r4nqy

Michael Hunt

2025-07-23

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

```
1 + 1
```

```
[1] 2
```

Introduction

This is a compilation of the methods for data analysis that you are likely to find useful in completing your undergraduate studies.

Data analysis in the life sciences is only part of the wider process of forming and then answering as best we can well-formed questions about the way this or that aspect of the natural world works.

1 A recommended analysis workflow

This is intended as a rough outline of the sequence of steps one commonly goes through when working on scripts:

- Before you start on the script, ask yourself: am I working in a Project?. If not, fix this!
- What is my question? AM I clear about what I want the script to do?
- Load packages
- Load data
- Inspect data
- Clean up or tidy or manipulate the data in some way - whatever it takes to get it in the form needed for the analysis steps
- Summarise the data
- Plot the data
- Do statistical analysis
- Decide what all this has told you and report it in plain English.

Details will differ from script to script, but this sequence of steps is very common.

1.1 Are you working within your Project?

Before we even think of the script, we need to make sure that we are working within our Project. If we are not doing this, bad things will happen. If you are, the Project name will be at the top right of the RStudio window. If you are not, save the script you are working on, and go to File/Open Project and open your Project. If you haven't even got a 'Project' or don't know what that means then just make sure that everything you need for whatever you are working on is in one folder and then turn that folder into a Project. (So a 'Project' is just a regular folder that has been given superpowers.) You do that by going to File/New Project/Existing Directory. Then you navigate to your folder and click on Create Project. RStudio will then restart and you will see the name of your newly anointed Project folder at the top-right of the RStudio window. You know that a folder is a 'Project' because it will have a .Rproj file inside it.

If all this sounds complicated, don't worry. It really isn't. Just get someone to show you how to do it and you will be fine.

Now, to the script itself:

1.2 Statement of the question(s) to be investigated

Without thinking this through, you won't know what your script is for...

What is the analysis that will follow for? What question are you trying to answer? What hypotheses are you trying to test?

Suppose we were trying to test the hypothesis that there is no difference between the petal widths of the *setosa*, *versicolor* and *virginica* species of iris. All we have to go on are the petal widths of the plants we happened to measure. From these measurements we want to make a statement about these three species in general.

1.3 Open a notebook

In RStudio, go to File/New File/ R Notebook. Delete everything below the yaml section at the top. This strangeley named section is the bit between the two lines with three dashes in. For the most part, we will not need to worry about this section. We just should not delete it entirely. What is useful to do is to amend the title to something sensible, and to add `author: "your name"` and `date: "the date in any old format"` lines, so that your yaml will look something like this:

```
---
title: "A typical workflow"
author: "Who wrote this?"
date: "Today's date"
output:
  html_document:
    df_print: paged
---
```

Delete everything beneath this yaml section. The big empty space that then leaves you with is where you write your code. Remember that in notebooks, the code goes in 'chunks' that are started and finished with by lines with three backticks. Any other text goes between the chunks and you can format this text using the simple rule of Markdown, available in the RStudio Help menu. Thus your script will end up looking something like this:

```
---
title: "A typical workflow"
author: "Who wrote this?"
date: "Today's date"
output:
```

```
  html_document:
    df_print: paged
---
```

1.4 First header

Any text we want to add. Note that a code chunk starts with `{r} and ends with`

```r
library(tidyverse) # some actual R code
```

1.5 Second header

Any text we want to add to explain what this next chunk does

````
```{r, include=FALSE}
library(tidyverse) # some actual R code
```
````

1.6 Set-up chunk

Just put this chunk in at the top. Worry about it later. Or don't worry about it at all, if you prefer. It is there to suppress warnings and messages from appearing in the rendered version of your script.

````
```{r, include=FALSE}
# makes the rendered version look prettier
knitr::opts_chunk$set(message=FALSE,warning=FALSE)
```
````

1.7 Load Packages

You will nearly always want the first five packages, and often you will appreciate the sixth, `janitor`. Others, such as `vegan` will be useful from time to time, depending on what you are doing. If any of these lines throw an error, it is most likely because you have not yet installed that package. Do so in the console pane (not in this script!) using the function `install.packages("name of package")`. Then run this whole chunk again.

```
library(tidyverse)
library(here)
library(ggfortify)
library(readxl)
library(cowplot)
library(janitor)
library(vegan)
```

1.8 Load data

There are several ways to do this, so details will differ depending on what file type your data is saved in and where it is stored.

Here are two examples. In each case code here presumes that the data is stored in a subfolder called 'data' within the Project folder, and we use the function `here()` from the `here` package. In my experience this dramatically simplifies the business of finding your data, wherever your script is. It makes it easier for you to share your script with others and be confident that what worked for you will work for them. It *does* require that you are working within your project.

1.8.1 If from a csv file

If you have your data in a `data` subfolder within your project, this chunk will work. Just substitute the name of your data file

```
filepath<-here("data","iris.csv")
iris<-read_csv(filepath)
glimpse(iris)
```

```
Rows: 150
Columns: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
$ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
$ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
$ Species      <chr> "setosa", "setosa", "setosa", "setosa", "setosa", "setosa~
```

1.8.2 If from an Excel file

You will need to use `read_excel()` from the `readxl` package, and you have to specify the name of the worksheet that holds the data you want. You can, if you want, specify the exact range that is occupied by the data. However I suggest you avoid doing this unless it turns out that you need to do so. If your data is a nice, neat, rectangular block of rows and columns, you should find that you don't need to specify the range.

```
filepath<-here("data","difference_data.xlsx")
iris<-read_excel(path = filepath,
                 sheet = "iris", # delete the comma if you choose not to specify the range i
                 range= "A1:F151" # optional - try leaving it out first. Only include if nec
                 ) |>
  clean_names()
glimpse(iris)
```

```
Rows: 150
Columns: 6
$ id           <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17~
$ sepal_length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
$ sepal_width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
$ petal_length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
$ petal_width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
$ species      <chr> "setosa", "setosa", "setosa", "setosa", "setosa", "setosa~
```

1.8.3 If from a URL

You can load data into R directly from a URL if you are given one, and that is in fact how you will mainly access data to be used in this statistics text.

here, we load data from a file stored in a ''repo' on my github account:

```
iris<-read_csv("https://raw.githubusercontent.com/mbh038/r4nqy/refs/heads/main/data/iris.csv
  clean_names()
glimpse(iris)
```

```
Rows: 150
Columns: 5
$ sepal_length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
$ sepal_width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
$ petal_length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
```

```
$ petal_width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
$ species      <chr> "setosa", "setosa", "setosa", "setosa", "setosa", "setosa~
```

1.9 Clean / Manipulate the data

Often we need to do some sort of data 'wrangling' to get the data into the form we want. For example we may wish to tidy it (this has a particular meaning when applied to data sets), to remove rows with missing values, to filter out rows from sites or time periods that we don't want to include in our analysis, to create new columns and so on.

For example, lets create a new data frame for just the *setosa* species of iris:

```
setosa <- iris |>
  filter(species == "setosa") # filter picks out rows according to criteria being satisfied
glimpse(setosa)
```

```
Rows: 50
Columns: 5
$ sepal_length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
$ sepal_width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
$ petal_length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
$ petal_width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
$ species      <chr> "setosa", "setosa", "setosa", "setosa", "setosa", "setosa~
```

or maybe we just want the column that contain numeric data and not the one containing the species identifiers, which is text:

```
iris_numeric <- iris |>
  select(-species) # select() retains or leaves out particular columns. Here, we leave out th
glimpse(iris_numeric)
```

```
Rows: 150
Columns: 4
$ sepal_length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
$ sepal_width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
$ petal_length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
$ petal_width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
```

1.10 Summarise the data

How big is the difference between the mean of this group over here and that group over there, and how big is that difference compared to the precision with which we know those means? We nearly always want to do this as a first way to get insight into whether we will or will not reject our hypothesis. For example, let's find the mean petal widths of the three species, the standard errors of those means and save the results to a data frame called `petal_summary`

```
petal_summary<-iris |>
  group_by(species) |>
  summarise(mean.Pwidth = mean(petal_width),
            se.Pwidth = sd(petal_width/sqrt(n())))
petal_summary
```

```
# A tibble: 3 x 3
  species    mean.Pwidth se.Pwidth
  <chr>            <dbl>     <dbl>
1 setosa           0.246    0.0149
2 versicolor       1.33     0.0280
3 virginica        2.03     0.0388
```

We can look at this table and already get an idea as to whether the petal widths are the same or are different for the three species.

1.11 Plot the data

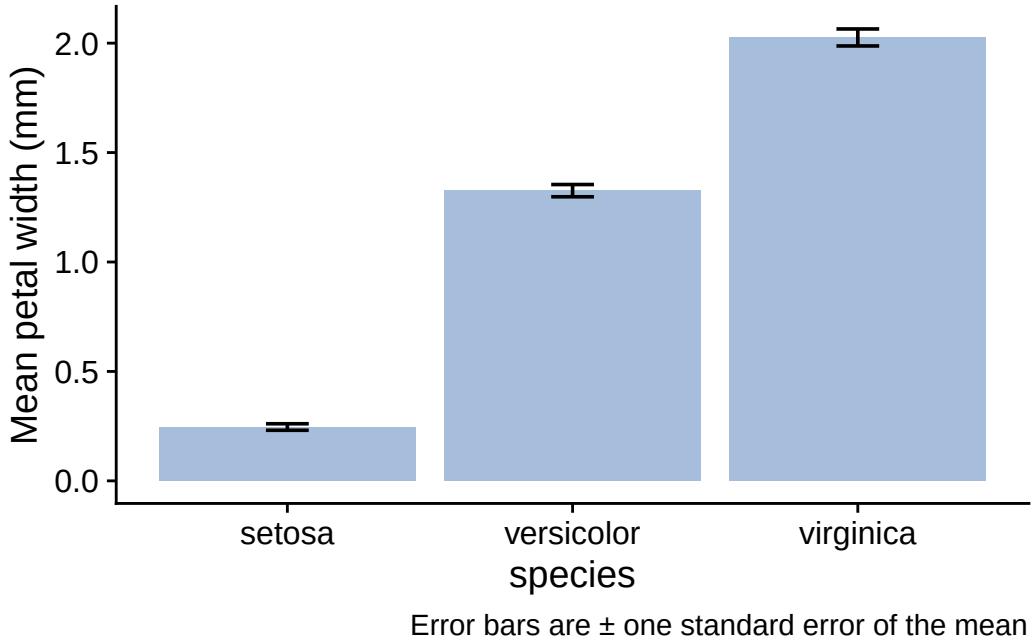
The next step is usually to plot the data in some way. We would typically use the `ggplot2` package from `tidyverse` to do this.

1.11.1 Bar plot with error bars

We could plot a bar plot with error bars, working from the summary data frame that we created:

```
petal_summary |>
  ggplot(aes(x = species, y = mean.Pwidth)) +
  geom_col(fill="#a6bddb") + # this is the geom that gives us a bar plot when we have already
  geom_errorbar(aes(ymin = mean.Pwidth - se.Pwidth, ymax = mean.Pwidth + se.Pwidth), width =
  labs( x = "species",
```

```
       y = "Mean petal width (mm)",
       caption = "Error bars are ± one standard error of the mean") + # important to say wha
  theme_cowplot()
```



Note that we have given the bars a fill colour - we got this color from this site due to the cartographer Cynthia Brewer, who is behind the various incarnations of the `Brewer` package in R, which is great for getting colours that work well. We have used the same colour for each species since the x-axis labels already tell us which bar relates to which species. To use a different colour for each bar would imply there is some extra information encoded by colour. Since there is not, it serves no purpose to have different colours, and potentially confuses the reader. Remember always that a plot is intended to convey a message. Anything that detracts from that message should be avoided, however pretty you think it is.

A couple of points could be made about this type of plot:

First, what about those error bars? Three types of error bar are in common usage and there are arguments in favour and against the use of each of them:

- the *standard deviation* tells us about the spread of values in a sample, and is an estimate of the spread of values in a population;

- the *standard error of the mean*, as used here, is an estimate of the precision with which the sample means estimates the respective population means for each of the species.

- the *confidence interval*, typically a *95% confidence interval*, gives us the region within which we are (say) 95% confident that the true species mean petal width might plausibly lie.

Which type of error bar is best to use depends on what story you want to tell. Here, because we are interested in whether there is evidence of a difference in the mean petal width of different species, we have gone for the standard eror of the mean.

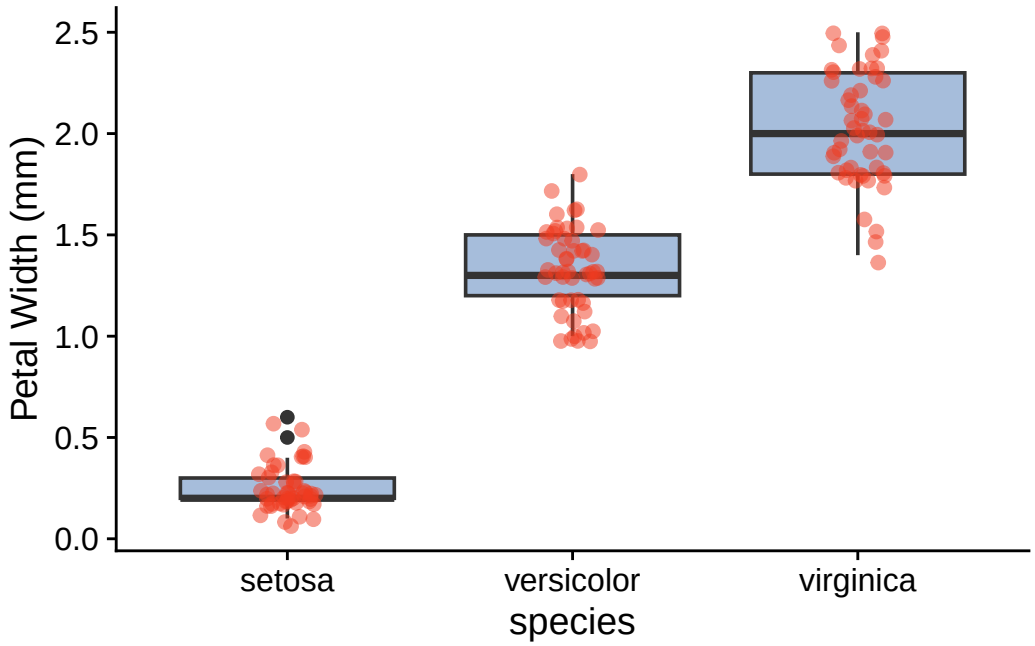
Regardless of which error bar you use and why, you should always tell the reader which one you have gone for, as we have in the caption to the figure.

A second point about this bar plot is that it doesn't tell us very much, and indeed nothing that we didn't already know. It only conveys the mean and standard error values for each species, which is information we already have, arguably more compactly and in more easily readable form, in the table we created. Further, it potentially obscures information that might come from knowing the *distribution* of the data.

Here are three other plot types that do show the distribution of petal widths for each species and thus add extra information to what we already know from the summary table

1.11.2 Box plot

```
iris |>
  ggplot(aes(x=species, y=petal_width)) + # what we want to plot
  geom_boxplot(fill="#a6bddb",notch=FALSE) + # what kind of plot we want
  geom_jitter(width=0.1, colour = "#f03b20",alpha=0.5) +
  labs (x = "species",
        y = "Petal Width (mm)") +
  theme_cowplot() # choose a theme to give the plot a 'look' that we like
```

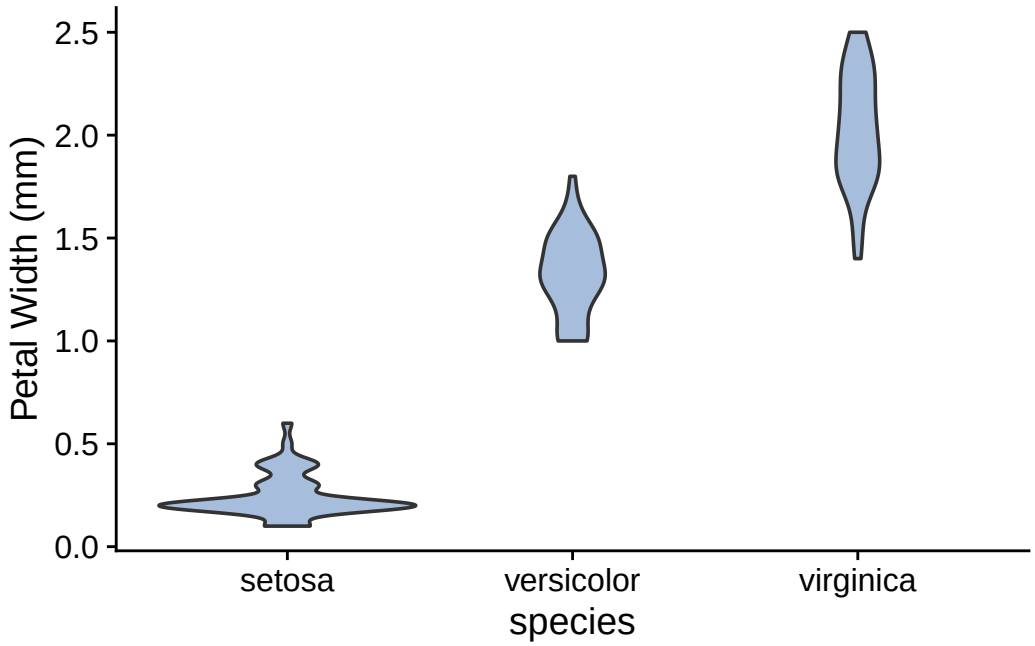


Here, we have added the points themselves on top of the box plot. When there are not too many data points, this can be useful. The 'jitter' adds some horizontal or vertical jitter, or both, so that the points do not lie on top of each other. In this case we see that the variability of petal widths is not the same for each species and that the data are roughly symmetrically distributed around the median values in each case. This information is useful in helping us determine which statistical test might be appropriate for these data.

1.11.3 Violin plot

A useful alternative to the box plot, especially when the data set is large, is the violin plot:

```
iris |>
  ggplot(aes(x = species, y = petal_width)) + # what we want to plot
  geom_violin(fill="#a6bddb",notch=TRUE) + # what kind of plot we want
  #geom_jitter(width=0.1, colour = "#f03b20",alpha=0.5) +
  labs (x = "species",
        y = "Petal Width (mm)") +
  theme_cowplot() # choose a theme to give the plot a 'look' that we like
```

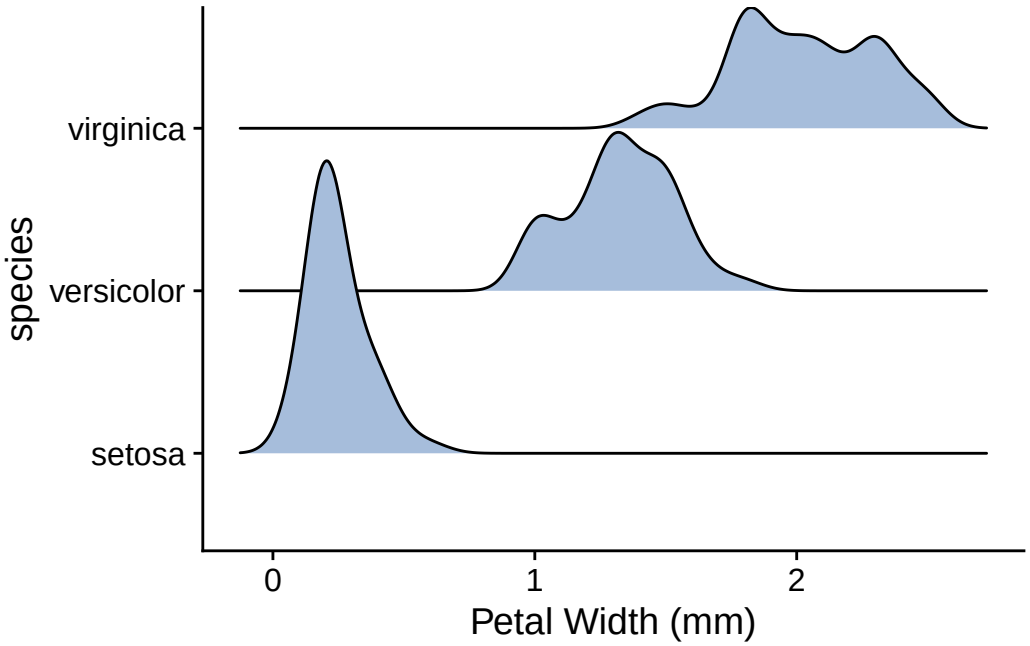


The widths of the blobs (I am probably supposed to call them 'violins'!) show us the distribution of the data - where they are widest is where the data are concentrated, while the height of the blobs shows us the range of variation of the data. The positions of the blobs tells us the mean petal widths of the different species and gives us an idea of the differences between them.

1.11.4 Ridge plot

A bit like a violin plot. This needs the package `ggridges` to be installed.

```
library(ggridges)
iris |>
  ggplot(aes(x = petal_width,y = species)) + # what we want to plot
  geom_density_ridges(fill="#a6bddb") + # what kind of plot we want
  #geom_jitter(width=0.1, colour = "#f03b20",alpha=0.5) +
  labs (x = "Petal Width (mm)",
        y = "species") +
  theme_cowplot() # choose a theme to give the plot a 'look' that we like
```



Having seen the summary and one of these plots of the data, would you be inclined to reject, or fail to reject, a null hypothesis that said that there was no difference between the petal widths of the three species?

1.12 Statistical analysis

Only now do we move on to the statistical analysis to try to answer our intial question(s). But by now, after the summary and plot(s), we may already have a pretty good idea what that answer will turn out to be.

The exact form of the analysis could take many forms. In a typical ecology project you might carry out several types of analysis, each one complementing the other. Here, an appropriate analysis might be to use the linear model in the form of a one-way ANOVA, since we have one factor (species) with three levels (*setosa*, *versicolor* and *virginica*) and an output variable that is numeric and likely to be normally distributed. We can use the `lm()` function for this.

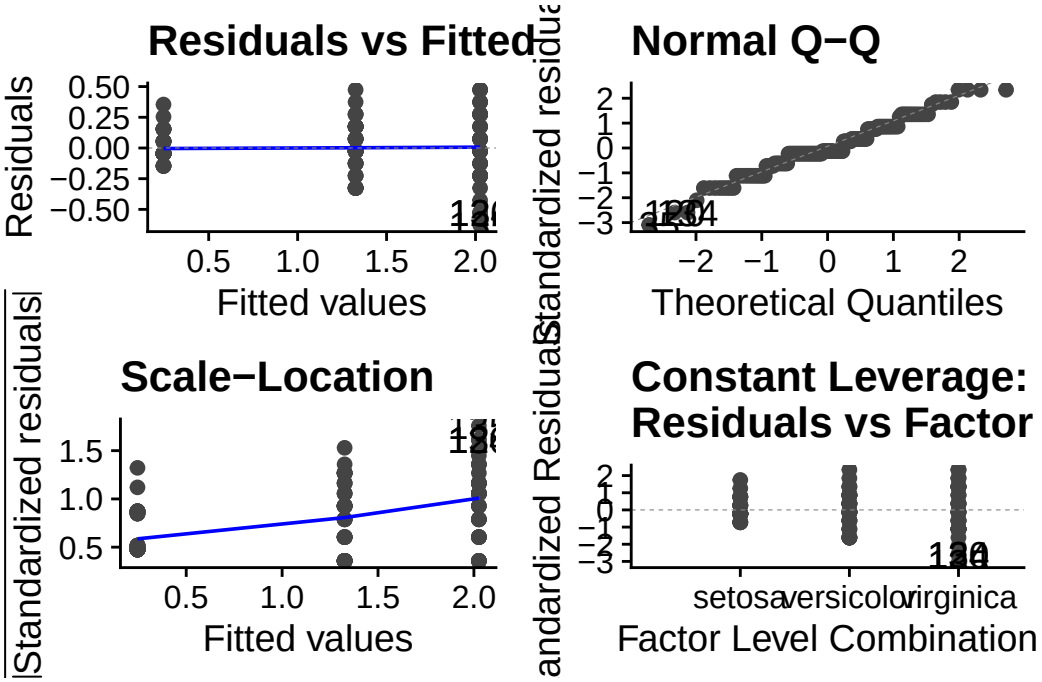
1.12.1 Create the model object

```
pw.model <-lm (petal_width ~ species, data = iris)
```

1.12.2 Check the validity of the model

We won't go into this here, but an important step is to check that the data satisfy the often finicky requirements of whatever statistical test we have decided to use. The `autoplot()` function form the `ggfortify` package is great for doing this graphically.

```
autoplot( pw.model) + theme_cowplot()
```



Here we note in particular that although the spread of data within each level is not roughly the same (top left figure)), the QQ plot is pretty straight (top-right figure). This means that the data are approximately normally distributed around their respective means. Taken together, this means that these data satisfy reasonably well the requirements of a linear model, so the output of that model should be reliable.

1.12.3 The overall picture

Typically, statistical tests are testing the likelihood of the data being as they are, or more 'extreme' than they are, *if* the null hypothesis were true. Thus, the null hypothesis is central to statistical testing.

The null hypothesis is typically that the 'nothing going on', 'no difference' or ' no association' scenario is true. In this case, it would be that there is no difference between the petal widths of the the three species of iris being considered here.

Typically too, a test will in the end spit out a p-value which is the probability that we would have got the data we got, or more extreme data, *if* the null hypothesis were true. Being a probability, it will always be a value between 0 and 1, where 0 means impossible, and 1 means certain. The closer the p-value is to zero, the less likely it is we would have got our data if the null hypothesis were true. At some point, if the p-value is small enough, we will decide that the probability of getting the data we actually got if the null hypothesis were true is so small that we reject the null hypothesis. Typically, the threshold beyond which we do this is when $p = 0.05$, but we could choose other thresholds. (Sounds arbitrary - yes, it is, but the choice of 0.05 is a compromise value that makes the risk of making each of two types of error - rejecting the null when we should not, and failing to reject it when we should, both acceptably small. This is a big topic which we won't explore further here.)

In the end, whatever other information we get from it, the outcome of a statisical test is typically that we either reject the null hypothesis or we fail to reject it. If we reject it then we are claiming to have detected evidence for an 'effect' and we go on to determine how big that effect is and whether it is scientifically interesting. If we fail to reject the null, that does not necessarily mean that there is no 'effect' (difference, trend, association etc). That might be the case, but it might also just mean that we didn't find evidence for one from our data.

It is all a bit like in a law court where the 'null hypothesis' is that the defendant is innocent, and at the end of the proceedings this null is either rejected (Guilty!) because the evidence is such as to make it untenable to hold onto the null hypothesis, or not rejected, because the evidence is not strong enough to convict, in which case the defendant walks free - but is not declared innocent. Formally, the court has simply found insufficient evidence to convict. In the latter case, the court would have failed to reject the null hypothesis. Crucially, it would *not* have declared that the defendant was innocent. In the same way, in a scientific study, we either reject or fail to reject a null hypothesis. We never *ever* accept the null hypothesis as true.

Actually, many researchers are unhappy wih this so-called 'frequentist' narrative and have sought to use an alternative 'Bayesian' approach to testing hypotheses. In this approach we can accept hypotheses and we can bring in prior knowledge. This is an interesting topic, but a very big one so we will not pursue it further here.

With all that behind us, we are in a better place to understand what the output of the test is telling us.

For the 1-way ANOVA, as with other examples of the linear model, this output comes in two stages:

1.12.4 Overall picture

Is there evidence for a difference between at least two of the mean values?

To see if there is evidence for this, an ANOVA test calculate the ratio between the dfference *betweeN* the groups compared to the differences *within* the groups. it calls this ratio F. The bigger F is, the more likely we are to reject the null hypothesis that there is no difference between he groups.

```
anova(pw.model)
```

```
Analysis of Variance Table

Response: petal_width
           Df Sum Sq Mean Sq F value    Pr(>F)
species     2 80.413  40.207  960.01 < 2.2e-16 ***
Residuals 147  6.157   0.042
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The F value is huge. The null hypothesis of this test, as with many tests, is that there is no difference between the petal widths of the three populations from which these samples have been drawn. In that case, the F value would be one. The p-value is telling us how likely it is that we would get an F value as big or bigger than the one we got for our samples if the null hypothesis were true. Since the p-value is effectively zero here, we reject the null hypothesis: we have evidence from our data that there is a significant varation of petal width between species.

The degrees of freedom `Df` tells us the number of independent pieces of information that were used to calculate the result. Let's not dwell on this here, but there are two that we have to report in this case: the number of levels minus one ie 3-1 = 2, and the number of individual data points in each level minus one, times the number of levels ie (50-1) x 3 = 147.

1.12.5 Effect size

Now that we have established that at least two species of iris have differing petal widths, we go onto investigate where the differences lie, and how big they are. This is important: effect sizes matter. It is one thing to establish that a difference is statistically significant (and typically even the tiniest difference can show up as significant in a study if the sample size is big enough), it is quite another to establish whether the difference is big enough to be scientifically interesting.

```
summary(pw.model)
```

```
Call:
lm(formula = petal_width ~ species, data = iris)

Residuals:
   Min     1Q Median     3Q    Max
-0.626 -0.126 -0.026  0.154  0.474

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        0.24600    0.02894    8.50 1.96e-14 ***
speciesversicolor  1.08000    0.04093   26.39  < 2e-16 ***
speciesvirginica   1.78000    0.04093   43.49  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2047 on 147 degrees of freedom
Multiple R-squared:  0.9289,    Adjusted R-squared:  0.9279
F-statistic:   960 on 2 and 147 DF,  p-value: < 2.2e-16
```

The output here is typical of that from a 1-way ANOVA analysis in R. Each line refers to one of the three levels of the factor being investigated, which is petal width in this case. By default, those levels are arranged alphabetically, so in this case the order is *setosa*, *versicolor* then *virginica*. The first row is always labelled `(Intercept)`, so here that row is referring to *setosa*. This level is used as the 'control' or reference level- the one with which the others are compared. If we are happy to have *setosa* as that control then we can just carry on, but if we are not, then we have to tell R which level we want to play that role. We'll go through how to do that later on.

In the Estimate column the value 0.246 cm in the first row refers to the actual mean petal width of the *setosa* plants in the sample. If we go back to the summary table we created earlier on, or look at one of the plots we created, we see that that is the case.

For all other rows, the value in the Estimate column is not referring to the absolute mean petal width but to the difference between the mean petal width for that species and the mean petal width of the control species. So we see that the mean petal width of the *versicolor* in our sample is 1.08 cm greater than that of *setosa* and so is equal to .246 + 1.08 = 1.326 cm, while that of the *virginica* is 1.78 cm greater and so is equal to 2.026 cm. Check from the table of mean values we created and the plots that this is correct.

Here though, we are not interested in absolute values so much as we are in differences, which is why that is what the summary table here gives us. Look again at the differences between the mean petal widths for *versicolor* and *virginica* and that for *setosa* and compare them with the standard erros of those differences, which are given in the second column of the table. These

standard errors are much smaller than the differences, meaning that we can have confidence that the differences are statistically significant.

This is borne out by the p-value in the right hand column of the table. The null hypothesis of this table is that there is no difference in petal width between populations of the different species from which these samples have been drawn.

Lastly, the adjusted R^2 tells us the proportion of variation of petal width that is accounted for by taking note of the species. Here, the value is 0.93, which tells us hat little else besides species determines the relative petal widths. Ther are no other variables that we need to have taken into account.

1.13 Report in plain English

You would say something like

We find evidence that petal widths are not the same acros thhree species of iris, with *virginica* > *versicolor* > *setosa*. (ANOVA, df = 2, p < 0.001)

2 Two-sample *t*-test

2.1 Preliminaries

In this exercise we find out how to use R to run a two-sample *t*-test, to determine whether there is evidence to reject the hypothesis that two samples are drawn from the same population.

Two-sample *t*-tests are used when we have two independent sets of numerical data, and we want to know whether the data provide evidence that the sets are drawn from different populations.

The exercise is based on Chapter 5: Beckerman, Childs and Petchey: Getting Started with R.

2.2 Motivation and example

In our example we will consider concentrations of airborne ozone (O_3) at ground level, as measured in gardens around a city. This is of interest because ozone levels can affect how well crops grow, and can impact on human health.

We have measurements of airborne ozone levels in ppb taken at two samples of locations in the city: some randomly selected from among gardens in the eastern residential sector and some randomly selected from among gardens in the western sector, close to a zone of heavy industry.

Our question is:

Is there evidence for a difference between airborne ozone concentrations in the east and the west of the city?

From which our null hypothesis is:

There is no difference between airborne ozone concentrations in the east and the west of the city.

and our alternate, two-sided hypothesis is:

There is a difference between airborne ozone concentrations in the east and the west of the city.

2.3 Pros and cons of the *t*-test

2.3.1 Pros

- It can be used when the data set is small.
- It can still be used when the data set is large. So...if in doubt, just use the *t*-test, (Kind of, the data do need to fulfil some criteria, but being few in number is fine. See below).

2.3.2 Cons

- It assumes that the data are drawn from a normally distributed population. There are various ways to test if it is plausible tha this is the case, and you should try at least one of them, but with small samples, just where the *t*-test is most useful, it can be difficult to tell. In the end we can also appeal to reason: is there good reason to suppose that the data would or would not be normally distributed?
- When comparing the means of two samples both samples should have approximately the same variance, which is a measure of the spread of the data. You need to check that this is at least *approximately* the case, or have reason to suppose that it should be. (Note: in an actual *t*-test, it is possible to ignore this requirement - see below).
- When we have more than two samples and we use the *t*-test to look for a difference between any two of them, it becomes increasingly likely, the more pairs of samples we compare, that we will decide that we have found a difference because we got a *p*-value that was less than some pre-determined threshold (which could be anything, but is most often chosen to be 0.05) even if in reality there is none. This is the problem of high false positive rates arising from multiple pairwise testing and is where ANOVA comes in. *t*-tests are only used to detect evidence for a difference between two groups, not more. ANOVAs (or their non-parametric equivalent) are used when we are looking for differences between more than two groups.

2.4 The workflow

2.4.1 Open your project

Open your RStuff (or whatever you have called it) project using File/Open Project, navigating to the project folder, then clicking on the `... .Rproj` file you will find there.

If your Rstuff folder is not already a Project, then make it one using File/New Project/Existing Directory - then navigate to your Rstuff folder.

2.4.2 Create a new script

Create a nw notebook script using File/New File/R Notebook Delete everything from below the yaml section at the top. This is the bit between the pair of lines with three dashes. In the yaml, amend the title and add lines `author: "<your name>"` and `date: "<the date>"`. Inside the quotes, add your name and the date.

Now add code chunks to carry out the steps listed below. In between the chunks, add as much explanatory text as you want so that next time you come back, you understand what each code chunk is doing. You can format this text using the simple markdown rules to be found in Help/markdown Quick Reference

2.4.3 Load packages

We typically include a chunk at or near the top of a script that loads any packages we are going to use. If we load all of them in this one chunk it is easy to see at a glance which ones have been loaded.

```
library(tidyverse)
library(here)
library(mbhR)
# if that last line doesn't work, uncomment the next line by deleting the # and run it to ins
# remotes::install_github("mbh038/mbhR")
```

2.4.4 Read in and inspect the data

```
# there should be an 'ozone.csv' file in your data folder
# if not, you should be able to get it from the data folder on Teams or Moodle
filepath<-here("data","ozone.csv")
ozone<-read_csv(filepath)
#glimpse(ozone)
```

What kind of data have we got?

You might also wish to inspect the data using `summary()`. If so, include a code chunk to do this.

2.5 Step One: Summarise the data

With numerical data spread across more than one level of a categorical variable, we often want summary information such as mean values and standard errors of the mean for each level. We can do this by using the `group_by()` and then `summarise()` combination. This first group the data however you want to, then calculates whatever summary information you have requested for each group.

Here we will calculate the number of replicates, the mean and the standard error of the mean for both levels of `garden.location` ie east and west, then store the result in a data frame called `ozone.summary`

```
ozone.summary<-ozone |>
group_by(garden.location) |>
summarise(n = n(),
          mean.ozone = mean(ozone),
          se.ozone = sd(ozone)/sqrt(n()))
ozone.summary
```

```
# A tibble: 2 x 4
  garden.location     n mean.ozone se.ozone
  <chr>           <int>      <dbl>    <dbl>
1 East               10       77.3     2.49
2 West               10       61.3     2.87
```

From these data, does it look as though there is evidence for a difference between ozone levels in the East and the West? Clearly, the ten gardens in the east had a higher mean ozone concentration than the ten in the west. But is this a fluke? How precisely do we think these sample means reflect the truth about the east and the west of the city? That is what the standard error column tells us. You can think of the standard error as being an estimate of how far from the true ozone concentrations for the whole of the east and the whole of the west our sample means, drawn from just ten locations in each part of the city, are likely to be.

Bottom line: the difference between the sample means is about six times the size of the standard errors of each. It really does look as thought east of the city has a higher ozone concentration than the west.

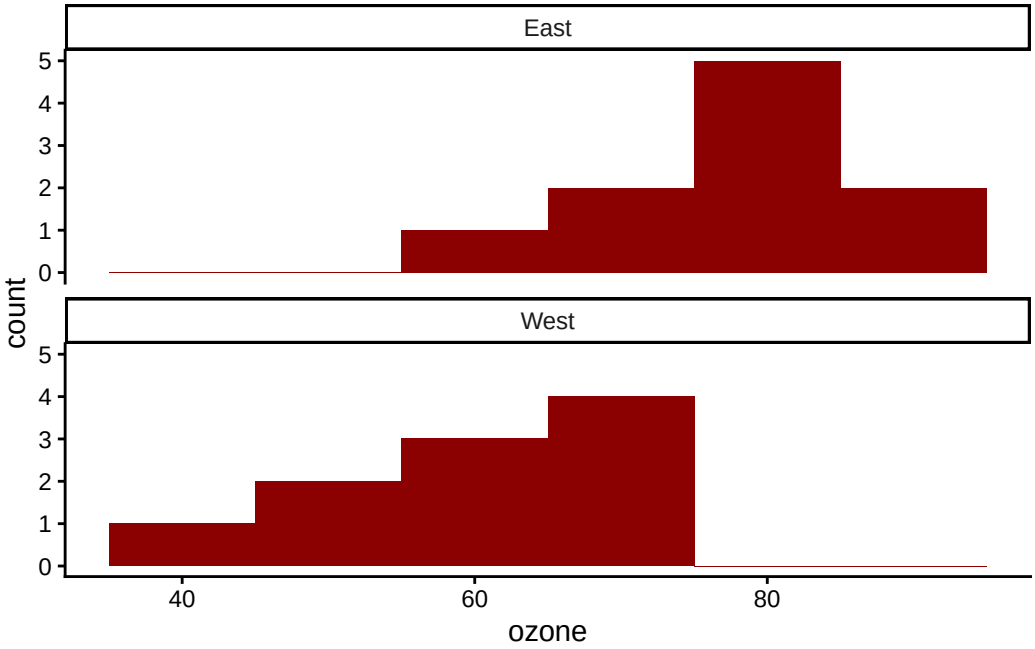
2.6 Step Two: Plot the data

Remember, before we do any statistical analysis, it is almost always a good idea to plot the data in some way. We can often get a very good idea as to the answer to our research question just from the plots we do.

Here, we will

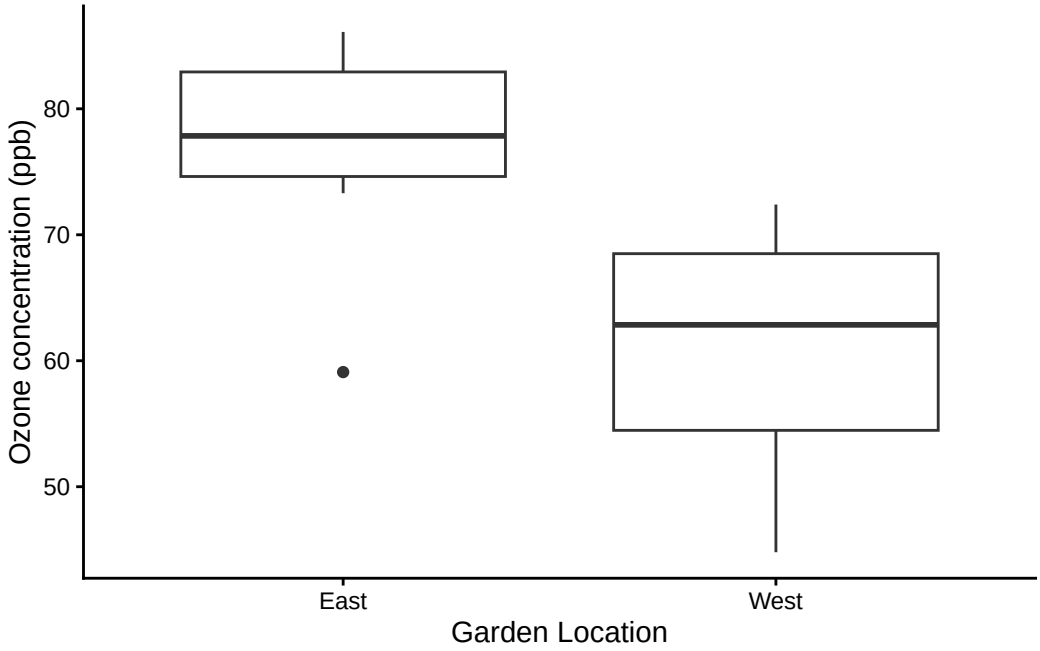
- use `ggplot()` to plot a histogram of ozone levels
- use the `facet_wrap()` function to give two copies of the histogram, one for east and one for west, and to stack the histograms one above the other.
- make the histogram bins 10 ppm wide.

```
ozone |>
  ggplot(aes(x=ozone)) +
  geom_histogram(binwidth=10,fill="darkred")+
  facet_wrap(~garden.location,ncol=1) +
  theme_classic()
```



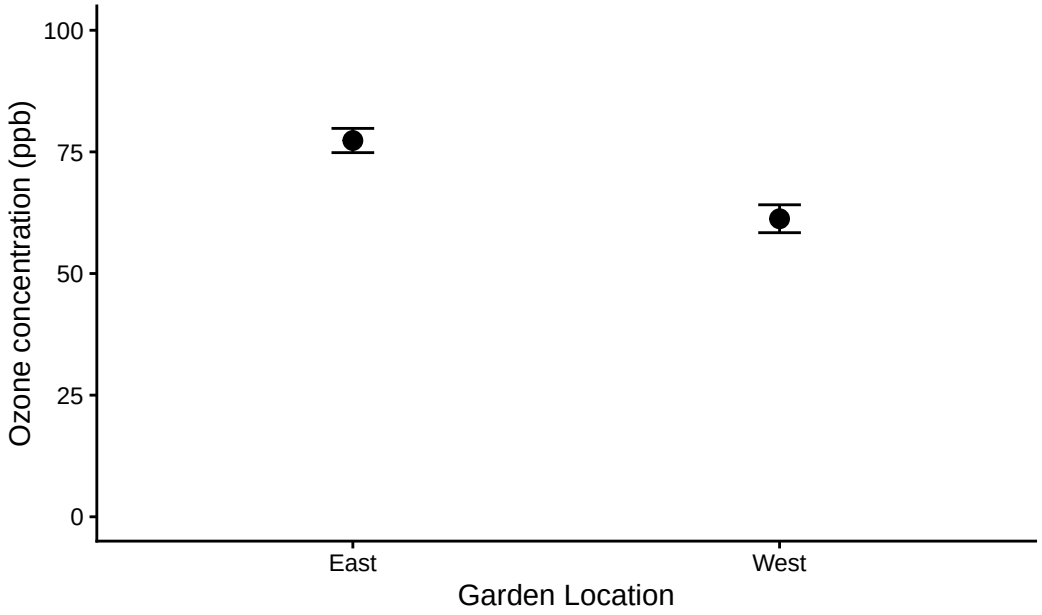
Instead of histograms, we could have drawn box plots:

```
ozone |>
  ggplot(aes(x=garden.location,y=ozone))+
  geom_boxplot()+
  labs(x="Garden Location",
       y="Ozone concentration (ppb)") +
  theme_classic()
```



or as a dot plot with standard errors of the mean included:

```
# for this chart we will use the summary table that we created above.

ozone.summary |>
  ggplot(aes(x=garden.location,y=mean.ozone))+
  geom_point(size=3) +
  geom_errorbar(aes(ymin=mean.ozone-se.ozone,ymax=mean.ozone+se.ozone),width=0.1)+
  ylim(0,100) + # try leaving this line out. What happens? Which is better?
  labs(x="Garden Location",
       y="Ozone concentration (ppb)",
       caption="The data points show mean values, the error bars show plus or minus one stand
  theme_classic()
```



data points show mean values, the error bars show plus or minus one standard error of the mean

Do the data look as though they support the null hypothesis or not?

In addition, do the data look as though each group is drawn from a normally distributed population? One of the types of graphs gives you no indication of that while the other two do. Which is the odd one out? Even when looking at the other two figures, when there are so few data it's kind of hard to tell, no?

Let's now do some stats.

2.7 Step Three: Check that the data meet the criteria required for a *t-test*

2.7.1 Are the data normally distributed?

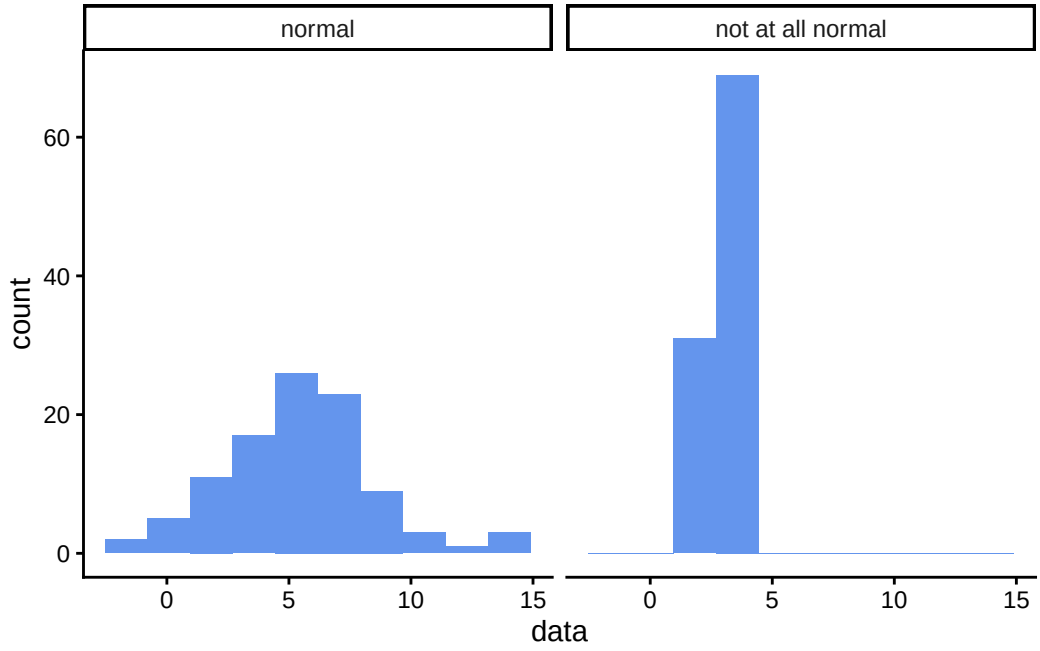
We can go about establishing this in three ways: using an analytical test of normality, using a graphical method and by thinking about what kind of data we have. Let's consider these in turn.

2.7.2 Normality test - analytical method

There are several analytical tests one can run on a set of data to determine if it is plausible that it has been drawn from a normally distributed population. One is the Shapiro-Wilk test.

For more information on the Shapiro-Wilk test, type `?shapiro.test` into the console window. For kicks, try it out on the examples that appear in the help window (which is the bottom right pane, Help tab). One example is testing a sample of data that explicitly *is* drawn from a normal distribution, the other tests a sample of data that definitely *is not*. What p-value do you get in each case? How closely do the histograms of each sample resemble a normal distribution?

```
#first we create a data frame containing the two example data sets
example1<-rnorm(100, mean = 5, sd = 3) # first example from the help pane
example2<-runif(100, min = 2, max = 4) # second example from the help pane

df<-tibble(data=c(example1,example2), distribution=c(rep("normal",100),rep("not at all normal

# then we plot a histogram of each data set
ggplot(df,aes(x=data)) +
  geom_histogram(bins=10,fill="cornflowerblue") +
  facet_wrap(~distribution) +
  theme_classic()
```




```
# and finally we run a Shapiro-Wilk normality test on each data set
shapiro.test(example1) # 100 samples drawn from a normally distributed population
```

```
    Shapiro-Wilk normality test

data:  example1
W = 0.98099, p-value = 0.1588
```

```
shapiro.test(example2) # 100 samples drawn from a uniformly (ie NOT normally) distributed po
```

```
    Shapiro-Wilk normality test

data:  example2
W = 0.93363, p-value = 8.047e-05
```

For the examples above, we see that Shapiro-Wilk test gave a hig p-value for the data that we knew *were* drawn from a normal distribution, an a very low p-value for the data that we knew were not.

The Shapiro-Wilk test tests your data against the null hypothesis that it is drawn from a normally distributed population. It gives a p-value. If the p-value is less than 0.05 then we reject the null hypothesis and cannot suppose our data is normally distributed. In that case we would have to ditch the t-test for a difference, and choose another difference test in its place that could cope with data that was not normally distributed.

Why don't we do that in the first place, I hear you ask? Why bother with this finicky t-test that requires that we go through the faff of testing the data for normality before we can use it? The answer is that it is more powerful than other, so-called non-parametric tests that *can* cope with non-normal data. It is more likely than they are to spot a difference if there really is a difference. So if we can use it, that is what we would rather do.

So, onwards, let's do the Shapiro-Wilk test on our data

We want to test each garden group for normality, so we group the data by location as before and and then summarise, this time asking for the p-value returned by the Shapiro-Wilk test of normality.

```
ozone |>
  group_by(garden.location) |>
  summarise('Shapiro-Wilk p-value'=shapiro.test(ozone)$p.value)
```

```
# A tibble: 2 x 2
  garden.location `Shapiro-Wilk p-value`
  <chr>                            <dbl>
1 East                            0.0953
2 West                            0.599
```

For both groups the p-value is more than 0.05, so at the 5% significance level we cannot reject the null hypothesis that the data are normally distributed, so we can go on and use the t-test. Yay!

2.7.3 Graphical methods - the quantile-quantile or QQ plot.

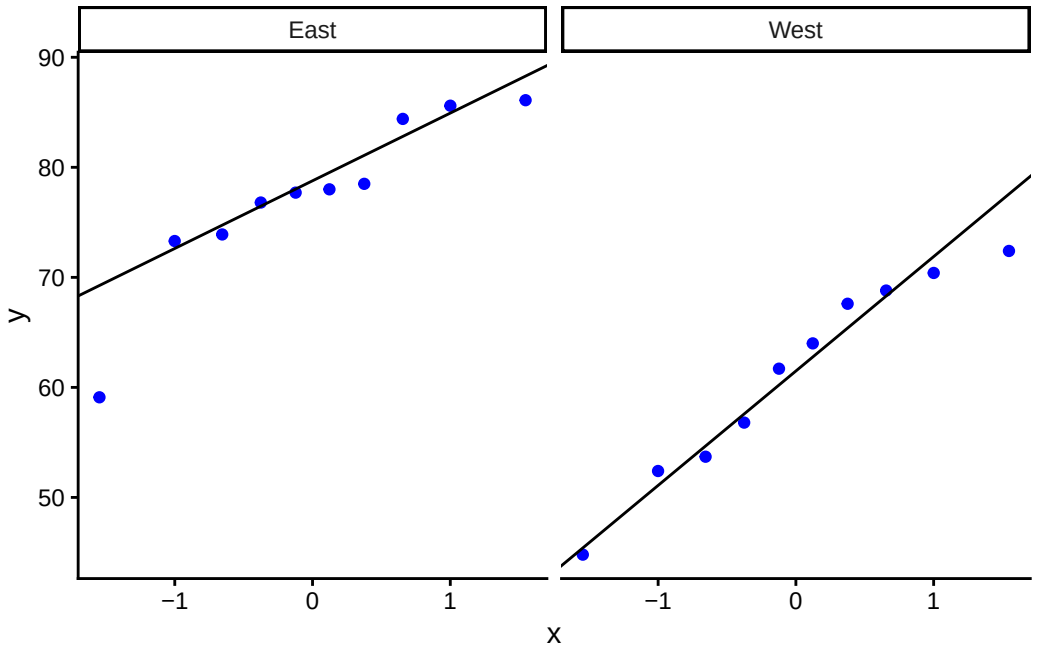
Confession: I don't normally bother with numerical tests for normality such as Shapiro-Wilk. I usually use a graphical method instead.

For an overview of how normally distributed and non-normally distributed data looks when plotted in histograms, box plots and quantile-quntile plots, see this review

We have already seen two ways of plotting the data that might help suggest whether it is plausible that the data are drawn from normally distributed populations. Histograms and box plots both indicate how data is distributed, and for normally distributed data both would be symmetrical. Well, they would be, more or less, if the data set was large enough but for small data sets it can be quite hard to tell from either type of plot whether the data are drawn from a normally distributed population.

A better type of plot for making this call is the quantile-quantile or 'QQ' plot which basically compares the distribution of your data to that of a normal distribution. If your data are approximately normally distributed then a qq plot will give a straight(-ish) line. Even with small data sets, this is usually easy to spot.

```
ozone |>
  ggplot(aes(sample=ozone)) +
  stat_qq(colour="blue") +
  stat_qq_line() +
  facet_wrap(~garden.location) +
  theme_classic()
```



Nothing outrageously non-linear there, so that also suggests we can safely use the *t*-test.

2.7.4 The 'thinking about the data' normality test

As you might have guessed, this isn't a test as such, but a suggestion that you think about what kind of data you have: is it likely to be normally distributed within its subgroups or not? If the data are numerical values of some physical quantity that is the result of many independent processes, and if the data are not bounded on either side (say by 0 and 100 as for exam scores) then it is quite likely that that they are. If they are count data, or ordinal data, then it is quite likely that they are not.

This way of thinking may be all you can do when data sets are very small and any of the more robust tests for normality presented here leave you not much the wiser.

2.7.5 Now for the actual two-sample *t*-test

So, it looks as though it is plausible that the data are drawn from normal distributions. That means we can go on to use a parametric test such as a *t*-test and have confidence in its output.

We can use the `t.test()` function for this. This needs to be given a formula and a data set as arguments. Look up `t.test()` in R's help documentation, and see if you can get the *t*-test to tell you whether there is a significant difference between ozone levels in the east and in the west of the city.

```
t.test(ozone~garden.location,data=ozone)
```

```
    Welch Two Sample t-test

data:  ozone by garden.location
t = 4.2363, df = 17.656, p-value = 0.0005159
alternative hypothesis: true difference in means between group East and group West is not equ
95 percent confidence interval:
  8.094171 24.065829
sample estimates:
mean in group East mean in group West
             77.34              61.26
```

Note the ~ tilda symbol. This means 'is a function of'. So this line means: do a *t*-test to see if there is a significant difference between the ozone levels in the two garden locations.

2.7.6 Interpret the output of the *t*-test.

Study the output of the *t*-test.

- What kind of test was carried out?
- What data was used for the test?
- What is the test statistic of the data?
- How many degrees of freedom were there? Does the number make sense? In a *t*-test the 'degrees of freedom is one less than the number of data points.
- What is the p-value?
- What does the p value mean?
- What is the confidence interval for the difference between ozone levels in east and west? Does it encompass zero?
- Is there sufficient evidence to reject the null hypothesis?
- What does the word 'Welch' tell you - look it up in the help for `t.test()`.

3 Tests for difference - non parametric

A common scenario is that we have two sets of measurements, and we want to see if there is evidence that they are drawn from different populations. For some data types we can use a *t*-test to do this, but for others we cannot.

A *t*-test requires in particular that the two sets of data are normally distributed around their respective means. With *ordinal* data this makes no sense. The mean is undefined as a concept for such data.

To see this , reflect that for a collection X of numerical data, say, 5, 3, 3, 4, and 5 we would calculate the mean as:

$$\bar{X} = \frac{5+3+3+4+5}{5} = \frac{20}{5} = 4$$

But trying doing the same to five responses of a Likert scale survey. Say the responses you had to five Likert items (individual questions) were "strongly disagree", "strongly agree", "mildly disagree", "strongly disagree" and "don't care either way". If you tried to calculate a 'mean' response you would be attempting to add up all these responses and to divide the 'sum' by five, like this:

$$\text{mean response} = \frac{\text{stongly disagree} + \text{strongly agree} + \text{mildly disagree} + \text{stongly disagree} + \text{don't care either way}}{5}$$

This sum makes no sense, I hope you will agree. It makes no sense, not because we are using words to describe our responses, but because, these being ordinal data, we do not know the size of the gaps between the different points on the scale. Is the difference in agreement between the lowest two, "strongly disagree" and "mildly disagree" the same as the gap between the highest two, "mildly agree" and "strongly agree"? We don't know, mainly because 'agreement' is not even something that is well-defined in a quantifiable way - it can't be measured easily using something like a weighing machine. And if this is the case then we shouldn't really be adding these responses up or dividing them by anything.

Nevertheless, ordinal data are very common, since they are typically what is generated by survey data where, for example, repondents may answer a series of questions ('items'), each with typically five possible responses, but maybe more or fewer, these responses being ordinal in the sense that there is a definite order to them. They might encompass responses like those

above, say, or something similar like "very unhappy" to "very happy". They are also common in clinical and veterinary practice where ordinal pain scores are widely used - patients being asked (if they are human) or assessed as to their level of pain on some numerical scale such as 1-10. Note that even if the pain value is recorded as a number it is actually a label, that could just as well have been recorded as one of a series of letters, A, B, C etc or emojis, or any symbol you like. You can't take the average of a set of faces!

Thus, formally, we need another kind of test for a difference. Broadly, we need to use some form of *non-parametric* test where we do not assume that the data has any form of distribution, and where, often, we do not use the actual values of the measurements in our dataset but instead use only their *ranks*. The smallest value would be given rank 1, the next rank 2 and so on.

There are many non-parametric tests out there. Here we will look at only one - the **Wilcoxon Rank Sum Test**, often referred to as a **Mann-Whitney U** test for a difference. We can use this for the scenario we have painted above, where we have two sets of data and we wish to know if these provide evidence that the populations from which the samples have been drawn are in fact different.

3.1 Example

This example uses actual data gathered by an Honours Project student.

The student wished to assess peoples' sense of wellbeing using two different sets of questions designed to assess this. The scales chosen were the Warwick–Edinburgh Mental Well-being Scale (WEMWBS) and the New Ecological Paradigm (NEP) Scale. The student wished in particular to determine whether this sense of well-being was affected by whether a person often and actively frequented the coast and made it and the sea a substantive part of their life in one way or another. ie to find out whether there was evidence to support the notion that it could be good for your mental wellbeing to be by the sea and to make it part of your life.

Each scale used consists of 15 questions or 'Likert items', each of which is answered on a 5 point ordinal scale, where a score of 1 indicates lowest wellbeing and a score of 5 indicates highest wellbeing. Thus each respondent could score anything from 15 to 75.

The student got responses from 374 people, 86 of whom were not "marine" users, while the other 288 say that they *were* marine users. The total scores from each respondent were recorded for each type of survey and stored in the file `wellness.csv` which you should find on the module Moodle site / Teams page. Please put this file in the `data` folder of your R project.

3.2 Script

The first few chunks of this script carry out the same old-same old that we see in script after script: load packages, load data, summarise data , plot data.

You can run this script by running each chunk in sequence, which you do by clicking the green arrow in the top-right corner of each chunk.

Try also to 'Knit' the script by clicking on the Knit button at the top of the script pane.

If you want to create your own script to use with your own data then you can copy and paste into it any code chunks that would be of use and adapt them as necessary.

3.2.1 Load packages

```
library(tidyverse)
library(here)
```

3.2.2 Load data

Our data set is in a `.csv` file which we have placed in the data folder within our project folder.

Note that this data set has been stored in 'tidy' form: each variable appears in only column, and each observation appears in only one row.

```
filepath<-here("data","wellness.csv")
wellness<-read_csv(filepath)
glimpse(wellness)
```

```
Rows: 748
Columns: 4
$ id          <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,~
$ scale       <chr> "WEMWBS", "WEMWBS", "WEMWBS", "WEMWBS", "WEMWBS", "WEMWBS"~
$ marine      <chr> "Yes", "No", "No", "No", "Yes", "Yes", "Yes", "No", "Yes",~
$ total_score <dbl> 48, 51, 37, 39, 38, 40, 54, 39, 54, 39, 51, 50, 49, 51, 54~
```

3.2.3 Summarise the data

We'll calculate the median score (50th percentile) and the 25th and 75th percentile scores. For ordinal data, these summary statistics are well defined, whereas means and standard deviations are not.

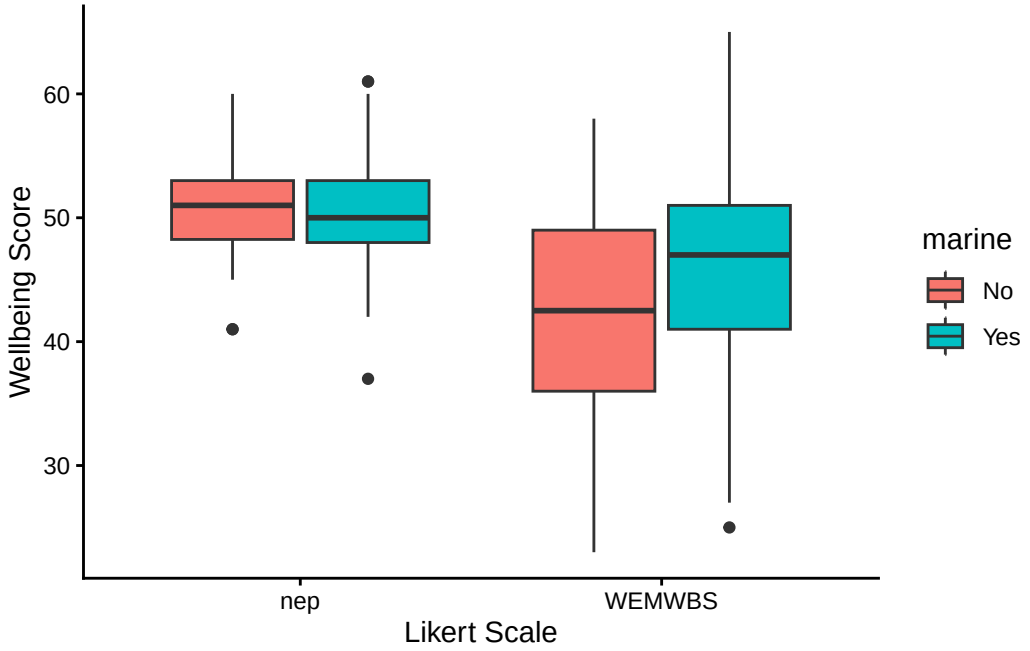
```
wellness |>
  group_by(scale,marine) |>
  summarise(median.score=median(total_score),iqr_25=quantile(total_score,0.25),iqr_75=quanti
```

```
# A tibble: 4 x 5
# Groups:   scale [2]
  scale  marine median.score iqr_25 iqr_75
  <chr>  <chr>         <dbl>  <dbl>  <dbl>
1 WEMWBS No             42.5   36       49
2 WEMWBS Yes            47     41       51
3 nep    No             51     48.2     53
4 nep    Yes            50     48       53
```

3.2.4 Plot the data

Box plots are particularly suitable for ordinal data since they show the 25th and 75th percentiles of the data (the bottom and top of the box) plus the 50th percentile aka the median, which is the thick line across each box. All of these percentiles are well defined quantities for ordinal data.

```
wellness |>
  ggplot(aes(x = scale,y = total_score,fill = marine)) +
  geom_boxplot() +
  labs(x = "Likert Scale",
       y = "Wellbeing Score") +
  theme_classic()
```



Looking at the plot, what do you think each scale suggests about whether proximity to the sea makes a difference to wellbeing?

3.2.5 Wilcoxon-Mann-Whitney U test

First let's pull out the scores as measured by the WEMWBS scale and do a test for a difference between the scores of marine users and those of non-marine users. We can use the `filter()` function to do this.

```
WEMWBS<-wellness |> filter(scale=="WEMWBS")
wilcox.test(total_score~marine,data=WEMWBS)
```

```
    Wilcoxon rank sum test with continuity correction

data:  total_score by marine
W = 9077.5, p-value = 0.0001687
alternative hypothesis: true location shift is not equal to 0
```

The null hypothesis of this test is that there is no evidence that the data are drawn from different populations. In this case, the p-value is very small, so we can confidently reject that null hypothesis and assert that there is evidence, according to the WEMWBS scale that marine use makes a difference to peoples' sense of wellbeing.

Does it make it worse or better? - we can see from the summary table and from the box plot that higher scores are associated with those people who were exposed to a marine environment.

We might report this results as follows, first using a plain English statement of the main finding, and then reporting the type of test use, the value of the test statistci that it calculated and the p value. In this case, because the p value is so small, we would not report its exact value, but simle give an indication of how small it is:

We find evidence, according to the WEMWBS scale, that the wellbeing score is 4.5 or about 10% higher for people exposed to a marine environment (Mann-Whitney U, W = 9077.5, $p < 0.001$).

3.3 Exercise

Adapt the code of the last chunk so that you can do the same test but for data as recorded by the nep scale

3.4 When should I use this Wilcoxon-Mann-Whitney U test?

The test we have used here is an example of a *non-parametric* test. This means that it does not assume that the data follow a known mathematical distribution and, further, that it can be used with ordinal data.

We used the Mann-Whitney U test in particular because we were testing for a difference, and because the factor of interest - marine exposure - had just two levels - Yes or No. This test is only suitable when there are just two levels, so you can think of it as as a non-parametric alternative to a t-test.

In another setting where we still had just one factor (eg zone of a rocky shore) but there were more than two levels (eg low, mid and high zones of the shore) and we decided that we wanted to do a non-parametric test for a difference, then we would probably use the Kruskal-Wallis test, which you can think of as the non-parametric alternative to a one-way ANOVA.

In this example we used the Mann-Whitney U test because the data were ordinal and thus not suitable to use with a parametric test (but see below!). Where we can, we usually try to use a parametric test as they are more powerful than their non-parametric equivalents, meaning, if there is a trend or a difference in the data, they are better able to detect it. However those parametric tests (t-test, ANOVA, pearson correlation, PCA, GLM to name but a few) typically require not only that the data are numerical but also a host of other things, including that they follow a particular distribution, usually (but not always) the normal distribution, and this is often not the case with real biological data. Often, especially with count data, there are

lots of zeros, or the data distribution is heavily skewed, usually to the right. In these cases, providing the data are independent of each other, we can usually still use a non-parametric test such as we have here. it might not be the most powerful test we can use (GLMs are typically way better if you can use them), but it will work.

3.5 Hang on!

The eagle eyed among you may have spotted a massive flaw in the line of argument presented above. We said that ordinal data can't be added up, can't be used to calculate averages and so on. Thus we can't run parametric tests on them and have to look for alternatives, namely, non-parametric tests.

And yet, these non-parametric tests are usually run on the output of Likert *scales* such as we have considered here, where for each person we have a number of Likert *Items* (ie individual questions) that together constitute the *scale*, that each generate a score 1-5, then we *add up the scores* to get a total score. But that means we are adding up ordinal data!!!

It turns out that you actually get much the same results with Likert scale data if you analyse them using supposedly inappropriate parametric tests such as a 2-sample t-test as you do if you use a non-parametric test such as the one we considered here, the Mann-Whitney test.

A study by De Winter and Dodou (2010) shows this convincingly.

de Winter, J. F. C., & Dodou, D. (2010). Five-Point Likert Items: t test versus Mann-Whitney-Wilcoxon (Addendum added October 2012). Practical Assessment, Research, and Evaluation,15, 1–16. https://doi.org/10.7275/bj1p-ts64

For an enlightening discussion of this paper, see this blog by Jim Frost

4 Analysis of Variance aka ANOVA

Material used from Chapter One of Grafen and Hails: Modern Statistics for the Life Sciences

4.1 What is ANOVA?

```
fertilizer <- fertilizer |>
  mutate(FERTIL=as.factor(FERTIL))
```

4.1.1 The basic principles of ANOVA

In a simple case we consider the comparison of three means. This is done by the analysis of variance (ANOVA). In this case we will go through an example in detail and work out all the mechanics, but once we have done that and seen how the output is derived from the input we will not need to do it again. We will use R to do the heavy lifting. We will just need to know when it is appropriate to use ANOVA, how to get R to do it and how to interpret the output that R produces.

4.1.2 The Scenario

Suppose we have three fertilizers and wish to compare their efficacy. This has been done in a field experiment where each fertilizer is applied to 10 plots and the 30 plots are later harvested, with the crop yields being calculated. We end up with three groups of 10 figures and we wish to know if there are any differences between these groups.

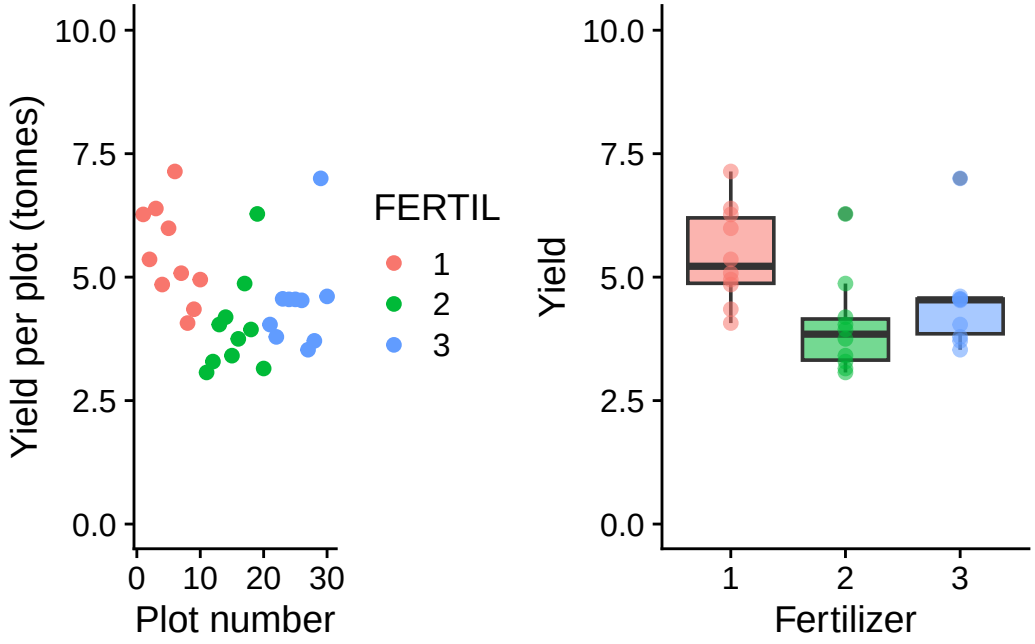
When we plot the data we see that the fertilizers do differ in the amount of yield produced but that there is also a lot of variation between the plots that were given the same fertilizer.

```
g1<-ggplot(fertilizer,aes(x=FERTIL,y=YIELD, fill= FERTIL,alpha=0.1))+
  geom_boxplot()+
  geom_point(aes(colour=FERTIL))+
  scale_y_continuous(limits=c(0,10))+
  labs(x='Fertilizer', y='Yield')+
```

```
  theme_cowplot()+
  theme(legend.position = "none")
```

```
g2<-ggplot(fertilizer,aes(x=plot,y=YIELD,colour=FERTIL))+
  geom_point()+
  scale_y_continuous(limits=c(0,10))+
  labs(x='Plot number',y='Yield per plot (tonnes)')+
  theme_cowplot()
```

```
grid.arrange(g2,g1,nrow=1)
```



4.1.3 What does an ANOVA do?

An ANOVA (ANalysis Of VAriance) analysis attempts to determine whether the differences between the effect of the fertilizers is significant by investigating the variability in the data. We investigate how the variability *between* groups compares to the variability *within* groups.

4.1.4 Grand Mean

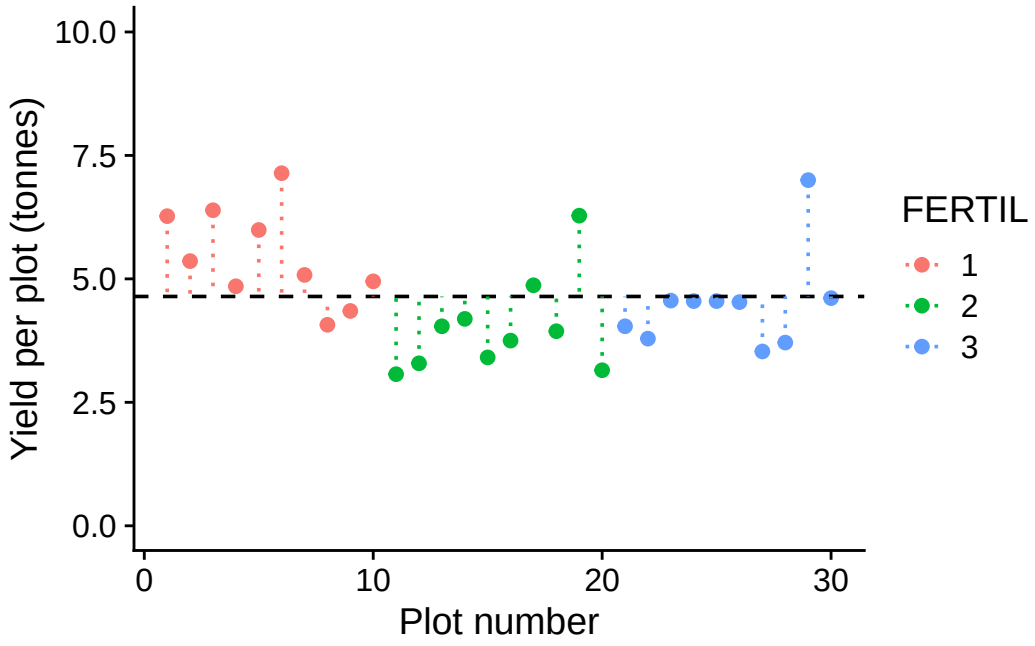
First we calculate the 'grand mean', the mean of the yields across all 30 plots:

```
grand_mean=mean(fertilizer$YIELD)
grand_mean
```

```
[1] 4.643667
```

4.1.4.1 Deviations from the grand mean

```
SST.plot<-g2+geom_hline(yintercept=grand_mean,linetype='dashed')+
  geom_segment(aes(x = plot, y = YIELD, xend = plot, yend = grand_mean),linetype='dotted')
SST.plot
```



4.1.4.2 Mean value of yield for each fertilizer

```
f_means<-fertilizer |>
  group_by(FERTIL) |>
  summarise(fmean=mean(YIELD))
f_means
```

```
# A tibble: 3 x 2
  FERTIL fmean
  <fct>  <dbl>
1 1       5.44
2 2       4.00
3 3       4.49
```

```
fertilizer<-mutate(fertilizer,fmean=c(rep(f_means$fmean[1],10),rep(f_means$fmean[2],10),rep(

f1<-filter(fertilizer,FERTIL==1)
f2<-filter(fertilizer,FERTIL==2)
f3<-filter(fertilizer,FERTIL==3)
```

```
g3<-ggplot()+

  geom_point(data=f1,aes(x=plot,y=YIELD))+
  geom_segment(aes(x=min(f1$plot),y=f1$fmean[1],xend=max(f1$plot),yend=f1$fmean[1]))+
  geom_segment(aes(x = f1$plot, y = f1$YIELD, xend = f1$plot, yend = f1$fmean[1]),linetype='

  geom_point(data=f2,aes(x=plot,y=YIELD))+
  geom_segment(aes(x=min(f2$plot),y=f2$fmean[1],xend=max(f2$plot),yend=f2$fmean[1]))+
  geom_segment(aes(x = f2$plot, y = f2$YIELD, xend = f2$plot, yend = f2$fmean[1]),linetype='

  geom_point(data=f3,aes(x=plot,y=YIELD))+
  geom_segment(aes(x=min(f3$plot),y=f3$fmean[1],xend=max(f3$plot),yend=f3$fmean[1]))+
  geom_segment(aes(x = f3$plot, y = f3$YIELD, xend = f3$plot, yend = f3$fmean[1]),linetype='

  scale_y_continuous(limits=c(0,10))+
  labs(x='Plot number',y='Yield per plot (tonnes)',title="SSE: Error sum of squares")+

  theme_cowplot()
```

4.1.5 Measures of variability

4.1.5.1 SST - Total sum of squares

```
SST=sum((fertilizer$YIELD-grand_mean)^2)
SST
```

```
[1] 36.4449
```

SST is the **total sum of squares.** It is the sum of squares of the deviations of the data around the grand mean. This is a measure of the total variability of the data set.

4.1.5.2 SSE - Error sum of squares

```
SSE<-fertilizer |>
  group_by(FERTIL) |>
  mutate(fmean=mean(YIELD)) |>
  mutate(se=(YIELD-fmean)^2) |>
  summarise(sse=sum(se),.groups = 'drop') |>
  summarise(SSE=sum(sse),.groups = 'drop') |>
  pull(SSE)
```

SSE is the **error sum of squares**. It is the sum of the squares of the deviations of the data around the three separate group means. This is a measure of the variation between plots that have been given the same fertilizer.

4.1.5.3 SSF - Fertilizer sum of squares

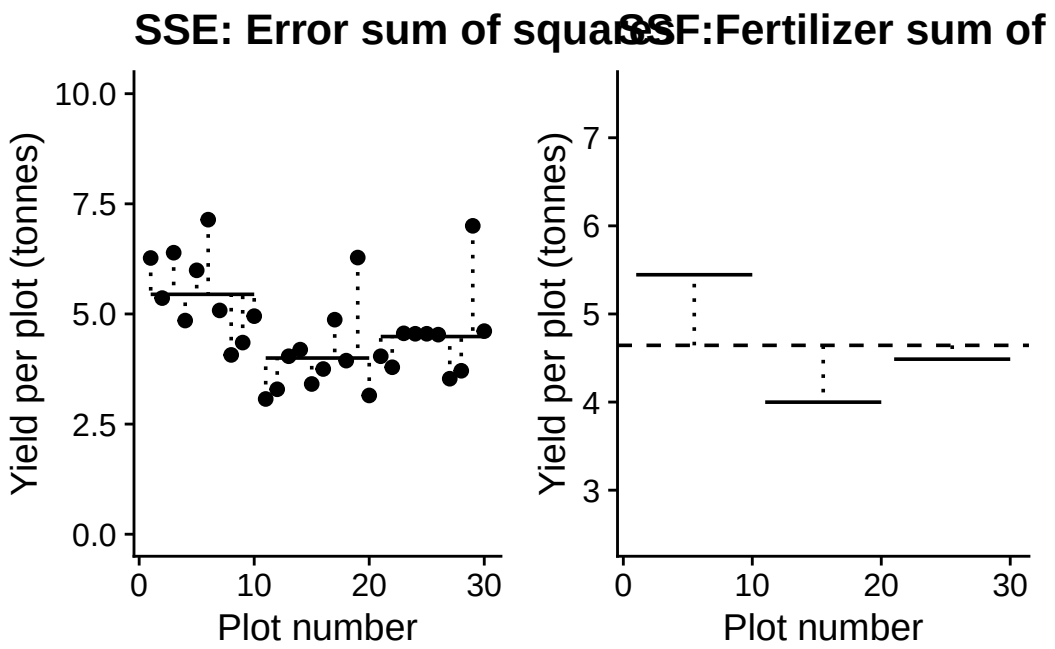
```
SSF<-fertilizer |>
  group_by(FERTIL) |>
  mutate(fmean=mean(YIELD)) |>
  mutate(se=(fmean-grand_mean)^2) |>
  summarise(sse=sum(se),.groups = 'drop') |>
  summarise(SSF=sum(sse),.groups = 'drop') |>
  pull(SSF)
```

SSF is the **fertilizer sum of squares**. This is the sum of the squares of the deviations of the group means from the grand mean. This is a measure of the variation between plots given different fertilizers.

```
g4<-ggplot()+

  geom_hline(yintercept=grand_mean,linetype='dashed')+

  geom_segment(aes(x=min(f1$plot),y=f1$fmean[1],xend=max(f1$plot),yend=f1$fmean[1]))+
  geom_segment(aes(x = mean(f1$plot), y = grand_mean, xend = mean(f1$plot), yend = f1$fmean[
```

```
  geom_segment(aes(x=min(f2$plot),y=f2$fmean[1],xend=max(f2$plot),yend=f2$fmean[1]))+
  geom_segment(aes(x = mean(f2$plot), y = grand_mean, xend = mean(f2$plot), yend = f2$fmean[

  geom_segment(aes(x=min(f3$plot),y=f3$fmean[1],xend=max(f3$plot),yend=f3$fmean[1]))+
  geom_segment(aes(x = mean(f3$plot), y = grand_mean, xend = mean(f3$plot), yend = f3$fmean[

  scale_y_continuous(limits=c(2.5,7.5))+
  labs(x='Plot number',y='Yield per plot (tonnes)',title="SSF:Fertilizer sum of squares")+
  theme_cowplot()
```

```
grid.arrange(g3,g4,nrow=1)
```



When the three group means are fitted, there is an obvious reduction in variability around the three means compared to that around the grand mean, but it is not obvious if the fertilizers have had an effect on yield.

At what point do we decide if the amount of variation explained by fitting the means is significant? By this, we mean, "When is the variability between the group means greater than we would expect by chance alone?

First, we note that SSF and SSE partition between them the total variability in the data:

4.1.6 SST = SSF + SSE

```
SST
```

```
[1] 36.4449
```

```
SSF
```

```
[1] 10.82275
```

```
SSE
```

```
[1] 25.62215
```

```
SSF+SSE
```

```
[1] 36.4449
```

So the total variability has been divided into two components. That due to differences between plots given different treatments and that due to differences between plots given the same treatment. Variability must be due to one or other of these components. Separating the total SS into its component SS is known as partitioning the sums of squares.

A comparison of SSF and SSE is going to indicate whether fitting the three fertilizer means accounts for a significant amount of variability.

However, to make a proper comparison, we really need to compare the variability per degree of freedom ie the variance.

4.1.7 Partitioning the degrees of freedom

Every sum of squares (SS) has been calculated using a number of independent pieces of information. In each, case, we call this number the number of degrees of freedom for the SS.

For SST this number is one less than the number of data points n. This is because when we calculate the deviations of each data point around a grand mean there are only *n-1* of them that are independent, since by definition the sum of these deviations is zero, and so when *n-1* of them have been calculated, the final one is pre-determined.

Similarly, when we calculate SSF, which measures the deviation of the group means from the grand mean, we have k-1 degrees of freedom, (where in the present example k, the number of