# A Brief Overview of R


***Patrick Burns***

***http://www.burns-stat.com***

**June 2009**

This is the section pertaining to R of a talk that was presented 2009 June 03 at the Thalesians.

http://www.thalesians.com

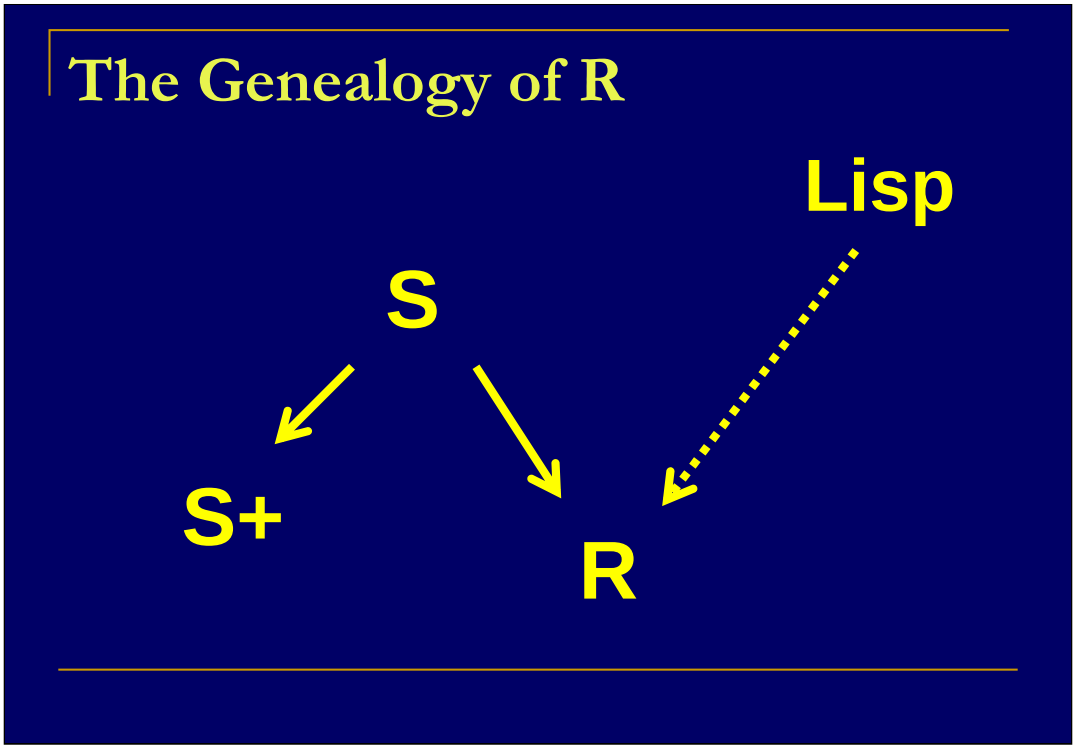
## The R Language

- **Data analysis**
- **Graphics**
- **Good alternative to spreadsheets**
- **Free and open source**
- **http://www.r-project.org**

The R language was designed for the purpose of data analysis. A large part of good data analysis is graphics.

Much of what is currently done in spreadsheets would be better done in R. See “Spreadsheet Addiction” for why spreadsheets are inherently unsafe.

http://www.burns-stat.com/pages/Tutor/spreadsheet_addiction.html




In the mid 1970's the Data Analysis group at Bell Labs started a research project into a computing environment for data analysis.

In the mid 1980s that research project started to escape out into the wild in the form of the S language.

In the late 1980s S spawned a commercial product called S-PLUS.

In the early 1990s R was written. That first version of R might be characterized as Scheme with a large crust of syntactic sugar on top that made it look like the S language.

In the mid 1990s the R Project formed.

R has been growing ever since.

## Language characteristics

- Rich in data structures
- Vector oriented
- Syntax similar to C
- Influenced by functional programming
- There exists object-orientation

A key difference with C is that indexing starts at 1, not 0.

The functional programming influence means that it is difficult to destroy your data accidentally.

## Data structures

- **Atomic vectors**
  - **Logical**
  - **Numeric**
  - **Complex**
  - **Character**
- **All these have NA (missing value)**

Atomic vectors are each only of one type. You can have logical values in a vector or numbers in the vector, but not both.

## Data structures

- **Lists**
  - **Components can be anything, including a list**
- **Functions**
  - **May have default values for arguments**
  - **Objects in the language**

Lists allow quite general data structures.

Default values of arguments in functions provides both convenience and flexibility.

## Data structures

- **Attributes**
  - **A named list of meta-data about the object**
- **“class” attribute implies object orientation**
- **“dim” attribute implies matrix or higher dimensional array**

Attributes are a stroke of genius. They expand our universe from the earth with the sun and moon and a few planets spinning around it to a place with billions of galaxies.

Attributes provide the possibility of virtually any sort of data structure.

Some functions change their behavior based on the `class` of the object given as an argument.

If an object has a `dim` that is c(3, 2), then we know it is a matrix with three rows and two columns.

If an object has a `dim` that is c(3, 2, 4), then it is a three-dimensional array with three rows, two columns and four slices.

## Subscripting

- **Can subscript with:**
  - **Positive integers**
  - **Negative integers**
  - **Characters**
  - **Logicals**

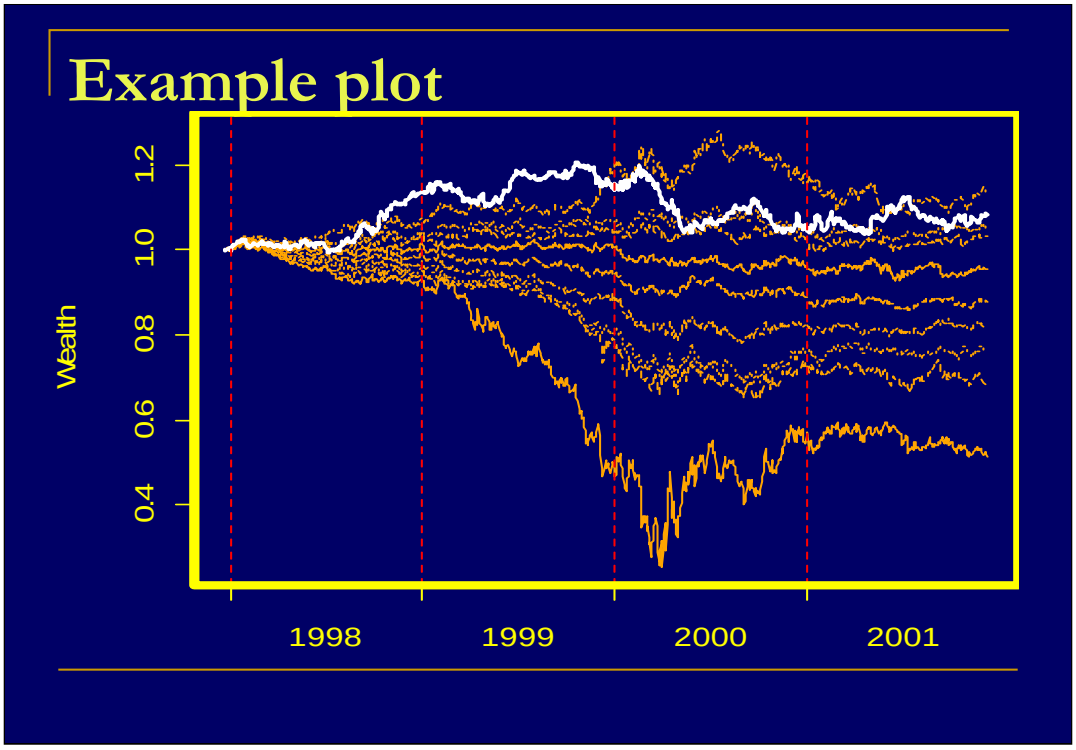
Subscripting is very important in R, partly because it is so flexible.

```
X[1:5] # selects the first five elements of X

X[-1:-5] # selects all but the first five elements of X

X[c('dog', 'cat')] # selects the elements with those names

X[c(TRUE, FALSE)] # selects the odd numbered elements
                  #  the logical vector is replicated to
                  #  the length of X
```




This plot has a custom x-axis that allows us to see quite well when events are happening.

The usual alternatives would be one of two cases:

1) The x-axis is treated as numbers from 1 to 1000.
2) There is a large number of dates written (possibly diagonally). You probably can't read the dates, and even if you can, you can not easily relate them to what is happening in the plot.

A key advantage of R is that it allows you to create graphics that do what they should, and not just what some programmer thinks you need.

## Example plot

```
matplot(quantile.matrix, axes=FALSE,
  type='l', col='orange')
lines(orig, col='white')
axis(2)
custom.timeaxis(quantile.matrix)
box()
```

The previous plot was created with these commands[1]. Graphics don't have to be created with just one command, they can be built up with a number of commands.

The key bit here is a function that I wrote to create the x-axis as I wanted it to appear.

Note that when doing typical data analysis, a single graphics command will generally get you what you want – this technique of using multiple commands is mainly for presentation graphics. It would also be possible to write a function that did this sort of plot, so we'd be back to one command again.

[1] The first command should also include the argument:

`ylab='Wealth'`

But that didn't fit in the slide well.

## Example: random portfolio

```
IBM  MSFT   C
237   84   -391
```

**Named numeric vector:**

```
rp <- c(IBM=237, MSFT=84, C=-391)

rp <- c(237, 84, -391)
names(rp) <- c("IBM", "MSFT", "C")
```

A natural way to represent a single random portfolio is as a named numeric vector.

Such an object can be created all in one go, or the data can be assigned to the object and then the `names` attribute added later. The resulting object is exactly the same in either case.

## Example: random portfolios

- **We want a set of random portfolios**
- **List with length equal to number of random portfolios**
- **Attributes include “class” and “call” and a timestamp**

A natural representation of a set of random portfolios is as a list with the length of the list equal to the number of random portfolios, and each component of the list being a named numeric vector describing a single random portfolio.

The ‘`call`’ (attribute in this case, often a component of a list) is an image of the command that created the object. Objects that include their call are self-describing, and there are additional advantages.

## R mailing lists

- R-help
- R-devel
- R-sig-finance
- R-sig-hpc
- …

There are several mailing lists for R.

‘sig’ stands for Special Interest Group.

And ‘hpc’ is High Performance Computing.

## The dark side of R

- Inconsistencies
- Redundancies
- RAM bound
- Slightly challenging to search on
- Drinking from a firehose

There are many inconsistencies in R. A key reason for this is that the S language was a research project long after it was being used for substantial work. Backward compatibility has been a big issue for 20 years now.

'The R Inferno' tries to help you past the inconsistencies.

http://www.burns-stat.com/pages/Tutor/R_inferno.pdf

There are redundancies both because of the on-going research problem, and because there may be several contributed packages that do essentially the same thing.

The freedom of having any data structure we like has the price that the data needs to all be in RAM. That is a problem for large data -- generally there are workarounds, though.

If you think you want to know all of R, give up that idea.

## The future of R

- **Three companies producing supported versions**
- **New statistical techniques appear first in R**
- **So bright I gotta wear shades**

For those of you who don't listen to Radio Paradise enough, the allusion in the last line is to a song by Timbuk3.

The momentum of R in the last few years is phenomenal.

I'm particularly surprised by the momentum in finance – the area that I know best.

Further reading:

"An Introduction to the S Language" provides a few more arguments of why R is a good thing.

http://www.burns-stat.com/pages/Tutor/slanguage.html

"A Guide for the Unwilling S User" is a brief introduction to using R or S+.

http://www.burns-stat.com/pages/Tutor/unwilling_S.pdf