Practical 1: Differences between groups - ANOVA

Introduction

In this practical you will examine a dataset, erect a hypothesis and analyse data to test your hypothesis. First you will develop some hypotheses and think about how to test these hypotheses. You will then use **R** to test your hypothesis, explore and analyse the data.

The Dataset

The dataset we will be using initially is Doughnuts.xls. This dataset contains information about the amount of fat that doughnuts contain from 4 different outlets Examine the dataset critically – what kind of variables does it contain?.

Testing your hypotheses with R

Quick R tips

1. There is no need to remember R commands by heart. Remember you can copy code from this handout and paste it directly into your script file in R Studio, so you don't need to type everything out. You can also search for example code snipets and use those as a base. Try out ChatGPT too but remember all the caveats we discussed in lecture 1.

2. You cannot break R! If you want to know what something does, type it in and press enter. If you get an error it doesn't matter, just try again.

3. R is case sensitive. So be careful about capitalising words and watch carefully for typos, and missing brackets or punctuation. This includes the names of the files you're trying to import.

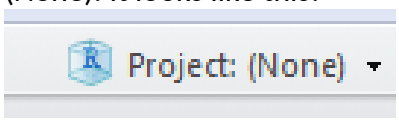
4. You can use the up and down arrows on your keyboard to toggle through things you have previously typed into R at the console. This is often quicker than typing everything in again.
5. The # character (alt+3 on a Mac keyboards) allows you to create a comment to annotate your script file i.e. explain what you are doing but you don't actually want R to read it. Any line or text starting with # will not be processed by R as a command.

1. Setting up and inputting data into R

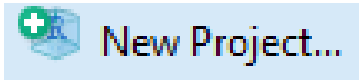
To work with data files, R needs to know where your file is. By default, it will look in the same folder as the one that the script is running in – if your file is not there it will give an error. This means that we need to keep track of where all the data files and R projects are located on your computer. This will also allow you to find things again later.

We begin by creating a folder in which you will keep all your statistical information. Start by making a folder on your desktop (or elsewhere but remember where) called "DataHandling". This will be where all of your analyses live.
Then, you need to open R Studio, which will open R. You might already have this from previous modulest, but if you don't, please go to https://www.rstudio.com/products/rstudio/download/ and follow the instructions on screen appropriate to your operating system. If you already have R Studio installed, please update it before we start.
If you're working in a lab PC the first time you use it you will need to install R Studio using MyTrinity Apps https://www.tcd.ie/itservices/our-services/mytrinityapps/
NOTE!!! If you have a Chromebook or are having trouble installing Rstudio, you can use R online via https://posit.cloud/ The free plan gives you 25 hours per months, which should be more than enough for this module. The interface looks exactly the same as the desktop version. You will however need to upload your data files to the web via the upload button in the bottom right panel.
To open R Studio, all you need to do is double click the R Studio icon.

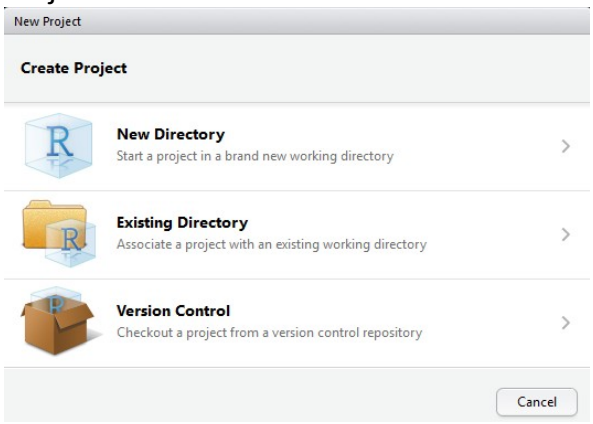
To begin, we will set up an R Project. A "project" is a group of documents related together, which includes the initial data, the Script file (which we will generate below), and all the information to link them together. You can share projects between people you collaborate with, and they should work for everyone the same way! On the right-hand side of R Studio, you should see a button that says Project: (None). It looks like this!



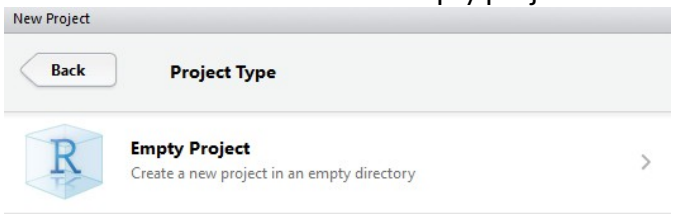
You'll need to click this button and then click New Project, which looks like this:



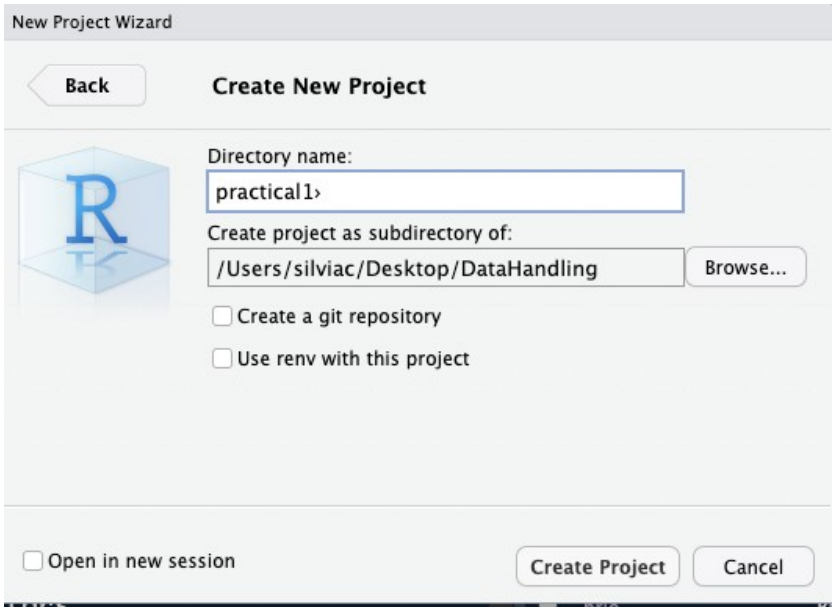
You'll see a new menu, and you should click "New Directory". This will put your R Project into a new folder!

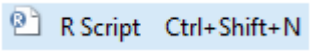


You will then want to start an empty project.

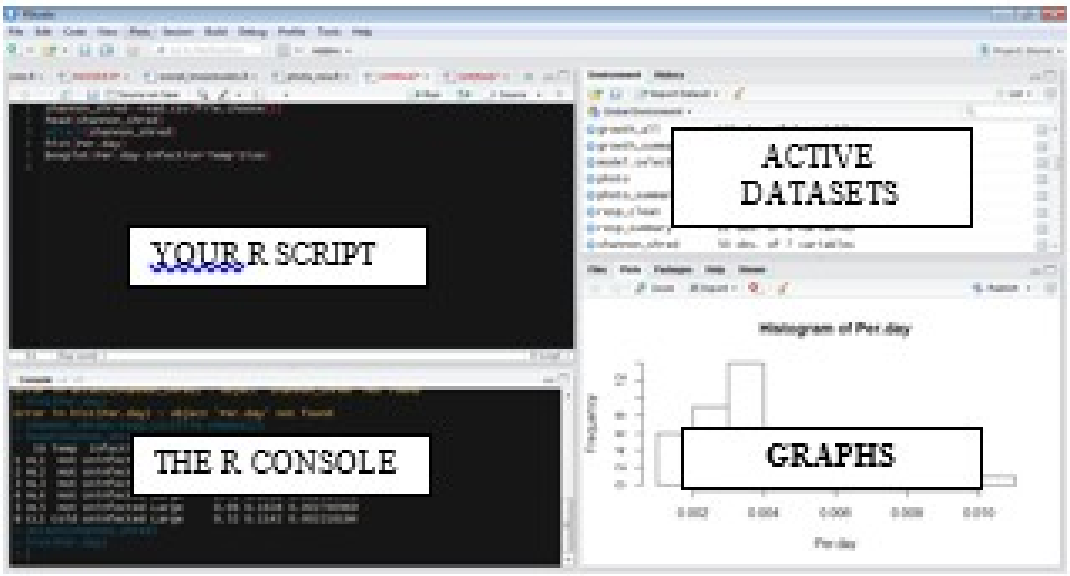


Finally, you can make the R Project. You then tell R where to create the new folder for this analysis, which is in the "DataHandling" folder you made at the start. You can use the "Browse" button to navigate to the folder on your desktop and click into it.



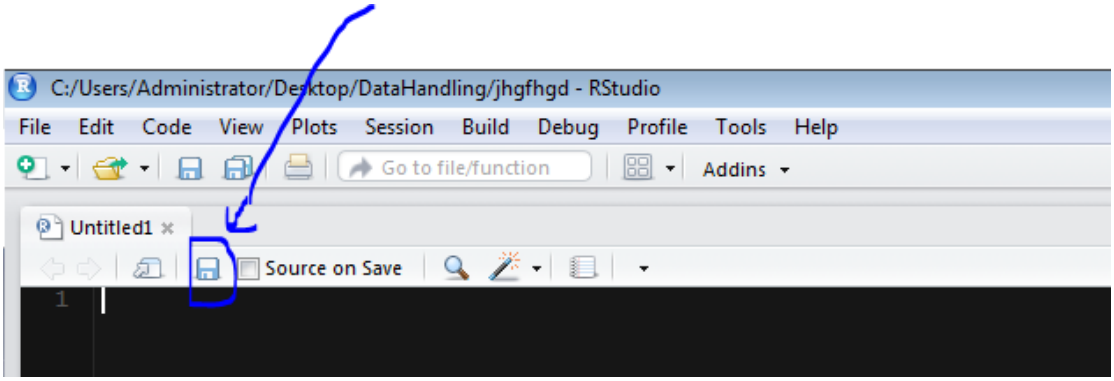
R Studio opens with three windows, but we'd like there to be four. PLEASE click the new file button () and then click "R Script" which looks like this (

). This will give you four windows which will look somewhat like the below image.



R is a command line programme which means you tell it what to do by typing commands (or code) into it. To use it you can type commands into the R script, or copy and paste the commands from this file. You then need to run these commands for the program to execute them.

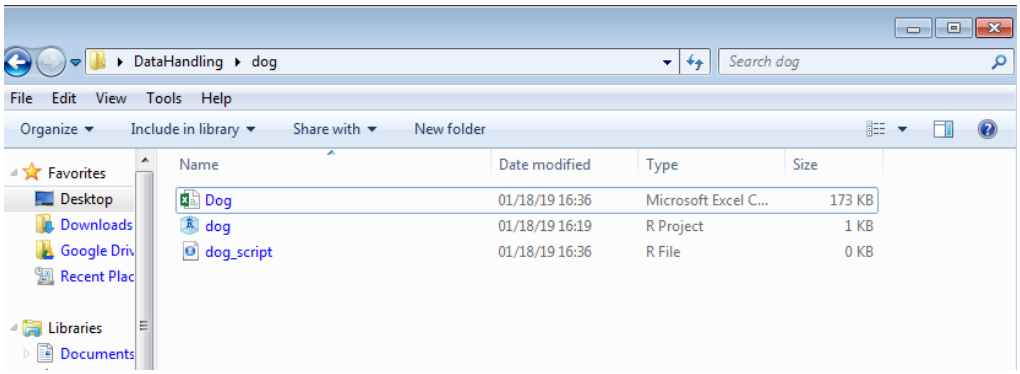
Save your Script file frequently, it's a valuable resource and means that you can reuse bits of code you've used before. Click the save icon above the script, and save the script as something obvious, e.g. doughnuts_script.



You have an Excel file called Doughnuts.xls. You now need to save this file in a different format and place in order to use it in R. Open the dataset and save it as a .csv files ("comma separated values"). You now have your original Excel file and a new file saved in a different format.

You will want to then place the Doughnuts.csv file into your practical1 folder (within your DataHandling folder) next. You can drag and drop, or copy and paste, the file into the folder where you have your R Project. Based on the example above, I put mine into the folder: C:\Users\[username]\Desktop\DataHandling\practical1

At this point, you are ready to go! You should have three things in your folder:
1. A place to put it all: An R Project File called practical1
2. The data for the analysis: a CSV file called "Doughnuts"
3. A way to analyse things: a script file doughnuts_script



You then have to tell R Studio to bring in the data you've added. Type the command below into the script window (top left) you've opened and run the command by clicking the RUN button (Run). If you're using a PC, you can hit CTRL+ENTER at the same time while you're typing on the line, which will also run the bit of code highlighted. It should show up in the bottom left window. If you're using a Mac, you can press ⌘ + ENTER at the same time. Note sometimes Word sometimes messes up quotation marks so if you get an error when you paste the line below, type the "" manually.

```
Doughnuts<-read.csv("doughnuts.csv", header=TRUE,
stringsAsFactors=FALSE)
```

`read.csv` is simply a command which tells R the file you want to read in is a csv file. "`Doughnuts.csv`" is the name of data you just saved from the excel sheet. `header=TRUE` tells R that the top row of data is the name of each column. `stringsAsFactors=FALSE` is important in the future and will stop your data from being formatted incorrectly.

You can look at the data by typing in the following commands:

```
head(Doughnuts)#this will show you the first few lines of
the data
names(Doughnuts)#this will show you the names of the
columns
Doughnuts #this will print out the whole dataset
```

2. Looking at your data visually in R

First look at all of the data on FAT content together (i.e. the variable called **Fat**). We can use the command hist to make a histogram.

```
Doughnuts$Fat<-as.numeric(Doughnuts$Fat)
```

This command tells R that the values in this column are numbers.
Using "as.numeric" tells R how to class the variable, which is as a number. If you are ever unsure what class your variables are at any given time, you can use the command below to double check the structure of your data.

```
str(Doughnuts)
```

Most importantly for this analysis, we also need to tell R what is a "factor" – in this case Outlet is our factor.

The command is

```
Doughnuts$Outlet<-as.factor(Doughnuts$Outlet)
```
It is easy then to check if R has understood this by typing and seeing what R says
```
class(Doughnuts$Fat)
```

We can now plot a histogram of the Fat variable and a QQ norm to check for normality

```
hist(Doughnuts$Fat)
qqnorm(Doughnuts$Fat)
qqline(Doughnuts$Fat, col = "blue", lwd = 2)
```

These help us check if the data is normal. We conclude that data is close to a normal distribution.

Then we can plot a boxplot of the 4 groups (Fat split by outlet)

```
boxplot(Doughnuts$Fat~Doughnuts$Outlet)
```
We will learn in practical 4 how to make pretty plots. For now we just use these basic commands and concentrate on the analysis.

3. Summary statistics in R

You can also use R to get summary statistics (mean, median, standard deviation, range etc.) for your variables. For example, to calculate the mean amount of fat for each outlet

```
mean(Doughnuts$Fat[Doughnuts$Outlet=="1"])
sd(Doughnuts$Fat[Doughnuts$Outlet=="1"])
```

```
var(Doughnuts$Fat[Doughnuts$Outlet=="1"])

summary(Doughnuts)
```

4. Performing an analysis of variance

The command for ANOVA in R is `aov`
And in this case, we tell R to define the numeric variable (Fat) and the categorical variable (factor – outlet).
`aov.Doughnuts=aov(Fat~Outlet,data=Doughnuts)`
We then ask R to provide an ANOVA table and followed by that the grand mean and the mean of each group
The command is
`summary(aov.Doughnuts)`

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Outlet	3	1636	545.5	5.406	0.00688 **
Residuals	20	2018	100.9		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 >

Interpretation of the table:
Df = degrees of freedom; numerator j(number of groups)-1 = 4-1 = 3
Denominator n-j (total number of observations – number of groups = 24-4 = 20.
Sums of squares due to outlet calculated from BETWEEN group variability (SS_B),
Sums of squares due to residuals calculated from WITHIN group variability (SS_W)
Mean square is the Sums of squares divided by the degrees of freedom
F value is the ration of the Mean square due to outlet and residuals i.e the ratio of between and within group variability.
The p value is associated with the F ratio and the degrees of freedom and is compared to a cut-off value in the F table.
This result tells us that we CAN reject the null hypothesis and that the mean fat content DOES differ between the 4 outlets at a probability of p = 0.00688.
Remember that is this point, we cannot say anything more than that the groups differ.
To see which groups are different from the others, we use Tukey's Honestly Significant Difference method:
`TukeyHSD(aov.Doughnuts)`
Which should give you the output:

```
Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Fat ~ Outlet, data = Doughnuts)

$Outlet
    diff        lwr       upr     p adj
2-1   13  -3.232221 29.232221 0.1461929
3-1    4 -12.232221 20.232221 0.8998057
4-1  -10 -26.232221  6.232221 0.3378150
3-2   -9 -25.232221  7.232221 0.4270717
```

```
4-2  -23 -39.232221 -6.767779 0.0039064
4-3  -14 -30.232221  2.232221 0.1065573
```

So we can see there is a difference between groups 4 and 2 but not the others.

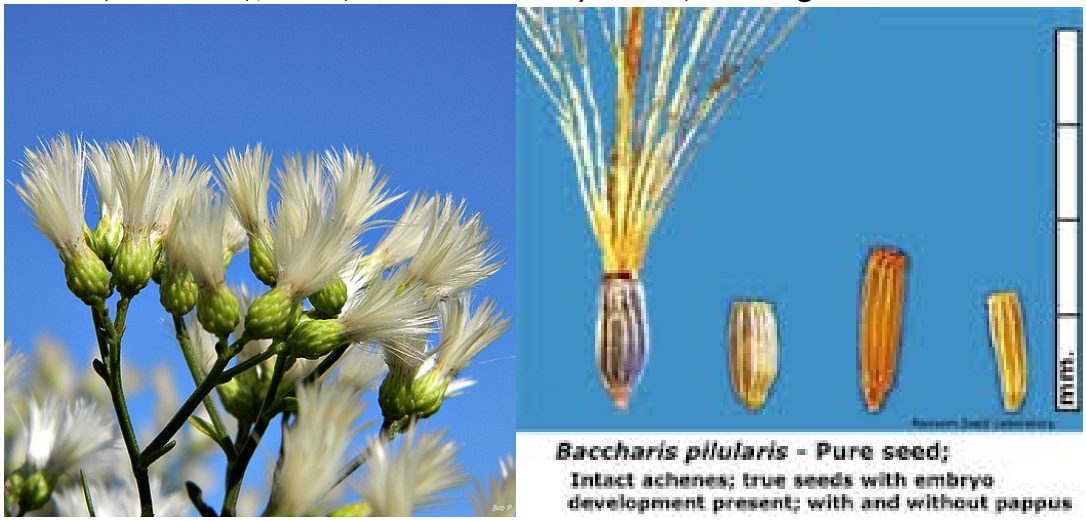
Now you are provided with 3 other data sets that will allow you to perform analysis of variance. Remember to submit your R script (the .R file in your folder) through Blackboard at the end of the session.

For each data set undertake the following

1. Erect a hypothesis (Null and Alternate)
2. Plot the data and assess normality
3. Make sure that the variables are correctly described (numeric, group etc)
4. Provide summary statistics
5. Perform an Analysis of Variance and interpret the output

Data set 1: Seeds

This data set consists of four variables. Plant (*Baccharis*) ID, Numbers of seeds (raw data), Site (6 sites coded by letter) and log10 seeds.



Data set 2: Pigeons

This data set consists of two variables. Five groups of pigeons (Group variable coded 1-5) with a morphometric measurement (beak) for each bird (Length variable).

Data set 3: Ascaris in a mouse model

This data set consists of 5 variables. These are mouse ID, mouse inbred strain (9 in total), numbers of *Ascaris* larvae in the left lung, right lung and the total number of larvae.