# 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 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 Zeileis and Grothendieck (2005) and the manual pages.

*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 friends can be used as usually and `zoo()` has to be 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. Therefore, the 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")
```

By specifying the `format` argument, the first column is transformed to an index of class `"Date"`. The data in `demo2.txt` look like

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

and need a bit more attention because the first column is useless.

```
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 query 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 ar 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 everything is observed

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

whereas the union uses all dates and fills the gaps (by default) with `NA`s

```
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))
```

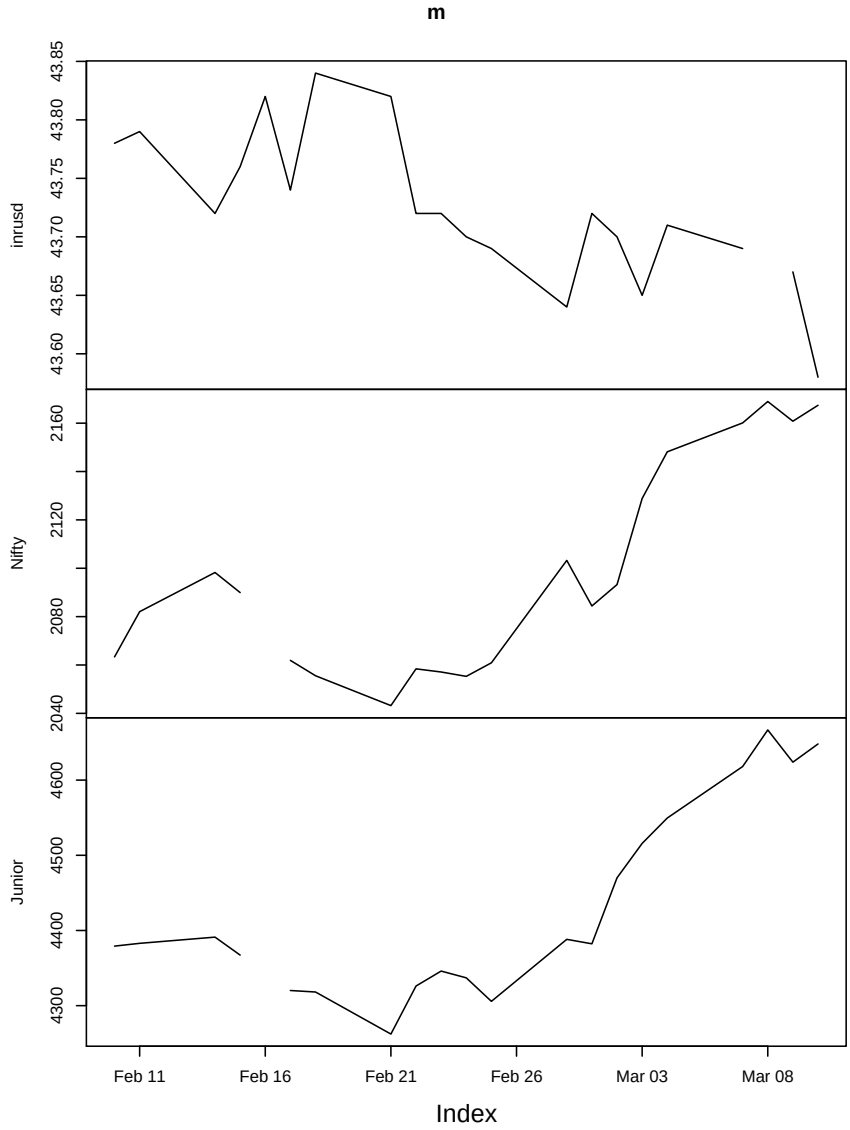
```
           inrusd lag(inrusd, -1)
2005-02-10 43.78     NA
2005-02-11 43.79  43.78
2005-02-12    NA  43.79
2005-02-14 43.72     NA
2005-02-15 43.76  43.72
2005-02-16 43.82  43.76
2005-02-17 43.74  43.82
2005-02-18 43.84  43.74
2005-02-19    NA  43.84
2005-02-21 43.82     NA
2005-02-22 43.72  43.82
2005-02-23 43.72  43.72
2005-02-24 43.70  43.72
2005-02-25 43.69  43.70
2005-02-26    NA  43.69
2005-02-28 43.64     NA
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
2005-03-05    NA  43.71
2005-03-07 43.69     NA
2005-03-08    NA  43.69
2005-03-09 43.67     NA
2005-03-10 43.58  43.67
2005-03-11    NA  43.58
```

### *Visualization*

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

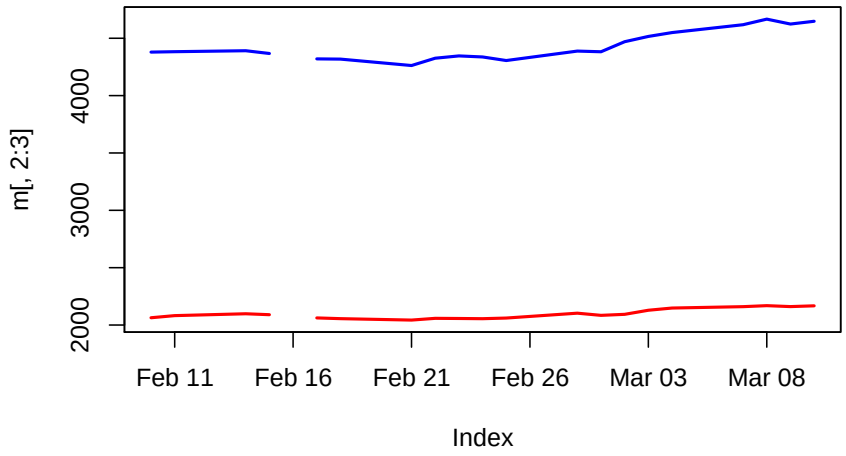
```
R> plot(m)
```




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 2167.40 4648.05
```

### *Handle missing data*

Various methods for dealing with `NA`s are available, e.g., linear interpolation

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

or '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
```

### *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> rapply(r, width = 10, FUN = sd)
```

```
           inrusd    Nifty     Junior
2005-02-18 0.1484024 0.6827704 0.7022275
2005-02-22 0.1497484 0.6168169 0.9586918
2005-02-23 0.1516702 0.8414873 0.9659141
2005-02-24 0.1517071 0.8838981 0.9710362
2005-02-25 0.1389399 0.7149261 0.8474269
2005-03-01 0.1160404 0.7217292 0.9557652
2005-03-02 0.1117103 0.7051424 0.8886226
```

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  3.0044525  5.7528657
```

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 -1.28095332 -1.48835360
2005-02-25 -0.34273997  0.25993286 -0.28757311
2005-03-04  0.04576659  4.14642260  5.50769882
2005-03-11 -0.29785794  0.89212859  2.14194498
```

### *Query Yahoo! Finance*

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

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

## References

Zeileis A, Grothendieck G (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/.