Using R as a Research Tool.
Part 2: R Markdown and Basic Statistics.

Dr Susan Johnston: Susan.Johnston@ed.ac.uk

Demonstrator: Gergana Daskalova.

5th November 2019

1 Introduction

This tutorial is based on information from R4all: `http://r4all.org/` and Our Coding Club: `https://ourcodingclub.github.io/`. Data, code, answers and an **R** Markdown Cheat Sheet is included in the github repository: `https://github.com/susjoh/Intro_to_Stats_in_R`.

This practical will explore how to write reports in **R** Markdown and how to conduct simple statistical tests in **R**. By the end, you should be able to:

- Write, embed and render code and results into an HTML document.
- Carry out basic statistics and visualisations, including:
  - Chi-squared (χ^2) test with `chisq.test()`
  - 2-sample t-test with `t.test()`
  - Linear regression with `lm()`

2 Writing reports with R Markdown.

R Markdown is a tool for writing reproducible reports in **R**. It uses the `knitr` library by Yihui Xie to produce documents with embedded code and figures in HTML, Word and PDF format. It can also be used to create webpages and presentations.

2.1 Creating an R Markdown Document.

Open RStudio and create a new markdown document by going to **File > New File > R Markdown...**. In the window, name your document, select **HTML** and click **OK**. RStudio should automatically create a template as in Figure 1.

To render the document, click the button that says `Knit` (you may have to save it first). As you can see, it produces a formatted HTML document with embedded figures that reflect the text and code within the markdown file. Take some time to familiarise yourself with how the script matches to the output.

A convenient feature of **R** Markdown in RStudio is the **Show document outline** in the top right corner of the window. For very long scripts, adding #### or ---- to a comment line will include that comment in the document outline. Try it out. It is possible to carry out basic formatting as well as adding headings, tables, lists and external figures. For the very basics of Markdown, go to **Help > Markdown Quick Reference**. You can also access more detailed cheatsheets through **Help > Cheatsheets**.

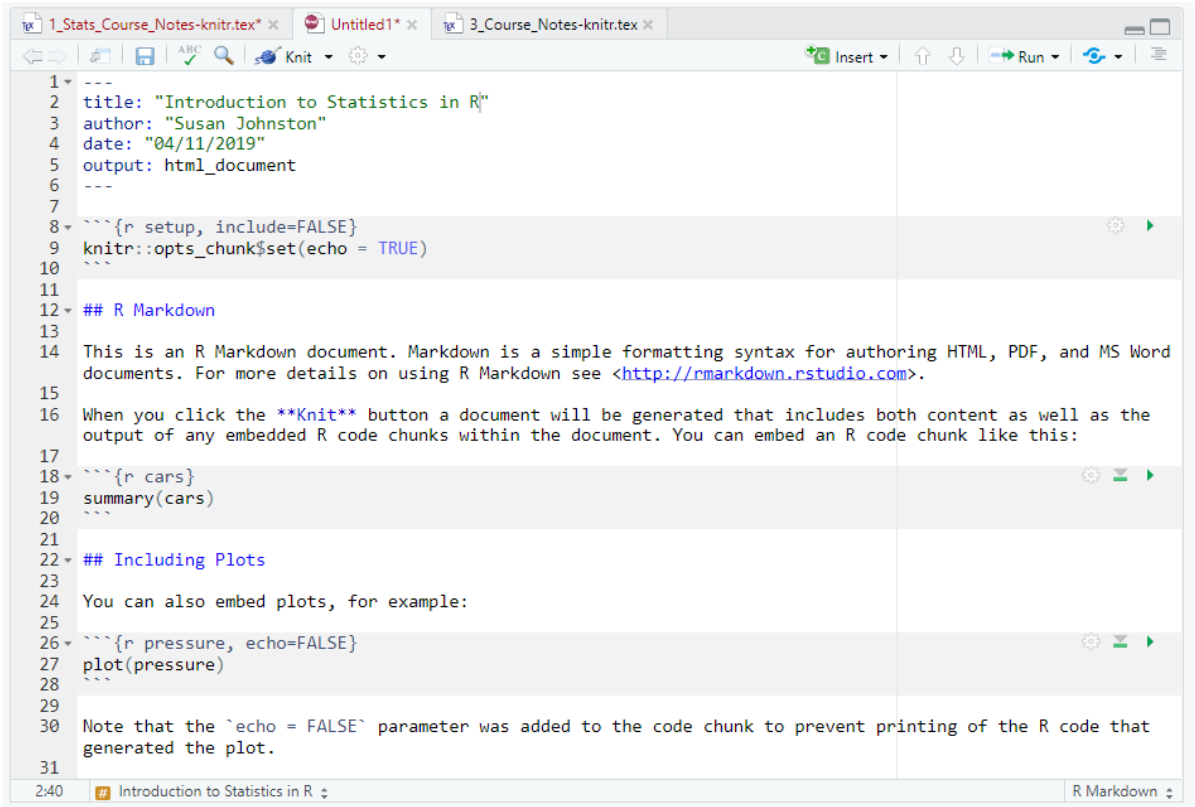


Figure 1: R Markdown Template.

Exercise 1.

1. Create a new **R** Markdown Script with the following header:

```
--
title: "Introduction to Statistics in R"
author: "Your Name"
date: "05 November 2019"
output:  html_document
--
```

Add some text & familiarise yourself with basic formatting (e.g. *italic*, **bold**). Add headers for "Introduction", "Chi-square test", "Two-sample t-test" and "Linear regression".

2.2 Embedding code in R Markdown.

Code can be embedded into the document in two ways:

Code Chunks:

````
```{r}
head(iris)   # this is a dataset included in base R
```
````

Code chunks can be named, e.g. ```` ```{r iris} ````

Inline code:

```
Two plus two equals `r 2 + 2`.
```

.
Which will print "Two plus two equals 4.".

2.3 Structuring an R Markdown document.

As you have seen in the example, there can be several code chunks within a document. Each chunk is run sequentially and saved in the working environment. Therefore libraries

and new objects created in a chunk will remain in the workspace for all subsequent chunks. When clicking "Knit", RStudio will assume that the directory containing the file is the working directory.

The first chunk in your Markdown document is a good place to load libraries and data. In this practical, we will use the libraries ggplot2 and dplyr. Put this chunk after the header of your document and click Knit:

````
```{r}
library(ggplot2)
library(dplyr)
```
````

You may notice that this results in some unsightly code that is not relevant to the document. It is possible to control what is reported in the document using additional markers within the chunk description. For example, the following will allow the command to run invisibly:

````
```{r echo = FALSE, message = FALSE, warning = FALSE}
library(ggplot2)
library(dplyr)
```
````

It can also control how results and figures are presented. Try knitting:

````
```{r}
data(iris)
str(iris)
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) + geom_point()
```
````

and then replacing it with the following, and knitting again:

```
```{r echo = F, results = "hide", fig.width = 4, fig.height = 3}
data(iris)
str(iris)
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) + geom_point()
```
```

The second option will hide the code (`echo = F`) and code output (`results = "hide"`) but show the figure (dimensions specified with `fig.width` and `fig.height`). Some other common options are given in here:

Rule	Example (default)	Function
`eval`	`eval=TRUE`	Is the code run and the results included in the output?
`echo`	`echo=TRUE`	Is the code displayed alongside the results?
`warning`	`warning=TRUE`	Are warning messages displayed?
`error`	`error=FALSE`	Are error messages displayed?
`message`	`message=TRUE`	Are messages displayed?
`tidy`	`tidy=FALSE`	Is the code reformatted to make it look "tidy"?
`results`	`results="markup"`	How are the results treated? `"hide"` = evaluate, but don't show the results. `"asis"` = show the results without formatting. `"hold"` = results are only compiled at the end of the chunk (e.g. if chunk contains many commands)
`fig.width`, `fig.height`	`fig.width=7`	What width/height (in inches) are figures?
`fig.align`	`fig.align="left"`	Do you want to align figures `"left"`, `"center"`, or `"right"`?

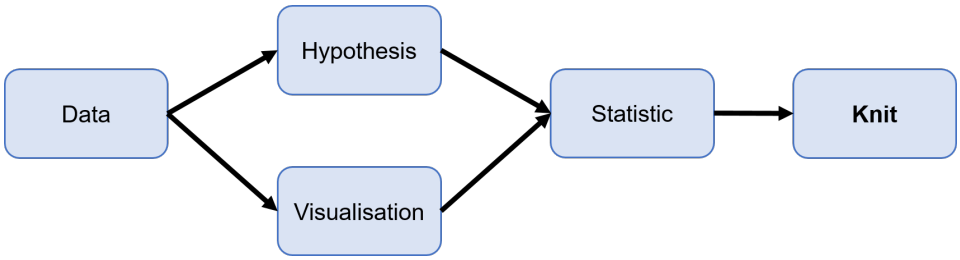
2.4 Important Points to Note.

- Using R Markdown essentially opens a new R session to run all of the code within your **.Rmd** document. Creating an object in a chunk means that it can be called in all subsequent chunks unless removed from the workspace.
- Code in **R** Markdown chunks can be run in the console as in the previous practical. The default option is to show the code output inline in the document. Some people like this, others don't: to switch this function on and off, go to **Global options > R Markdown > Show output inline for all Markdown documents** and select your preferred setup.

- R Markdown documents are **not** flexible to errors - the code must be error free, or it will not render.
- Want to learn more? More detailed information on this can be found at the Coding Club tutorial at `https://ourcodingclub.github.io/2016/11/24/rmarkdown-1.html`.

3 Basic statistics.

For the rest of the practical, it is expected that you create an R Markdown document that contains text, code, graphs and inline reporting of results. We will approach this using the approach of examining the data through visualisation, form our hypothesis, carry out the statistical test and then `Knit` this information into our document.



In the interests of time, this practical will focus on the **how** rather than **why** we will do these tests. We highly recommend doing further reading in your own time, such as "Getting Started with R" (2nd Edition, 2017) by Beckerman, Childs and Petchey (`http://www.r4all.org`), which is available on PDF through the library service.

4 Chi-squared (χ^2) contingency table

A χ^2 contingency table analyses count data, and looks at the association between two or more categorical variables. In this example, we will examine the differences in the frequency of red and black ladybirds (*Adalia bipunctata*) in rural and industrial habitats. Our question is: are dark morphs more likely to reside in dark (industrial) backgrounds? The null hypothesis is that there is no association between ladybird colour morph and habitat type [1].

[1]The approach and dataset here is based on the example presented in the book "Getting Started with R" (2nd Edition, 2017) by Beckerman, Childs and Petchey (`http://www.r4all.org`)

Load the data file `ladybirds.csv` into **R** using `read.csv()` and examine it using `glimpse()` from the `dplyr` package, or `head()` as before.

```
library(dplyr)

ladybirds <- read.csv("data/ladybirds.csv", header = T)
glimpse(ladybirds)

Observations: 20
Variables: 4
$ Habitat      <fct> Rural, Rural, Rural, Rural, Rural, Rural, Rural, Rural...
$ Site         <fct> R1, R2, R3, R4, R5, R1, R2, R3, R4, R5, U1, U2, U3, U4...
$ morph_colour <fct> black, black, black, black, black, red, red, red, red,...
$ number       <int> 10, 3, 4, 7, 6, 15, 18, 9, 12, 16, 32, 25, 25, 17, 16,...
```

There are multiple lines for each category, with the column number giving the count details. We ultimately want four numbers, corresponding to the 2×2 categories: red industrial, black industrial, red rural and black rural.

This can be done using the `dplyr` functions `group_by()` and `summarise()`.

```
> totals <- group_by(ladybirds, Habitat, morph_colour)
> totals <- summarise(totals, total.number = sum(number))
> totals

# A tibble: 4 x 3
# Groups:   Habitat [?]
  Habitat    morph_colour total.number
  <fct>      <fct>               <int>
1 Industrial black                 115
2 Industrial red                    85
3 Rural      black                  30
4 Rural      red                    70
```

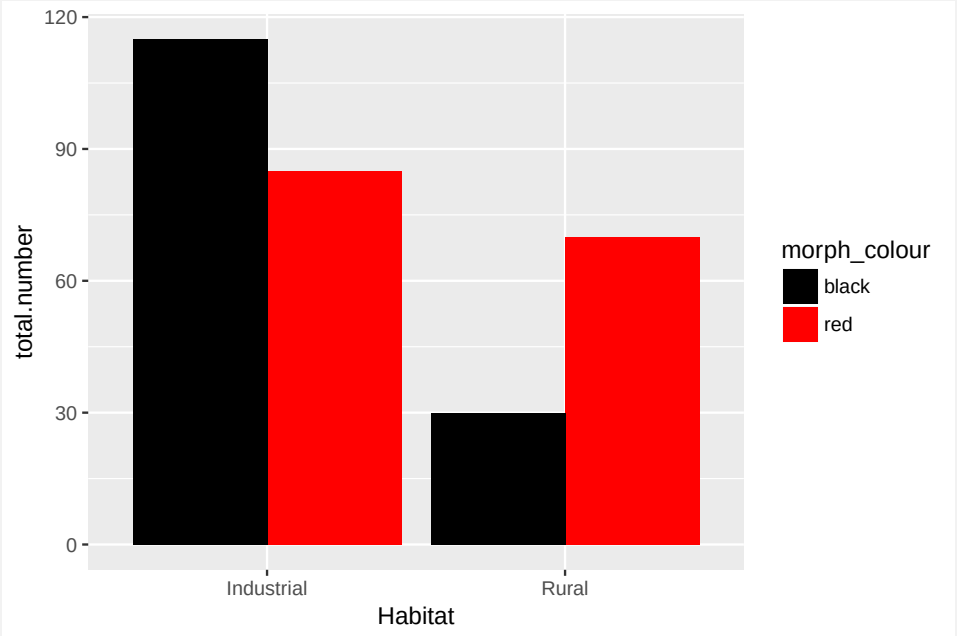
4.1 Plot the data

One visualisation for this type of data is a bar chart using `geom_col()` in `ggplot2`.

```
> library(ggplot2)
> ggplot(totals, aes(x = Habitat, y = total.number, fill = morph_colour)) +
+   geom_col()
```

There are two edits we can make to this to improve the visualisation: to add `geom_col(position = "dodge")` to place bars side by side, and to use `scale_fill_manual()` to change the colours of the bars to black and red to match the colour of the morphs in real life:

```
> ggplot(totals, aes(x = Habitat, y = total.number, fill = morph_colour)) +
+   geom_col(position = "dodge") +
+   scale_fill_manual(values = c(black = "black", red = "red"))
```



Does it look like there is a trend here? What would your prediction be based on the figure?

4.2 Test the hypothesis with `chisq.test()`.

The χ^2 is run using the function `chisq.test()`. As this is a 2×2 contingency test, we must convert the data into a matrix. Looking at the data `ladybirds`, a matrix can be made using the function `xtabs()`, which is similar to creating pivot table cross-tabulation in Excel:

```
> lady.mat <- xtabs(number ~ Habitat + morph_colour, data = ladybirds)
> lady.mat

            morph_colour
Habitat      black red
  Industrial   115  85
  Rural         30  70
```

Now run the test:

```
> chisq.test(lady.mat)

        Pearson's Chi-squared test with Yates' continuity correction

data:  lady.mat
X-squared = 19.103, df = 1, p-value = 1.239e-05
```

This provides a statistic and p-value indicating that there is a very small probability that the observed pattern arose by chance. Therefore, we can reject the null hypothesis. We can extract more information from the statistic if we save the `chisq.test(ladymat)` as an object:

```
> lady.chisq <- chisq.test(lady.mat)
```

Running `lady.chisq` gives the same output as before, but we can explore the object in detail using the `$` notation:

```
> names(lady.chisq)

[1] "statistic" "parameter" "p.value"   "method"    "data.name" "observed"
[7] "expected"  "residuals" "stdres"

> # str(lady.chisq) # not run here to save space - please run it!
>
> lady.chisq$statistic

X-squared
 19.10289

> lady.chisq$p.value

[1] 1.238571e-05
```

In the **R** Markdown document, it is possible to quote statistics inline using the `` `r` `` notation e.g. `` `r lady.chisq$statistic` `` and `` `r lady.chisq$p.value` `` will print the χ^2 statistic and P value inline, respectively.

Exercise 2.

Create a short report in the **R** Markdown document with an inline report of the test statistics and P-value. This can be done as follows:

1. Add a code chunk for loading and manipulating the data above, and running the χ^2 test.
2. Edit the previous chunk options so that it does **not** print the code or results to the compiled document (hint: define `echo` and `results` in the chunk options).
3. Write a few lines of text stating the hypothesis, the test statistic and interpretation. E.g.
   The null hypothesis is... Ladybird morphs are not equally distributed in the two habitats (Chi squared = ..., df = ..., P = ...), with black morphs being more frequent in the ... habitat.
4. Add a code chunk to output the figure that illustrates the results of the χ^2 test.

5 Two-sample t-test.

A two-sample t-test is one of the most conceptually simple and commonly used hypothesis tests. It determines whether the mean of two groups of numeric values are significantly different from each other, or are due to random chance. Here, we will use data from two *Iris* species, *I. virginica* and *I. versicolor* to determine if their Sepal lengths are significantly different (Figure 2).

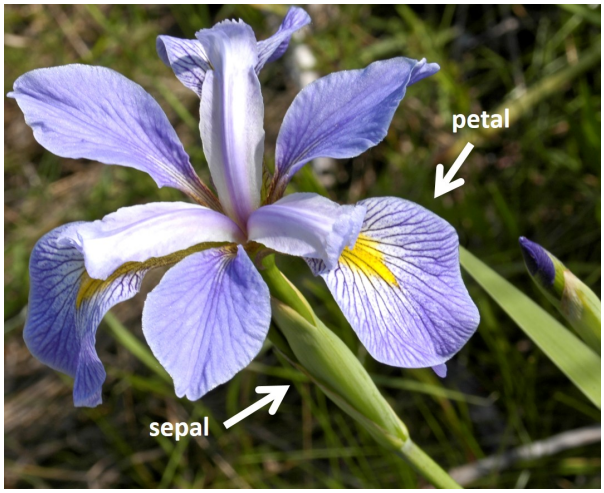


Figure 2: *Iris virginica.*

This test makes two assumptions about the data - that both groups are normally distributed and that the variances are equal in each category. For the interests of time, we will assume that both of these assumptions are met[2].

First, load the data:

```
sepals <- read.csv("data/iris.edited.csv", header = T)
glimpse(sepals)

Observations: 100
Variables: 2
$ Species      <fct> versicolor, versicolor, versicolor, versicolor, versic...
$ Sepal.Length <dbl> 7.0, 6.4, 6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5.0,...
```

[2]These assumptions can be tested using `shapiro.test()` and `var.test()` on specific specific value vectors. If you would like to try this and need help, use the `?` command or ask the demonstrators.

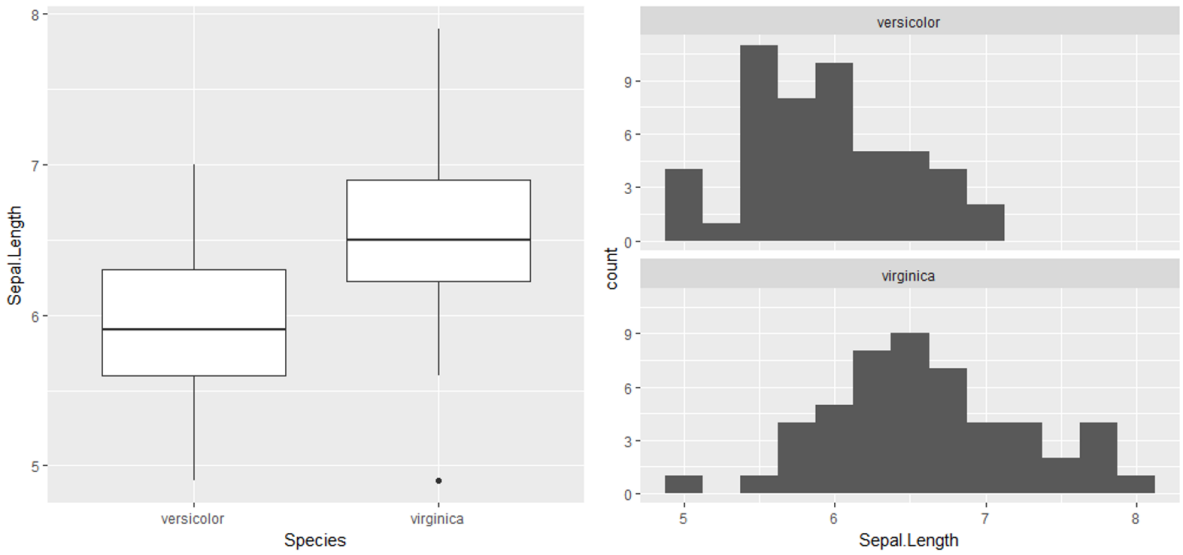
The first step is to visualise the data. One approach is to use a boxplot, which is more visually appealing (but based on the median rather than the mean) - another is to use histograms. These can help us to assess if the means seem different between the two categories, and if the data is normally distributed with a similar variance. It can also provide an indication of whether the null hypothesis can be accepted or rejected.

```
# boxplot

ggplot(sepals, aes(Species, Sepal.Length)) + geom_boxplot()

# histogram with facet_wrap

ggplot(sepals, aes(Sepal.Length)) +
geom_histogram(binwidth = 0.25) +
facet_wrap(~Species, ncol = 1)
```



To carry out the t-test, we will use the `t.test()` function. We can find out the details of the test using `?t.test`. The syntax requires a formula `Sepal.Length ~ Species` and the data frame (`data = sepals`). This should reflect the hypothesis - how does sepal length vary as a function of species?

```
> sepal.test <- t.test(Sepal.Length ~ Species, data = sepals)
> sepal.test

        Welch Two Sample t-test

data:  Sepal.Length by Species
t = -5.6292, df = 94.025, p-value = 1.866e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8819731 -0.4220269
sample estimates:
mean in group versicolor  mean in group virginica
                   5.936                    6.588
```

The function has automatically used the Welch version of the t-test, which relaxes the assumption of equal variances - this is fine for the purposes of this practical (see Beckerman et al. for a discussion of this in more detail). The output provides the `t`, `df` and `p-value` for the test, as well as the mean value in each of the two groups. The 95% confidence interval shows the interval between the difference between the two means - if this overlapped 0, then we would retain the null hypothesis.

Therefore, given the output, we can reject the null hypothesis, and can conclude that *I. virginica* has longer sepals than *I. versicolor*.

Exercise 3.

1. Create a short report in the **R** Markdown document with an inline report as for Exercise 2.

2. Use the Markdown Quick Reference in RStudio to add the image of *Iris virginica* to the Markdown document ("`data/Irisvirginica.jpg`")

6 Simple Linear regression.

The last model we will tackle is a linear regression. This is the most basic of a class of models called '**general linear models**' which also includes multiple regression and ANOVA.

For this, we will use one of the first datasets upon which linear regression was performed, which is Francis Galton's human height dataset from 1886 [3]. This dataset contains information on mother and father heights, as well as the height and sex of their adult children.

The data can be called as follows:

```
heights <- read.csv("data/Galton.csv", header = T)
head(heights)

  family father mother sex height nkids mid.parent
1      1   78.5   67.0   M   73.2     4      72.75
2      1   78.5   67.0   F   69.2     4      72.75
3      1   78.5   67.0   F   69.0     4      72.75
4      1   78.5   67.0   F   69.0     4      72.75
5      2   75.5   66.5   M   73.5     4      71.00
6      2   75.5   66.5   M   72.5     4      71.00

> dim(heights)

[1] 898   7
```

There are a number of columns here, including identifiers for the family and the number of children. We will focus on `height` (the adult child height in inches) and `mid.parent` (the mid-parental height between the mother and father). As there is a strong sex difference in child height, we will filter the dataset to only include female offspring.

```
> heights <- filter(heights, sex == "F")
> dim(heights)

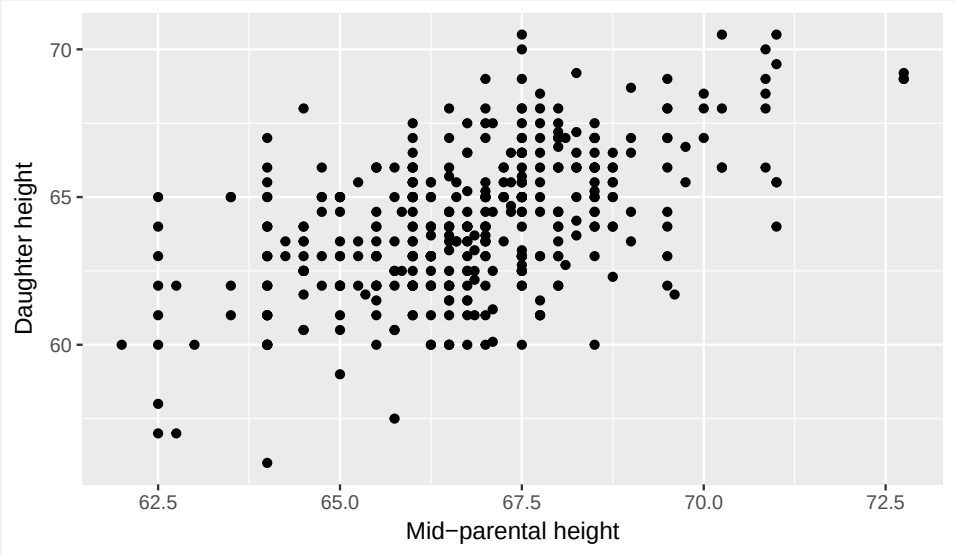
[1] 433   7
```

Our question is whether offspring height varies relative to the mean height of their parents. Here, offspring height is the response (dependent) variable, whereas mid-parental height is the explanatory (independent) variable. Both are continuous, numeric variables.

[3] Whilst Francis Galton made many important contributions to statistics, geosciences, genetics and psychology, like Ronald Fisher, he was also a keen eugenicist. The data is presented here as a historical scientific spirit, and in no way condones his views.

Visualise the data:

```
> ggplot(heights, aes(x = mid.parent, y = height)) +
+   geom_point() +
+   labs(x = "Mid-parental height", y = "Daughter height")
```



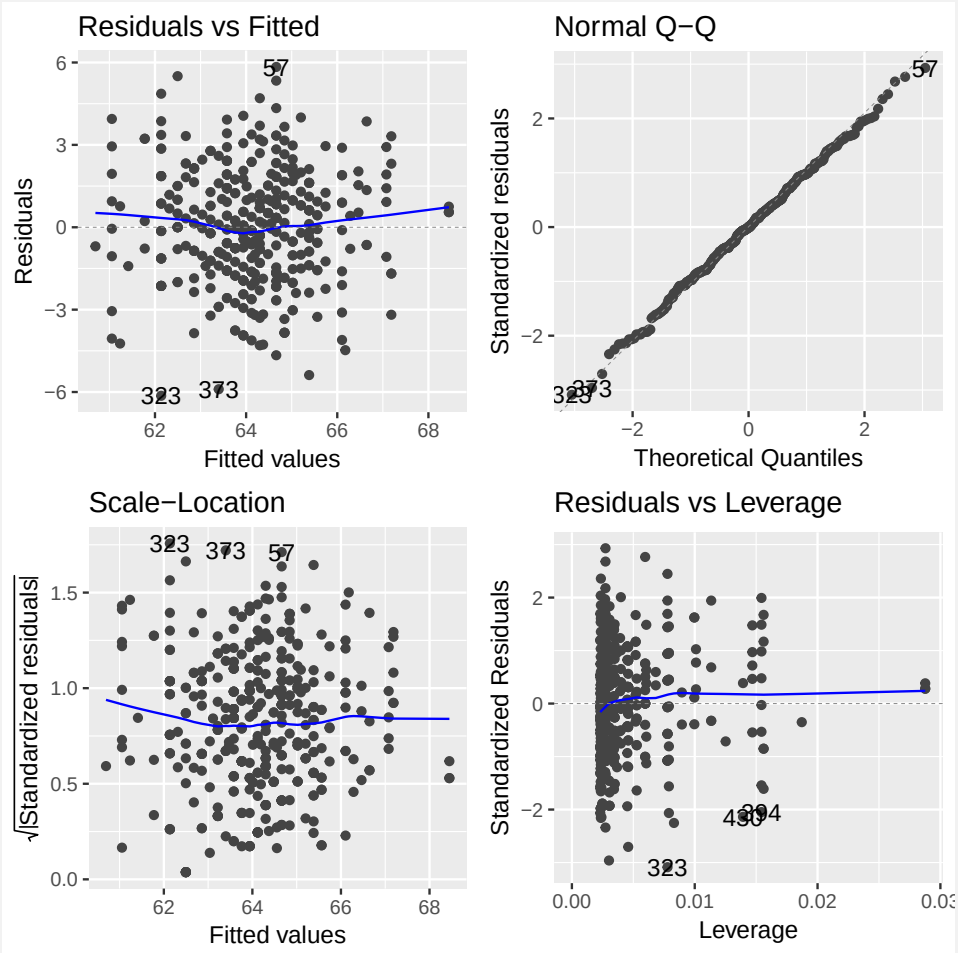
Visually, it looks like there is a positive relationship between mid-parental height and daughter height - taller parents have taller daughters. We can test this using the function `lm()` to fit the model. The syntax is similar to that of the t-test above:

```
> lm.heights <- lm(height ~ mid.parent, data = heights)
```

Before examining the output, it is important to check the assumptions of the linear model. This can be done using the `autoplot()` function in `library(ggfortify)` to examine the model residuals: [4]

```
> library(ggfortify)
> autoplot(lm.heights)
```

[4] You may need to install this package using `install.packages("ggfortify")`.



Generally these plots look good - although we have limited time to understand them in detail, we can quickly interpret them as follows:

- The **top left** plot determines if the line is an appropriate fit to the data. A relatively straight line is good - something humped would indicate that a non-linear relationship.
- The **top right** plot shows if the residuals are normally distributed - they should line up on the straight dashed line.
- The **bottom left** plot is a check of equal variances; again, a lack of pattern (i.e. a horizontal line) shows the model is an appropriate fit.
- Finally, the **bottom right** panel shows the residuals vs. leverage, to determine some of the data points have a particularly strong influence on the regression line. A straight line and all data within the Cook's distance lines (these do not appear in this plot) indicate that this is not an issue in the data.

Given that the data looks good, let's examine the output form the statistical model:

```
> lm.heights

Call:
lm(formula = height ~ mid.parent, data = heights)

Coefficients:
(Intercept)   mid.parent
    15.9671       0.7214
```

As you can see, calling the statistical model alone only provides us with two values - the model intercept and the slope. The slope is positive, and suggests that for for each unit increase in mid-parental height, there is an increase in daughter height by 0.72 inches.

However, calling this alone does not provide information on whether we can reject the null hypothesis. For this, we use the `summary()` function:

```
> summary(lm.heights)

Call:
lm(formula = height ~ mid.parent, data = heights)

Residuals:
    Min      1Q  Median      3Q     Max
-6.1365 -1.3989  0.0028  1.4207  5.8386

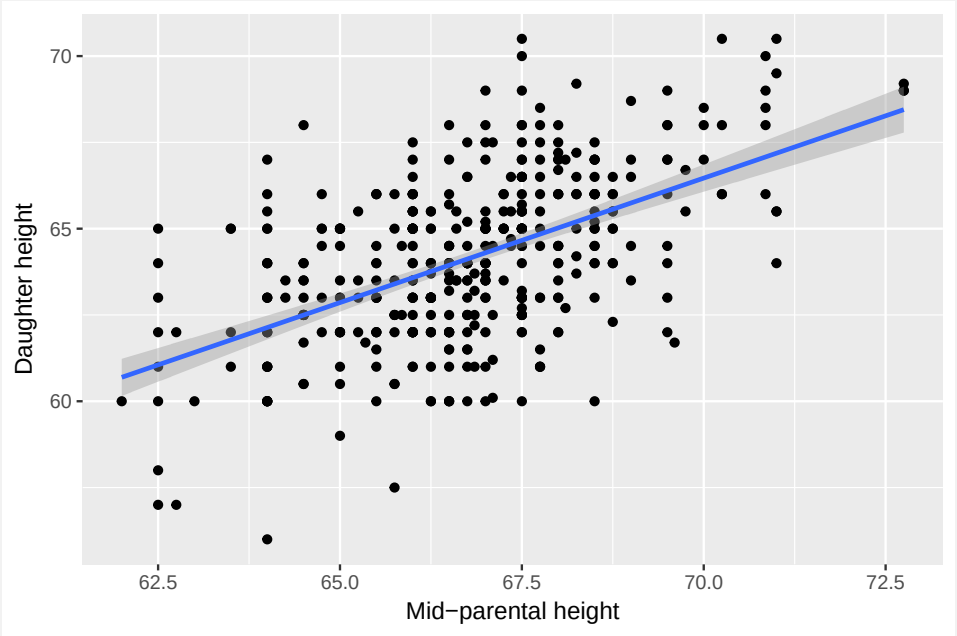
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 15.96711    3.60237   4.432 1.18e-05 ***
mid.parent   0.72140    0.05396  13.369  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.995 on 431 degrees of freedom
Multiple R-squared:  0.2931,        Adjusted R-squared:  0.2915
F-statistic: 178.7 on 1 and 431 DF,  p-value: < 2.2e-16
```

We can see here that the observed positive slope is highly significant (`<2e-16`).

Finally, we can add the slope to the plot using `stat_smooth(method = "lm")`:

```
> ggplot(heights, aes(x = mid.parent, y = height)) +
+   geom_point() +
+   stat_smooth(method = "lm") +
+   labs(x = "Mid-parental height", y = "Daughter height")
```



Exercise 4.

1. Create a short report in the **R** Markdown document with an inline report as for Exercises 2 and 3. Remember to report the slope, t-statistic, degrees of freedom and P-value.

Hint: extracting values from the model can use indexing as visited in the preparation tutorial for the previous session. The best approach is to create an object for the summary of lm.heights e.g. `summary.lm.heights <- summary(lm.heights)$coefficients` and calling values based on the indices (e.g. the t-value will be `summary.lm.heights[2,3]`.