"Putting the R in ScHARR"

Robert Smith

26. September 2019

Contents

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 ...

Thomas Bayley ...

Naomi Gibbs ...

Our series of Short Courses in R.

We have put together content for a series of short courses in R. These courses are designed to provide the user with the necessary skills to utilise R to answer questions using data. By the end of the series a user should be able to manage data efficiently, analyse relationships between variables and display this in aesthetically pleasing graphs, run simulations, feed back on methods and create apps to facilitate the decision making process. The objective is to give the user an understanding of what is possible, and the foundation on which to achieve it.

Course 1 - Intro to R

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.
- Know where to find further information (StatsExchange/Google).

Course 2 - Beautiful Visualisations

By the end of the half 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 - R for Health Economics

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 - R Shiny

By the end of the half 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 - Collaboration in R

By the end of the half 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 - Introduction to R.

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.

Basics

Our course starts in the R console, which those of you who are familiar with R but not RStudio will recognise. We will enter commands as input into the console, and receive output from the console. We will start with some simple basic operations, for which R is clearly very excessive.

Basic operations

Entering 1+1, we get the output [1] 2. The output is 2, but the [1] lets us know that the number 2 is the first number of output. If we had an output that was particularly large (e.g. 100 seperate numbers) then r may let us know that the first row displayed starts with the first value [1] and the second row starts with the [x]th value.

```
rm(list = ls())

# add 1 to 1.
1 + 1
```

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

```
# divide 12 by 4
12/4
```

```
## [1] 3
```

```
# times 3 by 7
3*7
```

```
## [1] 21
```

```r
# 10 to the power 3
10^3
```

```
## [1] 1000
```

```r
# root isn't a basic operation so we will look at later.
```

Objects

It is possible to create objects, an object can take a number of forms (e.g. a number, a vector of numbers, a matrix, a data-frame). We can then use these objects going forward rather than the values directly. Operations can be applied to these objects, and objects can be over-written. R is basically just objects and functions.

```r
# objects basics
x <-  3
y <-  5
x + y
```

```
## [1] 8
```

```r
x <-  4
x + y
```

```
## [1] 9
```

```r
z <- x + y
z
```

```
## [1] 9
```

Overwriting / Manipulating Objects

```r
a <- 10
a
```

```
## [1] 10
```

```r
a <- a + 1
a
```

```
## [1] 11
```

Exercises

set d equal to 10. divide d by 5 multiply d by 8 add 8 to d what is d?

Evaluations

We can perform evaluations, which provide a true or false answer. For example the input 4>2 returns "FALSE".

It can be very useful in cases where an outcome is binary (e.g. an individual dies or remains alive). Or where we want to change a continous variable to a binary.

```
# simple evaluations
# 4 is greater than 2
4 > 2
```

```
## [1] TRUE
```

```
# 4 is greater than 5
4 > 5
```

```
## [1] FALSE
```

```
# 4 is equal to 3, note double == for an evaluation
4 == 3
```

```
## [1] FALSE
```

```
# 4 is not equal to 3, note != is not equal to.
4 != 3
```

```
## [1] TRUE
```

```
# the character x is equal to the character x.
"dog" == "dog"
```

```
## [1] TRUE
```

```
"dog" == "cat"
```

```
## [1] FALSE
```

```
# the output from an evaluation can be stored as an object, x. This object can be subject to operations
b <- 4<2
b
```

```
## [1] FALSE
```

Different object classes and types

Objects don't have to take a single value. For example a single object may be the heights of each child in a group of children (in the example below a small class of 4).

We have been working with single values, which are vectors of 1. To illustrate the different classes we are going to create some vectors.

Object Classes

Different class include numeric, character, logical, integer & complex (ignore).

```
# numeric
height <- c(1.72,1.78,1.65,1.90) # 1:4
height
```

```
## [1] 1.72 1.78 1.65 1.90
```

```
# numeric
weight <- c(68,75,55,79)
weight
```

```
## [1] 68 75 55 79
```

```
class(weight)
```

```
## [1] "numeric"
```

```
# character
first_name <- c("Alice","Bob","Harry","Jane")
first_name
```

```
## [1] "Alice" "Bob"   "Harry" "Jane"
```

```
#first_name + 1 # error

# factor
sex <- factor(x = c("F","M","M","F"))
sex
```

```
## [1] F M M F
## Levels: F M
```

```
# logical
tall <- height > 1.8
```

Operations on different data structures

Operations on Vectors

```
#Adding:
c(1,2,3) + 1
```

```
## [1] 2 3 4
```

```r
c(1,2,3) + c(1,2,3)
```

```
## [1] 2 4 6
```

```r
#multiplication
heightft <- height*3.28
```

Exercise Create a vector called 'odds' with the numbers 1,3,5,7,9. Show what class odds is.

Evaluate which numbers in the odd vector are greater than 4.

Create a vector called 'fail' containing 1,3,5,'seven',9. Show what class fail is.

Create a vector that gives everyone's weight in pounds (2.2lbs to kg)

Basic object Types

There are multiple types of object in R. We can store objects together in a data-frame. In our example data-frame each column is a variable (height, weight, first_name), and each row is an individual.

Different object types include:

Vector - single variable a 1x1 vector. Vector all elements same data-type. Matrix - all same data-type. Dataframe - columns same data type. List - anything goes.

```r
# data frame- columns are variables, rows are observations.
df <- data.frame(height,weight,first_name,sex)
df
```

```
##   height weight first_name sex
## 1   1.72     68      Alice   F
## 2   1.78     75        Bob   M
## 3   1.65     55      Harry   M
## 4   1.90     79       Jane   F
```

```r
# we can select a single variable within the dataframe using the dollar sign.
df$height
```

```
## [1] 1.72 1.78 1.65 1.90
```

```r
# We can add a new variable easily, in this case based on other variables within the dataframe.
df$bmi <- df$weight / df$height^2
df
```

```
##   height weight first_name sex      bmi
## 1   1.72     68      Alice   F 22.98540
## 2   1.78     75        Bob   M 23.67125
## 3   1.65     55      Harry   M 20.20202
## 4   1.90     79       Jane   F 21.88366
```

Subsetting

We can subset our data, to reduce it to those we are interested in. This is useful when cleaning our data, and when changing a continuous variable to a categorical.

```
# Our data-frame contains the height, weight, first name and bmi of 4 individuals.
df
```

```
##   height weight first_name sex      bmi
## 1   1.72     68      Alice   F 22.98540
## 2   1.78     75        Bob   M 23.67125
## 3   1.65     55      Harry   M 20.20202
## 4   1.90     79       Jane   F 21.88366
```

```
#To subset a data frame we can use square brackets i.e df[row,column]
#Selecting a column(s)
df$height
```

```
## [1] 1.72 1.78 1.65 1.90
```

```
df[,"height"]
```

```
## [1] 1.72 1.78 1.65 1.90
```

```
df[,1]
```

```
## [1] 1.72 1.78 1.65 1.90
```

```
df[,1:3]
```

```
##   height weight first_name
## 1   1.72     68      Alice
## 2   1.78     75        Bob
## 3   1.65     55      Harry
## 4   1.90     79       Jane
```

```
df[,c(1,3)]
```

```
##   height first_name
## 1   1.72      Alice
## 2   1.78        Bob
## 3   1.65      Harry
## 4   1.90       Jane
```

```
#selecting a row(s)
df[1,]
```

```
##   height weight first_name sex     bmi
## 1   1.72     68      Alice   F 22.9854
```

```r
#We might also want to select observations (rows) based on characteristics of the data
#E.g. we might want to only look at the data for people who are taller than 1.75m

#create a logical variable called min_height which contains T/F for each individual being over 175cm.
min_height <- df$height >= 1.75
min_height
```

```
## [1] FALSE  TRUE FALSE  TRUE
```

```r
# Subset the data to include only those observations (rows) for which height > 175cm (using min_height)
df.at_least_175 <- df[min_height,]
df.at_least_175
```

```
##   height weight first_name sex      bmi
## 2   1.78     75        Bob   M 23.67125
## 4   1.90     79       Jane   F 21.88366
```

```r
#People smaller than 1.75m
# Subset the data to include only those who are not above min-height of 175cm.
smaller = df$height < 1.75
df[smaller,]
```

```
##   height weight first_name sex      bmi
## 1   1.72     68      Alice   F 22.98540
## 3   1.65     55      Harry   M 20.20202
```

```r
df[!min_height,]
```

```
##   height weight first_name sex      bmi
## 1   1.72     68      Alice   F 22.98540
## 3   1.65     55      Harry   M 20.20202
```

Not that there are other more advanced methods, which uses pipes and require less code (these are covered in more advanced courses).

Exercises select the 3rd row from the data frame

Select the weight variable from the data frame using your prefered method

Select alice's data from the data frame

Subset the data frame to show just the data for the females

type df[,-1] what does this give

Exercises

Exercise 1 Simple calculations

a) Calculate the following:

5*10 20/3

More complex calculations. b) Calculate x where a = 20 b = 9, c = 5, d = 1.2 *you are only allowed to type each number once*

$x = 4b + 7c + 3d$

$x = \frac{8b+4c-12d}{a}$

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

Exercise 2 x <- c(10,30,4,52,60,7,8,10,12,15,14,17,19,20,25,30) a) Which numbers in x are above 8 b) Which numbers are equal to 10. c) Which numbers are below 8 or above 30.

d) Can you create a matrix with numbers and characters. names <- c("Anne","Tom","Jamie","Max","Claire") ages <- c(12,16,25,34,28) cbind(names,ages) What happens if you try to use the ages?

e) Create a dataframe for five individuals (Andrew, Betty, Carl, Diane and Elisa) who are aged (62,80,24,40,56) and have gender (male, female, male,female, female).

f) Use evaluations and subsetting to find the characteristics of the individual who can claim their free bus pass (age 65+).

g) Create a variable in the dataframe called life expectancy, set this to 83 for females and 80 for males.

h) Create another variable called lyr (life years remaining) which is the number of years to life expectancy for each individual