"Putting the R in ScHARR"

Robert Smith

29. August 2019

Contents

Introduction to R . . . 3

Who we are: . . . 3

Course 1 Aims: . . . 4

Course 2 Aims: . . . 4

Course 3 Aims . . . 4

Course 4 Aims . . . 4

Course 5 Aims . . . 4

Course 1 **5**

Install and navigate R Studio. . . . 5

Using R to do basic calculations . . . 5

Importing Data . . . 6

Loops . . . 7

Using R Markdown **16**

Introduction to R

This series of short courses are designed to equip the participant with a basic set of tools to undertake research using R. The aim is to create a strong foundation on which participants can build skills and knowledge specific to their research and consultancy objectives. The course makes use of the authors' experiences (many of which were frustrating) of working with R for data-science and statistical analysis. However there are many other resources available, and we would particularly recommend the freely available content at *R for Data Science* as a good place to recap the materials taught in this course. The hard copy of Hadley Wickham and Garrett Grolemund's book of the same name (and content) is available at *Amazon.com*. Alternatively, a user guide is available on the CRAN R-Project website here, although the author finds this less easy to follow than Hadley Wickham's book described above. Further details of where to go to answer more specific questions are provided throughout the course.

Requirements: It is assumed that all participants on the course have their own laptop, and have previously used software such as Excel or SPSS. Some basic understanding of statistics and mathematics is required (e.g. mean, median, minimum, maximum).

Who we are:

Robert Smith is a PhD Candidate at ScHARR, funded by the Wellcome Trust Doctoral Training Centre in Public Health Economics and Decision Science. His focus is on the methods used to estimate the costs and benefits of public health interventions, with a specific interest in microsimulation modelling (done in R). He has become increasingly intersted in the use of R-Markdown and R-Shiny to make research more transparent and to aid decision makers. While doing his PhD, Robert has been involved in projects with the WHO, Parkrun and Western Park Sports Medicine.

Paul Schneider ...

Sarah Bates ...

Course 1 Aims:

By the end of the 1 day short course, the attendee should be able to:

- Install and navigate R Studio. Set working directory.
- Understand the types of objects and basic operations in R.
- Read in data from csv and excel files.
- Summarise data.
- Perform basic regression analysis on data.
- Create basic plots and graphs.
- Use an 'if' function
- Create a loop
- Know where to find further information (StatsExchange/Google).

Course 2 Aims:

By the end of the 1 day short course, the attendee should be able to:

- Create beautiful graphs using ggplot.
- Use the 'apply()' function to improve run-time.
- Create a custom function.
- Know where to find further information (StatsExchange/Google).

Course 3 Aims

By the end of 1 day short course, the attendee should also be able to:

- Understand the strengths and limitations of R for HTA.
- Create a markov model from scratch given known parameters.
- Create a microsimulation model to incorporate hetrogeneity between groups.
- Understand the importance of tranparency of coding. In particular commenting.

Course 4 Aims

By the end of the 1 day short course, the attendee should be able to:

- Understand the benefits and limitations of R-Shiny.
- Have a basic understanding of the principles behind R-Shiny.
- Create an R-Shiny application from scratch.
- Integrate beautiful plots into R-Shiny.
- Develop a user interface for an existing complex model in R-Shiny.

Course 5 Aims

By the end of the 1 day short course, the attendee should be able to:

- Understand the strengths and limitations of R-Markdown.
- Create replicatable HTML, Word and PDF documents using R-Markdown.
- Include chunks of code, graphs, references and biblographies, links to websites and pictures.
- Change replicate analysis for new or updated datasets, or create templates for routine data.
- Create markdown presentations.

Course 1

Install and navigate R Studio.

R is a free software environment for statistical analysis and data science. It is a collaborative effort started by Robert Gentleman and Ross Ihaka in New Zealand, but now made up of hundreds of people. R is made freely available by the Comprehensive R archive Network (CRAN), in the UK it can be downloaded through the University of Bristol here.

Downloading R gives you the basic software environment, but an incredibly popular add-on called 'RStudio' is required to follow this course. You should download the free 'RStudio Desktop Open Source Licence' version for the laptop you will be attending the course with from RStudio.com.

If you have time before the course, it would be hugely beneficial to get familiar with RStudio.

Objectives:

Download R from https://www.stats.bris.ac.uk/R/.
Download RStudio from https://www.rstudio.com/products/rstudio/#Desktop.

Using R to do basic calculations

Anything you can do with a calculator, you can do in R.

```
2+2
```

```
## [1] 4
```

<- can be used to assign values to objects.

```
a <- 2
b <- 2

a+b
```

```
## [1] 4
```

Exercise

Calculate the following, where a = 20 b = 9, c = 5, d = 1.2

You are only allowed to type each number once:

$x < -4b + 7c + 3d$

$\frac{12 \times (a+b)}{x}$

Functions

Functions can be applied to make data manipulation easier. Many functions are included in baseR. For example the 'log' function takes the log of a number. We can start by looking up the help file and see that the default for base for log is exp(1).

```r
help(log)

# base exp(1) is the default
log(x = 9,base = exp(1))
```

But what we can change that base to be anything we want:

```r
log(x=9, base = 3)
```

```
## [1] 2
```

Importing Data

In almost every case, you will want to import some data to analyse. This data will come in a variety of formats. R studio has a nice feature in Environment>Import_Dataset which enables you to select the file you wish to download (similar to STATA) from your computer. The data is then imported into R and the code available in the console.

It is possible to import files in the following formats:

Type	Suffix
R	.R
CSV	.csv
Excel	.xls/.xlsx
SPSS	.spv
Stata	.dta

Alternatively packages can be installed to import data in almost any format. For example the `readr` package can read in spreadsheets from text files or tab delimited files.

CSV

We can import the file using the full path with the file name and suffix included such as below:

```r
read.csv("C:/Users/Robert/Google Drive/MyProject/Data/rawdata.csv", header=FALSE)
```

Setting Working Directory

Or, if we know that we are going to be downloading a lot of files from one folder we can set a working directory. The working directory is the defined folder in which R will then import and export files from and to. This allows users to send whole files to others who can replicate the work by simply changing the working directory to the new file location.

The current working directory can be found by typing:

```r
getwd()
```

```
## [1] "C:/Users/Robert/Documents"
```

A new working directory can be set by clicking on the tab (Session) then (Set_Working Directory), or by the command setwd:

```
setwd(filename)
```

Downloading files from the internet

Sometimes it is more practical to download files directly from the internet. There are lots of different packages out there to do this. The one I use was developed by Hadley Wickham, called readr. Here we are going to download some data from the github page for the course.

```
# load the readr package, if this is not installed then install it.
library(readr)

#use the function read_csv
data <- read_csv("https://raw.githubusercontent.com/RobertASmith/Intro_to_R/master/who_complete.csv")
```

Downloading files directly to R within the same script as the analysis can be useful since it reduces the risk of you accidently changing the file. Just be careful that the data will always be available.

Loops

There are numerous different ways to create loops, we are going to look at the simplist, the **for** loop which executes the command for each number given.

```
 # Loop printing 1 to 10

for(i in 1:10) { # for each value of i from 1 to 10.
  print(i)       # print the value of i
}                # close the loop
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

We can also create an object, in this case a vector with the numbers we would like to loop through:

```
# create an object of odds
odds <- c(1,3,5,7,9)

# Loop printing all the values in the odds vector.
for(i in odds) { # for each value of i in the odds vector.
  print(i)       # print the value of i
}                # close the loop
```

```
## [1] 1
## [1] 3
## [1] 5
## [1] 7
## [1] 9
```

If we create a vector, *ourdata*, beforehand we can fill it with the output from each iteration of the loop. This is particularly useful if we want to record the output of a simulation.

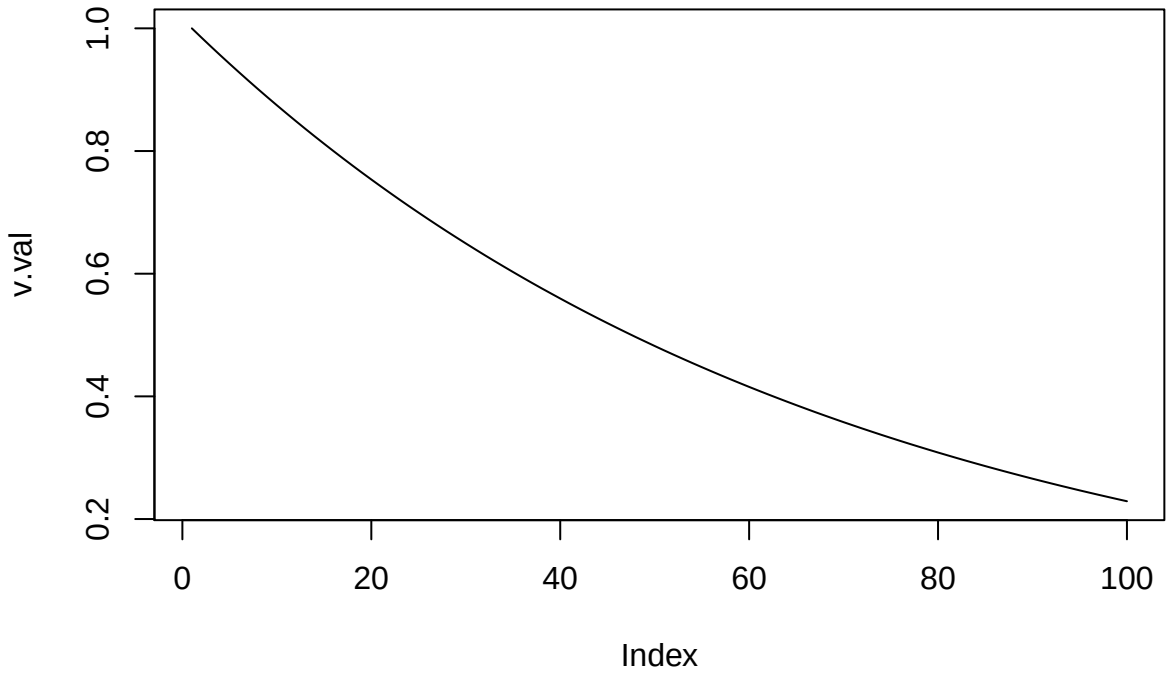
```
#create an vector of empty data
ourdata <- vector(mode = "numeric",length = 10)

# Fill our vector with values
for(i in 1:length(ourdata)) { # for each value of i in our data object.
  ourdata[i] <- i             # print the value of i
}                             # close the loop

print(ourdata)                # print the vector
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

We can do much more complex calculations within loops, the sky is the limit, but here are two examples.

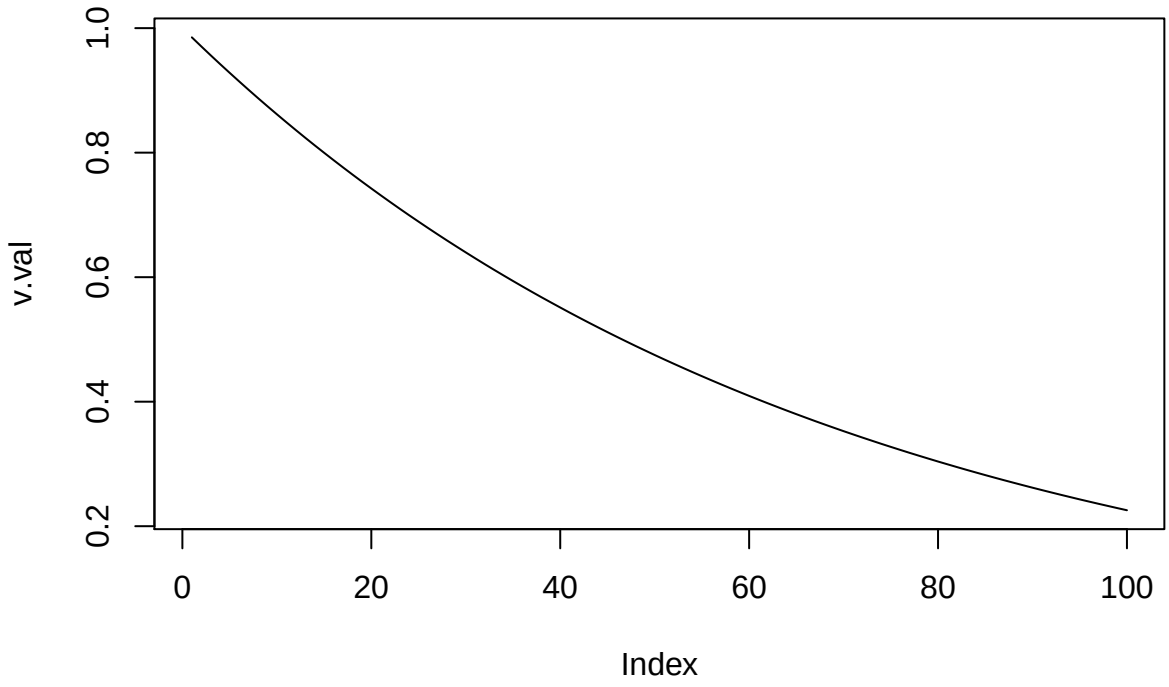
Example 1 Example 1 is an example of discounting each year simply by multiplying the previous year value by (1/(1+d.r)).

```
#create an vector of empty data
v.val <- vector(mode = "numeric",length = 100)

# start with 1
v.val[1] <- 1

# Discount our value at 1.5% each year
for(i in 2:length(v.val)) {            # for each value of i in the odds object.

  v.val[i] <- v.val[i-1]* (1/1.015)    # it takes the value of the previous year value 1/(1.015)

  }                                      # close the loop

plot(v.val,type="l")                          # print the vector
```



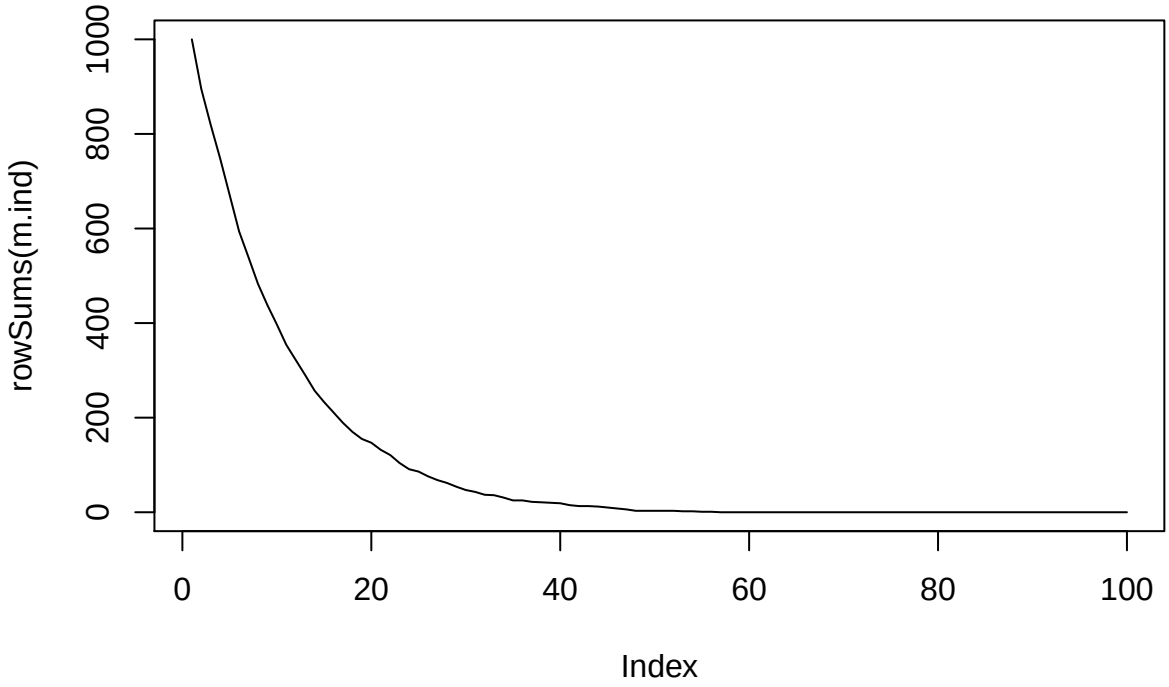
An important lesson though, it is often possible to achieve a goal without loops, which is much faster in R when running big simulations. In this case we could have achieved the same result with the following, much simpler code:

```
v.val <- (1/1.015)^(1:100)

plot(v.val,type = "l")
```



Example 2 Example 2 records the survival of 1000 individuals who die with probability 0.1.

```
#create an vector of empty data
m.ind <- matrix(data = NA,
                nrow = 100,
                ncol = 1000)

# everyone starts alive (1)
m.ind[1,] <- 1

# Each person has a probability of death of 0.1
for(i in 2:nrow(m.ind)) {

  # value is 1 if alive in previous period and random number > 0.1
  m.ind[i,] <- m.ind[i-1,] * runif( n = 1000, min = 0, max=1 ) > 0.1
}                                       # close the loop

plot(rowSums(m.ind),type="l")   # print the sums of the rows of the matrix (% alive)
```



Example 3

Example 3 runs a simple simulation testing the claims by the FIRE movement, Financial Independence Retire Early, that 4% is a safe withdrawl rate on a portfolio made up of US equities.

The Observer Retirement planning

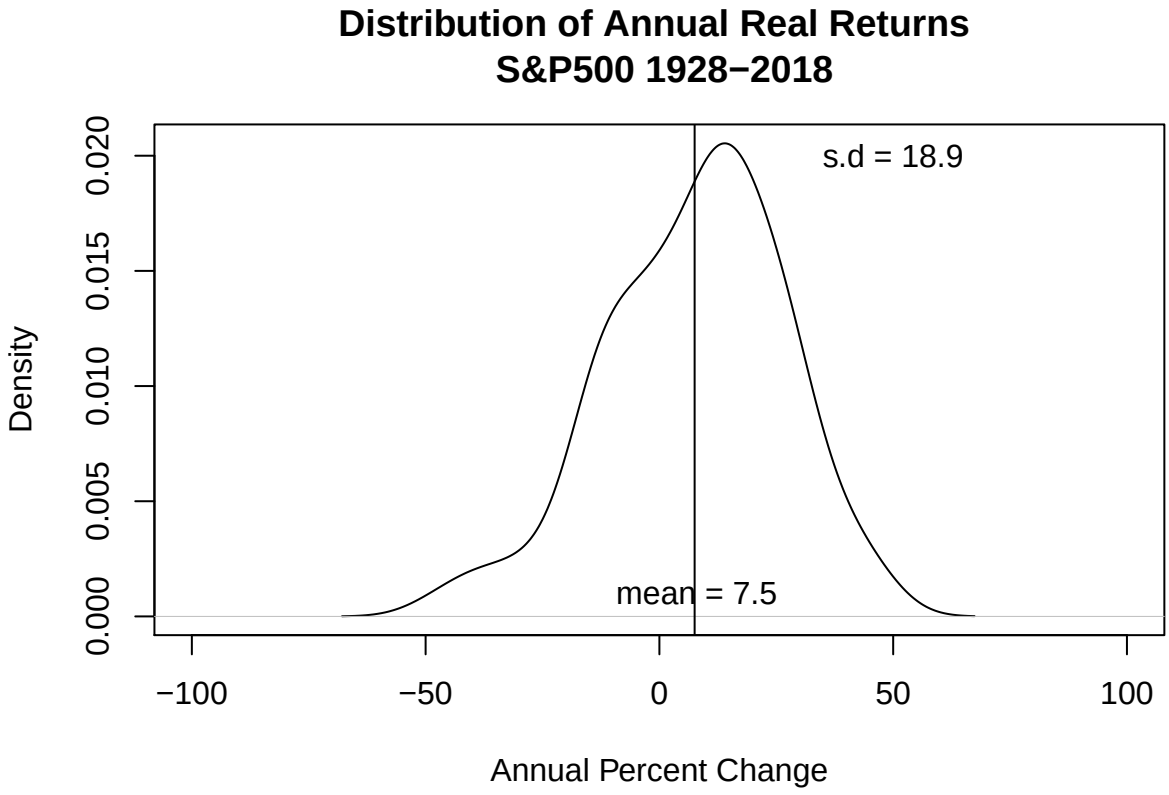
Meet the people trying to save enough to retire by 40

Followers of the Fire - Financial Independence, Retire Early - movement say it's possible to amass enough cash to quit work and follow your dreams in mid-life

We start by clearing the work environment and loading pre-downloaded packages and the required data.

```
#===
# FINANCIAL INDEPENDENCE RETIRE EARLY
#===

# clear the work environment
rm(list=ls())

# load necessary pre-downloaded packages
library(readxl)
library(fitdistrplus)

# Read in data from source, an excel file using data from https://www.macrotrends.net/2526/sp-500-histo

sp500 <- as.data.frame(read_excel("Data/snp.xlsx", sheet = "Sheet1"))
```

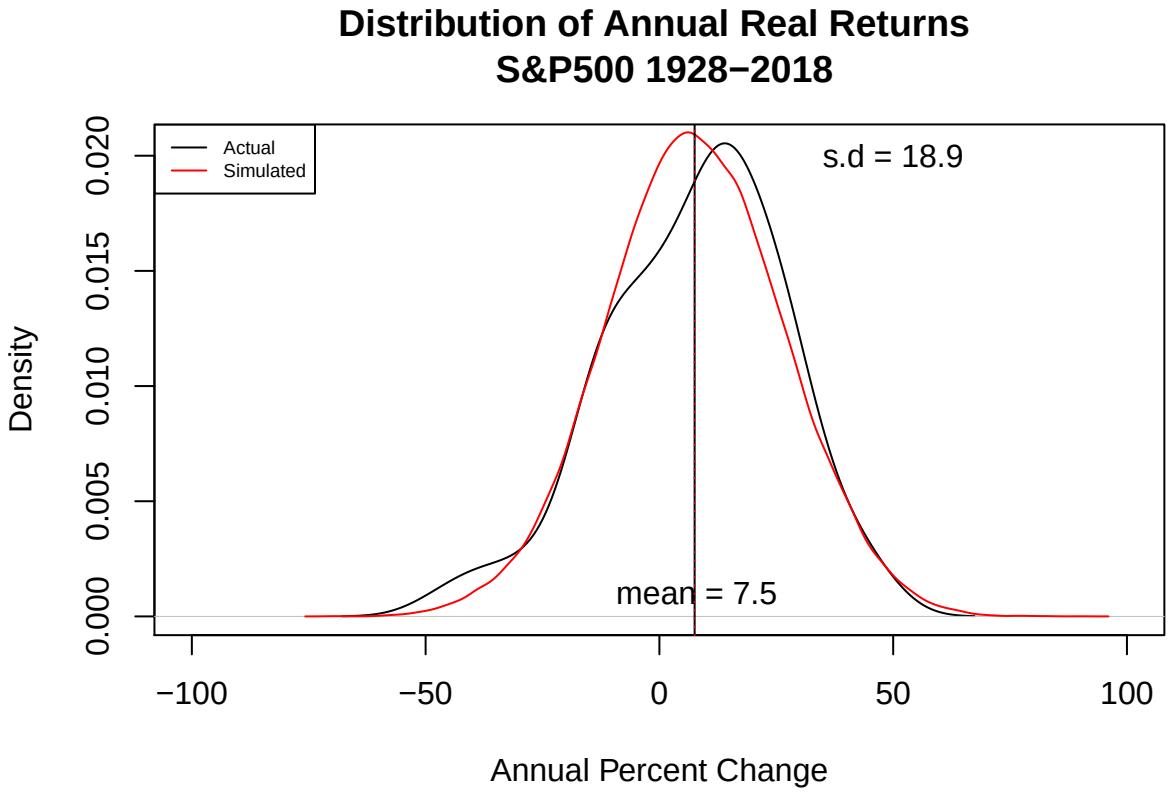
Then we can look at real returns of the S&P500 index since 1928.



Since the returns follow what is close to a normal distribution with a mean of approximately 7.53 and a standard deviation of approximately 18.94, it is possible to create a sample of 50,000 using the **rnorm** function in R.

```
set.seed(100)

# create vector of returns with mean = mean and s.d  = standard deviation.
v.rrsp500 <- rnorm(n = 50000,
                   mean = mean,
                   sd = sd)
```

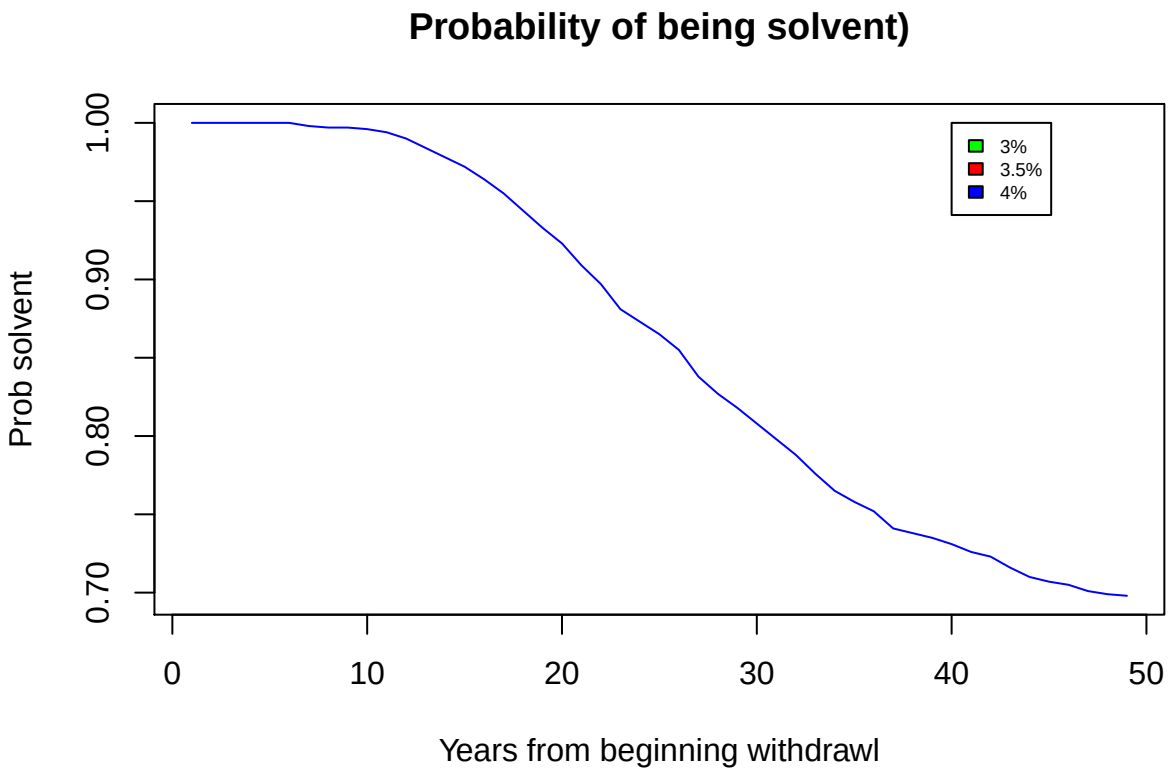
We can then plot these to show that the two are very similar, enough for this simple analysis anyway.



Then we can run a simulation to see what would happen to a £1million pound portfolio invested solely in the S&P500. We want to see what happens if a person withdraws £40,000, 4% of the original £1million, every year but keeps excess money the market may return to them invested.

```
# create a set of returns for 1000 people for 50 years
sim.real.returns <- matrix(data = v.rrsp500,
                           nrow = 50,
                           ncol = 1000)

#create a results matrix, 50 years 1000 individuals
results <- matrix(data = NA,
                  nrow = 50,
                  ncol = 1000)

#create a vector to store the % with net worth >0
solvent <- vector(mode = "numeric",length = 50)

# everybody starts with £100000
results[1,] <- 1000000

# Loop for 4%

for(x in 2:50){ # loop from 2 to 50 years
  results[x,] <- (results[x-1,] - results[1,]*0.04)*(1+sim.real.returns[x,])
  solvent[x] <- sum(results[x,] >0)/1000
}
```

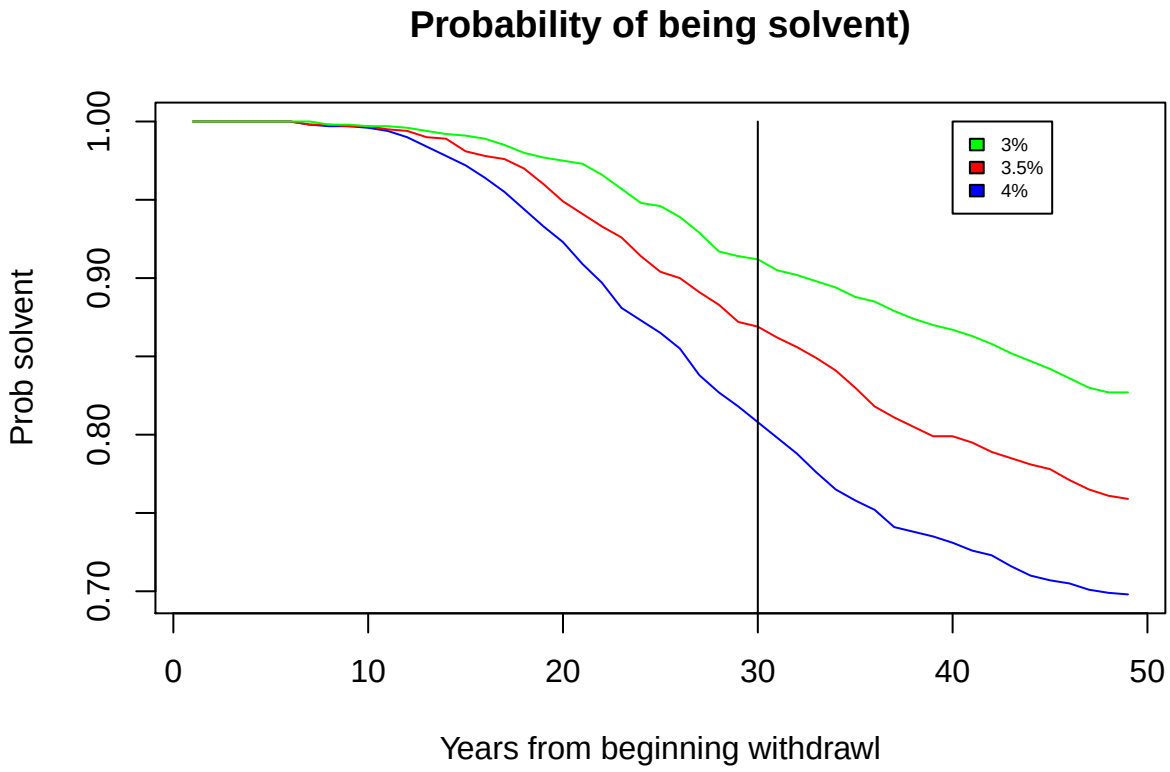
Since we have a sample of 1000, we can plot the probability of being solvent (not running out of money), over time. As we can see, the risk of running out of money is very low at first, but over 40 years they are quite high, close to 1/4 of the time.



We can then replicate the analysis for other rates of withdrawl and add to the graph. For example 3.5% and 3%.

```
plot(solvent[2:50],col="blue",type="l",
     main = "Probability of being solvent)",
     xlab = "Years from beginning withdrawl",
     ylab = "Prob solvent")
     legend(x = 40,y=1,legend =c("3%","3.5%","4%"),
            fill = c("green","red","blue"),cex=0.6)

# Loop for 3.5%

for(x in 2:50){
  results[x,] <- (results[x-1,] - results[1,]*0.035)*(1+sim.real.returns[x,])
  solvent[x] <- sum(results[x,] >0)/1000
}

lines(solvent[2:50],col="red")

# Loop for 3%

for(x in 2:50){
  results[x,] <- (results[x-1,] - results[1,]*0.03)*(1+sim.real.returns[x,])
```

```
  solvent[x] <- sum(results[x,] >0)/1000
}

lines(solvent[2:50],col="green")


lines(x=c(30,30),y=c(0,1))
```



Reducing our withdrawl to 3% reduces our probability of running out of money over 30 years quite substantially. From around 18% to around 8%.

The purpose of this example was not to provide financial advice, but to show how simple this type of analysis is in R.As always there are numerous limitations with the analysis:

- The annual returns are not completely independent. The stock market has booms and busts.
- Individuals may choose to spend less in recessions, or have alternative incomes.
- Individuals will probably have more diversified portfolios.

Using R Markdown

This document has been created in R Markdown.

It is possible to embed an R-Shiny App into an R-Markdown Document.

```
#{r fig.height=8, fig.width=8} #knitr::include_app(url = "https://iolmap.shinyapps.io/parkrun/")
#
```

Or just a simple website:

```
knitr::include_url(url = "https://www.youtube.com/watch?v=KznAcOK7cTQ")
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```