Using **R** for Statistical Analysis

Michael T. Heaney
Lecturer in Politics and Research Methods
University of Glasgow
Web: www.michaeltheaney.com
Email: michaeltheaney@gmail.com

Prepared for:

Fairview International School
11 January 2024

Session Description

This session will orient students to using **R** for statistical analysis. They will become familiar with the **R** environment, including the functionality of base **R** and the installation/use of user-written packages. Students will learn to prepare a simple data set in a commonly available spreadsheet programme, such as Google Sheets, and then read and analyse the data in **R**. Analysis will include core descriptive statistics (e.g., mean, standard deviation) and ordinary least squares (OLS) regression. Extensions and applications will be discussed. Access this handout and other materials at www.michaeltheaney.com (under Teaching tab).

1. Motivation: Why learn about statistics?

- Statistics provides techniques to test the implications (hypotheses) of many scientific theories. If we have data relevant to a theory (or theories), we may be able to assess the degree to which the data provide support for (or against) the theory (or theories).

- Statistics provide a common language and methodology that is used widely across the natural sciences, social sciences, and many professional fields. Statistics may help professionals in diverse fields to communicate with one another better.

- Understanding statistics helps to read a wide range of materials critically, from textbooks to scholarly articles to newspapers. Understanding statistics makes it less likely that others can trick you with faulty or weak evidence.

- Statistics are much in demand for employment in many different areas.

- Statistics can be fun. (They can be frustrating too.)

2. Motivation: Why learn about R, specifically? What are some challenges of doing so?

- There are many computer programmes available for statistical analysis. **R** has some desirable features that make it worth learning.

- **R** is **freely** produced by a nonprofit organization, The R Project for Statistical Computing. Many other programmes are quite expensive, with licenses that may cost hundreds of pounds each. Download **R** for free to your computer using this link: https://www.r-project.org/

- **R** is widely used in both academic and professional communities.

- **R** provides a highly flexible framework in which almost all mathematical and statistical procedures can be performed.

- Base **R** – included with all downloads – makes it possible to execute many mathematical and statistical procedures.

- More advanced procedures are available using thousands of user-written packages that are available for free download. These packages make it possible to use cutting-edge statistical techniques. The downside is that most of these packages are poorly written and documented. It is advisable to identify packages that are backed by supportive communities of users. One example is the **tidyverse** suite of packages. Learn more here: https://www.tidyverse.org/

- Many books have been written on how to use **R**. They are available for free or for purchase, depending on your preference. For free: https://r4ds.hadley.nz/ For purchase: https://www.amazon.co.uk/Statistics-Using-R-Integrative-Approach/dp/1108719147/ref=sr_1_8?crid=2CFQ2Z6BXHHDK&keywords=statistics+using+r&qid=1704935641&sprefix=statistics+using+r%2Caps%2C84&sr=8-8

- It is possible to use R effectively without knowing much about computer programming or statistics. At the same time, more advanced applications benefit from understanding programming languages such as **C++** and advanced courses on statistics.

3. The basics of base R

- Please open **R** on your computer.

- Let's begin with some basic and commonly used commands in **R**.

- The combine function: c(element, element, . . ., element) creates a vector. For example:

```
c (9, 8, 7, 6, 5, 4, 3, 2, 1)
```

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

- It is possible to add vectors:

```
c(2,2,2) + c(2,2,2)
```

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

- Likewise it is possible to subtract, multiply, or divide vectors using -, *, and /.

```
c(6,6,6) / c(2,2,2)
```

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

- Other standard mathematical functions work as well. To calculate the log base 10 of 1000, type:

```
log(1000,10)
```

```
[1] 3
```

- To use an exponent, just use the carrot ^.

```
10^3
```

```
[1] 1000
```

- You can generate variables with the <- operator. It is roughly equivalent to the equals sign. It is pronounced "gets".
- Note that it works in either direction: -> or <-
- If you wish to generate a vector x=1, then:

```
x <- 1
```

- To check the value of x, just type x:

```
x
```

```
[1] 1
```

- Or, alternatively:

```
1 -> x
```

- It is possible to ask R a true/false question about your vector by making a logical statement.

```
x>1
```

```
[1] FALSE
```

```
x>0
```

```
[1] TRUE
```

```
x==0
```

```
[1] FALSE
```

```
x!=0
```

```
[1] TRUE
```

- It is possible to combine variables in a vector. For example:

```
five <- 5

six <- 6

seven <- 7

eight <- 8

nine <- 9

five

[1] 5

six

[1] 6

seven

[1] 7

eight

[1] 8

nine

[1] 9

count_five_to_nine <- c(five, six, seven, eight, nine)

count_five_to_nine

[1] 5 6 7 8 9
```

- If we want, we can draw some subset of elements from this vector.
- In order to draw the first three elements from the vector, use square brackets to the right of the vector:

```
count_five_to_nine[1:3]

[1] 5 6 7
```

- Or, we could draw the last three elements from the vector:

```
count_five_to_nine[3:5]
```

```
[1] 7 8 9
```

- Or, we could ask for all of the elements of the vector:

```
count_five_to_nine[1:5]
```

```
[1] 5 6 7 8 9
```

- Which is equivalent to simply typing:

```
count_five_to_nine
```

```
[1] 5 6 7 8 9
```

- Since we've made a few different objects, you may want to look to see a list of the objects that we've created. To do so, use the list command: ls()

```
ls()
```

```
[1] "count_five_to_nine" "eight"        "five"        "nine"        "seven"        "six"
"x"
```

- If you want to delete an object, use the remove command: rm()

- Let's remove x.

```
rm("x")
```

- Now repeat the ls() command to see that it was, in fact, removed.

```
ls()
```

```
[1] "count_five_to_nine" "eight"        "five"        "nine"        "seven"        "six"
```

- If you need help, it is easy to get it: help(topic)

```
help(c)
```

- To save a specific object, use the "save" command. For example, to save the object "five":

```
save(five, file="five.RData")
```

- To save your entire workspace, use the save.image command:

```
save.image(file= "Entire.RData")
```

- When you're done, quit: q()

- You can reload your old workspace by using the load command.

```
load(file.choose())
```

4. Loading a simple data set

- The following data have been fabricated. They are not real data.

case	firvar	secvar	thivar
1	22	1.355	0.009
2	45	1.241	0.001
3	11	1.999	0.023
4	31	1.345	0.334
5	5	1.227	0.003
6	66	1.789	0.066
7	20	1.111	0.887
8	10	1.357	0.721
9	52	1.823	0.228
10	99	1.001	0.611

- Please open Google Sheets on your computer.
- Create a new file called: **testfile**
- Enter the data into the sheet as it appears above.
- Download it to your computer as a **.csv** file.
- Edit the name of the file so that it just reads **testfile.csv**
- Now, prepare **R** to read and store your file by creating a working directory:

```
setwd("C://Your_Directory_Name")
```

- To make sure that worked, use the get working directory command:

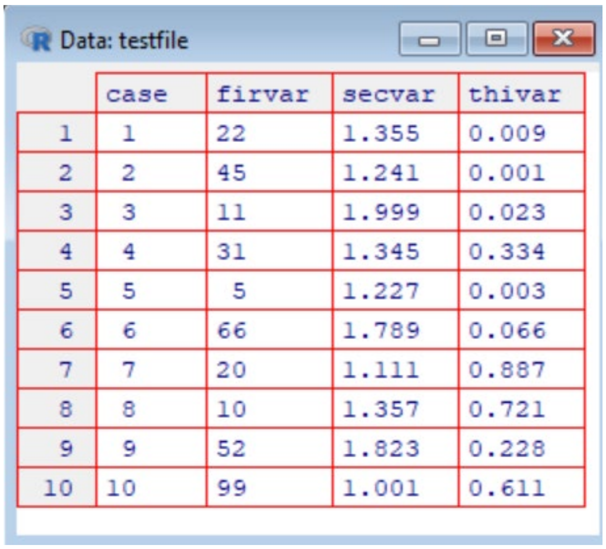
```
getwd()
```

- Now read the data into R:

```
testfile <- read.csv(“testfile.csv”, header=TRUE)
```

- To take a quick look at the file, use the **View** command.

```
View(testfile)
```



	case	firvar	secvar	thivar
1	1	22	1.355	0.009
2	2	45	1.241	0.001
3	3	11	1.999	0.023
4	4	31	1.345	0.334
5	5	5	1.227	0.003
6	6	66	1.789	0.066
7	7	20	1.111	0.887
8	8	10	1.357	0.721
9	9	52	1.823	0.228
10	10	99	1.001	0.611

- Descriptive statistics can easily be calculated using the **summary** command.

summary(testfile)

```
      case            firstvar            secvar             thivar
 Min.   : 1.00    Min.   : 5.00    Min.   :1.001    Min.   :0.0010
 1st Qu.: 3.25    1st Qu.:13.25    1st Qu.:1.230    1st Qu.:0.0125
 Median : 5.50    Median :26.50    Median :1.350    Median :0.1470
 Mean   : 5.50    Mean   :36.10    Mean   :1.425    Mean   :0.2883
 3rd Qu.: 7.75    3rd Qu.:50.25    3rd Qu.:1.681    3rd Qu.:0.5417
 Max.   :10.00    Max.   :99.00    Max.   :1.999    Max.   :0.8870
```

- Of course, these results are meaningless because the data are fabricated.

5. Examining a real data set

- Do mice exhibit empathy?

- Dale J. Langford, Sara E. Crager, Zarrar Shehzad, Shad B. Smith, Susana G. Sotocinal, Jeremy S. Levenstadt, Mona Lisa Chanda, Daniel J. Levitin, and Jeffrey S. Mogil, "Social Modulation of Pain as Evidence for Empathy in Mice," *Science* 312 (30 June 2006): 1967-1970.

- Please download the following data set and call it **mice.csv** from www.michaeltheaney.com (see Teaching tab):

case	painbehaviour	bothinpain
1	36.7	1
2	81.1	1
3	66.7	1
4	66.7	1
5	44.4	1
6	54.4	1
7	63.3	1
8	62.2	1

9	58.9	1
10	50	1
11	54.4	1
12	57.8	1
13	56.7	0
14	51.1	0
15	50	0
16	51.1	0
17	44.4	0
18	2.2	0
19	41.1	0
20	33.3	0
21	25.6	0
22	22.2	0
23	14.4	0
24	3.3	0
25	64.4	0

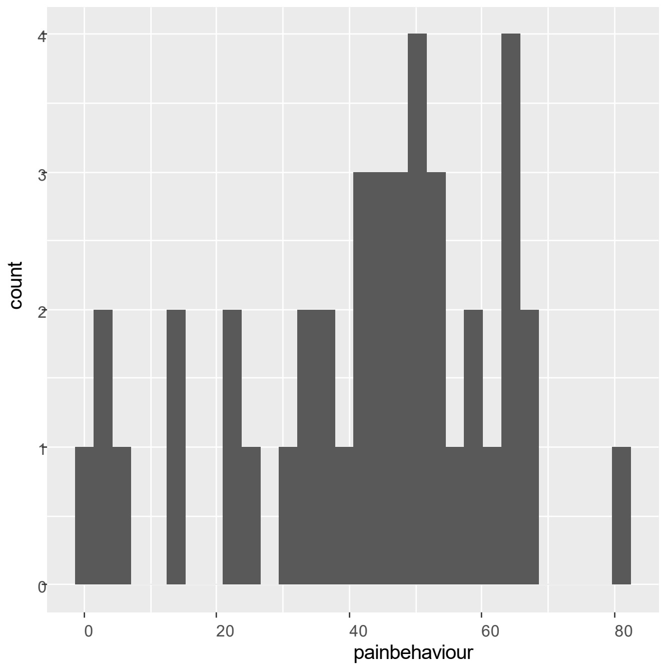
```
mice <- read.csv("mice.csv", header=TRUE)
```

- What is the distribution of pain behaviour in the data?
- Download **ggplot2**, a **tidyverse** package.

```
install.packages(ggplot2)
```

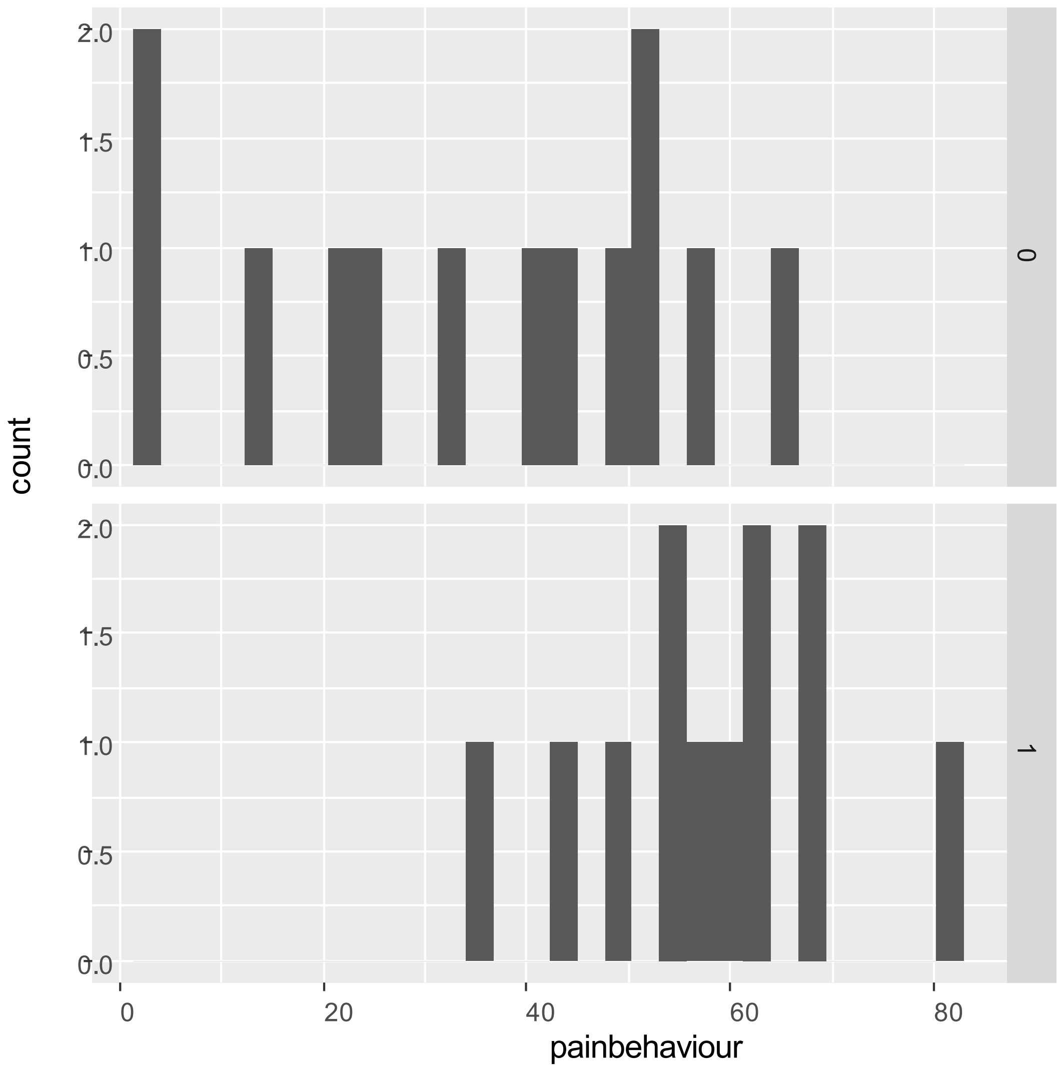
```
library(ggplot2)
```

```
ggplot(data = mice, aes(x=painbehaviour)) + geom_histogram()
```



- Is there a difference between the two groups of mice?

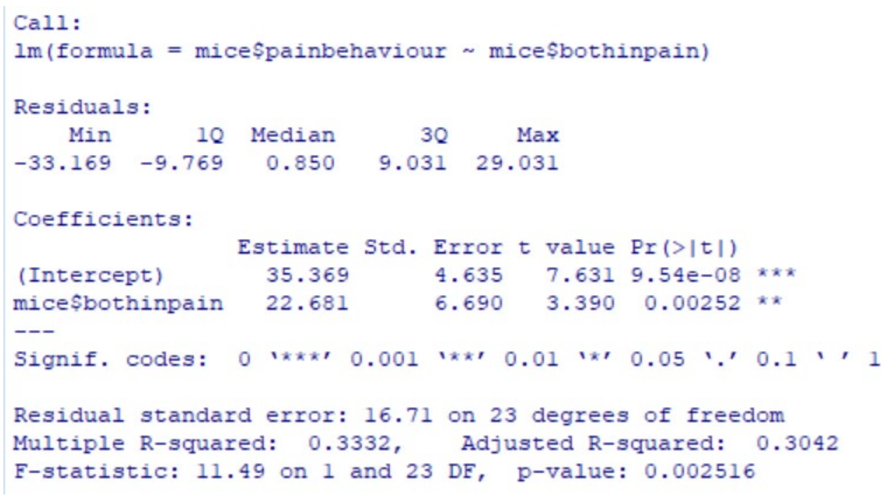
```
ggplot(data = mice, aes(x=painbehaviour)) + geom_histogram() +
facet_grid(mice$bothinpain)
```



- Learn to refine these plots using **help(ggplot2)**.
- It appears that a mouse expresses more pain when their cellmate also expresses pain (condition=1) than when their cellmate is not similarly situated(condition=0).
- Nevertheless, it is important to test this difference to see if it is statistically significant.
- A **t-test** is one way to do that.

```
Model_01 <- lm(mice$painbehaviour ~ mice$bothinpain)

summary(Model_01)
```



```
Call:
lm(formula = mice$painbehaviour ~ mice$bothinpain)

Residuals:
    Min      1Q  Median      3Q     Max 
-33.169  -9.769   0.850   9.031  29.031 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)       35.369      4.635   7.631 9.54e-08 ***
mice$bothinpain   22.681      6.690   3.390  0.00252 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 16.71 on 23 degrees of freedom
Multiple R-squared:  0.3332,	Adjusted R-squared:  0.3042 
F-statistic: 11.49 on 1 and 23 DF,  p-value: 0.002516
```

- The t-statistic on bothinpain is **t=3.290**, which is statistically significant with **p ≤ 0.05**. Thus, the data support the hypothesis that the “both in pain” condition yields greater displays of pain than when the cellmate is not also in pain. The researchers claim that this is evidence of empathetic behaviour in mice.

6. Examining a data set with multiple independent variables.

- Consider the following data set: Centellas, Miguel, 2017, "Global Indicators 2015 Dataset (Cross-Sectional)", https://doi.org/10.7910/DVN/ZN6MWY, Harvard Dataverse, V1, UNF:6:uK57C2iuZxQb7GrduW3EAA== [fileUNF]

- Could we develop a linear model to explain one of these indicators (**Y**) as a function of several independent variables (**X**s)?

- Access the data and codebook at www.michaeltheaney.com (under Teaching tab).

- First, read the data using **read.csv**.

- Then, develop a model using the following command:

```
Model_02 <- lm(Y ~ X1 + X2 + . . . + Xk)
```

- Think about: What do you want to explain (**Y**)? What factors will do the explaining (**X**s)?

- Work in teams to develop several models. Use a script file to save your programming.

- Which is most compelling to you and why?