# Introduction to R & RStudio

Dr Peter McKenna | p.mckenna@hw.ac.uk

29/11/2021

## Contents

# 1 Essentials

Before beginning today you must install two programs:

- `R`
- `RStudio`

CLICK HERE for installation instructions for both programs by operating system. Please follow the installation instructions carefully.

All the code from today's session is available on my personal Github: CLICK HERE TO ACCESS CODE.

In this tutorial you will read text and code chunks. Code chunks look like this, and contain the code for each of the operations we are going to perform today.

```
# this is what a code chunk looks like
```

# 2 Introduction to R and RStudio

- `R` is a programming language
- Powerful tool for performing mathematical and statistical operations on data
- Become popular in recent years for data science due to versatility
- Today I will scratch the surface of R's functionality

# 3 Today's intro content

- Get you a little familiar with operations of `base R`
- Walk through some programming examples
- Introduce you to the `tidyverse`: a popular data science ecosystem
- Perform data manipulation using the `tidyverse` functions

# 4 Introduction to R

- Let's start by opening a new project
- Projects act as directories, so all the RStudio files you create in one project remain part of that project
- Once you have saved you project in a memorable place open a new script (Ctrl + Shift + N)

## 4.1 Create subdirectories

- Before we get started let's create some folders in our directory

```
dir.create("data_raw")
dir.create("data_tidy")
dir.create("fig_output")
```

- To get started create a new project in RStudio and save to a meaningful location
- These folders will come in handy further into our exercises.

## 4.2 Install and load packages

- the following packages need to be installed an loaded
- R will ask if it can create a directory to store the packages - click `OK` to continue

```
# Install packages

install.packages('tidyverse') # install the tidyverse package(s)
```

- Do not be concerned by RStudio's request to store packages in a local library.
- This is so you can use these packages in future without the need to install.
- You may also see several warning messages about the package version or conflicts.
- You can ignore these so long as the package is successfully installed.

After a package is installed is must be loaded into RStudio for use.

```
# Load packages

library(tidyverse) # load the tidyverse
```

- I want to say a little about commenting here.
- Using the `#` key allows you to add inactive text to your script.
- So, you can add comments using `#` to keep a log of your activity

### 4.2.1 Running the Code

- To run the code hit `Ctrl` + `ENTER`.
- You can perform this action on multiple lines by highlighting them.
- Text will appear in the console section. This is fine, RStudio is just installing the package dependencies.
- Save the script (`Ctrl` + `S`) as **ddi-r-intro**.

## 4.3 Some Basics

```
# some simple operations

3+5
```

```r
12/7

x <- 5 # we use the assign operator "<-" to assign values to an object

7 -> x

x

# some simple mathematics

x <- 23

y <- 67

area_rect <- x*y

# BMI

height <- 1.8

weight <- 80

bmi <-
  (weight/(height*height))
```

## 4.4 Using functions

```r
# square root

sqrt(4)

a <- 4

sqrt(a)

# rounding numeric values

round(3.14159)

args(round)

help(round)

?round

round(3.14159, digits = 2)

round(3.14159, 2)

round(digits = 2, x = 3.14159)
```

```r
round(55.15, -2)

round(55.15, -3)
```

## 4.5 Vectors and data types

```r
# create vector for test score

test_scores <-
  c(3,5,2,8,7)

test_scores

# create vector for students

students <-
  c("Ronald", "Sophie", "Max", "Teresa", "Mandy")


# check length of test_score

length(test_scores)

length(students)

# check data type

class(test_scores)

class(students)

str(test_scores)

str(students)

# appending

students <-
  c(students, "Dennis") # add Dennis to the students vector

students
```

### 4.5.1 Mixed data types

- Let's create and examine an object with both numeric and character units

```r
test_answers <-
  c("a", 5, 8, TRUE, 3i)

typeof(test_answers)
```

```
class(test_answers)

str(test_answers)

test_answers2 <-
  c(FALSE, 4, 7, TRUE)

str(test_answers2)
```

### 4.5.2 Subsetting vectors

```
# subsetting from our vectors

test_answers

test_answers[2] # return 2nd element

test_answers[c(3,2)] # return elements in the order you specify

students[c(5,4,2)]

# what about elements that don't exist?

students[c(9,23,4,1)]
```

### 4.5.3 Conditional subsetting

- Let's amend the test_score vector slightly before we continue

```
# let's use the append the test_scores vector

test_scores <-
  c(test_scores, 10, 2, 1, 3, 6)
```

- Types of conditional subsetting

```
ls() # clear the consolse

test_scores

test_scores > 3 # boolean assessment

test_scores[test_scores > 3] # subsets the vector

# using boolean operators

test_scores[test_scores <= 2 | test_scores == 6] # `|` is the OR operator

test_scores[test_scores <= 3 & test_scores > 7] # `&` is the AND operator
```

```r
# boolean operators on character vector

students

students[students == "Max" | students == "Mandy"]

# using the %in% operator

students %in% c("Teresa", "Ronald")

students[students %in% c("Teresa", "Ronald")]
```

# 5 End of Intro to R

---

# 6 Introduction to `tidyverse`

- In this introductory class you are going to learn how to
  - read a .csv data file into R Studio
  - wrangle and explore the data using the *Tidyverse* functions
  - explore the data's distribution
  - generate plots

# 7 The `tidyverse`

- Today we are using `tidyverse` methods to explore and wrangle the data
- The Tidyverse is a set of R packages that allows for more user-friendly programming, relative to what's called *base R*
- For more about the Tidyverse see Hadley Wickham's free online text R for Data Science
- We are covering content from R for Data Science chapters 1,2,3 & 11.

## 7.1 Main verbs of R's Tidyverse

- The `tidyverse` packages allow us to use the following verbs:
  - `filter` : extract rows
  - `select` : extract columns
  - `pivot_wider` : spread rows to columns
  - `pivot_longer` : gather columns into rows
  - `mutate` : compute and append new/existing columns
  - `summarise` : summarise data based on stated criteria
- These verbs make the syntax of the previous operations (e.g., subsetting) easier to interpret

### 7.2 The Pipe operator

- You are going to see a lot of this symbol

```
%>%
```

- This is the *pipe* operator
- Do not fear the pipe operator
- It means "then do this"

## 8 Working with data objects

### 8.1 Download the data

```
# download data file from my personal github

download.file("https://github.com/petermckenna/ddi/blob/main/exp_data.csv", "data_raw/exp_data.csv", mo
```

### 8.2 Read in the data

- Now that we have downloaded the data we nened to read it in
- Today we are going to read in a .csv file
- RStudio can read in all data types with the package `readr` (e.g., .xls)

```
data <-                       # assign ( <- ) the label "data" to next set of commands
  read_csv("exp_data.csv") # read in csv file called "exp_data.csv"
```

### 8.3 Experiment data info

- `id` = participant number
- `like` = rating of assistants likeability from 1 = "Did not like at all" to 5 = "Liked it a lot"
- `voice_type` = speaker voice type including two levels: Female; Male.
- `conv_len` = duration of the conversation between the user and the smart speaker measured in sec

### 8.4 Inspecting the data

- Let's go through each of the commands below
- Checking data with these commands helps you to get a better understanding of the data
- `head` & `tail` for example help you quickly determine if the dataset has been read in properly from top to bottom

```
head(data)    # shows top 6 rows

tail(data)    # shows bottom 6 rows

dim(data)     # number of rows and columns
```

```
names(data)   # list vector names

# you can use square brackets to give an index to certain elements of the data

names(data)[2] # give the name of column 2

View(data)    # opens subset of the data in a new tab; note the capital 'V'

glimpse(data) # short by-vector summary
```

# 9 Data wrangling and manipulation with the `tidyverse`

## 9.1 Using `filter` to subset by row

```
# filtering by single element

data %>%
  filter(id == 1:5) # participants 1 to 5

data %>%
  filter(voice_type != "female") # "!=" denotes exclude

# filtering by multiple elements

data %>%
  filter(id == 1:5,
         voice_type == "female")
```

## 9.2 Using `mutate` to append object

```
# say we knew that participants 1:6 were female and the rest were male

# we can use an ifelse statement to create a new vector `gender` based on this criteria

data %>%
  mutate(gender = ifelse(id %in% 1:6,
                         "female",
                         "male"))
```

## 9.3 Using `select` to subset by column

```
data %>%
  select(id, like) # you can use a colon to select multiple vectors

# using do not select
```

```
data %>%
  select(-id) # everything except `id`

# you can also reorder existing cols with select

data %>%
  select(like, conv_len, id)

# or with vector numbers instead of names

data %>%
  select(4, 2, 3, 1)
```

## 10 Summarising data

- In research, we are interested in summary statistics
- I'll give you a brief overview of how to generate summaries using the `tidyverse`
- To do so, we use the functions `group_by` and `summarise` with other existing mathematical functions.

```
data %>%
  summarise(med_like = median(like)) # here we generate the median of `like` and call the summary "med_


data %>%
  summarise(med_rating = median(like),
            avg_conv = mean(conv_len),       # new vector `avg_conv` is the mean of conv_len
            sd_conv = sd(conv_len))          # new vector `sd_conv` is S.D. of conv_len

# grouping data

  data %>%                                 # create new object `bp` from data
  group_by(voice_type) %>%                 # group by `voice_type`
  summarise(avg_conv = mean(conv_len),     # new vector avg_conv is the mean of conv_len
            sd_conv = sd(conv_len))        # new vector sd_conv is the S.D. of conv_len
```

## 11 Plotting Data

- Let's make a plt of user conversation length based on the sex of the smart speaker voice.
- We use the `ggplot2` package functions to do so
- Note, this package does not support the `%>%` operator, using `+` to chain commands instead

```
data %>%
ggplot(., mapping = aes(x = voice_type,
                        y = conv_len,
                        fill = voice_type)) +  # notice mapping comes first here
  geom_bar(stat = "summary",
           fun.y = "mean") +                   # apply black outline to bars
  labs(title = "User Conversational Length by Agent Voice Type\n",
       x = "\nSmart Speaker Voice Type",
```

```
      y = "Conversation Length (seconds)\n") +
  theme_classic(base_size = 15) +                # use ggplot's calssic theme size 20 text
  theme(legend.position = "none")                # remove the legend - it's not useful here
```

## 12 Additional resources

R for Data Science – useful for learning *Tidyverse* syntax

R Graphics Cookbook, 2nd edition – very useful starting guide for ggplot2