COMP226: Slides 04

Timeseries in R

Rahul Savani

rahul.savani@liverpool.ac.uk

Overview

This is the last set of slides for the first topic: **Intro to R**

- What is a **time series**?
- The **quantmod** package
- **Downloading and charting** financial data with quantmod
- **xts** time series objects in R

Time series

Definition

A **time series** is a **sequence of measurements indexed by time**. The measurements can be

- at **regular intervals**
- or **unevenly spaced** (e.g., trade times)

Time series analysis is prominent in Economics and Finance

Our motivation

- Any strategies we develop will be based on **historical data**
- This data will always be indexed by time
- It could be **daily** price and volume data or **tick data** (trades or orders that can happen at any time)

The quantmod package

- So far we have used only functions from R's **base** package, which is loaded automatically
- We will use several other packages that provide useful functions, e.g.,
- **quantmod** http://cran.r-project.org/package=quantmod
- Quantmod is good for
  - **getting daily financial data** and
  - **charting (plotting) financial time series**

Downloading data with quantmod

See `quantmod.R`:

```
library(quantmod) # this loads in the package

# if you haven't installed the package do so with
# install.packages("quantmod")

getSymbols("AAPL") # by default assigns to variable AAPL
# AAPL is the symbol for Apple Inc.
# it's listed on the NASDAQ stock exchange

print(head(AAPL)) # first few rows of the data
print(first(AAPL)) # first row
print(last(AAPL)) # last row
```

Downloading data with quantmod

```
> AAPL
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2007-01-03     86.29     86.58    81.90      83.80    44225700         81.03
2007-01-04     84.05     85.95    83.82      85.66    30259300         82.83
2007-01-05     85.77     86.20    84.40      85.05    29812200         82.24
2007-01-08     85.96     86.53    85.28      85.47    28468100         82.64
2007-01-09     86.45     92.98    85.15      92.57   119617800         89.51
2007-01-10     94.75     97.80    93.45      97.00   105460000         93.79
```

We have just loaded in an **xts time series** object

```
> class(AAPL)
[1] "xts" "zoo"

> first(AAPL)
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2007-01-03  12.32714  12.36857     11.7   11.97143   309579900       8.01682
> last(AAPL)
           AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
2019-02-04    167.41   171.655   167.28     171.25    31495582        171.25
```

Other packages we will use

- xts (eXtensible Time Series)
- Performance Analytics: Econometric functions for performance and risk analysis.
- TTR (Technical Trading Rules): Functions and data to construct technical trading rules with R.

CRAN task views

Collections of R packages related to a particular topic

http://cran.r-project.org/web/views/

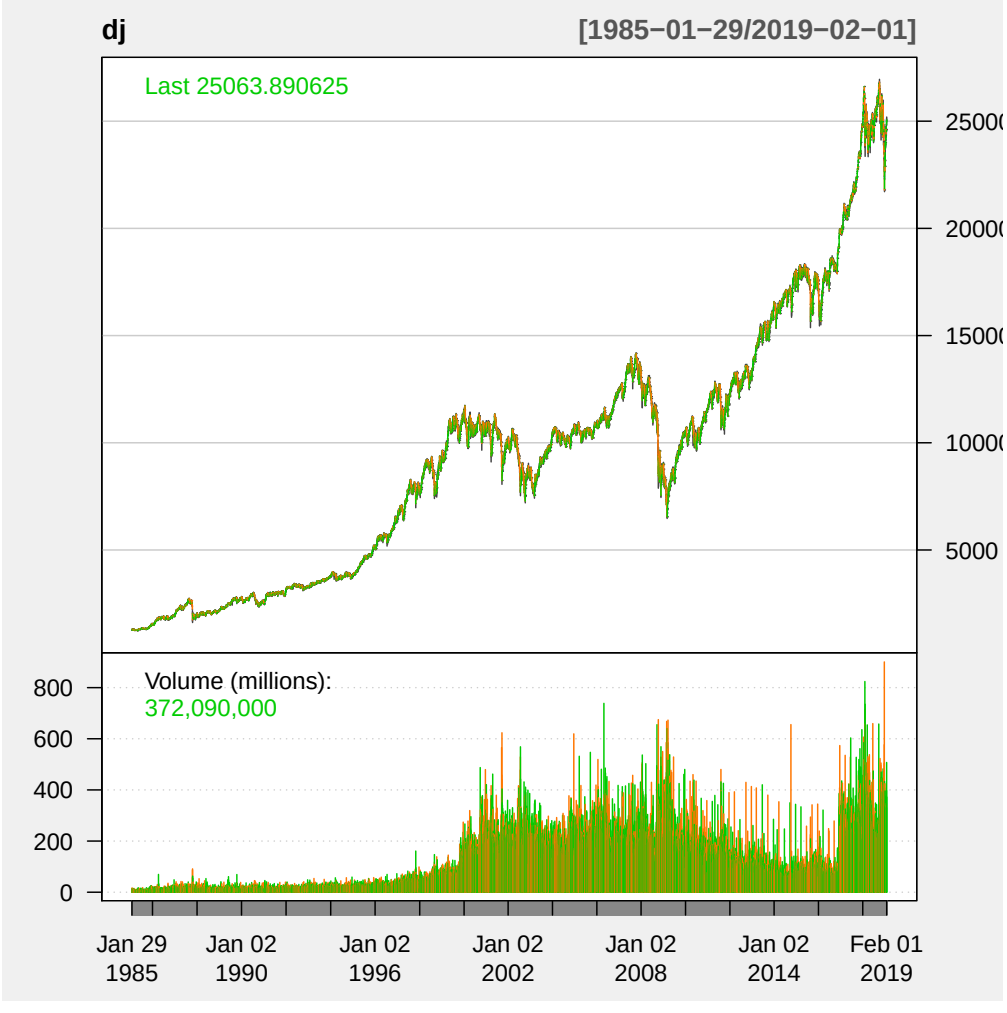
The most relevant ones are:

- http://cran.r-project.org/web/views/Econometrics.html
- http://cran.r-project.org/web/views/Finance.html
- http://cran.r-project.org/web/views/MachineLearning.html
- http://cran.r-project.org/web/views/Optimization.html
- http://cran.r-project.org/web/views/ReproducibleResearch.html
- http://cran.r-project.org/web/views/TimeSeries.html

Charting data with quantmod

```
library(quantmod) # library loads the package

dj <- getSymbols("^DJI",auto.assign=FALSE,
                                from="1900-01-01")
# here we specified a start date
# we also turned auto.assign off
# (otherwise it would have put the data
# in a variable called DJI)

pdf("DJI.pdf")
chartSeries(dj,theme="white")
dev.off()
```

Charting data with quantmod



Times and dates in R

There are many time and date classes in R, e.g.

- `POSIXct`, `POSIXlt`, `Date`, `chron`, `timeData`, `yearmon`, `yearqtr`

We will use mainly

- **Date** (date)
- **POSIXct** (date-time)

Times and Dates

```
> as.Date("2013-01-14")
[1] "2013-01-14"
```

```
> as.POSIXct("2013-01-14 14:24:22")
[1] "2013-01-14 14:24:22 GMT"
```

Sequences of dates

`seq` even works with dates:

```
> seq(as.Date("2009-01-01"),by=1,length.out=3)
[1] "2009-01-01" "2009-01-02" "2009-01-03"
```

Time series packages in R

There are two time series packages in R we will focus on:

- `xts`
- `zoo`

`xts` **extends** `zoo`

This means that all `zoo` functions work on `xts` objects

What is zoo?

- class of **indexed totally ordered observations**
- particularly aimed at irregular time series
- must contain data of **one** type (numeric/logical/character)
- zoo's key design goals are
    - independence from particular index/date/time class;
    - consistency with ts and base R by providing methods to **extend standard generics (e.g. plot)**

What is xts?

- **time-based** extension of `zoo` which is popular and robust
- **Requires indexing** bases on recognised time-based class, e.g. `POSIXct`, `Date`, `chron`, `timeData`, `yearmon`, `yearqtr`
- time-based subsetting
- hidden attributes (accessed with `xtsAttributes()`)
- smart conversion tools
- time-based tools like fast aggregation

Create an `xts` object

Two options:

- `as.xts()`
- `xts()` constructor

Example using `xts` constructor

```
library(xts)
prices <- c(90,100,120,80,70,60)

len <- length(prices)
dates  <- seq(as.Date("2009-01-01"), by=1,length.out=len)

price_series <- xts(prices, dates)
```

Example using `as.xts()`

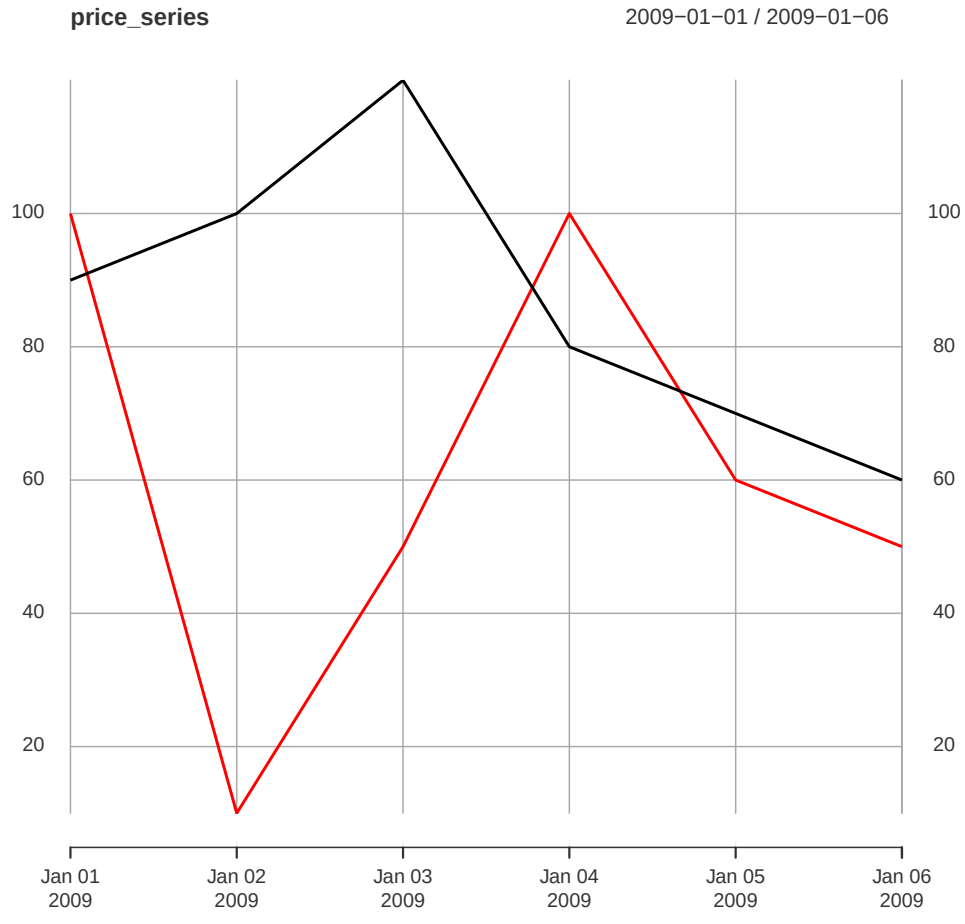
```
library(xts)
prices <- c(90,100,120,80,70,60)

len <- length(prices)
dates  <- seq(as.Date("2009-01-01"), by=1,length.out=len)

# as.xts requires names to be the time/dates
names(prices) <- dates

price_series <- as.xts(prices)
```

Two-column example with as

```
library(xts)

prices <- c(90,100,120,80,70,60)
volumes <- c(100,10,50,100,60,50)

len <- length(prices)
dates  <- seq(as.Date("2009-01-01"), by=1,length.out=len)

x <- cbind(prices,volumes) # x is a matrix with two columns

# as.xts requires rownames to be the time/dates
row.names(x) <- as.character(dates)

price_series <- as.xts(x)

plot(price_series)
plot.zoo(price_series) # alternative plotting function
```

```
> price_series
            prices volumes
2009-01-01     90     100
2009-01-02    100      10
2009-01-03    120      50
2009-01-04     80     100
2009-01-05     70      60
2009-01-06     60      50
```

```
> plot(price_series) # uses plot.xts
```



```
> plot.zoo(price_series) # alternative plotting function
```