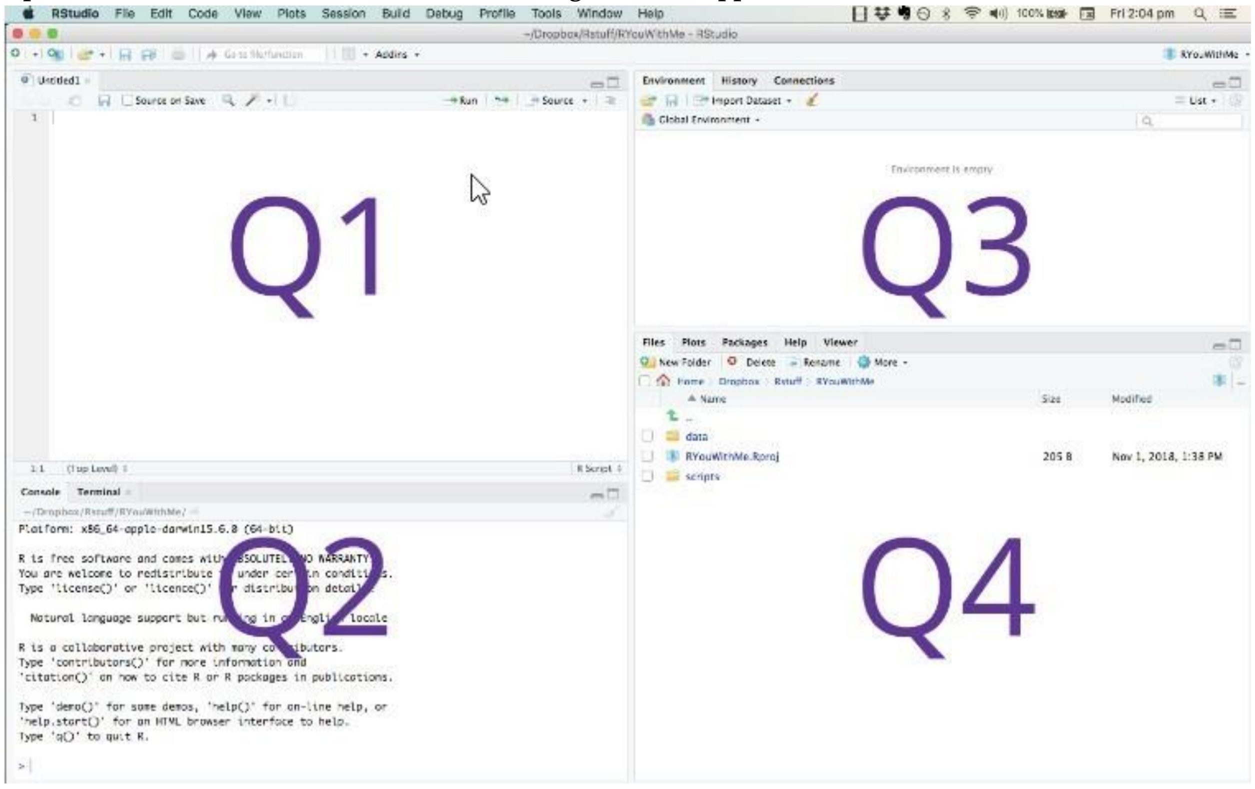
Practical 3 – Introduction to R (Lab Version)

If you are waiting for the lab to start – kill some time by filling in our fun anonymous questionnaire which hopefully we will be able to run some statistical tests on later in the practical. It is at:

https://forms.office.com/e/UCWxNfT11s

Although you will be able to look through the data that last's years students left when they did some answers to the questionnaire here. However, we will go into the data in much greater depth next week and it will be more interesting if we can use the data that this year's cohort leave.

Open RStudio – this should make the following interface appear



RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
~/Dropbox/Rstuff/RYouWithMe - RStudio
Untitled1
Source on Save
Run
Source
Q1
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Q3
Console Terminal
~/Dropbox/Rstuff/RYouWithMe/
Q2
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home Dropbox Rstuff RYouWithMe
Name Size Modified
data
RYouWithMe.Rproj 205 B Nov 1, 2018, 1:38 PM
scripts
Q4

```
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Q1 - contains: ***script, data, command to run script***
Your written R Scripts will appear here, but also data (for instance a spreadsheet you might have imported) – also any variables can be inspected here. Data
variables and code will appear in their separate tabs
Q2 - contains: ***console***
This is where you can run commands. In the tab "console" you can run R commands over any data imported or variables created, in the "Terminal" console you can make operating system commands
Q3 - contains: ***environment***
Essentially this is a visual representation of all the current variables available in the session, clicking on any one of them will make that variable appear in Q1

Q4 - contains: ***files, plots, packages, help as tabs***
Files just shows the files in a folder, plots displays any graphs generated by your code (unless specifically sent to a file as a graphic or a pdf), and packages shows the packages currently being used

---

Part 1: Using the Console

At this point you should start by just playing with R commands in the Q2 quadrant

After R is started, there is a console awaiting for input. At the prompt (>), you can enter numbers and perform calculations.

```
> 1 + 2
[1] 3
```

Task 1:

Try to do some simple calculations here. For instance convert pounds to dollars or rupees. Convert today's temperature into Fahrenheit.

Variables

We assign values to variables with the assignment operator "=", however, in R, a more common assignment operator is "<-"
Just typing the variable by itself at the prompt will print out the value. We should note that another form of assignment operator "<-" is also in use.

```
> x = 1
> x
[1] 1
```

Task 2

See if you can convert the weight 15st 7lbs into kg. You will need to do two calculations. Convert stones to pounds (multiply by 14) then add the pounds, then turn it into kilograms by dividing by 2.2. See if you can do that. You might need to create some variables (e.g stones, pounds) to be able to do this.

Functions and Data Types

R functions are invoked by its name, then followed by the parenthesis, and zero or more arguments. Lets start off with a very basic but extremely powerful function c(). It will convert a set of values (separated by commas) into a data type called a vector. A vector, in other programming languages would be called an "array" – that is to say, a set of variables of the same data type.

```
> c(1, 2, 3)
```

```
[1] 1 2 3
```

But here is where things get interesting. Lets convert this into a variable for instance

```
>myvector<-c(1,2,3)
```

But see what happens if you multiply that by 2 by typing:

```
> myvector*2
```

What do you see? Essentially each item in the vector is multiplied by 2! This is quite different from what you might expect in other programming languages. For instance in javascript – to do that you would need to write

```
let myvector=[1,2,3];
for (let i=0;i<myvector.length;i++) console.log(myvector[i]*2);
```

Here you can see what makes R so powerful, but also, a little opaque. When an operation is applied to an array in other languages – normally some kind of loop syntax is required. , whenever that operation is applied.

Now lets try the function paste() which is used in R to concatenate strings. Try

```
> paste("a", "b")
```

The output gives us:

```
[1] "a b"
```

Nothing very surprising there. But now try this

```
> paste(myvector, "ok")

[1] "1 ok" "2 ok" "3 ok"
```

Again, the function is run over every element in the vector individually.

Task 3

Ask (at least) two people sitting next to you their names. Then create a vector of their names. Then use the paste function to say “Hi!” to all those people

If you are finding this a bit strange– don’t worry – that is R – it is a strange language, even though it is fantastically powerful – and for statistics and data science there is nothing more powerful. But if you are thinking at this point: “But I am doing cybersecurity – what use is this to me?” – I will answer –

- do you not think security teams make statistics of DDoS attacks?

- Are not the number of requests calculated on a time axis – will it not be important to plot these as graphs?

All of this will need to be done with some understanding of statistics – and while there are other routes, I would suggest that your ability to represent them will be massively enhanced by knowledge of R.

Task 4

However, the ultimate goal of R is to make sense of data – to give it meaning. Now let us try a more real world objective. Choose your favorite sport – and your favorite player or team in it. If its football, find how many goals your team has scored in the last 5 games. If it is cricket, how many runs a team or player has scored in their last 5 innings. Then turn it into a vector – for instance

```
goals<-c(0,2,1,3,2)
```

or

```
runs<-c(22,44,0,106,45)
```

Then run the summary function over it. What you should get is something like this:

```
summary(goals)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0     1.0    2.0     1.6   2.0        3.0
```

What was the average score, in your case? What was the maximum, and what was the minimum?

We will cover the meanings of 1st Qu and 3rd Qu when we have done a a bit more statistical theory.

As an aside at this point, if you need to access individual elements of a vector – you can use the traditional square bracket notation. In the example above we could use *goals*[1] to get the first element of the *goals* vector. Note this is also, a change from most other programming languages which zero index their arrays (that is to say, they start at 0). R always uses a 1 to reference the 1st element of a vector.

However, rather than summarize, we might wish to get these results separately. Here we can introduce the functions mean() for the the average, median(), in very basic terms to get the middle of the distribution, and sd(), to get the standard deviation, the best known measure of the variability of a distribution. (For instance, student heights will have a very low standard deviation because it is unlikely that we find anyone below 1.2 m or anyone above 2 m. However, the standard deviation in marks at the end of the course might vary a lot – some people might score very highly, others very low, and so this will have a much wider standard deviation).

Task 5

Get the average (mean), the median and the standard deviation (sd) of the vector of goals or runs that you did in task 4

The other data structures we might wish to look at are the ***matrix***, the ***list*** and the ***dataframe***. Of these the most relevant for us is the dataframe – and it is the one into which much of the data we look at will be put.

To help us get started with data frames, the best way might be to open a built-in dataframe in R which is **mtcars**

To see what's in mtcars, just type that name at the prompt

```
>mtcars
```

Here you will see all the data in that dataset. However to see it more easily, type in **head(mtcars)** or **tail(mtcars)** – this will get you the first and last 5 rows in the dataframe respectively. But you can also run the summary function on this data too. If you run that, you will see the summary of the data on a column by column basis. But you can also query the data on a column by column basis. You can see the columnames by typing the command **names(mtcars)** – however you can query individual columns by using the dollar notation. For instance type:

```
>mtcars$mpg
```

This then reads all the values in the mpg column in the mtcars data frame as a vector. Moreover, you could find the average mpg of all the cars in this sample by typing

```
>mean(mtcars$mpg)
```

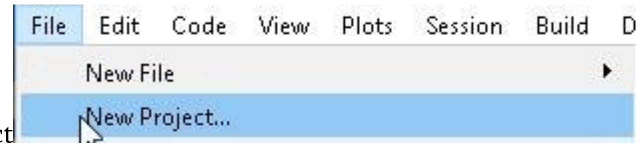
Task 6

1. Get the mean horse power of the cars
2. Find the standard deviation of horse power in the distribution
3. What is the highest horse power in the distribution
4. Do a summary of mtcars

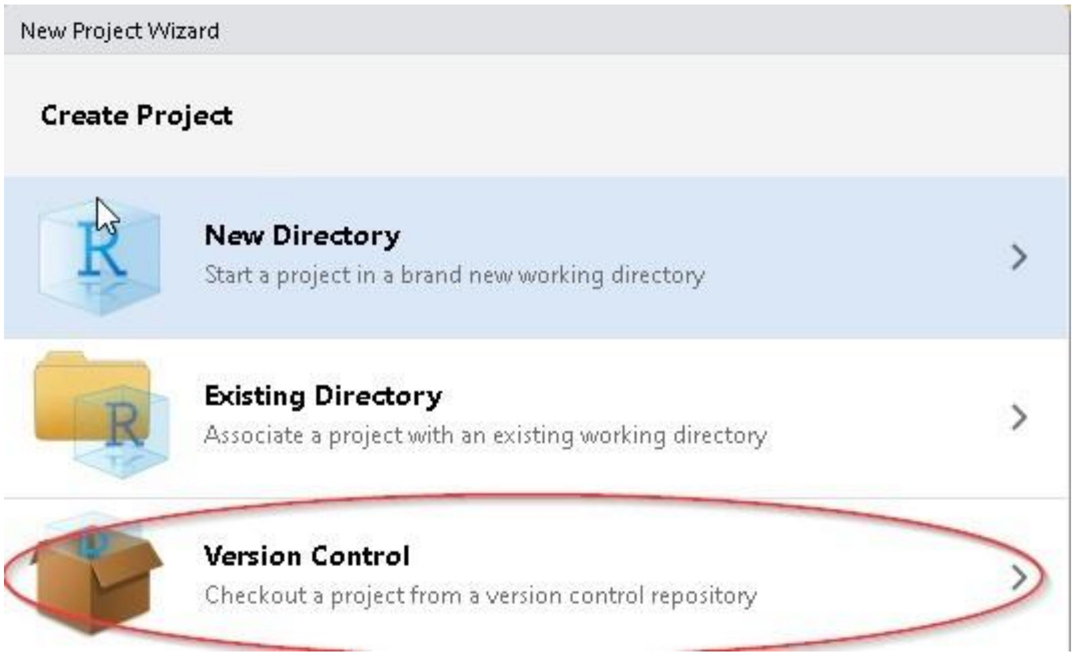
Task 7

Using R Studio with GIT

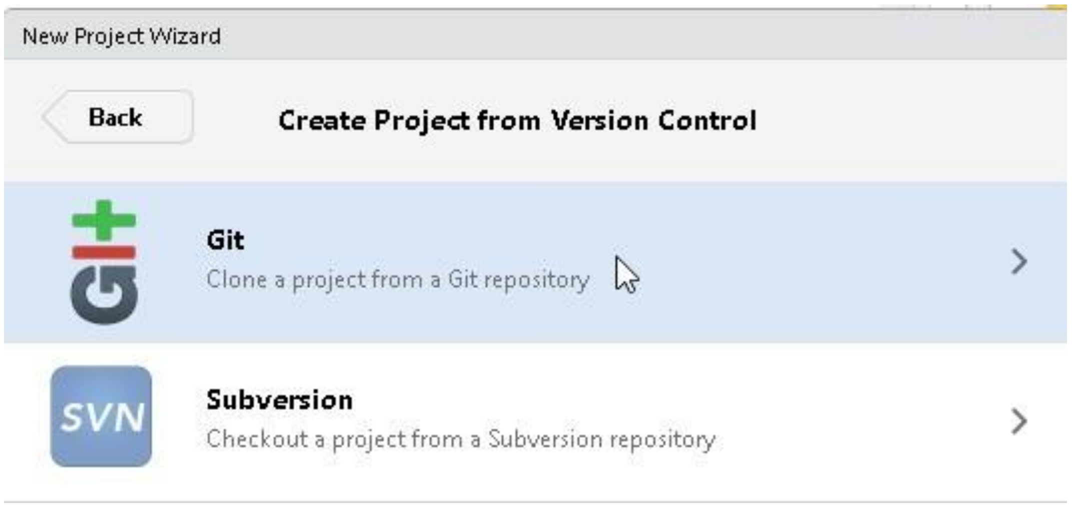
1. Next I would like you to combine R Studio with GIT. So go to github and create a new repository, and after you have done so, and added your git developer token to the clone url (you can temporarily do all that in notepad or TextEdit) - then do the following.



2. Click on File-> New Project
3. Then choose "Version Control"


New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository

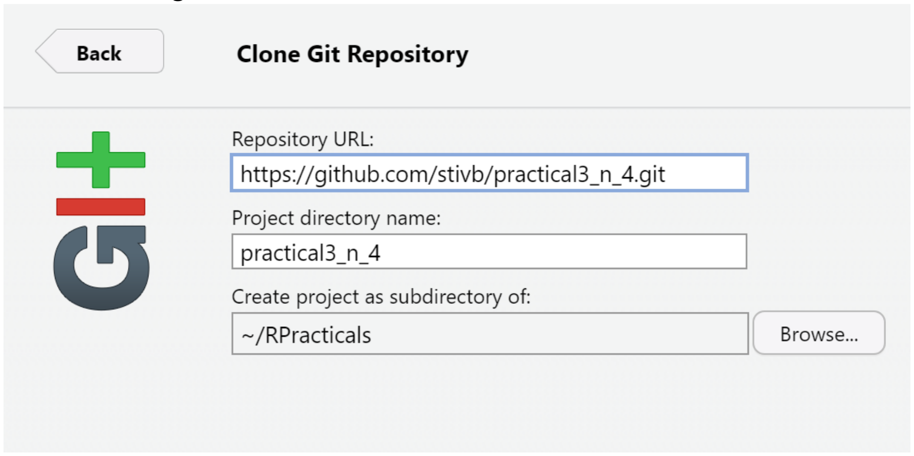
4. Then choose GIT


New Project Wizard
Back
Create Project from Version Control
Git
Clone a project from a Git repository
Subversion
Checkout a project from a Subversion repository

5. Then in the repository url put this git url from the clone command (without git clone at the beginning).
https://github.com/stivb/practical3_n_4.git

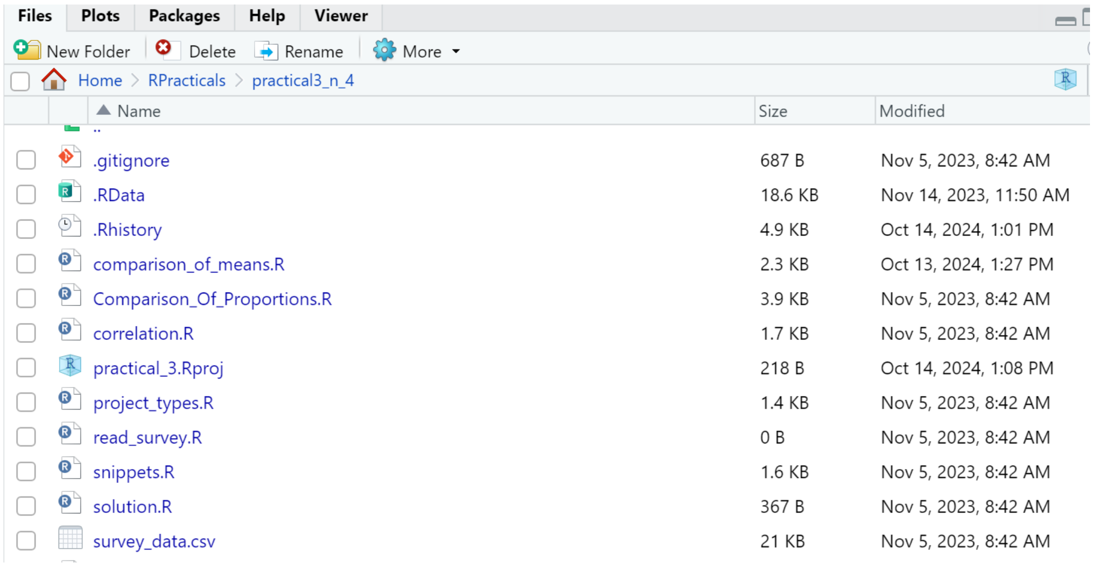
6. In the directory name, you can choose the name you want for this folder (rather than having it automatically named which is what happens when you do a simple git clone command) and finally say where you want this repo to be. (Under which subdirectory – yours does not need to be the same as mine).

7. This is a public project so you should not need a password
8. When the project opens, go into the files area. (In the Q4 Window). It should look like somethingn like this:



Repository URL:
https://github.com/stivb/practical3_n_4.git
Project directory name:
practical3_n_4
Create project as subdirectory of:
~/RPracticals

9. After having done that, in the files section in the bottom right quadrant you should see the following



Home > RPracticals > practical3_n_4

Name	Size	Modified
..		
.gitignore	687 B	Nov 5, 2023, 8:42 AM
.RData	18.6 KB	Nov 14, 2023, 11:50 AM
.Rhistory	4.9 KB	Oct 14, 2024, 1:01 PM
comparison_of_means.R	2.3 KB	Oct 13, 2024, 1:27 PM
Comparison_Of_Proportions.R	3.9 KB	Nov 5, 2023, 8:42 AM
correlation.R	1.7 KB	Nov 5, 2023, 8:42 AM
practical_3.Rproj	218 B	Oct 14, 2024, 1:08 PM
project_types.R	1.4 KB	Nov 5, 2023, 8:42 AM
read_survey.R	0 B	Nov 5, 2023, 8:42 AM
snippets.R	1.6 KB	Nov 5, 2023, 8:42 AM
solution.R	367 B	Nov 5, 2023, 8:42 AM
survey_data.csv	21 KB	Nov 5, 2023, 8:42 AM

10. Click on **project_types.R** - this uses the built-in dataset mtcars to demonstrate research questions that you might want to use in your group projects

[redacted] A nice feature of R Studio is that you can run single lines in an R script or groups of lines without running the whole script. I have grouped the lines into different colors below. Select one by one, the yellow, green, cyan, magenta, blue, red, dark yellow, grey, dark cyan, dark green and black groups of lines and see what effect they have when you choose to run them.

```
#histogram to show normalcy of data
h <-hist(mtcars$mpg, breaks = 20, xlab = "Miles per Gallon", ylab = "Frequency", main = "Histogram of MPG Values")
# Add a normal distribution line
x <- seq(min(mtcars$mpg), max(mtcars$mpg), length = 100)
y <- dnorm(x, mean = mean(mtcars$mpg), sd = sd(mtcars$mpg)) * length(mtcars$mpg)
box.size <- diff(h$mids[1:2])
y <- y * box.size
lines(x, y, col = "red")

#comparison of means
t.test(mtcars$mpg ~ mtcars$am)# for normal data
wilcox.test(mtcars$mpg ~ mtcars$am # for non parametric data
boxplot(mtcars$mpg ~ mtcars$am, xlab = "Gear", ylab = "Miles per Gallon", main = "Boxplot of MPG by Gear")

#comparison of proportion
pt <- table(mtcars$cyl,mtcars$am)
chisq.test(pt)
percentages <- prop.table(pt, margin=2) * 100
barplot(percentages, col = c("red", "blue", "yellow"), xlab = "Manual 0 vs Automatic 1", ylab = "Percentage", main = "Stacked Bar Chart of Cylinders by Transmission Type", ylim = c(0, 100), legend.text = c("4", "6", "8"), args.legend = list(x = "topright"))

#correlation
cor.test(mtcars$mpg, mtcars$hp)
cor.test(mtcars$mpg, mtcars$hp, method="spearman")
plot(mtcars$hp, mtcars$mpg, xlab = "Horsepower", ylab = "Miles per Gallon", main = "Scatterplot of MPG vs. Horsepower")
# Add a trend line
abline(lm(mpg ~ hp, data = mtcars), col = "red")
```

These demonstrate skills that you might need depending on your project.
Histograms to check if you data is normal or non normal.
Then various tests:

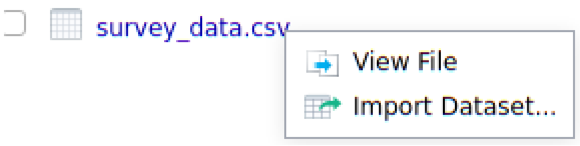
Test	Goal	Data Expected
T-Test	Comparison of Means	Normal Data
Wilcox Test	Comparison of Means/Medians	Non-Parametric Data

Chi Squared Test	Comparison of Proportions	A table of counts (or *contingency table)*
Correlation Test (Pearson)	Correlation	Normal Data
Correlation Test (Spearman)	Correlation	Non-Parametric Data

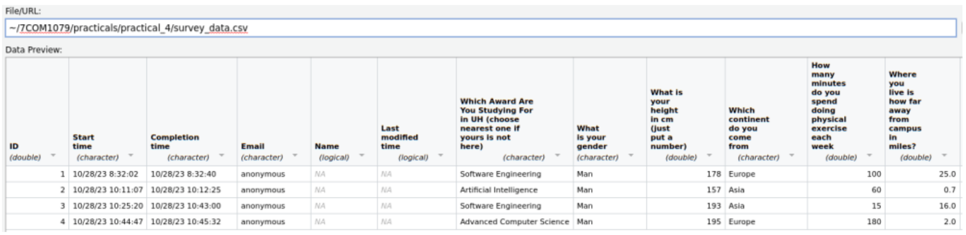
Then various chart

Goal	Visualisation
Comparison of Means	Boxplot
Comparison of Proportions	Stacked Barchart
Correlation	Scatterplot

If you think you have understood this, then try to look at the fun survey data set up for last year's students the form in the file survey_data.csv. Click on that and choose "Import Data Set"



You should then see something like this:



File/URL:
~/7COM1079/practicals/practical_4/survey_data.csv

Data Preview:

ID (double)	Start time (character)	Completion time (character)	Email (character)	Name (logical)	Last modified time (logical)	Which Award Are You Studying For in UH (choose nearest one if yours is not here) (character)	What is your gender (character)	What is your height in cm (just put a number) (double)	Which continent do you come from (character)	How many minutes do you spend doing physical exercise each week (double)	Where you live is how far away from campus in miles? (double)
1	10/28/23 8:32:02	10/28/23 8:32:40	anonymous	NA	NA	Software Engineering	Man	178	Europe	100	25.0
2	10/28/23 10:11:07	10/28/23 10:12:25	anonymous	NA	NA	Artificial Intelligence	Man	157	Asia	60	0.7
3	10/28/23 10:25:20	10/28/23 10:43:00	anonymous	NA	NA	Software Engineering	Man	193	Asia	15	16.0
4	10/28/23 10:44:47	10/28/23 10:45:32	anonymous	NA	NA	Advanced Computer Science	Man	195	Europe	180	2.0

Then click on the Import button, and then the data can be viewed in R Studio.

To start with, just run `summary(survey_data)` to see what kind of overall results you get.

Then have a think about how you might find out the following research questions:

RQ	RQ TYPE

Is there a difference in average height between men and women on the course?	Comparison of means (normal data)
Is there are difference in average amount of travelling done by men and women on the course	Comparison of means (probably non-parametric data)
Is there a difference in the proportion of men vs women between cybersecurity majors and artificial intelligence majors (or other awards)	Comparison of proportions
Is there a correlation between the amount spent on transport and distance from the university	Correlation (probably non parametric data)
Is there a correlation between students height and the amount they spend on food	Correlation (probably normal)

For each example, which is the dependent and which is the independent variable? What is the result in each case? Is the result robust in each case? And what does it look like in a graph? We might not be able to answer all this in this practical but in the coming weeks hopefully we will.