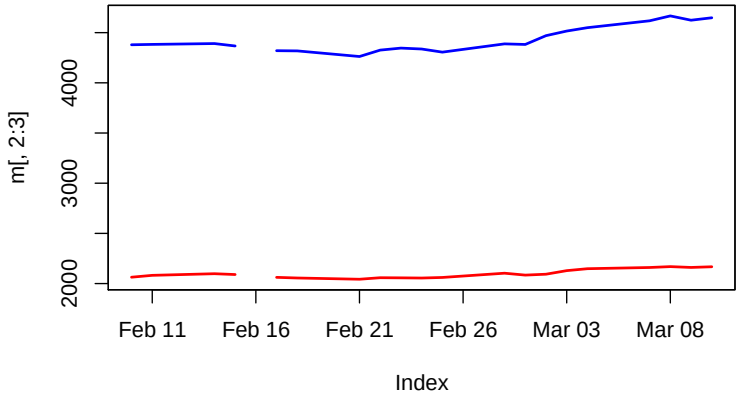
By default, the `plot` method generates a graph for each series in `m`

```
R> plot(m)
```



but several series can also be plotted in a single window.

```
R> plot(m[, 2:3], plot.type = "single", col = c("red", "blue"),
+     lwd = 2)
```



Select (a few) observations

Selections can be made for a range of dates of interest

```
R> window(z, start = as.Date("2005-02-15"), end = as.Date("2005-02-28"))
```

```
             Nifty  Junior
2005-02-15 2089.95 4367.25
2005-02-17 2061.90 4320.15
2005-02-18 2055.55 4318.15
2005-02-21 2043.20 4262.25
2005-02-22 2058.40 4326.10
2005-02-23 2057.10 4346.00
2005-02-24 2055.30 4337.00
2005-02-25 2060.90 4305.75
2005-02-28 2103.25 4388.20
```

and also just for a single date

```
R> m[as.Date("2005-03-10")]
```

```
           inrusd  Nifty  Junior
2005-03-10  43.58 2167.4 4648.05
```

Handle missing data

Various methods for dealing with `NA`s are available, including linear interpolation

```
R> interpolated <- na.approx(m)
```

'last observation carried forward',

```
R> m <- na.locf(m)
R> m
```

```
           inrusd   Nifty  Junior
2005-02-10  43.78 2063.35 4379.20
2005-02-11  43.79 2082.05 4382.90
2005-02-14  43.72 2098.25 4391.15
2005-02-15  43.76 2089.95 4367.25
2005-02-16  43.82 2089.95 4367.25
2005-02-17  43.74 2061.90 4320.15
2005-02-18  43.84 2055.55 4318.15
2005-02-21  43.82 2043.20 4262.25
2005-02-22  43.72 2058.40 4326.10
2005-02-23  43.72 2057.10 4346.00
2005-02-24  43.70 2055.30 4337.00
2005-02-25  43.69 2060.90 4305.75
2005-02-28  43.64 2103.25 4388.20
2005-03-01  43.72 2084.40 4382.25
2005-03-02  43.70 2093.25 4470.00
2005-03-03  43.65 2128.85 4515.80
2005-03-04  43.71 2148.15 4549.55
2005-03-07  43.69 2160.10 4618.05
2005-03-08  43.69 2168.95 4666.70
2005-03-09  43.67 2160.80 4623.85
2005-03-10  43.58 2167.40 4648.05
```

and others.

Prices and returns

To compute log-difference returns in %, the following convenience function is defined

```
R> prices2returns <- function(x) 100 * diff(log(x))
```

which can be used to convert all columns (of prices) into returns.

```
R> r <- prices2returns(m)
```

A 10-day rolling window standard deviations (for all columns) can be computed by

```
R> rollapply(r, 10, sd)
```

```
               inrusd     Nifty    Junior
2005-02-17 0.14599121 0.6993355 0.7878843
2005-02-18 0.14527421 0.6300543 0.8083622
2005-02-21 0.14115862 0.8949318 1.0412806
2005-02-22 0.15166883 0.9345299 1.0256508
2005-02-23 0.14285470 0.9454103 1.1957959
2005-02-24 0.13607992 0.9453855 1.1210963
2005-02-25 0.11962991 0.9334899 1.1105966
2005-02-28 0.11963193 0.8585071 0.9388661
2005-03-01 0.09716262 0.8569891 0.9131822
2005-03-02 0.09787943 0.8860388 1.0566389
2005-03-03 0.11568119 0.8659890 1.0176645
```

To go from a daily series to the series of just the last-traded-day of each month `aggregate` can be used

```
R> prices2returns(aggregate(m, as.yearmon, tail, 1))

             inrusd    Nifty   Junior
Mar 2005 -0.1375831 3.004453 5.752866
```

Analogously, the series can be aggregated to the last-traded-day of each week employing a convenience function `nextfri` that computes for each `"Date"` the next friday.

```
R> nextfri <- function(x) 7 * ceiling(as.numeric(x - 1)/7) + as.Date(1)
R> prices2returns(aggregate(na.locf(m), nextfri, tail, 1))

               inrusd      Nifty     Junior
2005-02-18  0.11411618 -1.2809533 -1.4883536
2005-02-25 -0.34273997  0.2599329 -0.2875731
2005-03-04  0.04576659  4.1464226  5.5076988
2005-03-11 -0.29785794  0.8921286  2.1419450
```

Query Yahoo! Finance

When connected to the internet, Yahoo! Finance can be easily queried using the `get.hist.quote` function in

```
R> library("tseries")
```

From version 0.9-30 on, `get.hist.quote` by default returns `"zoo"` series with a `"Date"` attribute (in previous versions these had to be transformed from `"ts"` 'by hand').

A daily series can be obtained by:

```
R> sunw <- get.hist.quote(instrument = "SUNW", start = "2004-01-01",
+     end = "2004-12-31")
```

A monthly series can be obtained and transformed by

```
R> sunw2 <- get.hist.quote(instrument = "SUNW", start = "2004-01-01",
+     end = "2004-12-31", compression = "m", quote = "Close")
```

Here, `"yearmon"` dates might be even more useful:

```
R> time(sunw2) <- as.yearmon(time(sunw2))
```

The same series can equivalently be computed from the daily series via

```
R> sunw3 <- aggregate(sunw[, "Close"], as.yearmon, tail, 1)
```

The corresponding returns can be computed via

```
R> r <- prices2returns(sunw3)
```

where `r` is still a `"zoo"` series.

Query Oanda

Similarly you can obtain historical exchange rates from http://www.oanda.com/ using `get.hist.quote`.

A daily series of EUR/USD exchange rates can be queried by

```
R> eur.usd <- get.hist.quote(instrument = "EUR/USD", provider = "oanda",
+     start = "2004-01-01", end = "2004-12-31")
```

This contains the exchange rates for every day in 2004. However, it is common practice in many situations to exclude the observations from weekends. To do so, we write a little convenience function which can determine for a vector of `"Date"` observations whether it is a weekend or not

```
R> is.weekend <- function(x) ((as.numeric(x) - 2)%%7) < 2
```

Based on this we can omit all observations from weekends

```
R> eur.usd <- eur.usd[!is.weekend(time(eur.usd))]
```

The function `is.weekend` introduced above exploits the fact that a `"Date"` is essentially the number of days since 1970-01-01, a Thursday. A mor intelligible function which yields identical results could be based on the `"POSIXlt"` class

```
R> is.weekend <- function(x) {
+     x <- as.POSIXlt(x)
+     x$wday > 5 | x$wday < 1
+ }
```