# Statistics for Biologists

Dr Ellen Bell

2023-11-08

# Contents

# Chapter 1

# Welcome to the Statistics Coursebook

The aim of this course is to give a basic introduction to key ideas and techniques for statistical analysis. Most people find statistics difficult to learn - statistical analyses, however, play a central role in experimental biology, and a thorough knowledge of statistical techniques is essential in most areas of research. In order to conduct these analyses we will be learning and using the coding language R on the posit Cloud platform (more about this in a bit).

The key to success with statistics is practice. Whilst practical classes and lectures are useful in demonstrating how statistical analyses work and the kinds of problems that can be tackled, it is only with experience that much of statistics becomes understandable. The more you do the easier it will become.

## 1.1 Learning Objectives

- To develop a sound understanding of the principles of statistical testing and probability.
- To be able to use and understand univariate tests of differences and associations.
- to develop confidence in understanding the outputs of univariate tests of differences and associations.
- To learn and gain confidence in use of R for data visualisation and data analysis.

## 1.2 Teaching Layout

This is a 10 credit module which will run from Weeks 2 - 7

- 1 hour lectures run in weeks 2 - 7 which will cover elements of statistical theory
- 3 hour practical sessions run in weeks 2 - 6 which give you a chance to explore some data sets, visualise data and conduct some statistical analyses in R. These are where you will be getting some real hands on experience of data management and analysis.

## 1.3 How to use this workbook

This workbook contains all of the information you require for the practical sessions (it was originally written by Rob Freckleton before being modified by Jenny Gill and then by myself).

The practical sessions have been designed so that the data sets are small, tidy and fairly easy to manage. Should you wish to revisit these data sets, I highly recommend that you download them from the server based platform that we will be using (posit Cloud) and save them locally on your own devise.

## 1.4 Why are we learning R

It is a common misconception, that to be a good biologist you need to 'know' the mechanics of how life works. Questions like; "How does a cell undergo respiration?", "How do kidneys filter blood?", "How do some plants fix nitrogen?" and "How do honey bees communicate the location and quantity of resources to each other?", spring to mind. While these questions are important, they are also, now, fairly well understood. But how did biologists come to their understanding of these mechanisms? The answer to this lies in data.

Good science is based on empirical observations and these observations should be reliably collected and reproducible. Exploration of theories through observations and experimentation leads to the collection of data and analysis and interpretation of data feeds into the bedrock of our understanding of how life works, i.e. the biological sciences.

Hopefully you can see why data handling, analysis and interpretation are important. So why are we teaching you how to handle data using R?

To some of you, the use of programming languages, such as R, will be new. However if you can get into the habit of manipulating and analysing data in R you will be well set on the path to becoming an efficient and effective data analyst. Being confident with data is a key skill in the sciences and will serve you well in many career paths. In addition, knowledge and experience of programming languages such as R are fast becoming key skills in their own right, in science, industry, government and beyond.

In terms of the use of R for data handling and analysis, you may notice that the term **reproducible** reoccurs throughout this workbook. In the same way that

your methods of data collection should be conducted and recorded in such a way that they may be reproduced by others, your data manipulation and analysis must also be conducted and recorded so as to be reproducible. Using R within the posit Cloud interface makes it easy to record your data manipulation and analysis workflow and, if done well, makes it very easy for others to see and repeat what you have done.

## 1.5 Introduction to posit Cloud

So what really is R and what is the difference between R and posit Cloud?

Essentially R is a programming language that is commonly used in statistical computing, data handling, data visualisation and data analysis. Posit Cloud is a cloud based interface for a piece of software called RStudio (we wont be using the non cloud based RStudio here, so we wont explore this software further, however, it is something you may wish to consider downloading and installing on your own device). Posit Cloud uses the R programming language but has a nice user friendly interface and is a great tool for learning how to conduct analysis in R.

We are using the posit cloud rather then RStudio because it means that no one has to worry about installing extra software on their own computers and everyone will be working with the same software versions. You will all need to create your own free accounts on posit Cloud, but first have a look at the short video below introducing you to the interface.

### 1.5.1 Task 1; Create your own posit Cloud account

Now that you've watched the video, create your own Free posit Cloud account here.

### 1.5.2 Creating your first posit Cloud Project

Once you have created an posit Cloud account you should be presented with this window

Under **Spaces** go to **Your Workspace** and under **Projects** create a **New Project > New R studio Project**.

Lets name this project `testing_R`, notice that I have no spaces in my project name. Instead of a space I have used an underscore, there are a number of good habits you should try and adopt when naming projects or files and not including spaces is one of them, we will go over this in more depth in lectures and later chapters. You can rename your project by clicking on **Untitled Project** at the top of the window, and typing in your new project name.

You will see that your new project has three panels with tabs showing the Console, Environment and Files for your project.

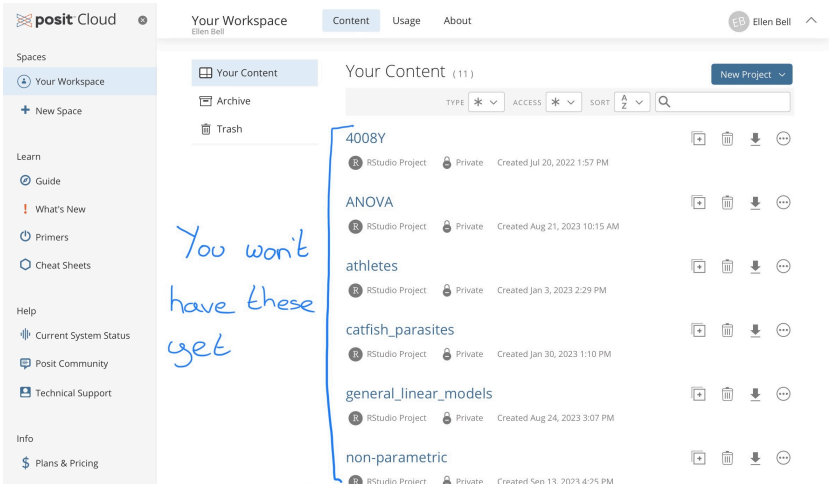


Figure 1.1: Your workspace in posit Cloud

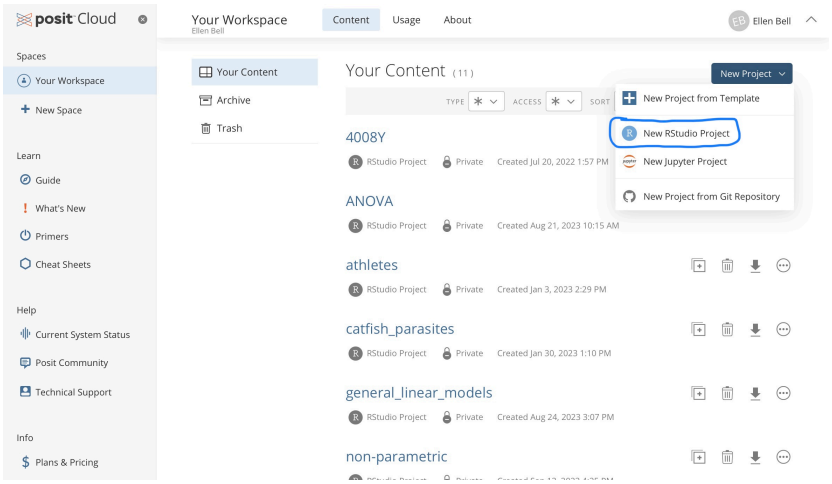


Figure 1.2: Creating a new project in posit Cloud

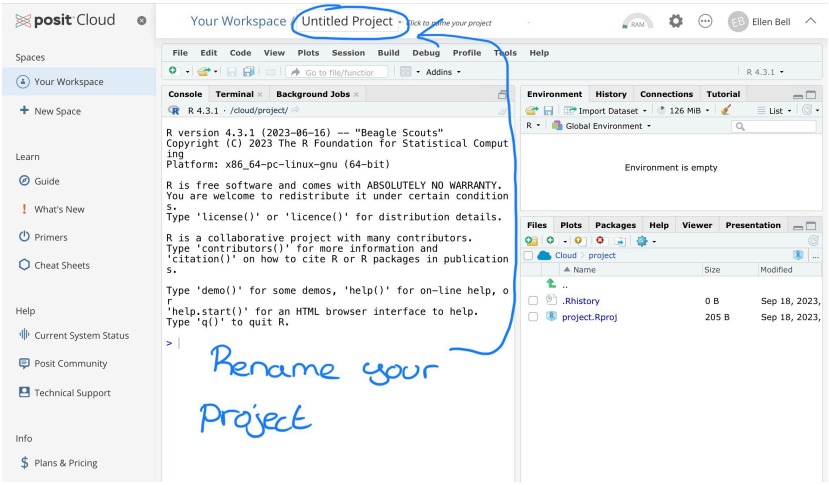


Figure 1.3: Renaming your project

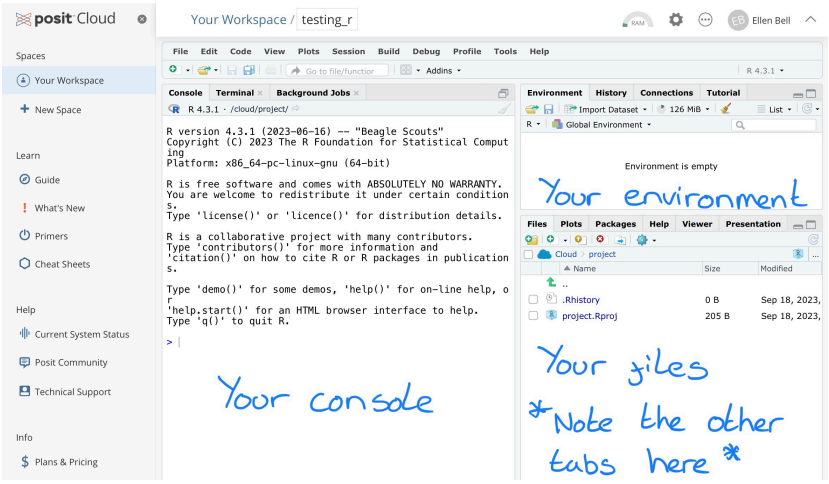


Figure 1.4: The three basic panels of an R Studio workspace

### 1.5.3 A warning on managing your work flow

Your free posit Cloud account comes with some restrictions which you should be aware of. These are;

- You may have up to 50 projects in total
- You are limited to 25 project hours per month
- You have up to 1GB of RAM and 1 CPU per project

For the purposes of our work here these restrictions should not be a problem. But I strongly suggest you don't leave your coursework to the last minute. There are some options of last resort should this happen, you could install RStudio on your desktop for example, or you could open a new free posit Cloud account with a different email address and copy your script over. But with both of these options we run the risk of things getting complicated. Lets avoid that where we can! There are some ways you can increase the efficiency of your workspace which are documented below in Chapter 1.5.4.

### 1.5.4 Maximising the efficiency of your workspace

We know there are limited project hours on our free posit Cloud accounts. But equally we will only be working with very small data sets, so we can tinker with resource usage. To reduce your resource usage. In your project, go to settings (the cogwheel sign), Resources, and drag all of the sliders down to the minimum setting, apply the changes, this will mean the project has to restart. This will mean that running the project for 1 hour will consume 0.5 project hours. We wont be doing anything particularly computationally intensive so there is no need for RAM of CPUs to be higher.

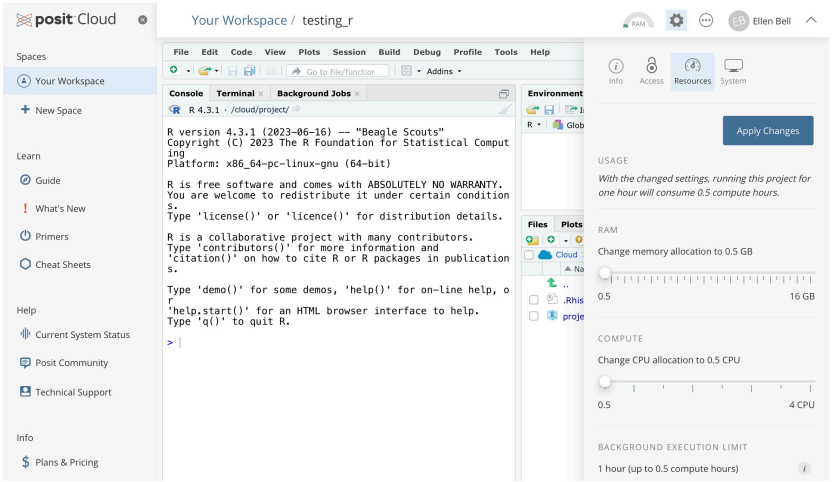


Figure 1.5: How to make your project more efficient

## 1.6 Before you leave!

Log out of posit Cloud!

# Chapter 2

# Some fundamentals to effective use of R

If you are already familiar with R this chapter will be a very swift refresher, you should still work through it though, it will also provide a good overview of posit Cloud.

Login to you posit Cloud account and go into your `testing_R` project (see Chapter 1.5 if you haven't done these steps yet)

## 2.1 Entering commands directly into the console

Lets start by playing with some commands in the console. Type or copy and paste the simple calculation (or command) shown below, into the console. Your text should appear next to the `>` symbol. Press `Enter` on your keyboard, this will instruct R to run the command.

```
5 + 9
```

The two final lines of your output should look like this:

```
> 5 + 9
[1] 14
```

So… your initial command `5 + 9` is shown after the `>`, symbol and the resulting output from R is shown after `[1]`. Don't worry too much about the syntax of `>` and `[1]` here. You can think of `>` as meaning that R is ready to receive a command and `[1]` as R telling you that the answer to the first part of your question is here (in this case `14`).

Try out some other commands… What happens when you input the following?

```
362 * 12
```

```
55 / 5
```

```
(40 / 990) * 100
```

```
4^2
```

See if you can work out what 30% of 735 is using the R console

## 2.2 Using an R script

I have already mentioned the **reproducibility** factor as an advantage of using posit Cloud. This is because you can record and run all of your commands from an R script within posit Cloud. This means that you have a written record of your analysis workflow, what you did to your data at which stage, and you can do all of this without altering the original data files! This is super important because it means that if you revisit your work in a few weeks/months/years you can see exactly what you did AND if someone else needs to rerun any of your analysis or use your workflow on some other data, they can!

So lets go about setting up your new script. You currently have three panels in posit Cloud. If you go to **file > New File > R Script** a new panel will open.

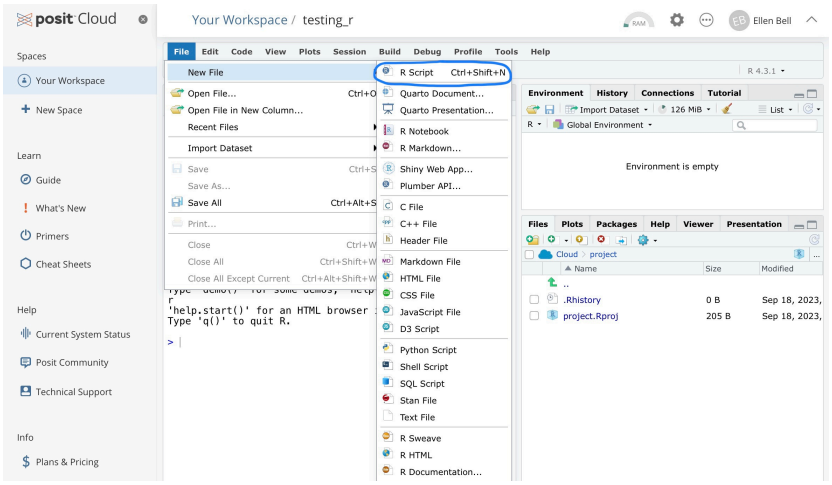


Figure 2.1: How to create a new R script

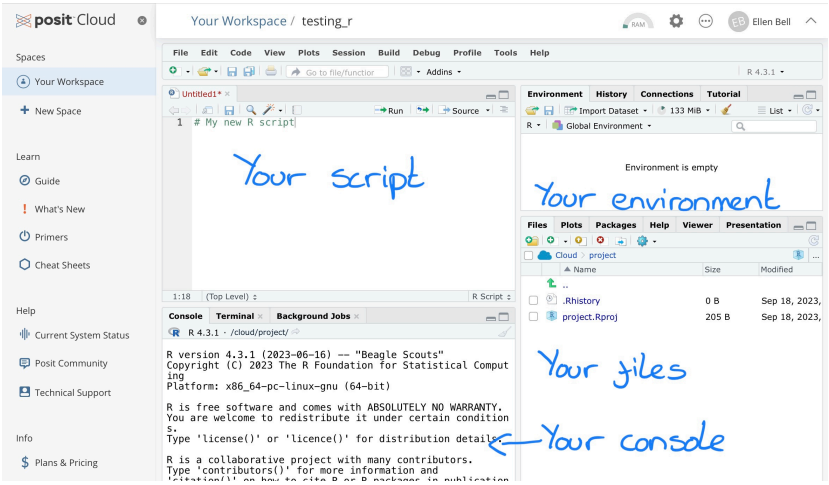


Figure 2.2: The four panels of an R Studio workspace

This new panel is essentially a text file where you can write your commands into a script and then send them down to the console when you want to run them. Lets have a play. Copy the below into your new R script.

```
362 * 12

55 / 5

(40 / 990) * 100

4^2
```

These are all commands you have run before but now if you save your script you will have a text based document with your progress saved. It doesn't make a huge amount of sense to do this now because these are just a un-associated set of calculations and we are just playing with the interface, but you could if you wanted to.

Ok, so now we are ready to execute our commands. You can do this with each calculation individually, line by line, or you can run the whole script.

To run your script line by line, place your cursor on the line you wish to run and you can either;

- Click on the **Run** button on the top right of the script panel
- Press **ctrl + Enter** (or **Command + Enter** on mac)

Or if you wish to run the entire script you can either;

- Manually highlight the script and click the **Run** button or press **ctrl + Enter** (or **Command + Enter** on mac)
- Press **ctrl + A** (or **Command + A** on mac) and then click the run button or press **ctrl + Enter** (or **Command + Enter** on mac)

## 2.3 Creating objects

As we have just seen, R can be used to perform calculations. However, we can use R to perform tasks that are much more complex. In order to do this we are going to have to learn about objects.

Create a new R script in your current and save it under the name `objects`. You should see it appear as `object.R` under the **Files** tab in the bottom right panel. Run the following command using your script.

```
A <- 50
```

What you have essentially done here, is told R to create an object called `A` and to store the number `50` inside it using the syntax `<-`. After running the command you should see your object `A` and `50` pop up under **Environment** and **Values** in the top right hand panel, so you can see that `50` has been stored in `A`.

Lets create some more objects;

```
B <- 6
```

Now see what happens if you run the following command;

```
A * b
```

Oh dear... we have an error...

```
> A * b
Error: object 'b' not found
```

This is because R has absolutely no flexibility with typos. It is looking for an object called `b` when there is no object called `b` in your environment. There is an object called `B`, but to R, `B` and `b` are completely different things. Try running this instead;

```
A * B
```

Hopefully you should now have an output that looks like this

```
> A * B
[1] 300
```

This is a very simple example. But lets try to demonstrate why saving values within objects may be useful. Lets say you are interested in looking at the number of students who get freshers flu in Week 1 of teaching at UEA. You have a class size of 200 students and 15 of them report that they have contracted freshers flu. Use the following script to calculate the percentage of students with freshers flu.

```
# Create two objects, one for your total class and one for those that have flu
# Notice that neither of my object names contain spaces
# If you need a space always use an underscore or full stop
total_class <- 200
with_flu <- 15

# Calculate the percentage of students with flu in week 1
percentage_with_flu <- (with_flu/total_class)*100
```

If you look in your Environment in the top right panel you should see the percentage of students with flu calculated and stored under `percentage_with_flu`. You can use the following command to see it printed in the console.

```
print(percentage_with_flu)
```

Some students have been late in reporting their symptoms. A week later you hear that `7` more students also had freshers flu in Week 1. If you had been typing into the console and not using objects you would have to type this script out all over again. But you don't need to. You just need to change your entry for `with_flu` from `15` to `22`. Do that now and re-run the script. By editing the `15`, R will overwrite the object `with_flu` to represent the new value of `22`. If you run the last line of code then the percentage of students with flu will also update.

This may seem like a very simple example, and it is, but we are still building up your knowledge of R so you will have to take my word for it that objects will be very helpful to you as we progress :). You will also see, that objects can contain a range of different types of data. We will cover some of these in lectures, but for now lets leave objects and have a think about functions and packages.

## 2.4 What are functions and packages?

If you have the coding skills it is possible to do pretty much whatever you like with your data in R. However, why reinvent the wheel trying to write your own complex scripts, when a lot of very clever coders have already written lots of functional bits of code, known as **functions**, for general use. Functions can be thought of as a piece of code that is designed to perform a set task. R comes with

lots of functions already built in, but there are also lots of additional functions that are stored in **packages**.

Packages are containers that can hold sets of functions or data and as the course progresses you will use a range of packages that contain useful functions and data. For example the data visualisation package (which you will become very familiar with later on) `ggplot2` contains ranges of functions which allow you to define how your plot or graph will look, for example `geom_bar` is a function within the `ggplot2` package that contains the instructions required to build a bar chart.

### 2.4.1 Using functions

We actually have already had some exposure to functions. You used the print function earlier when you asked R to print out the value contained within the `percentage_with_flu` object. But lets have a go at using another function.

```
# First of all we need some data
# Notice the syntax, by using c() I have told R to prepare for a list of values
# Each value is separated by a comma
data <- c(2,4,6,8,10,12,14,16,18,20)

# Now I want to know what the total value of data is if I add all the values stored wi
total <- sum(data)
```

In the above piece of code we have created some data, as a list of values, and stored them under the object name data. We have then used the function `sum()`. We have essentially told R that we wish it to perform the function `sum()` on everything contained within the object `data` and store it in another object called `total`. You can see the value within `total` in your environment, can you work out how to get R to print the value of `total` out?

If you ever need help working out how to use a function you can use another function `help()`. Try inputing the code below into your console.

```
help(sum)
```

This should bring up a help file in your bottom right hand panel that describes and explains the use of the function `sum()`.

### 2.4.2 Installing packages

The `sum` function is part of the base R package that comes with any R installation. However there are lots of other useful functions that are held in additional packages. In order to use these functions, you must have the necessary pack-

ages installed in your work space. I already mentioned that we would be using a package called `ggplot2`, lets try installing and loading it in your work space.

With any package you wish to use in posit Cloud you will need to go through a two step process, first of all you need to **install** the package using the `install.packages()` function (see below for usage and then you will need to **load** it using the `library()` function. We only need to install and load a package once per posit Cloud project but I would generally keep the commands for installing and loading packages at the top of the script I'm working on.

Try copying and running the following lines in your script;

```
install.packages("ggplot2") # Install the ggplot package
library(ggplot2) # Load the ggplot package
```

Now we can try to make a simple plot using some functions from the `ggplot2` package and using an example data set called `iris` that is stored in the base R `datasets` package.

Copy the following command into your script and run it

```
# First load the base R datasets in your workspace
library("datasets")

# Then call the ggplot function and tell it that the data set you wish to use is called iris
# We then need to tell ggplot how to map our variables
# The aes function is an aesthetic tool that allows us to define variables for the x and y axes
# We then use the geom_point function to instruct R to built a scatter plot
# A second use of the aes function tells R to color points by species
ggplot(data = iris,aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(aes(colour=Species))
```

Those sharped eyed among you may have noticed that the short script defining our plot goes over two lines and contains a `+` and an indentation on the second line. The `+` and indentation is essentially allowing you to pipe one line of code into the next. This allows you to add more detailed instructions on how you would like your graph to be built and linking together multiple functions. Don't worry if this is a lot to take in at this stage, as we progress with `ggplot2` we will revisit this in more detail.

## 2.5 Analysis hygiene

Hopefully you can see how having a script could make your work all the more reproducible. There are a few good "hygienic" practices that, if you can get into the habit of using, you're future self will thank you.

### 2.5.1 What's in a name?

While you're running analyses you will find youself naming lots of things, including; variables, files, objects, etc. I am going to take a moment to go over a few good rules to follow.

- Keep names meaningful - Over the course of your MSc you will create a lot of files and folders/directories. You will also frequently be naming data variables and objects. You need to be able to easily recognise what is what so informative names are better then arbitrary names. So `girraffe`, `elephant`, `hippo` are better names then `1`, `2` and `3`.
- Make versions obvious - There will be occasions when you want to save new information in a file or object but keep the original untouched. You may be experimenting or you may want the option to revisit to your original work. In these instances it makes sense to save your edits to a new version of the original file or object. If you do this and especially if you are sharing it with others make it clear. I tend to add my initials and a version identifier (e.g. v1, v2) as a suffix to the file name.
- Keep names short - Typing takes time, its boring, and the more you type the greater the risk of a typo. So keep names short. So `trunk_length` is better than `elephant_trunk_length_cm`, there is important information in the second name but you can store this as a comment or a note.
- Avoid spaces - Lots of software packages hate spaces, including R. If you can get out of the habbit of using spaces and instead use `_` your future self will thank you.
- Abide by the rules for R - R does have some rules of its own for naming. Names must **not** start with a number, but numbers may be used provided the first character is a letter. You can also use `.` or `_`, but avoid spaces at all costs. You may use upper or lower case letters, but remember R is case sensitive so will see `Elephant` and `elephant` as different. I try to keep to a lower case and use `_` instead of spaces, it keeps everything uniform and saves me time (this style is known as snake_case).

### 2.5.2 Directories and their structure

Working with text based interfaces like R, where you are writing and running commands, means that we need to have some understanding of the underlying computer architecture. You need to have some appreciation for how files are organised within folders/directories and why its important to know which directory you are working from with R.

Under **Files** in the bottom right panel of your screen you will see a file with the ending .Rproj. This is an R project file which tells R where you are working in the server. It means that R will automatically treat your project location as the **working directory**. If you wish to access files in any of the folders you have

just created you will need to tell R the **path** that it needs to follow in order to access these files. There are two types of path;

- **absolute** paths, which refer to the same location in a file system relative to the root directory (don't worry if that doesn't make too much sense at this stage). For example, in Figure 4.2, if you wanted to access files in the `/work` directory but were already in any of the other directories like `/dev`, an absolute path would work from wherever you were working this would be `/home/jono/work`.
- **relative** paths, which refer to a location in the file system relative to the directory you are currently working in. For example, in Figure 4.2, if you were working in the `/home` directory and wanted to access files in `/work` a relative path would look like `jono/work`. This is fine from the `/home` directory, but the path would not work if you were working in, for example `/lib`.

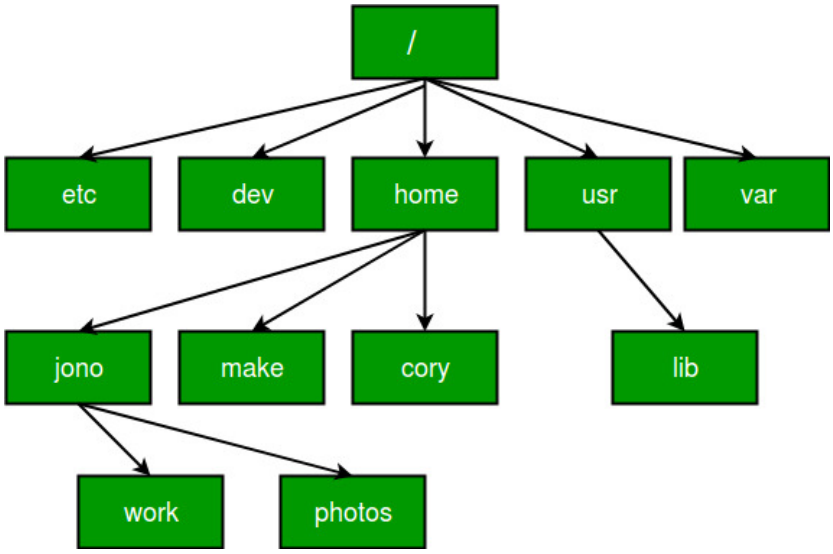


Figure 2.3: Absolute vs Relative Paths

Here we will only be dealing with **relative** paths because you will always be working from within your R project directory. So for example if you wish to access a file in your `data` folder in the R project you would need to provide the path `("data/any_sub_folders_you_may_have_made/the_file_you_wish_to_access")`. We will do this in the next step so don't worry if you don't quite get it yet.

### 2.5.3 Workspace set up

Its always good to keep a clean, well organised and tidy workspace, generally I will start a new project in posit Cloud if I am working on a new series of

questions with a new data set or sets.

We will be moving towards doing some fairly comprehensive data handling so setting up your workspace is going to become more and more important. If you can get into the habit of setting up your workspace properly at the start of each project you will thank yourself later. A well developed project will contain several files including;

- Input data
- Analytical scripts
- Outputs that may include plots and tables

Whenever you create a new RStudio Project, under **Files** in the bottom right panel create a **New Folder**, for each of the following;

- data
- scripts
- figures

**Make sure you copy the names over exactly if you're using bits of code from this workbook, it will minimise errors**

A tidy workspace will help you find the things you need when you need them and will reduce the risk of loosing things.

### 2.5.4 Script set up

Having a uniform script set up can also help you in the future, especially if you are working with multiple scripts. Lets have a go at setting up a clean and tidy script. Create a new R script by clicking on the icon with a little green plus (as you did in Chapter 2.2), below **File** in the top left of the posit Cloud window and select **R Script**. Call your new script `script_hygene`. Its always good to add a few lines of description at the start of your script, this tells your future self and anyone else what the script is doing. Copy over the following.

```
# An example of a lovely clean script that will help me build excellent data hygiene h
# Your name
# Date
```

Then we need to install and load any packages that we might want to use, copy over the following as an example;

```
install.packages("tidyverse")
install.packages("janitor")
library(tidyverse) # Load the tidyverse package
library(janitor) # Load the janitor package
```

You're script is now ready for you to import some data and start analysing with the packages `tidyverse` and `janitor`.

### 2.5.5 Comment on your work

When you are writing a script you can leave notes for yourself. This can be extremely useful if you need reminding on what a particular piece of script is doing. This is a good learning tool while you are still learning how to use R, but its also a good habit to get into for later. As you progress with R you will inevitably start constructing quite intricate scripts, leaving good notes on what each part of the script is doing is very useful for you and for anyone else who may later come to use your script. You can leave notes or comments in scripts if you precede them with `#`. If R sees `#` it will ignore all text on that line that comes after it. For example, copy and paste the following into your script and try running it line by line.

```
# Four squared can be calculated with the following command
4^2
```

In this case I have reminded myself how to calculate 4 squared, everything after the `#` will be ignored by R when you try to run the line.

Comments are also super useful if you only need to use a line of code once. Like when you install packages. for example in your script set up, once you have installed your packages you may see the following;

```
#install.packages("tidyverse")
#install.packages("janitor")
library(tidyverse) # Load the tidyverse package
library(janitor) # Load the janitor package
```

Notice that their is a `#` at the beginning of the `install.packages` line. This is called commenting out. And we do it because you only need to install a package once in your RStudio project but it is useful to have a record of what packages have been installed. You may also need to reload packages after installation, hence why the `library` function hasn't been commented out.

## 2.6 What do I do if I get an error message?

Error messages are almost guaranteed when working regularly with R, they are super common, don't panic when you eventually run into one. In fact you may remember that we already encountered one earlier in this chapter, that particular error was caused by a typo. Generally the most common causes of error messages are;

- A typo (you can reduce risk of this by consistently labeling your files and objects)
- A mistake in syntax (missing or un-closed; `()`, `""`, or missing `,`)
- Missing dependencies (maybe you didnt install or load a package correctly, or maybe if your running a longer script, you are trying to call an object that you haven't created yet)

This is by no means an exhaustive list but when I run into errors its normally because of one of the above. The trick is don't panic and read the error message carefully, it often tells you what is wrong. Then look over your script slowly to troubleshoot. And if you are genuinely stuck and the error doesn't make sense, run it through Google, I can almost guarantee you wont have been the first person to have that error message.

Well done everyone. I hope you have enjoyed this weeks session. Next week is our first workshop and we will start to use some of these new found skills in tandem and start to look at using them on some new data.

## 2.7 Before you leave!

Log out of posit Cloud and make sure you save your script(s)!

## 2.8 References

Firke, S., 2021. Janitor: Simple Tools for Examining and Cleaning Dirty Data. https://github.com/sfirke/janitor. Wickham, H., Averick, M., Bryan, J., Chang, W., D'Agostino McGowan, L., François, R., Grolemund, G., et al., 2019. "Welcome to the tidyverse." Journal of Open Source Software 4 (43): 1686. https://doi.org/10.21105/joss.01686. Wickham, H., Chang W., Henry, L., Pedersen, T.L., Takahashi, K., Wilke, C., Woo, K., Yutani, H., and Dunnington, G., 2021. Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics. https://CRAN.R-project.org/package=ggplot2. Wickham, H., François, R., Henry, L., and Müller, K., 2021. Dplyr: A Grammar of Data Manipulation. https://CRAN.R-project.org/package=dplyr. Wickham, H., Hester, J., and Bryan, J., 2021. Readr: Read Rectangular Text Data. https://CRAN.R-project.org/package=readr.

# Chapter 3

# Workshop 1 - Explorartory Data Analysis

Welcome to the first of the practical sessions in this module. These sessions will all begin with an introduction summarising the theory covered in lectures and then the introduction of a practical session and accompanying data set. Hopefully you have all already read over Chapters 1 and 2, if not do that now before moving on to the rest of this chapter.

## 3.1 Introduction

Statistics are necessary because of the inherent variability between biological objects. For example, people vary in height, plants within a field vary in weight, the lottery numbers are not the same from one week to the next etc. This variation is termed natural variation. A second source of variation is errors that we make in making measurements, termed measurement error e.g. inaccuracies in weighing, taking length measurements.

This variation is important because we cannot be sure whether differences between different groups are due to biological differences or merely the result of chance variations.

On a day-to-day basis there are three important applications for statistical analyses;

1) Statistical analysis revolves around hypotheses. A hypothesis is a tentative proposition which is subject to verification through subsequent investigation. A properly stated hypothesis generates predictions that are then tested by data collection and statistical analyses.

2) Often we have some data, but no idea of whether patterns exist in the

data: e.g. long-term numbers of birds or crop yields. Exploring the data by looking for trends, comparing with weather data etc. will enable us to generate hypotheses to test.

3) Knowledge of statistical analysis is necessary in designing experiments that are robust. In particular to avoid confounding effects, and to control for randomly or systematically varying factors, e.g. environmental clines in field experiments.

The basic measurements or observations we take constitute our data. These may take a number of forms. There are many forms of data but broadly data can be broken down into four major categories. These are:

1) **Continuous data** These data are quantitative/numeric but not fixed and have an infinite number of possible values. These include familiar measurements of such as; weight, height, length etc.

2) **Discrete data** These data are also quantitative/numeric but contain values that fall under integers or whole numbers and cant be broken down into decimals or fractions. For example; shoe size, number of students in a class or days in the week are all discrete data.

3) **Nominal data** These data are qualitative/categorical and are generally used as labels but have no order associated with them. For example; hair colour, marital status and nationality are all nominal data types.

4) **Ordinal data** These are also qualitative/categorical data, but they are subtly different to nominal data in that they can be ordered along a scale. Examples of ordinal data include; letter grades in an exam (A, B, C, D etc), ranking in a competition (first, second, third, etc) or satisfaction scores on a survey (extremely satisfied, satisfied, neutral, unsatisfied or extremely unsatisfied).

It is important to realise which form of data we are dealing with as this can critically affect the type of statistical analyses we can perform. Before we can start any statistical analysis, we can usefully say a great deal about our data through some simple statistical measures and through graphical exploration. One of the first things you should do is to draw a histogram of the frequency distribution of your data. That is you group the observations into intervals (e.g. 0-1cm, 1-2cm, 2-3cm etc.) and then count up how many observations lie in each interval. There are two things to look out for:

1) What is the **shape** of the data? Often your data will follow a 'bell'-shaped normal distribution. The term 'normal' is a technical term, rather than indicating what the data should look like.

2) Are there any **extreme values** in the data? Do any of the values you have measured look suspiciously high or low?

Commonly, for example, you may find that the data you have are skewed. The normal distribution is symmetrical however, often this may not be the case. The

distribution may be, instead, 'L'- shaped (called positive skew) or 'J'-shaped (negative skew). It is essential to determine if this is the case prior to any further analysis.

We have to find some way to describe or summarise our data. There are two basic kinds of measure that we commonly employ. These are termed 'measures of central tendency' and 'measures of spread'.

Measures of central tendency effectively determine the location of the distribution of data, e.g. where the peak is in the normal distribution. There are 3 common measures of central tendency:

1) **Mean** or average of the data. This is an intuitively obvious measure; the average of a sample is simply the sum of all the observations divided by the number of samples.

2) **Median** of the distribution: this is defined as the value of the observation which lies midway in the dataset (i.e. 50% of the values are lower than this value and 50% are higher). Non- parametric statistics often use the median rather than the mean. However its worth noteing that the median is also less sensitive to extreme values.

3) **Mode** of the distribution: This metric is quoted less often and is defined as the interval class containing the largest number of individuals.

Measures of dispersion measure the amount of variation in the data around the mean value. Whilst the mean, median and mode describe where the data are located, measures of dispersion describe the spread and the shape of the data around these values.

We may compare the spread of distributions in a number of ways:

1) **Range** This is simply the maximum value minus the minimum value. The range is not a very good estimate of spread because it is strongly influenced by extreme (high or low) values.

2) **Standard Deviation** or the Variance of the data. These two measures are related (variance is the square of standard deviation). The standard deviation is more usually quoted. We will talk in detail about these in the next few weeks. For now remember that bigger standard deviations imply more spread in the data.

3) **Skewness** It is also possible to measure the degree to which distributions are skewed (i.e. 'L'- or 'J'-shaped). One simple way to do this is to look at the relative order of the mode, median and mean, or indeed just the order of the mean and the median. There is also a statistic (called the skewness coefficient) that measures the skewness of data relative to a normal distribution.

This practical is intended to serve as an introduction to the rest of the course. The aims here are to:

1) Start using some basic functions in R
2) To introduce simple methods for exploring and describing data, preliminary to more detailed statistical analysis.

## 3.2 Practical 1 - Exploring distributions in R

In terms of statistical analysis, the basic message to take home from today's session is that all forms of data analysis, no matter how complex, should begin with a graphical visual inspection. There are a number of reasons for this, including checking whether the data you have collected are consistent (i.e. are there any extreme values and if so, are these correct), checking the spread of the data (i.e. is the frequency distribution of measured values skewed one way or the other), and checking whether, on inspection, you can see possible differences in the average values for different groups of data or treatments (i.e. that may later be tested by statistical analysis). Checks like this are therefore important for ensuring the validity of assumptions made in later analyses (e.g. concerning the spread of data), and making sure that the results of analyses relate to what you actually observe in the data.

### 3.2.1 Task 1: Setting up our workspace and script

I have created teaching workspaces in posit Cloud for all of our workshops, to join the class workspace, you will need to have a free posit Cloud account already set up and then you can click here. You will be asked if you would like to join the space, click `Yes`. In the left hand panel you will see your workspace and then below that a space called `BIO-7026A`, click on this. You will then see a project set up called `bio-7026A_classroom`. This is essentially the classroom/workspace were we will be working during each practical. This is set up on UEAs educators lisence so has no usage limits for you to worry about.

We need to make sure our workspace and script are correctly set up, for todays session we will need to make sure that the packages `tidyverse`, `janitor` and `PerformanceAnalytics` are installed and loaded and I have already made the requisite `data`, `scripts` and `figures` folders which are set up in **Files**. Have a look at Chapter 2.5.3 and Chapter 2.5.4 if you need a reminder of the significance of this or to remind your self on how to install and load packages. Take a moment to explore your folders and workspace.

I strongly recommend that you create a separate script for each workshop, this will mean that you have a premade set of notes for each session and it will help you with future analyses.

If you work through each workshop sequentially you will find that the chunks of code offered can be copied over to your script and if copied and annotated in an ordered way will make up the script you need for each analysis. Sometimes you will need to modify the chunk or rerun the chunk after its been modified,

but much of the code you will need will be presented to you in each workshop (some later workshops may require you to flip back to former chapters).

### 3.2.2 Task 2: Importing the data

Today we will be working with crop data collected from dry weights of wheat plants from plots with or without weeds. The data were collected over two years by dividing a field sown with a crop of winter wheat into 140 plots. The plots were then either left to grow normally, or a mixture of three species of weeds (poppies, sterile brome and cleavers) was sown at the same time as the crop. At the end of the growing season, the wheat was harvested from an area of 0.75m x 0.75m within each plot. The table records the dry weight of wheat removed from each plot in year 1 or year 2 from the weedy or non-weedy plots.

You can download our data set here. Or by downloading it from blackboard.

Once you have downloaded the data you will then need to upload it to your `data` folder in posit Cloud. To do this, go to the project you have just set up in posit Cloud. Under the files tab in the the bottom right panel you will see a little icon with a yellow upwards pointing arrow. If you click on this icon you will be given the option to select and upload a file from your home computer. Do this and file your data within your `data` directory.

Now we can load the data, try running the following chunk;

```
weedy <- read_csv("data/weedy.csv")
```

- Can you identify the function in this line of code?
- What does the `weedy <-` argument do?

Once you have loaded the data, you should see the following in your console;

```
> weedy <- read_csv("data/weedy-data.csv")
Rows: 280 Columns: 2
  Column specification
Delimiter: ","
chr (1): treatment
dbl (1): yield

  Use `spec()` to retrieve the full column specification for this data.
  Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

### 3.2.3 Task 3: Checking the data

Whenever we start analysis on a new data set we need to start by checking the data. This is to make sure that we are aware of any missing data, any anomalous data points (that could be errors) and that R has interpreted the kind of data we have entered correctly. At some point whatever data set you are working

on will almost certainly have been entered manually into a spreadsheet. This creates space for errors to be introduced even if the orrigional data sheets are correct. Try running the following functions, use the # to note down what each of these functions does in your script.

```
# Some functions to play with
nrow(weedy)
ncol(weedy)
colnames(weedy)
str(weedy)
head(weedy)
tail(weedy)
view(weedy)
glimpse(weedy)
tabyl(weedy,treatment)
```

- Can you see how using functions like these to check the data before you start analysing is an important step?

Now that we have explored some super useful functions to check the data, lets actually have a look at the data themselves. Use the above functions to answer the following questions;

- What variables are present in this data set?
- What types of data are represented by these variables (are the data continuous, discrete, nominal or ordinal?
- What do you notice about the way the data has been laid out? Is it long or wide?

### 3.2.4 Task 4: When in doubt - make a histogram

The shape of the frequency distribution of the data is a key element in determining how we analyse data. We need to determine (i) whether the data are distributed symmetrically; and (ii) whether there are any extreme values in the data. To do this we are going to create a frequency distribution of the data and display this as a histogram. To recap, a frequency distribution is obtained by grouping the data into intervals and counting the number or proportion of observations that fall within each interval. So how do we make histograms in R? When you used the `tabyl()` function above you might have noticed that there are four categories listed under treatment (your experimental treatments). For now we only want to work with one of these treatments so run the following chunk;

```
weedfree1 <- weedy %>% # pipe the weedy data frame into the next function
  filter(treatment == "weedfree1") # filter the piped data frame and only keep treatme
```

A note on syntax: you might have noticed the %>% syntax here. This is a piping variable. This is essentially performing the same function as the + that we used when using ggplot in Chapter 2.4.2. This allows you to take an object and to add more detailed instructions on the job you would like performed, this is a very simple use of a pipe but we will be building more complex examples later on.

Try using some of the data checking functions that we used earlier to check your new data frame `weedfree1`. Are you happy that you have filtered out one of the four treatments from the `weedy` data frame?

Now we are ready to make our first histogram. For this we will call on some functions within the `ggplot2` package. We had a quick play with ggplot in the last chapter so this wont be completely unfamiliar to you. Try running the next chunk;

```
ggplot(data = weedfree1, aes(x=yield)) +
  geom_histogram(bins = 6, fill="cornflowerblue")
```

- Can you work out what each line of code is asking for?
- What are your functions here? What do the + signs do?
- What happens when you change the bin number (have a play with this)?
- Use # to add notes to your script.

Now we have looked at the code we can start to ask questions about the distribution;

- Do the data appear to follow a normal distribution?
- Are there any extreme values?
- What would the vegetation in these 70 plots look like, in comparison to one another?
- Now try using the filter and ggplot functions to make histograms for your other treatments, how do these histograms compare?

## 3.3 Descriptive statistics in R

We discussed a range of measures of central tendency and spread in lectures. Here we will learn how to derive some of these quickly and easily in R. But first just to recap;

- **Mean:** This is simply the average of the data and measures central tendency, i.e. where the data are located.
- **Standard Deviation:** This measures the spread of the data about the mean, and is calculated as the square root of the variance, which itself is the average of the squared deviations about the mean.

- **Skewness:** Although not often quoted, this statistic allows us to determine the degree to which the distribution of the data is "L" or "J" shaped.
- **Median:** This is an alternative measure of the central tendency. It is defined as the middle value of a set of ordered data. If we have an even number of observations then it is defined as the average of the middle two datapoints.
- **Mode:** This is the most frequently occurring value in the dataset.

### 3.3.1 Task 5: Descriptive statistics

We are going to use some functions from within the `tidyverse` package to summarise our yeild data, using descriptive statistics.

Try running the following chunk;

```
weedy_summary <- weedy %>%
  group_by(treatment) %>%
  summarise(n=n(),
            mean=mean(yield))
weedy_summary
```

- Add comments to each line of code to describe what each piece does.

So we have a mean for our data, but we can produce many more descriptive statisitcs in one place. Try editing your script so that it reads as follows;

```
weedy_summary <- weedy %>%
  group_by(treatment) %>%
  summarise(n=n(),
            mean=mean(yield),
            sd=sd(yield))
weedy_summary
```

Hopefully you can see we now have a standard deviation (sd) of the yield. Now modify your code with the following self explanatory functions; `median()` and `skewness()`.

### 3.3.2 Task 6: Finding the range and the mode

Finding the range and the mode in R is slightly more complex then using a single function.

We can find the range by adding the following line to the `summarise()` function above;

```
range=max(yield)-min(yield) # subtracting the smallest yield value from the largest yield value t
```

To calculate the mode we will have to create our own function (R doesnt have a base function to calculate the mode). Don't worry if you dont 'get' this right away.

If you run the following chunk of code independtly to the above `summarise()` function, you will create your own function called `mode()` in base R that calculates the mode;

```
mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# This is a stage by stage translation of the above code.
# mode <- function(v): First we define a function named mode that calculates the mode of a given
# uniqv <- unique(v): This line creates a new vector uniqv by removing duplicate values from the
# tabulate(match(v, uniqv)): The match() function is used to find the position of elements in the
# which.max(...): This function is applied to the result of the previous step (the frequency coun
# uniqv[...]: Finally, the uniqv vector is indexed using the position of the mode (identified by
# So the mode function first finds the unique values in the input vector, then counts the frequen
# A note of caution if there are multiple modes, this function will only return the first one it
```

With the above function, we can now compute the mode of our dataset. Try incorporating the following into your `summarise()` function.

```
mode=mode(yield)
```

### 3.3.3 Task 7: Interpretation

We have done quite a lot of data exploration today, now we need to finish up by checking our interpretation.

Take a look at your `weedy_summary`, just focus on the first row for now, this should be `weedfree1`, and try to answer the following questions;

- What is the mean of the distribution?
- Is there any evidence that the data are skewed? (remember a normal distribution has a skewness coefficient of zero).
- How do the median and mode compare to the mean (are they greater or less than the mean)?

Having completed some interpretation of the data on the weed free plots from year one, we are going to repeat the interpreatation with the data from one of

the weedy plots.

- Remind yourself of what the histogram looks like for weedy1 then go back to your `weedy_summary`
- Are the data in weedy1 skewed and, if so, in which direction?
- What would the vegetation in these 70 weedy plots look like, in comparison to one another?
- How does the mean of these weedy1 data compare with that of the weedfree1 data?
- How does the standard deviation compare between weedy1 and weedfree1?
- What do these differences in mean and SD suggest about effects of weeds on wheat growth?
- Repeat the above with the data from year 2.
- How do the differences between the weedy and non-weedy plots compare between the two years' experiments?

## 3.4 Conclusion

Exploring the distribution of your data helps you to both understand the data and select the correct statistical tests. In analysing your own data, you should repeat the procedures that we have gone through here so that you know the shape of the frequency distribution for each variable, as well the mean and the standard deviation of the data. In next week's practical the importance of checking these characteristics will become apparent.

## 3.5 Before you leave!

You may wish to keep your script for personal reference. You should see your script saved in the appropriate folder in the `Files` panel, it will have a `.R` suffix. If you tick the box next to your `.R` file you can then click `More`, `Export` and `Download` to download your script to your local machine. You can then save this somewhere sensible and use it in your own posit Cloud space or RStudio space if you want to revisit the material. I highly recommend doing this as the posit Cloud workspaces will only remain open for this academic year.

Please remember to log out of posit Cloud!

## 3.6 References

Wickham, H., Averick, M., Bryan, J., Chang, W., D'Agostino McGowan, L., François, R., Grolemund, G., et al., 2019. "Welcome to the tidyverse." Journal of Open Source Software 4 (43): 1686. https://doi.org/10.21105/joss.01686.

# Chapter 4

# Workshop 2 - Sampling Distributions and Testing for Differences

## 4.1 Introduction

In the lecture last week we noted that one of the most important applications for statistical analyses is assessing whether differences exist between groups. In the next two sessions, we are going to concentrate on a series of statistical analyses that allow us to determine whether there are differences between groups of data. In developing an understanding of this form of statistical analysis, we have to be clear about two statistical concepts. These are the ideas of sampling and of probability.

### Sampling

What is the density of trees in the Amazon? We cannot measure this density precisely – we have estimate from samples. When presented with a large number of units that we could potentially measure, we can only usually measure a subset. In statistical language, this larger number is known as the population, which is the collection of units to which we want to generalise a set of findings. The subset taken from this population is termed a sample, and is defined as any portion of the population selected for study. This distinction is important, and is emphasised by using different symbols for the population and sample parameters (mean and standard deviation in particular). Importantly, if we calculate a statistic, then we are trying to estimate a population parameter.

## Sampling error

The key goal in sampling is to select a portion of the population that displays all of the characteristics of the population. For example, if we take a sample of 100 British women, then we would expect measurements taken from these (height, weight etc.) to be representative of the population of British women. Critically, however, this sample would not have exactly the same mean, standard deviation etc. There is always some error in such estimates. Statistical analysis is designed to cope with such error.

## Probability

In a sample, individuals (e.g. tall women, dense forest patches) that are more abundant will be encountered more frequently. The frequency distributions we looked at last week effectively represent the probability distribution of our observations. The height of a bar represents the probability of observing that class.

The normal distribution was introduced last week, as well as the standard deviation that measures the spread of this distribution.

## Variance

$$S^2 = \frac{\sum(x - \bar{x})^2}{(n-1)}$$

Variance ($S^2$) is calculated as the sum of squared deviations from the mean. The standard deviation (S) is then calculated as the square root of variance. It is easy to see how these both measure the spread in the data: the variance is the difference of each data point from the mean, squared (to remove the sign) and averaged. Square-rooting variance to give standard deviation allows this number to be presented in the correct units.

## Standard Deviation

$$SD = \sqrt{\frac{\sum(x - \bar{x})^2}{(n-1)}}$$

The standard deviation (usually abbreviated to SD or S) defines some important probabilities for the normal distribution, as shown in the graph on the left. For example, 95% of the data lie within the range given by $\pm 1.96$ standard deviations (an approximation of $\pm$ 2 standard deviations can therefore often be used as a rough guide to where 95% of the data lie).

We noted earlier that if we take a sample of a given size from a large population then we will not get the same estimate of the mean and SD each time. If we wanted to test a hypothesis about the mean then this could be a problem: how do we know whether variations between groups are due to chance variations

or real differences? Luckily we have a way of assessing this: if we estimate a mean from a sample of a population, then the mean and variance can be used to estimate how variable we would expect the sample mean to be, on average. This depends on:

1) The sample size: low sample sizes give more variable estimates of the population mean.

2) The variability in the population: the more variable the population, the more variable the sample mean will be.

## Standard Error of the Mean

$$SEM = \frac{SD}{\sqrt{n}}$$

Standard error of the mean of standard error (SEM or SE) is the parameter which measures how well the sample mean is likely to reflect the population mean. It does this by dividing sample SD (the spread in the data) by the size of the sample (bigger samples are more likely to provide accurate estimates of the mean). Large SE's can therefore result from widely spread data, small sample sizes, or both. A large sample from a narrow distribution will give a small SE, indicating that the sample mean is very likely to be close to the population mean.

## Confidence Intervals

$$CI = \bar{x} \pm (t * SEM)$$

If we have estimates of the mean and SD of a population, we can state the probability that the population mean is a given value. This is estimated by the t-distribution, which accounts for error in the estimates of the mean and the population SD. A good example of how this works is the estimation of confidence intervals. If, from a sample size of 20, we estimate a mean of 50, and a SEM of 5, then the t distribution can be used to calculate where we would expect the true population mean to be 95% of the time (which turns out to be between 40 and 60, i.e. the difference between the population and sample mean would be greater than 10 only 5% of the time).

This is a powerful statement: from the sample data we can calculate the probability with which we would expect the population mean to be greater or less than a given value. Two things are important here:

1) Sample size: this is used to calculate the SEM and in selecting the appropriate value of t. For a sample size of n, the value of t has n-1 degrees of freedom (d.f.). All statistical tests use either d.f. or sample size to account for the likely accuracy of estimates.

2) Probability: in statistical analysis we say that an event is unlikely, and hence likely to be of significance, if there is less than a one in twenty probability of the result arising by chance (i.e. the probability of the event is 5% or less). Note that this means that if the probability of an event is <0.05, then this result can have arisen by chance one in twenty times.

## T-tests

**One sample t-test**

$$t = \frac{\bar{x} - \mu}{SEM}$$

The one-sample t-test is a simple form of the t-test. It compares a sample of data to a pre-defined value. If the probability of the sample mean ($\bar{x}$) being the same as the pre-defined value ($\mu$) is <0.05, then we can state that there is a statistically significant difference between our sample mean and that value.

**Two sample t-test**

2-sample t-statistic

$$t = \frac{\bar{x}_1 - \bar{x}_2}{SE}$$

Pooled estimate of variance

$$s^2 = \frac{\sum(x_1 - \bar{x}_1)^2 + \sum(x_2 - \bar{x}_2)^2}{(n_1 - 1) + (n_2 - 1)}$$

Pooled SE

$$SE = \frac{SD}{\sqrt{n}_1 + n_2}$$

The most common use of the t-statistic is in comparing two samples to assess whether they are drawn from the same distribution, i.e. whether or not they are different. To do this, the 2-sample t-test first compares the means of the two samples; a small difference between these means makes it more likely that they are from the same distribution, however, this will also depend on the spread of the data. The t-test therefore also calculates a pooled estimate of the variance by taking an average of the squared deviations of the two samples. This pooled estimate of variance is then used to calculate the pooled standard error for the difference in the means. The t-statistic is then the difference between the means divided by the pooled SE.

Both the one- and two-sample t-tests have some very important assumptions:

1) The data are drawn from a normal distribution.
2) In the two-sample analysis, it is assumed that the two samples are drawn from distributions with the same variances.

Violation of either of these assumptions **could** lead to invalid results.

But don't panic - we will cover how to check these assumptions very straightforward.

## 4.2 Some key terms for todays practical

- **Standard deviation:** the standard deviation is a measure of the spread of the data. It is calculated from the variance, which is defined as the average squared deviation from the mean. Importantly, and in contrast with the standard error, the standard deviation is unaffected by the sample size.

- **Standard error:** the standard error is a function of the sample size, and is defined as the standard deviation divided by the square root of the sample size. Whilst the standard deviation measures the spread of the data, the standard error measures how the combination of the spread of the data and the number of measurements in our sample affect how well we are likely to have estimated the mean, i.e. it measures error in our estimate of the mean.

- **Degrees of freedom:** in most parametric statistical tests we need to account for sample size, because larger samples will generally give better estimates of the mean and variance. Most statistical tests use Degrees of Freedom (df) to represent sample size. The df for an estimate is the sample size minus the number of additional parameters (e.g. mean, SD) estimated for that calculation. As we estimate more parameters, the available degrees of freedom decrease. It can also be thought of as the number of observations (values) which are freely available to vary given the additional parameters estimated.

- **Statistical significance:** When we do a statistical test we cannot state categorically that a hypothesis is true or false - rather, we have to state the probability that the hypothesis is true. For example, in the one-sample t-test that we look at below, we calculate a difference between our sample mean and another value. Then, based on the combination of the size of this difference, the variability within our data and the sample size, we calculate the probability that this difference is zero (i.e. that our mean is not different from the value we specified). When we do this we need some cut-off value to be able to say if our probability is big enough to regard as important - and Fisher decided that this would be a probability of $P < 0.05$, that is a less than one-in-twenty probability that such a large difference would arise by chance alone.

## 4.3 Practical 2 - Sampling and Testing for Differences

Most data are collected as samples from larger collections or populations. The basis for all statistical analyses is attempting to infer population parameters (e.g. the population mean, variance, standard deviation etc.) from the sample data. This week's practical is intended as an introduction to analysing data using simple tests on samples from populations.

The importance of exploratory analyses of data was stressed last week: today we will apply some of these ideas in analysing data from a field study.

### 4.3.1 The Data

As many of you are new to Norfolk, we are going to analyse some data on the yields of sugar beet, which is something of a Norfolk specialty. Sugar beet is a root crop which is grown across East Anglia and into the Midlands. It is sown in March-April and then harvested in October. The roots are then used to make sugar. Being sown in April, the crop is rather vulnerable to attack by pests. In particular, plants are attacked by aphids, which transmit viruses to the crop. The main symptom of these viruses is a yellowing of the crop, and the viruses have the potential to reduce yields by up to 40%. However, sometimes the crop does not succumb to the virus and yields can be unaffected. These data were taken from 56 fields in Norfolk & Suffolk, half of which were determined to have viral infection, half of which did not. The data report the dry weight of sugar beet processed from each field at the same factory. The investigators wished to ask 2 questions of the data:

1) Did yields in this year deviate from the long-term average yields for the region?
2) Did the presence of viruses affect the yield of sugar beet in this year?

Before starting, answer the following questions:

- What constitutes the population(s) in this study?
- What hypotheses correspond to the two questions posed above (remember a good hypothesis should make a prediction)?
- What statistical tests would you use to test these hypotheses?

The data are stored in a .csv file called `sugar_beet_virus.csv` and can be found in this classroom workspace on posit Cloud in the `data` folder. For each field, we have a measure of yield and whether or not it was infected by virus.

### 4.3.2 Task 1 - Setting up your workspace

Go to the classroom workspace in posit Cloud and open up a **New** script, give this script a sensible name and save it in your `scripts` folder. You will need to make sure the packages `tidyverse` and `car` are installed and loaded to complete

todays workshop, check Chapter 2.5.3 and Chapter 2.5.4 if you are unsure how to do this.

You will probably have lots of bits and pieces stored in your environment from last weeks session. As we are starting a new analysis with new data it makes sence to begin with a nice clean environment. Don't worry about loosing things from your environment, provided you have **saved your script** in your `script` folder and have the raw data .csv file you will be able to reproduce all of those analyses from last week. If you run the following piece of code, everything stored in your environment will be removed;

```
# Clean up your environment
rm(list = ls())
```

### 4.3.3 Task 2 - Checking the data

Load the data into your workspace. Make sure you load it into an object named `sugar_beet_virus` and then try performing some routine checks to make sure R has interpreted the data variables correctly and all expected data is present and accounted for. You can use some of the functions we explored in Chapter 3.2.3 if you need some help with this.

The data consist of measures of yields per field and then an indicator of whether the field was infected by virus or not.

- How would you describe these the data types in these two variables?

### 4.3.4 Task 3 - Making some basic plots

We will now use the `filter()` and `ggplot` functions to split the data sets and make some histograms. Try using the following chunk to look at yields of sugar been from fields infected with virus;

```
sugar_beet_virus %>% # pipe our data set into the filter function
  filter(virus =="v") %>% # filter out data from virus infected fields
  ggplot(., aes(x=yield)) + # note that there is a . here instead of a data set. This is because
  geom_histogram(bins = 6) # plots a histogram with 6 bins
```

You might notice that we haven't assigned any of this to an object. We are also doing a lot of piping here. As you get more confident with code you will find that for quick explorations of the data, this method of analysis is much cleaner and a little faster. Now see if you can manipulate the above code to report a histogram for fields that aren't infected with virus.

- Are the data approximately normally distributed?

### 4.3.5 Task 4 - Produce some descriptive statistics

Try to calculate the mean, standard deviation and standard error of the mean for yields from virus infected fields and fields without virus detected in them. Try using the `group_by()` and `summarise()` functions to do this, the code from Chapter 3.3.1 will help you with two of these. See if you can figure out how to manipulate this code to give you the standard error of the mean. The equation for the standard error of the mean is given above.

- Is there any preliminary evidence that the virus has affected the yields?

It is incredibly important to begin all statistical analyses with a visual inspection of your data, to assess the extent of any differences or associations and **critically**, to assess whether the assumptions of particular tests are met. In parametric tests of differences, data should be approximately normally distributed. If the data are skewed, you can explore whether the skew is influencing the outcome of the test, by comparing tests with and without datapoints driving the skew and/or comparing parametric and non-paramteric tests (we will cover these later in the course).

Parametric statistical tests make certain assumptions about the shape and distribution of the data. Specifically, in the case of the one-sample t-test, our data should ideally be normally distributed. You will sometimes see the "Kolmogorov-Smirnov" test used to explore the assumption of normality. The null hypothesis is that the data are drawn from a normal distribution, so $p<0.05$ means that the data are statistically significantly different from a normal distribution. However, the real question is are your data sufficiently different from a normal distribution to make a parametric test result unreliable? There is no test for this, you have to explore your data and try different approaches.

### 4.3.6 Task 5 - Answering Question 1

**Did yields in this year deviate from the long-term average yields for the region?**

You should have answered above that in order to answer this question we require the "one-sample t-test". This test allows us to look at whether the mean of a sample is statistically significantly different from a pre-specified value. In this case, we are using the test to ask whether the mean yield from our sample of fields is different from the long-term average yield. The basis for the t-test is the existence of sampling error. The data we have are from a sample, not from the whole population. In this example we have 56 fields (the sample) from all the fields of sugar beet grown in Britain (the population). We want to use the sample to infer something about the population. We know (hope!) that the sample will be representative of the population, but we also know that our estimates of population parameters (mean, variance) will not be exactly the same as those of the population. The t-test is derived from statistical sampling

theory and allows us to specify, based on the sample size how likely it is (the statistical probability) that we have over- or under-estimated the population mean by a given amount.

- Given that the value of the t-distribution for a sample size of 28 (i.e. each half of the data) is 2.052, calculate the 95% confidence intervals (CIs) for the means of the two groups. You can use R to do this, or manipulate your earlier `summarise()` function to add the upper and lower CI it to your summary table.
- How many "degrees of freedom" are there?

The long-term average yield for the region is 10.7 tonnes.

- Do the 95% confidence intervals for the two groups include 10.7?

Now we are ready to try running a one sample t-test. Try running the following;

```
# One-sample t-test
one_sided_t <- t.test(sugar_beet_virus$yield, mu = 10.7)
# mu here is our test mean, this is the mean known yield of the population as a whole

# Printing the results
one_sided_t
```

- What does the one-sample t-test indicate about whether the mean yields of the pooled virus and non-virus affected fields differ from the long-term average?
- If you separate the yields into virus and non virus affected fields and rerun the one sided t-test on each, what the results tell you?

Note that the output in the one-sample t-test includes the 95% Confidence Interval of the Difference. This is calculated from the difference between the sample mean and the hypothesised mean, and so is different from the 95% CIs that you calculated previously.

### 4.3.7 Task 6 - Answering Question 2

**Did the presence of viruses affect the yield of sugar beet in this year?**

This time we need the two-sample t-test. The two sample t-test assesses whether the means of two samples are different from each other. It is therefore more commonly used than the one-sample test. The principle of the test is very similar to the one-sample test, but in the two-sample test we look at the standard error and confidence intervals for the difference between the means of the two samples. Specifically, we calculate a standard error for the difference between the two means, and then use this to generate a t-statistic. The assumptions of two-sample t-test are that the data for both samples are normally distributed and

that the variances of the two groups are similar. To test this second assumption we can use a test called Levene's test for Equality of Variances.

Try running the following;

```
# Using leveneTest()
leveneTest(yield ~ virus, sugar_beet_virus)
# Be careful with which variable you place on which side of the tilde (~). You want yo
```

The null hypothesis for Levene's test predicts that the variances of the two groups are equal.

- Is this hypothesis rejected?

If the Levene's test is not significant ($p>0.05$), you assume that variances are equal. If the Levene's test is significant ($p<0.05$), then you can not assume that variances are equal. In this second scenario we can still use two sample t-tests but we will need to include a correction to do that.

- Can we assume that variances are equal in this data set?

Try running the following;

```
# Two sample t test
t.test(yield ~ virus, data = sugar_beet_virus, alternative = "two.sided", var.equal = 
# Once again the tilde is used to denote dependent and independent variables
# We also specify which type of t-test we wish to perform and identify (based on our e
```

You should see a reported output from our two sample t-test and it should look something like this;

```
    Two Sample t-test

data:  yield by virus
t = 1.3987, df = 54, p-value = 0.1676
alternative hypothesis: true difference in means between group NV and group V is not e
95 percent confidence interval:
 -0.3030355  1.7016069
sample estimates:
mean in group NV  mean in group V
       10.053571         9.354286
```

The output that you are given has 3 major components;

1) The test statistic (t), degrees of freedom (df) and p-value.
2) The 95% confidence intervals based on the standard error of the difference (we will cover this more in the next lecture).
3) The means for the two groups.

- Is there evidence that the means of the two groups are different?
- Write a summary of this analysis, including the statistics you have calculated.

There was no significant difference between the yields of the two groups, despite the fact that the virus-infected fields had significantly lower yields than the long-term average. Thus, comparison of the viral fields' yields with the long-term average suggested that viral fields produced lower yields, but the difference between the viral and non-viral fields' yields in this one year was not statistically significant.

- Why might viral fields have significantly lower yields than the long-term average but viral and non-viral fields have similar yields in this year?
- How would you confirm these differences in the future?

## 4.4 Conclusion

We have begun to explore some inferential statistics, beginning with two variations of the t-test. Next week we will extend some of these ideas of analysing differences between groups to the analysis and design of experiments.

## 4.5 Before you leave!

Make sure you save your script and download it if you would like to keep a local copy.

Please log out of posit Cloud!

## 4.6 References

Wickham, H., Averick, M., Bryan, J., Chang, W., D'Agostino McGowan, L., François, R., Grolemund, G., et al., 2019. "Welcome to the tidyverse." Journal of Open Source Software 4 (43): 1686. https://doi.org/10.21105/joss.01686.

# Chapter 5

# Workshop 3 - Introduction to Design and Analysis of Experiments

## 5.1 Introduction

There are two simple requirements that should guide the design of experiments: randomisation and replication. Replication is necessary because biological systems are variable, and so we need multiple measurements to distinguish meaningful effects from random variations. As the number of replicates or measurements increases, the standard error of the mean of the data becomes smaller. The concept of randomisation is straightforward: if we set up an experiment to compare the behaviour of 2 species of lizard, for example, but place all our replicates for one species in one tank, and all our replicates for the other species in another tank, then we could not be certain whether the differences we observe are due to measurement error, or due to the differences between the tanks. Randomising an experiment can be done using tables of random numbers to allocate different treatments to your experimental units.

### Experimental Design

The layout in Figure 5.1 is an example of a randomised design: we have two treatments (A & B), with five replicates of each treatment. There are 10 plots in all, and the experiment was randomised. This type of arrangement is called a completely randomised design. There is a potential problem with the completely randomised design, however. This is because there is no restriction on the allocation of treatments to experimental units. All units from one treatment can, for example, end up on one side of the experimental area purely by chance.

In the example in Figure 5.1, for example, all the units on the left are treatment A. This would be bad if these units differed in levels of moisture, temperature, nutrients etc.

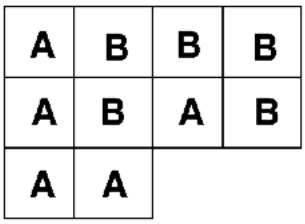


Figure 5.1: Completely randomised design

One of the easiest ways to control for this kind of problem is through a blocked design, as seen in Figure 5.2. This is achieved by first grouping the experimental units into blocks. Each block contains as many experimental units as there are treatments; the allocation of treatments to units is then randomised within each block. This design considerably reduces the possibility of producing an uneven arrangement of treatments. The blocked design is an improvement over the completely randomised design. It cannot, however, deal with trends across the experimental area.

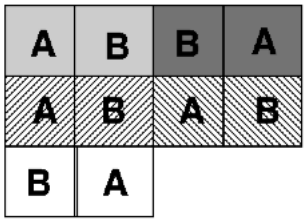


Figure 5.2: Blocked design

Trends across experimental areas can sometimes be controlled by a form of design known as a Latin Square, see Figure 5.3. The key to this design is that treatments are organised into rows and columns. In the example on the left, there are three treatments and each row and each column contains one of each treatment (a bit like sudoku). Blocked and Latin square designs may be mixed, but be warned, they can be difficult to analyse.

A multitude of different experimental designs exist: the examples above are the commonest forms of design, and illustrate how the experimenter can develop a design to cope with natural, uncontrollable variations in order to best study the biological factors of interest. Other designs include split plot designs that attempt to deal with practical restrictions. For example, if plot size were 3m

| | | |
|---|---|---|
| B | A | C |
| C | B | A |
| A | C | B |

Figure 5.3: Latin square design

x 3m but the equipment required to apply the treatment covers a minimum area of 12m x 3m, then we could not randomise the treatment. Instead we split the 12m x 12m plot into 4 plots. In Figure 5.4, the grey and white show one factor, which is applied in a split plot fashion; the other (1, 2, 3, 4) is completely randomised.

| | | | |
|---|---|---|---|
| 4 | 2 | 2 | 1 |
| 3 | 3 | 1 | 2 |
| 2 | 1 | 4 | 3 |
| 1 | 4 | 3 | 4 |

Figure 5.4: Split plots

When we take measurements from the same experimental unit (e.g. if we measure the effects of a treatment on the growth of an animal at several points through its life) we repeatedly take observations from the same unit. This type of design is called a repeated measures design. Analysis of this type of design accounts for both the treatment effect and the temporal sequence of measurements.

## Analysis of Variance

Last week we saw that the difference between two groups can be analysed using a t-test. The t-test can only be used to compare two groups. When our experiment contains 3 or more groups, we use a technique called ANalysis Of VAriance or ANOVA for short.

The null hypothesis in an ANOVA is that samples from different treatments are drawn from different populations with the same means. If our analysis indicates that the probability that this hypothesis is true is less than 1 in 20 ($p<0.05$),

then at least one of the treatments has a different mean from the others. ANOVA uses the data to calculate two estimates of the variance in the populations;

1) Within-treatment (error) variance is an estimate of the mean variance found within each treatment. It is calculated by summing the variances found within each of the samples (remember this is the 'Sum of Squares' - the sum of the squared deviations from the mean) and then averaging them (the Mean Squares - Sum of Squares/DF).
2) Between-treatment variance is an estimate of the variance between the sample means. It is calculated by summing the differences between each treatment mean and the overall mean of all of the treatments together, and then averaging them.

If the means of the treatments vary but the variance within the treatments is similar to the variance between treatments, then it will be unlikely that significant differences between the treatments can be identified. By contrast, if the within-treatment variance is small relative to the between-treatment variance, then it is more likely that statistically significant differences between the means will exist. We therefore need to be able to compare within- and between- treatment variances to determine whether the null hypothesis is likely to be true. This is done by calculating the ratio of these two variances, called the F-ratio. If the within and between treatment variance estimates are the same, then F = 1. If the between-treatment estimate of variance is greater than the within-treatment estimate of variance, then the value of F will be much bigger than 1. Clear differences between treatments therefore result in larger values of F. For a given value of F, and for a given number of treatments and replicates, we can then calculate the probability that the null hypothesis (no difference between treatment means) is true: again this critical level is 1 in 20, or $p<0.05$. There is a standard way of presenting the results of an ANOVA in a table, it looks something like this:

| Source | DF | Sum of Squares | Mean Square | F-Ratio | Probability |
|---|---|---|---|---|---|
| Between | 2 | 634.08 | 317.04 | 42.44 | 0.000 |
| Within | 21 | 156.88 | 7.47 | | |
| Total | 23 | 790.95 | | | |

There are four key things to look for;

1) **Mean Square (MS):** this is the average variance, and it is calculated **between** and **within** treatments, by dividing the respective Sums of Squares by DF.
2) **F-Ratio:** this is Between MS divided by Within MS. Remember large values (»1) imply differences.
3) **Degrees of Freedom:** if you have T treatments and r replicates (in the above table, T = 3 and r = 8), then Between-treatment DF is calculated as T-1. Within-treatment DF = T(r-1) and Total DF = Tr-1.

4) **Probability:** this is the probability of obtaining that value of F for those samples sizes.

With ANOVAs we are not limited to analysing the effects of just one factor: in an agricultural experiment we could, for example, vary the levels of two different nutrients. For example: Consider an experiment in which we varied Nitrogen (High and Low) and Phosphorus (High and Low). There are 4 Combinations of nutrient levels. This means that we are interested in three comparisons (see Figure 5.5);

1) HN vs. LN
2) HP vs. LP
3) HP and HN vs. LP and LN

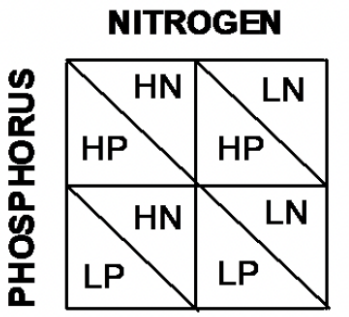


Figure 5.5: Blocked design

Out of these treatments, 1. and 2. are termed Main Effects: these are the basic factor levels that we have varied. 3. is termed an Interaction term.

We may find that increasing the level of either N or P on their own increases yields. However, these effects may not be independent. When combined, high levels of both N & P may increase yields by more than the sum of their individual effects (synergistic) or by less than the sum of their individual effects (antagonistic). Interaction effects are tested in exactly the same way as main effects, using an F-ratio. The ANOVA table is slightly modified:

| Source | DF | Sum of Squares | Mean Square | F-Ratio | Probability |
|---|---|---|---|---|---|
| Nitrogen | 1 | 376.04 | 376.04 | 108.74 | 0.000 |
| Phosphorus | 1 | 3.37 | 3.37 | 0.97 | 0.335 |
| Nitrogen * Phosphorus | 1 | 84.37 | 84.37 | 24.39 | 0.000 |
| Error | 20 | 69.19 | 3.45 | | |

In this example there is a significant effect of nitrogen on yields, but not phosphorus. In addition, the significant interaction term indicates that the response

of yields to nitrogen varies depending on the level of phosphorus. ANOVA can tell us when there are significant differences between the means of treatments, but it cannot identify which treatment differs from which other treatments. Post hoc tests are used to find where differences lie. Tukey's Honest Significant Difference is a widely used post-hoc test. Post-hoc tests are briefly explored in the practical. We will go on to look at how to test for relationships among different variables, as opposed to differences between them.

## 5.2 Practical 3 - Introduction to Analysis of Variance

### 5.2.1 Aims

The aim of this week's practical is to introduce the analysis of experimental data using one- and two- way analyses of variance (ANOVA). These analyses are used for testing for differences between treatments in fully randomised experiments. Note that other forms of experimental design, such as the randomised block and the Latin square require modified forms of ANOVA, although the basic ideas are the same.

### 5.2.2 The Data

We will be continuing with the rather agricultural theme for todays practical, the data are from one of the previous module leaders (Rob Freckleton) and are very appropriate for demonstrating key points around ANOVA. As an historical aside, this is also rather appropriate, as ANOVA and many other statistical analyses were designed by R. A. Fisher whilst working on long-term crop yield studies at Rothamsted.

Today we are going to look at data from two experiments:

- **Experiment 1.** Comparison of the yields of four different carrot varieties. In this experiment, six plots of each of the four varieties were grown, with the plots allocated to varieties in a completely random manner.
- **Experiment 2.** Effects of light and sex on feeding behaviour in chickens. This experiment aimed to determine the effect of light colour and sex on the rate at which chickens feed. Feeding rates were measured as the number of pecks made in a 15 minute period. There were 24 cages in total, half of which were illuminated with white light, and half with red light. Half of the 12 white-light cages were allocated a male chicken, and half a female chicken, and the same for the 12 red-light cages. The treatments were allocated to cages in a completely randomised manner.

### 5.2.3 Task 1 - Setting up your workspace

Log into posit Cloud and return to your instance in the class work space. Set up a new script in posit Cloud. Once again you will probably want to tidy your environment using the following (**make sure you have last weeks script saved in your `scripts` folder first**);

```
# Clean up your environment
rm(list = ls())
```

You will need to ensure that the packages `tidyverse` and `car` are installed and loaded to complete todays workshop, check Chapter 2.5.3 and Chapter 2.5.4 if you are unsure how to do this.

### 5.2.4 Task 2 - Checking the data

To start with we will just be focusing on the `carrot_yields.csv` data set, you can find this in the `data` folder of the workspace. Load it into your workspace, naming the object `carrot_yields` and perform your routine checks to make sure its all there and that R has correctly interpreted the variables. You can use some of the functions we explored in Chapter 3.2.3 if you need some help with this.

The data should consist on two columns, one column containing the dependent variable (carrot yields) and a second column containing the factor levels (varieties).

### 5.2.5 Task 3 - Making some basic plots

Now we can start to explore the data. Here we have four different varieties of carrot, we could use the `filter()` function to look at the distribution of each category seperatly but because we have four categoreies, this isnt terribly efficient. Instead the below code uses the `facet_wrap()` function, this splits your data plots according to a given variable (in this case Variety). Have a go at running the below code;

```
carrot_yields %>% # pipe your data set into ggplot
  ggplot(., aes(x = Yields)) + # Tell ggplot which variable is mapped onto the x axis
  geom_histogram(bins = 10) + # Plot a histogram with 10 bins
  facet_wrap(vars(Variety)) # Facet wrap the plot so that each variety is in its own panel
```

- Now try manipulating the code bin number and then compare your faceted histograms to a single histogram that includes all the data.

When you don't have many observations per category it can also be helpful to visualise the data a little differently. If we plot the data as point data for each

variety we can sometimes get a better idea of the spread of the data. Try using the following;

```
carrot_yields %>% # pipe your data set into ggplot
  ggplot(., aes(x = Variety, y = Yields)) + # Tell ggplot which variable is mapped ont
  geom_point() # Plot a scatter plot
```

- Visually inspect the data to assess their distributions, whether they are (approximately) normal and whether there appear to be differences between the means and variances of the varieties.

### 5.2.6 Task 4 - Produce some descriptive statistics

Try using the `group_by()` and `summarise()` functions to find the means and standard errors of the means for each of the yields of the four varieties.

- Do the treatment yields appear to differ and, if so, how do they differ?

### 5.2.7 Task 5 - Experiment 1 - One-way ANOVA

Now that we have explored the data, we can start to play with some ANOVAs. The null hypothesis for the one-way ANOVA is that all groups are drawn from populations with the same mean. Therefore, if the probability of the null hypothesis being true is small, then there is evidence that the samples are taken from populations with different means.

- Before running the ANOVA, write down what you understand by the terms: Sums of Squares, Mean square and F-ratio.

Remember that the analysis of variance (ANOVA) works by asking whether the variance between the experimental groups is as large, or larger, than the variance within the experimental groups. The F- ratio then compares these two quantities, by dividing the variance between groups by the variance within groups. The null hypothesis is that these are the same, i.e. F = 1. The assumptions of ANOVA are that the data for all treatments are normally distributed and that the variances of all treatments are equal.

First check whether the assumption of equal variances among treatments is met, by checking the output from a Levenes test. Run this test and check the result (look back at Chapter 4.3.7 if you need help doing this).

- The null hypothesis for this test is that the variances of all the groups are the same. Does the probability indicate that this null hypothesis is likely to be true?
- Are the assumptions of ANOVA met?

Now we can run our ANOVA

```
lsmodel01 <- aov(Yields ~ Variety, data = carrot_yields)
# Here we are creating an object to store our model in
# the aov() function will perform our ANOVA and we are identifying our predictor and response var
# As in earlier chapters your dependent (response) variable needs to be before the tilde and the
summary(lsmodel01)
# Summarises the results of your model stored in lsmodel01
```

The results should looks something like this;

```
> summary(lsmodel01)
            Df Sum Sq Mean Sq F value   Pr(>F)
Variety      3 1097.7   365.9    18.9 1.65e-05 ***
Residuals   16  309.7    19.4
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Can you see how this compares to the ANOVA tables shown above?
- Can you explain how the value of the F-ratio in this table is calculated?
- What would an F-ratio close to 1 indicate?
- What does the probability attached to the F-ratio indicate about the mean yields of the varieties?

The analysis indicates that there are some differences between the varieties of the groups but we do not know where these differences lie. To find this out, we are going to use a post hoc test. These operate like a series of pair-wise comparisons of the treatments, and will identify where significant differences exist between the means of the different varieties. To do this we will perform a Tuckey test, this is the most commonly reported post-hot test for ANOVA. Try running the following;

```
TukeyHSD(lsmodel01, conf.level=.95)
# Perform a d Tuckey test on our ANOVA, stored in lsmodel01 with confidence levels of 95%.
```

The output table indicates which pairs of treatment means are significantly different from one another. The `diff` collumn gives the difference in means between the two varieties, the `lwr` and `upr` columns give the upper and lower bounds for the 95% confidence interval for the difference and the `p adj` shows whether there is a significant difference between comparisons.

- Which varieties are significantly different from one another?
- Write a short summary of the results.
- How could you represent these results in a graph?

When we want to look at differences, a box plot or bar chart is often a good

option. We can use `ggplot()` to do this, try running the following;

```
ggplot(data = carrot_yields, aes(x = Variety, y = Yields)) +
  geom_boxplot()
```

- Does the plot you just produced match the conclusions you drew from performing an ANOVA and Tuckey test?

### 5.2.8 Task 6 - Experiment 2 - Two-way ANOVA

The analysis of Experiment 2 requires a two-way ANOVA since there are two factors in the experiment: light colour and sex. Note that you can have more than two factors in a multi-way ANOVA (although the more you have, the harder it can be to interpret the results).

Run through **Task 2** (5.2.4) and load the `pecking_rates.csv` file into your workspace, naming the object `chickens`. Run through some standard data checks, and explore the data using plots and descriptives statistics (as described in **Tasks 2-4**, 5.2.4, 5.2.5, 5.2.6).

Now try making a box plot to visualise these differences with the following code;

```
ggplot(data = chickens, aes(x = light, y = pecking_rates, fill = sex)) +
  geom_boxplot()
```

- What do you notice about the above code? How have we manipulated the `ggplot()` function to visualise two factors? Add some comments to your script to reflect the changes to the code.
- From this plot, do you think chicken peck rates vary between the sexes and light colours?

Hopefully you made some histograms when you were exploring the data, but if we are going to think about applying an ANOVA to these data we also need to check the other major ANOVA assumption.

Try running the following to check the assumption of equal variance;

```
leveneTest(pecking_rates ~ light * sex, data = chickens)
```

- What do you notice about the above code? Add comments to your script where appropriate.
- What does the output of the Homogeneity test tell you?

So lets try running a 2-way ANOVA, we can do this in a very similar way to before, but we are adding an additional factor plus an interaction `light:sex`. Try running the following;

```
lsmodel02 <- aov(formula = pecking_rates ~ light + sex + light:sex, data = chickens)
summary(lsmodel02)
```

The output looks more complex than the previous ANOVA output with a row for each factor plus a row for the interaction between factors.

- Do either or both of the main effects significantly influence the pecking rate?
- Is there evidence of a significant interaction between Colour and Sex?
- What does this interaction tell you about chicken pecking rates?
- What would a non-significant interaction indicate?
- Write down a short summary of the analysis.

## 5.3 Conclusion

This week we have looked at some different types of experimental design and explored how we can visualise differences in R and perform one and two-way ANOVAs.

## 5.4 Before you leave!

Make sure you save your script and download it if you would like to keep a local copy.

Please log out of posit Cloud!

## 5.5 References

Wickham, H., Averick, M., Bryan, J., Chang, W., D'Agostino McGowan, L., François, R., Grolemund, G., et al., 2019. "Welcome to the tidyverse." Journal of Open Source Software 4 (43): 1686. https://doi.org/10.21105/joss.01686.

# Chapter 6

# Workshop 4 - Testing for and Measuring Relationships

## 6.1 Introduction

In this week's practical we are going to test for relationships or associations between different variables. There are two basic methods for testing for associations:

1) Correlation analysis measures the strength of the relationship between two variables.
2) Regression analysis is used to estimate an equation that best describes the effect of one variable upon another.

In regression analysis, we are interested in the effect of an independent variable on a dependent variable. In scatter plots, the independent variable is always plotted on the horizontal x-axis, whilst the dependent variable is always plotted on the vertical y-axis.

### Direction of associations

Regressions or correlations can be positive (y-values increase as x- values increase), negative (y-values decrease as x-values increase) or there may be no relationship (no change in y-values as x-values increase). The commonest form of regression analysis is linear regression, which fits a straight line to the relationship. In linear regression analysis, there are three key steps:

1) Find the line that best describes the association between the two variables and assess whether the slope of this is significantly different from zero.

2) Determine the strength of the relationship (how well the data fit the line).
3) Determine the coefficients (slope and intercept) of the equation that describes the line.

The linear equation relates the dependent variable (y) to the independent variable (x), through two parameters:

- m is the slope: this is the rate of change in the y-variable as the x- variable increases. The larger the value of the slope, the faster y increases as x increases.
- c is the intercept: this is the value of y when x is equal to zero. The intercept therefore measures the position of the line on the y-axis.

The parameter m is of greatest biological interest to us in regression analysis, as it measures the extent of change in y that results from a unit change in x.

**The regression equation**

$$y = mx + c$$

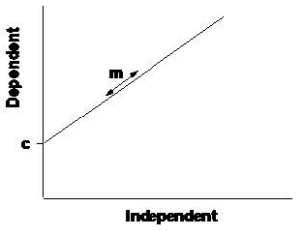


Figure 6.1: Linear regression

## Fitting the regression

Linear regression analysis basically works by finding the values of m and c that minimise the deviations of the observed data from the line (ie the distance of each point from the line). This is most commonly done using the least squares method, in which the squared deviations of the points from the line are minimised.

The deviations of points from the fitted line are termed residuals, and the best fit line is the one with the lowest residual variation. The residual variation is thus the amount of variation in the data that is not explained by the line. The fit of the line is measured by $r^2$, which is the proportion of the variance in y variable explained by x variable.

There are two ways to determine whether the relationship described by the fitted line is likely to have arisen by chance or not;

1) If there is no change in the y-values as x-values increase, the slope of the relationship is zero. We can therefore use a one-sample t-test to ask

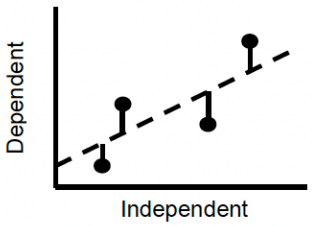


Figure 6.2: Fitting the regression

whether the slope of the relationship differs from a slope of zero. To do this, we need to compute a standard error for the slope, which is similar to the standard error of a mean. The SE of a slope measures our confidence in that slope, as it tells us where the true slope is likely to lie in relation to our estimate. Using n-2 DF we can calculate the 95% confidence intervals for the slope from;

$$m \pm t * se$$

If these CIs do not overlap zero, then there is a greater than 95% probability that the slope is different from zero, and we can then conclude that there is a significant relationship.

2) Alternatively, we can assess how different the slope is from zero by comparing the variance accounted for by the line with the total amount of variance in the y variable. If the line does not account for much of the variance in the data, then the overall variance will be much greater, and we cannot differentiate the slope from zero. This significance is assessed using an F-ratio in ANOVA.

In fact these two tests are equivalent! The value of F is always the square of the value of t that we can estimate from the SE of the slope.

## Assumptions

There are 3 main assumptions, which are **different** from the assumptions of the tests for differences;

1) The deviations of points from the line (residuals) are evenly distributed along the line.
2) The variance of these residuals is constant across the range of values of x (homoscedasticity).
3) The measurement error in the x-axis is zero (or at least negligible in comparison to the error in the y-axis).

### Correlation

In some circumstances we are not attempting to measure the effect of one variable on another, we are just interested in the strength of association between two variables x and y, without inferring anything about the nature of cause and effect. The association between two variables is measured using the correlation coefficient, which measures the degree to which the joint variation in the data (the covariation) differs from the total variation in the data. This coefficient, denoted as r, is therefore a ratio that varies between -1 and 1, and measures the strength of the association between x and y;

- r = 1 indicates that two variables are perfectly positively correlated.
- r = 0 indicates that two variables are entirely uncorrelated.
- r = -1 indicates that two variables are perfectly negatively correlated.

It is important to be aware that the existence of a high correlation between two variables does not imply a cause-effect relationship. Quoting correlation coefficients without an understanding of the underlying patterns of association between variables is therefore dangerous. The covariation between two variables may simply be due to the effects of a third, unmeasured variable. Regression analysis should, therefore, only be applied when the cause-effect mechanism that is assumed in the regression equation is valid.

## 6.2 Practical 4 - Regression & Correlation Introduction

This week we are going to look at statistical techniques involved in testing for associations between variables.

### 6.2.1 The Data

During the 1950s, radioactive waste leaked from a storage area into the Columbia River, Washington, USA. For nine counties downstream in Oregon, an index of exposure was calculated (based on distance from the leak site and average distance of the population from the river). The cancer mortality was also calculated for each county (deaths per 100,000 person-years, 1959-1964). Data were collected by Anderson and Sclove 1978

Because this data set is very small, we can enter this into R manually.

### 6.2.2 Task 1 - Setting up your workspace

Log into posit Cloud and return to your instance in the class work space. Set up a new script in posit Cloud. Once again you will probably want to tidy your environment using the following (**make sure you have last weeks script saved in your `scripts` folder first**);

```
# Clean up your environment
rm(list = ls())
```

You will need to make sure the packages `tidyverse` and `moderndive` are installed and loaded to complete todays workshop, check Chapter 2.5.3 and Chapter 2.5.4 if you are unsure how to do this.

As previously mentioned, the data set for this part of practical is relatively small, so we can enter it into R manually (this also gives you a chance to learn a little more code).

First of all we need to create a new object for each variable, and store our data accordingly. Try running the following;

```
exposure <- c(8.3, 6.4, 3.4, 3.8, 2.6, 11.6, 1.2, 2.5, 1.6)
cancer <- c(210, 180, 130, 170, 130, 210, 120, 150, 140)
```

Now we need to combine our individual variables into a single data frame, to do this we can use the function `tibble()`. Try running the following;

```
exposure_cancer <- tibble(exposure, cancer)
```

- Use some of the functions from Chapter 3.2.3 to exlore this data set.
- Are you happy you understand how we generated the new dataframe?

### 6.2.3 Task 2 - Making some basic plots

Lets start by taking a look at how your data are spread. We can do this by making a scatter plot. Last week we used `ggplot()` to make some boxplots and histograms, we can also use this function to make scatter plots. Try running the following;

```
# Create a scatter plot to look at the relationship between radiation
ggplot(data= exposure_cancer, aes(x = exposure, y = cancer)) +
  geom_point()
```

Hopefully R has produced a scatter plot in your plots panel exploring the relationship between radiation exposure and cancer mortality.

- Does there appear to be any relationship between the cancer rate and radiation exposure?
- What is the direction of the relationship?

### 6.2.4 Task 3 - Regression analysis

On the basis of medical evidence, we have strong reasons to suspect that elevated radiation can cause increased cancer mortality. As this hypothesis specifies a cause-effect relationship, we can use a regression analysis to assess the extent to which this radioactive leak can explain the observed cancer mortality rates.

We can modify the scatter plot you made in Task 2 to fit a regression line or line of best fit to your plot. Try adding the following line to your scatter plot code (dont forget to add a + to pipe to your new line of code);

```
 geom_smooth(method='lm')
# here we are instructing R to fit a line using the method `lm` which is an abbreviati
```

There are, of course, some assumptions about the data that must be checked before relying on the outputs from a regression analysis. The two key assumptions of linear regression are;

1) The deviations of points from the line (residuals) must be evenly distributed along the line.
2) The variance of these residuals must be constant across the range of values of x (homoscedasticity).

You can often assess these assumptions with visual inspection of the graphs, and it can also help to plot the residuals (difference between the actual value of y and the value predicted from the equation) against the predicted values (values of y predicted from the equation for each value of x). Positive residuals indicate actual values that are greater than predicted values (i.e. points that lie above the line), and negative residuals are less than the predicted values (i.e. points lying below the line). If the regression assumptions are met, then a plot of residuals against predicted values should give a random scatter of points with no systematic pattern or bias. A positive or negative trend in this plot would indicate heteroscedasticity, a wedge-shape would indicate a greater spread of data at one end of the line and a U-shape would indicate a non-linear relationship.

First of all, lets build our model, try running the following;

```
model01 <- lm(cancer ~ exposure, data = exposure_cancer)
# Here we are creating a new object called model01
# In this object we are placing a linear model as described by the lm() function
# We are then specifying that we want to analyse cancer (our response variable) as a f
summary(model01)
# Summary() will just print out a summary of our model for us to interpret.
```

Don't worry too much about interpreting the summary table yet, first lets check some of our assumptions. We can plot out our residuals against our predicted

values to check their distribution and homoscedasticity using the following piece of code, try running it now;

```
ggplot(model01, aes(x = .fitted, y = .resid)) +
  geom_point()
```

You can also check to see where these data points have come from with the following;

```
get_regression_points(model01)
# export regression points in a table of outcome/response variable, all explanatory/predictor va
```

By looking at these latest two outputs (your residuals vs predicted values plot and regression points table) try to answer the following;

- Do the residuals plot into a nice even band?
- Is there any sign of a wedge-shaped distribution with an increase in variance at one end?
- Is there any sign of a U-shape in the residual plot, indicating that non-linear regression might be needed?

It is important to check thoroughly the assumptions of regression, as datapoints exerting a disproportionate influence can have dramatic impacts on the results. If all assumptions are met, you can now explore the regression output, rerun the `summary(model01)` line. You should get an output that looks like this;

```
> summary(model01)

Call:
lm(formula = cancer ~ exposure, data = exposure_cancer)

Residuals:
    Min      1Q  Median      3Q     Max
-19.161 -11.934   3.741   8.969  17.226

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  118.449      8.365  14.161 2.08e-06 ***
exposure       9.033      1.480   6.103  0.00049 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.58 on 7 degrees of freedom
Multiple R-squared:  0.8418,    Adjusted R-squared:  0.8192
F-statistic: 37.24 on 1 and 7 DF,  p-value: 0.0004898
```

There is a lot of information here so we will break each section down.

First of all we have a reminder of the formula we gave to the `lm()` function;

```
Call:
lm(formula = cancer ~ exposure, data = exposure_cancer)
```

Then we have a summary of our residuals;

```
Residuals:
    Min      1Q  Median      3Q     Max
-19.161 -11.934   3.741   8.969  17.226
```

We have already explored our residuals in some depth so I wont go into a huge amount of detail around this summary, you can read around it if it interests you.

Then we have our coefficients summary;

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  118.449      8.365  14.161 2.08e-06 ***
exposure       9.033      1.480   6.103  0.00049 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

If we look at the `Estimate` column, the `(Intercept)` is the intercept in the traditional sense, this would be where the regression line meets the Y axis, when exposure is zero. So we can think of this as cancer mortality will be 118 deaths per 100, 000 when the radioactive exposure index is zero. Now if you look at the `exposure Estimate` this is 9.033, this represents the slope of the regression line. If we were to translate this into words we could think of this as for every unit increment in exposure, cancer mortality increases by c.9 deaths per 100, 000. We can test this theory mathematically.

Plug your intercept and slope values into the y = mx + c equation to estimate the mortality of a county with an exposure index of 6. You can do this by typing 9.003 * 6 + 118.449 into the console in R. Compare this output to the scatter plot for exposure and cancer mortality that we made earlier today.

Regarding the other values represented here, the `Std. Error` (Standard Error) measures the average amount that the coefficient estimates vary from the actual average value of our response variable. We can use these to calculate confidence intervals if required.

The `t values` represent our T statistic, this is simply the coefficient divided by the standard error. A large T statistic will have a small standard error proportionally to its coefficient, so the T statisitc is really saying that our coefficient is X standard errors away from zero and the larger the T statistic the more certain we can be that our coefficient is not zero. Finally we have our `Pr(>|t|)` which represents our p value. This is calculated using our T statistics and a T distribution (as seen in lectures). The significance codes denoted by `*` help you

interpret the size and significance threshold of the p value. These are essentially telling you how confident you can be that your coeffieient is not zero.

Next we have our Residual standard error and $R^2$ values;

```
Residual standard error: 14.58 on 7 degrees of freedom
Multiple R-squared:  0.8418,    Adjusted R-squared:  0.8192
```

The `Residual standard error` helps describe how well our data fit the model. If we think of our residuals as the distances our observed data points are away from the regression line then our residual standard error is the average amount that the observed data points differ from the predicted data points, shown by the regression line, in units of Y.

We then have two $R^2$ values. The `Multiple R-squared` value is mostly used if your model only has one predictor (as in this case). It tells us the percentage of variation within our responce (dependent) variable that is explained by our predictor (independent) variable. The `Adjusted R-squared` value is used when we have multiple predictor variables included in our model. It can be interpreted in the same was as the `Multiple R-squared` value but it shows the percentage of the variation in the response variable that is explained by all predictor variables. The differences in calculation are slightly nuanced, you dont need to worry about them here.

Finally we come to our F-statistic;

```
F-statistic: 37.24 on 1 and 7 DF,  p-value: 0.0004898
```

When we run a linear regression our null hypothesis is that there is no relationship between our two variables, you could also think of this as the null hypothesis stating that the coefficients for your variables are zero. So the alternative hypothesis would be that your coefficients for at least one of your variables is not zero. The `F-statistic` and associated `p-value` tests this hypothesis. Once again the larger your F statistic the more likely your result is to be significant and we can then use the p-value to measure the probability of seeing an F-statistic of that size if the null hypothesis were true.

Now we know how the summary output for our model is broken down we can start to interpret our results, from the `summary()` output for `model01` try to answer the following;

- How strongly is the cancer rate related to the radiation exposure? (Remember - the $R^2$ value tells you what proportion of the variation in y is explained by x, from 0 to 1).
- Is this relationship statistically significant (look at the p-value, which indicates the overall significance of the regression)?
- What is the slope of the regression line?
- How much does the cancer rate increase for an increase in exposure of 1 unit?

- For an increase in exposure of 20 units, how much would you expect the cancer rate to increase?
- What is the intercept of the slope?
- What does this intercept mean in real terms?
- What is the standard error of the slope and what does this tell you?
- Does the slope differ significantly from zero (look at the t-value of the slope, and the associated p value)? What does this tell you (bearing in mind that a slope of zero =no relationship).

We can also calculate confidence intervals for the slope and the intercept. For example, we can be 95% confident that the true values of our estimates lie somewhere between the upper and lower range of values given by the 95% CIs. The confidence intervals are calculated from:

**Slope:**

$$m \pm t * se$$

**Intercept:**

$$c \pm t * se$$

For 95% CIs, there is a 5% chance of the true value being outside these limits, thus there is a 2.5% chance it is less than the lower limit and a 2.5% chance it is greater than the upper limit. The appropriate t value for large degrees of freedom is 1.96, so if you want a crude estimate of the 95% CIs, t = 2, ie $\pm 2$ standard errors from the estimate.

- Calculate the 95% confidence intervals for the slope of the regression. What do these tell you? (Note: if you want to be precise, for 7 degrees of freedom, t = 2.365).
- What cancer rate would you expect to find in a hypothetical county with an exposure index of 5? You can estimate this from the graph and calculate it from the regression equation.

We often want to use fitted regressions to make predictions. These predictions can be made in two different ways;

1) **Predicted values:** predicting a mean value of y for a given value of x. If there were several counties with an exposure rate of 5, they would probably have slightly different cancer rates, due to natural variation. The cancer rate predicted from the regression equation is the mean rate for such counties. As this is a mean, it has a standard deviation, and the output also gives the 95% confidence interval (CI). We can be 95% sure that the mean lies between these upper and lower values.
2) **Prediction Intervals:** predicting a range (e.g. 95%) of values of y for a given value of x. We call this the 'prediction interval' (PI) to distinguish it from the 'confidence interval' (CI) for the mean. Individual values are scattered around the mean, thus the PI for an individual county is wider than the CI for the mean of many counties.

- What would you expect the cancer rate to be at an exposure index of 40?
- Is this a valid prediction?

### Task 4 - correlation analysis

In cases where there is no biological reason to expect a cause-effect relationship, we can still assess the degree of association between two variables, by measuring the extent to which they co-vary (i.e. as one variable changes, so does the other). The data listed below report the annual biomass of North Sea cod (in '000 tonnes), from 1980 to 2006.

In the `data` folder you should find a file called `cod_biomass.csv`. Load this file into your workspace using the `read_csv()` function. Use some of the commands from Chapter 3.2.3 to check the data and then use `ggplot()` to make a simple scatter plot. Try to answer the following questions;

- Does there appear to be a temporal trend in cod declines?
- In which direction is the trend?

Now we can calculate the Pearson (or product-moment) coefficient of correlation, using the command;

```
correlation01 <- cor.test(cod_biomass$year, cod_biomass$cod_biomass, method = "pearson")
# perform a correlation test on year and cod_biomass in the cod_biomass data set using Pearson's
correlation01
# print the information stored in correlation 01
```

- In the output the sample estimates is the correlation coefficient what does this value tell you?
- What is the statistical significance of this association?
- Could you have run a regression analysis on these data?

## 6.3 Conclusion

This week we have looked at how we can analyse and interpret relationships, either through regression analysis or through correlation analysis.

## 6.4 Before you leave!

Make sure you save your script and download it if you would like to keep a local copy.

Please log out of posit Cloud!

## 6.5 References

Wickham, H., Averick, M., Bryan, J., Chang, W., D'Agostino McGowan, L., François, R., Grolemund, G., et al., 2019. "Welcome to the tidyverse." Journal of Open Source Software 4 (43): 1686. https://doi.org/10.21105/joss.01686.

# Chapter 7

# Workshop 5 - Issues with Probability and Parametric Analysis

## 7.1 Introduction

Statistical analyses allow us to estimate the probability that a given effect will occur by chance alone. As a matter of convention, if the probability of a given effect occurring is less than 1 in 20 (i.e. $p<0.05$), then this effect is considered to be so unlikely to arise by chance that we can reject the null hypothesis of no difference or relationship.

There are two problems with using probability levels to determine whether an effect we see in our data is likely to be of significance;

1) Type I error occurs if we reject a null hypothesis when it is actually true (e.g. our t-test indicates a statistically significant difference, but in fact the populations do not differ).
2) Type II error occurs if we accept a null hypothesis when it is in fact false (e.g. our t-test does not indicate a statistically significant difference, when in fact the populations do differ).

Both of these problems arise because we cannot state for certain whether two populations differ or whether two variables are related: we can only use the information at hand to state the probability. For example, if we conduct 100 tests of the same null hypothesis we would expect 5 to be statistically significant, even if the null hypothesis were true. When conducting a large number of tests of the same null hypothesis, we could partially overcome this problem by lowering our critical p-value. Alternatively, a technique called a sequential

Bonferroni correction can be used to correct for a large number of tests of the same hypothesis. Often, however, the most sensible solution is to place more confidence in strongly (e.g. $p<0.001$) than weakly (e.g. $p<0.05$) significant results.

Ultimately, when we repeatedly test the same hypothesis, we cannot be completely certain that every effect we deem to be significant / non-significant is indeed so; we have to rely on an understanding of the system, and broad-scale patterns in our data, as much as rely on individual p values.

## Non-normal data & homogeneity of variance

The t and F distributions use information about the variance in estimates of population parameters, but they can only do this if a set of specific assumptions are met. All of the tests we have discussed so far rely on either data being normally distributed or residuals being evenly distributed. If this is not the case then all of the theory that predicts the likely behaviour of population parameters from our sampled estimates breaks down, and our tests are therefore invalid. This has been emphasised throughout, and we saw in an earlier practical how we could explore our data distribution using frequency histograms. Similarly, we saw that t-tests & ANOVAs assume that the variances of all our groups are the same, whilst the regression analysis assumed that the variance in our dependent variable did not change with the value of our independent variable. We saw earlier how Levene's test could be used to test this assumption of homogeneity of variance.

If we find that either the assumption of normality or of homogeneity of variance is violated, then often it is possible to overcome the problem through transforming our data. In Figure 7.1, the upper graph shows data that are extremely skewed; the lower graph shows that when these data are log- transformed, they then appear very normal.

Similarly, we can use a transformation of our data to meet the assumptions of regression analysis. In Figure 7.2, the upper scatter plot shows data that are inappropriate for linear regression, since the variance about the line increases with the value of the independent variable. The transformed data show a much more even distribution of residuals.

## Transformations

A number of useful transformations exist;

1) **Logarithmic:** remember that the logarithm of a number x is the number to which 10 (or e) has to be raised in order to give x: e.g. $\log10(100) = 2$ since, $100 = 10^2$; $\log10(1000) = 3$ since, $1000 = 10^3$, and so on.
2) **Logarithmic +1:** if we have some measurements of zero, or close to zero, then this may be a more effective transform that simply taking the logarithm. This is calculated as $\log(x+1)$.

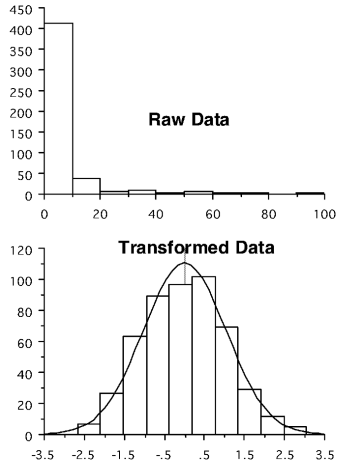


Figure 7.1: Impacts of log transformations - distribution

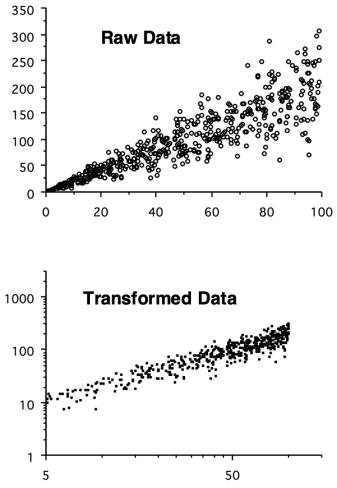


Figure 7.2: Impacts of log transformations - variance

3) **Square Root:** This operates in a similar fashion to the logarithmic transform and is especially used in regression analysis when the log-transform often goes a bit too far.
4) **Arcsine-square root:** This is used on proportional or percentage data, especially when the mean of the data is close to 0 or 1 (0 or 100%). This is calculated as: $\sin^{-1}(\sqrt{x})$. This helps to overcome the problem that such data are bounded, i.e. not continuous.

Transformation of data is a powerful and very important technique for overcoming the problem of non-normality and heterogeneity of variances. It is important to note, however that transforming data can affect our interpretation: e.g. if we compare the mean of untransformed data, and the back-transformed mean of our transformed data, then these will not be the same. This should be borne in mind in interpreting results.

## Non-parametric methods

In some cases it is not possible to transform data to a normal distribution. Data may also have been collected in ranks, or have measured an index of some variable we are interested in. To cope with these kinds of problems, a variety of statistical analyses have been developed that do not rely on assumptions about the distribution of data. They work somewhat differently, and in particular do not work by predicting the likely values of population parameters from statistics estimated from samples. In general, this is done in one of three ways;

1) **Median tests;** by analysing the median of the data, rather than the mean: you may remember that in the first practical we remarked that the median of our data is less sensitive to extreme values in our data and the skew of the distribution than the mean.
2) **Rank tests:** By ranking data, i.e. sorting them into order and exploring the position of each observation. For example, in tests of differences, one group may rank higher than another and, in tests of association, the ranks of observations will be similar on both axes if an association exists.
3) **Patterns of deviation;** Or we can look at patterns of deviation, for example to say whether a prediction tends to systematically over- or underpredict on the basis of positive and negative differences, or whether higher than average scores (positive deviations) or one variable tend to be associated with higher than average scores in another.

## Non-parametric tests for differences

The Mann-Whitney test is the non-parametric equivalent of the t- test. It is based on ranking data, and tests whether the location of one sample is different from that of another. This gives a statistic called U that, if greater than a critical value, indicates that there is a difference between the two groups.

The non-parametric equivalent to the one-way ANOVA (when we wish to test

for differences between several rather than just two groups) is called a Kruskal-Wallis test. This yields a statistic H that, if greater than a critical level, indicates the existence of differences between our groups.

The most common design of two-way ANOVA is randomised block design, and the non-parametric equivalent for this is called Friedman's method for randomised blocks. For a regular two-way ANOVA, the Schreirer-Ray-Hare extension to the Kruskal-Wallis test may be employed (see Sokal & Rohlf 1995; pp446-447).

### Non-parametric tests for association

The most commonly employed non-parametric tests for association are Spearman's and Kendall's correlation coefficients, based again on ranking data. These both yield coefficients that are comparable to the correlation coefficient, and if greater in magnitude than a critical level indicate the existence of a significant correlation.

For regression analysis, a technique called Kendall's robust line-fit method, details of which may also be found in Sokal & Rohlf (1995).

### Pitfalls of non-parametric tests

Most of the tests mentioned above rely on ranking methods - they therefore have to be able to deal with the existence of ties in the data, i.e. when two measurements have the same ranks. Often, when using these tests, you will see two outputs, one normal, and the other 'corrected for ties'.

As you can probably guess from the names of these non- parametric tests, many mathematicians have sought fame and fortune through designing a 'distribution-free analysis'. There are, therefore, many tests that do more or less the same thing, and which give basically the same results. Those mentioned above are generally the most commonly used. There are also methods for constructing models in which you can define the structure of the data (called generalised linear models), which we will explore in the next module.

## 7.2 Practical 5 - Transformations & Non-parametric Tests - An Introduction

The techniques we have so far used to test for differences and associations are all parametric tests, which estimate population parameters from samples in order to test hypotheses. As we have seen, however, these tests make some assumptions about data that, if violated, would lead to results being invalid. In the previous classes we saw how we could check the assumptions of our analyses in order to determine whether the results will be valid. In this practical we are going to introduce some techniques that can be used if we believe that the

assumptions of our parametric analysis have been violated. In the first section we will show how it is possible to transform data in order to overcome problems with the shape or variance of our data. In the second section we will introduce two simple non-parametric analyses, for use when our data either cannot be normalised, or when we have data (e.g. counts, scores, ranks) that would be inappropriate for analysis with parametric techniques.

### 7.2.1 Task 1 - Setting up your workspace

Log into posit Cloud and return to your instance in the class work space. Set up a new script in posit Cloud. Once again you will probably want to tidy your environment using the following (**make sure you have last weeks script saved in your `scripts` folder first**);

```
# Clean up your environment
rm(list = ls())
```

You will need to make sure that the package `tidyverse` is installed and loaded to complete todays workshop, check Chapter 2.5.3 and Chapter 2.5.4 if you are unsure how to do this.

The data for the first part of todays practical are the weights (g) of individual plants originating from seed taken from two sites. Each was grown singly in pots in a greenhouse, with the arrangement of pots being completely random.

The data set for this part of practical is relatively small, so as in the previous chapter, we can enter it into R manually.

First of all create two objects called `site1` and `site2` and fill them with the following data points (check Chapter 6.2.2);

```
Site 1; 0.142, 0.084, 0.029, 0.175, 0.047, 0.012, 0.042, 0.200, 0.973, 0.158
Site 2; 1.304, 1.647, 0.269, 0.209, 0.131, 0.818, 4.571, 9.377, 0.204, 0.169
```

Once you have your `site1` and `site2` objects you can then use the `tibble()` function to create a data frame called `seeds`.

Use `view()` to inspect your new data frame.

- What do you notice about the layout of these data?
- Are these data in a long or a wide format?
- What kind of format do we need them to be in?

Hopefully you identified that the data in `seeds` are in a wide format and we need them in a long format. Tidyverse has a lot of very helpful functions for wrangling data into the format that we need it to be in. Here, we can use the `pivot_longer()` function to manipulate our data. Try running the following, taking time to read each line of code and understand what each section does;

```
seeds2 <- seeds %>%
  pivot_longer(cols = site1:site2,
               names_to = "site",
               values_to = "weight")
# Here we are piping out initial seeds data set into the pivot_longer() function
# We are then telling the function which columns we with to be included within the seeds data fra
# The names_to argument is allowing us to name our categorical variable
# The values_to argument is allowing us to name our numeric variable
# We are then saving the output from this function to the object seeds2
```

Now use the `view()` function on your new `seeds2` data frame, can you identify what the `pivot_longer()` function has done?

### 7.2.2 Task 2 - Explore the data

Spend a little time exploring the data. Try using the `group_by()` and `summarise()` functions to find the means, standard deviation and the standard error of the plant weights for each site (see Chapter 3.3.1 if you need help with this). Build some histograms to explore data distribution and maybe consider building a boxplot to explore differences between sites (see Chapter 3.2.4 if you need help with this).

- Does there appear to be any difference between the two sites?
- Does the distribution of data appear to be normal? (remember we have only a small number of observations so we would hardly expect a perfect bell-shaped pattern).

Now try comparing the two sites with a levenes and a two sample t-test (check Chapter 4.3.7 if you're not sure how to do this).

- Does this test indicate that the two groups differ?
- What does the Levene's test tell you about the variances of the two groups?
- What are the assumptions of a t-test, and do you think that these data conform to them?

### 7.2.3 Task 3 - Data transformations

Basically there are two problems with these data;

1) The data (particularly those from site 2) are highly skewed
2) The variances for the two groups are very different, which could mean that a parametric test may fail to detect a difference, if it exists (although if the difference between the two groups is very large, a parametric test may detect the difference despite the assumptions not being met).

We can attempt to meet these assumptions by transforming the data. Since the data are spread over several orders of magnitude (e.g. in site 1 we have values ranging from 0.012 to 0.142: the former measurement is ten times the size of the latter), a logarithmic transformation is likely to be appropriate.

Thankfully this is a very easy addition we can make in our `seeds2` dataframe. Try running the following piece of code;

```
seeds2 <- seeds2 %>%
  mutate(log10 = log10(seeds2$weight))
```

Use the `view()` command to take a look at your `seeds2` dataframe.

- The log10 expression means to take the logarithm to the base 10. With this in mind, do you understand what the mutate() function has done?
- Add some appropriate comments to your script to describe the above code.

Repeat the data explorations that you carried our in Chapter 7.2.2 on the new log10 column.

- Is there still a difference between the two sites? (Remember that the true population mean for each group will lie within approximatly 2 standard errors either side of the mean)
- How does the distribution of data compare with what we saw before?

Carry our a levenes test and two sample t-test on the two log transformed groups.

- Does this test indicate that the two groups differ?
- What does the Levene's test tell you this time about the variances of the two groups?
- How do these results compare with the analysis of the untransformed data?

The transformation of the data has basically achieved what we wanted: the variances of the two groups are now not statistically different, and the transformed data are considerably less skewed than the raw data. A t-test comparison of these data is more likely to yield robust results.

### 7.2.4 Task 4 - Non-parametric analysis of the differences

One of the most striking features of the original data is that most of the larger values are for the data for site 2, whilst most of the smallest values are for site 1. This is very clear if we sort the data into order, and give each pot a rank:

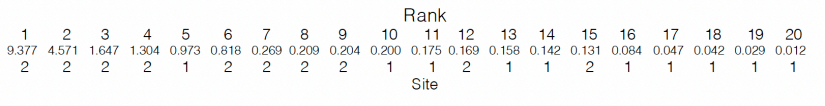


Figure 7.3: Rank Orders

Work out the mean ranks of the pots taken from the two sites (you can either do this manually or see if you can write a piece of R code to do this).

- Do these mean ranks tell you whether one site tends to produce bigger plants than the other?

In the top 10 positions, only 2 pots are from site 1, whilst in the last 10 positions, only 2 pots are from site 2. Clearly, irrespective of the spread or mean of these data, the plants from site 2 tend to be bigger than those from site 1. This is the basis for the first non-parametric test that we are going to look at, the Mann-Whitney test. This test simply compares the ranks of the data from our two groups, and determines whether one of the groups tends to occupy the top or bottom positions more frequently than we would expect by chance.

Copy over the following piece of code and run it from your script;

```
# Mann Whitney U / Wilcoxan
wilcox.test(weight ~ site, data=seeds2)
```

Although this is called a Wilcoxon test, if you run the test on two independent samples (as we have here), R runs a Mann-Whitney U test (confusing I know). Here we have performed a Mann-Whitney U test. The W value is our test statistic and p-value is fairly self explanatory. The key statistic to look at is W (if you are familiar with MannWhitney U tests, this is equivelent to the traditional U value): this measures the difference between the ranks for the two groups; a large value indicates a large difference in ranks.

- The null hypothesis is that there is no difference between the groups: does the probability value indicate that this is likely to be the case?

The non-parametric test works very differently to the t-test - in particular it ignores any information on the mean and variance in the data. Note that strictly speaking, it tells us whether the location of the data from two groups is different, not whether the means are different.

### 7.2.5 Task 5 - Non-parametric test for associations

Finally we are going to look at a non-parametric test that enables us to tell whether there is a significant association between two variables.

The data are taken from a study on aphids: `mother` records the total length of 15 aphid stem mothers, and `offspring` records the mean thorax length of their (numerous) parthenogenetic offspring, taken from visual estimates under a microscope.

```
mother; 8.70, 8.50, 9.40, 10.00, 6.30, 7.80, 11.90, 6.50, 6.60, 10.60, 10.20, 7.20, 8.6
offspring; 5.95, 5.65, 6.00, 5.70, 4.70, 5.53, 6.40, 4.18, 6.15, 5.93, 5.70, 5.68, 6.13
```

Use the above data to create a new data frame called aphids, with two 2 columns titled; mother and offspring. Create a scatter plot to see whether there appears to be any association between these two variables.

- What would be the problem with using parametric correlation or regression analyses to look at this association?

Basically there are two problems here;

1) There is error in both axes: the lengths of the mothers are estimated rather than accurately measured and the offspring lengths are averages.
2) The pattern of association is not linear: the significance test for a parametric correlation coefficient requires a linear association.

There are two (very similar) non-parametric measures of correlation. These again rely on ranks: i.e. they test whether high-ranking mothers produce high ranking offspring.

First of all try running the following to have R assign a rank to each of the variables;

```
aphids <- aphids %>%
  mutate(rank_m = rank(aphids$mother)) %>%
  mutate(rank_o = rank(aphids$offspring))
```

- Do you understand what each line of code is doing here? Try to add comments to it in your script.

If there is a correlation then we would expect the biggest mother to have the biggest offspring, and the smallest mother to have the smallest offspring. Create a scatter plot from your ranked variables.

- Does it appear that the rank of the offspring follows that of the mother?

We can see if there is a correlation using the Spearmans Rank correlation coefficient. Try running the following;

```
cor.test(aphids$mother, aphids$offspring, method = "spearman")
```

You will get the following warning;

```
Warning message:
In cor.test.default(aphids$mother, aphids$offspring, method = "spearman") :
  Cannot compute exact p-value with ties
```

This is because some of the ranks are tied in the offspring variable. We can edit our code so that we can still get an estimate of the correlation coefficient by editing our command to read as follows;

```
cor.test(aphids$mother, aphids$offspring, method = "spearman", exact = FALSE)
```

The coefficients that this analysis generates are structured in the same way as the normal (Pearson's) correlation coefficient: a magnitude of 1 indicates perfect positive correspondence; a value of 0 indicates no association.

- What does the coefficient tell you?
- Is it statistically significant?

## 7.3 Conclusion

Today we have explored how to deal with data that breaks the assumptions made by parametric tests. We have seen what the impact of log transformations and ranking data has on its structure and have played with a few non-parametric tests to test for differences and associations.

## 7.4 Before you leave!

Make sure you save your script and download it if you would like to keep a local copy.

Please log out of posit Cloud!

## 7.5 References

Wickham, H., Averick, M., Bryan, J., Chang, W., D'Agostino McGowan, L., François, R., Grolemund, G., et al., 2019. "Welcome to the tidyverse." Journal of Open Source Software 4 (43): 1686. https://doi.org/10.21105/joss.01686.

# Chapter 8

# Plot Beautification!!!

So far, in this workbook, we have used a number of different plots to explore our data. They have served their job well and given you a good overview of what the data look like. Which is absolutely fine, while you are tinkering and exploring with your data sets. However, that wont fly when it comes to sharing your analysis outputs; be that in a summative coursework piece, a presentation (verbal or poster) or a professional publication. Thankfully, `ggplot()` has some fantastic features for making plots look absolutely stunning, and we can quickly make a draft plot look absolutely fabulous very quickly.

This is a crash course in plot beautification using `ggplot()`. It is by no means an exhaustive list of all the aesthetic parameters you can edit, but it will give you a good grounding.

## 8.1 Deconstructing ggplot

Hopefully you have been saving the code from former workshops in a sensible place in our posit Cloud classroom. Back in Chapter 5.2.8 we made a boxplot comparing the pecking rates between male and female chickens. Initially, we will work with this same plot.

Log into posit Cloud and join the class workspace. Clean-up your enironment with the following command;

```
# Clean up your environment
rm(list = ls())
```

Now load your script from Practical 3, Chapter 5, install and load the `tidyverse` package and create a `chickens` object containing the `pecking_rates.csv`.

To get an idea of how ggplot works we are going to go back over some code you have already run. Try running the following, what happens?

```
# Making a box plot and saving it in an object
# Call the ggplot function and direct it to your data, define your x axis and y axis a
pecking_rates_box <- ggplot(data = chickens,aesdata = chickens, aes(x = light, y = pecl
print(pecking_rates_box) # Print the object your plot is stored in to view it
```

So you have given ggplot your data set and specified the axes. But you haven't told it how you would like the data to be visualised. So you should see something like this...

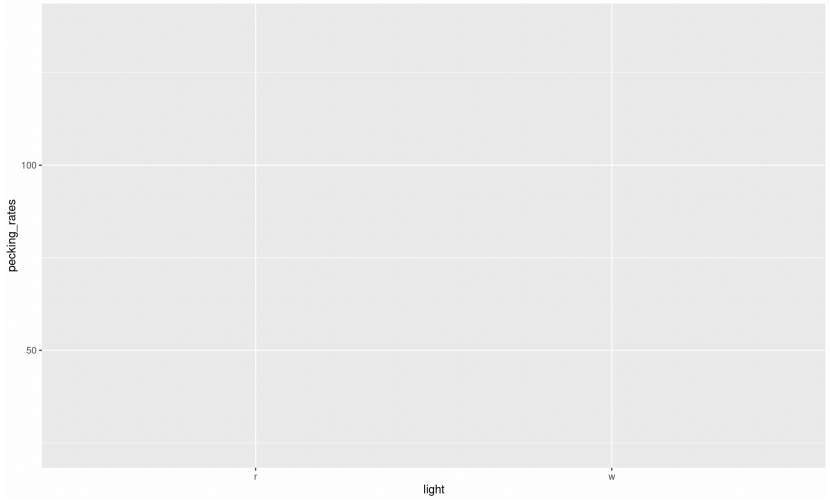


Figure 8.1: The most basic output from ggplot

Now modify the script so that it looks like this...

```
# Making a box plot and saving it in an object
# Call the ggplot function and direct it to your data, define your x axis and y axis a
pecking_rates_box <- ggplot(data = chickens,aesdata = chickens, aes(x = light, y = pecl
  geom_boxplot() # Tell ggplot that you want it to build a box plot
print(pecking_rates_box) # Print the object your plot is stored in to view it
```

Note the `+`, this is essentially another way of piping information from one function into the next. You could also have added `fill = sex` to an `aes()` function within the `geom_boxplot()` function.

Your figure should look like this...

Have a look at this figure, is there anything you would change, to make it more

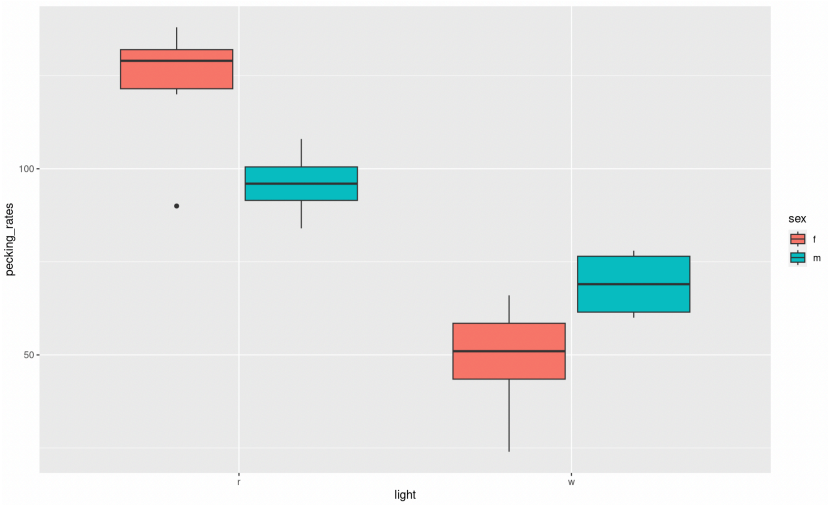


Figure 8.2: A basic ggplot box plot output

visually appealing or clear?

Here are some things I would change:

- Labels
- Colour scheme
- Spacing
- Theme

We will look into how to edit these elements now.

## 8.2 Labels

Clear accurate labeling is essential in the sciences, be it when your in the lab labeling up your samples or creating data visualisations.

In our current plot both our x and y axis could do with some relabeling, just because we dont use capitals when coding doesn't mean we don't follow the normal rules of English grammar when presenting data. Try adjusting your code so that it looks like the following;

```
# Making a box plot and saving it in an object
# Call the ggplot function and direct it to your data, define your x axis and y axis and colour y
pecking_rates_box <- ggplot(data = chickens,aesdata = chickens, aes(x = light, y = pecking_rates,
  geom_boxplot() + # Tell ggplot that you want it to build a box plot
  labs(x = "Light Colour", y = "Number of Pecks (per 15 minutes) \n") # Adjust your x and y axis
print(pecking_rates_box) # Print the object your plot is stored in to view it
```

Note the `\n` on your y axis label. This simply adds a new line to the label and spaces it nicely away from the y axis.

This looks a bit better but our key is still not very well labeled. Adjust your `labs` function so that it reads `labs(x = "Light Colour", y = "Number of Pecks (per 15 minutes) \n", fill = "Sex")`. This will change the label for your key.

Our labels still aren't quite right, I don't like the abbreviations on the x axis and our key could still use some work as there are no units given for the treatment temperatures. Try these changes;

```
# Making a box plot and saving it in an object
# Call the ggplot function and direct it to your data, define your x axis and y axis a
pecking_rates_box <- ggplot(data = chickens,aesdata = chickens, aes(x = light, y = pecl
  geom_boxplot() + # Tell ggplot that you want it to build a box plot
  labs(x = "Light Colour", y = "Number of Pecks (per 15 minutes) \n", fill = "Sex") +
  scale_x_discrete(labels = c("Red","White")) + # Rename the categories on the x axis
  scale_fill_manual(labels = c("Female", "Male")) # Rename your key labels
print(pecking_rates_box) # Print the object your plot is stored in to view it
```

Note that the `scale x discrete` and `scale_fill_manual` functions simply rename things in the same order as you present them, make sure you label your categories accurately or this can make a big mess later on.

If you try to remake your plot at this stage you will get the following error:

```
Error in `palette()`:
! Insufficient values in manual scale. 2 needed but only 0 provided.
Run `rlang::last_trace()` to see where the error occurred.
```

This is because `scale_fill_manual` is also expecting some instructions on how to colour your boxplot. You will need to complete Chapter 8.3 before you can successfully remake your boxplot.

## 8.3 Switching up colours

The colours of your boxes are the default colours given by `ggplot()`. We can modify these to make our plot look a bit more pleasing. Try editing the `scale_fill_manual` function to this `scale_fill_manual(labels = c("Female", "Male"), values = c("cornflowerblue", "coral"))`. Run it and see what happens to your plot.

What do you think of your new plot now? Do you think the changes to labels and colours are an improvement?

R has lots of colours, take a look at the linked reference lists and have a play with changing up some of your colours.

Colour can be a really useful tool to employ when making your plots visually appealing, however make sure you are mindful that some colour pallets can be difficult for some people to interpret. There are however, some really good tips out there for making sure your figures are accessible to everyone.

## 8.4 Spacing

Lets have a look at spacing. This is a box plot so we the main spacing option you are likely to want to play with is the width of your boxes. Here we can simply add an argument to the `geom_boxplot()` function. Edit this function so that it reads `geom_boxplot(width = 0.9)`. The width argument can be anything between 0.00 or 1.00. It changes the width of the boxes and this changes the spacing between them as well. Have a play with some values and see what happens.

We will come back to spacing when we look at our scatter plots later.

## 8.5 Themes

Another aspect of aesthetics that we can look at is themes. We have a reasonably attractive graph now, but its still got a grey background and the grid lines are unnecessary. To remove the grey background and implement the classic black on white aesthetic we can simply add a function that defines a pre-made theme. Adjust your script so that it includes the function `theme_bw()`, I suggest you add this as a new line, don't forget to pipe between your functions with a `+`. Run this chunk and print your new plot. How is that looking now?

Two things still jump out at me when looking at this plot. The grid lines are completely unnecessary and detract from the overall aesthetic and the text sizes could be larger. To make these edits we can simply add additional instructions to adjust the theme further. So although most of the work has been done by applying a the `theme_bw` we still need make some adjustments.

Add the following function to your growing ggplot chunk (don't forget to pipe `+` between functions);

```
theme(panel.border = element_rect(color="black"), # Specifies that the plot boarder is coloured b
        panel.grid.minor = element_blank(), # Removes minor grid lines
        panel.grid.major = element_blank()) # Removes major grid lines
```

Now we just need to adjust the text size. We can do this within the `theme` function as well adjust your theme so that it reads like this;

```
theme(panel.border = element_rect(color="black"), # Specifies that the plot boarder is coloured b
        panel.grid.minor = element_blank(), # Removes minor grid lines
```

```
        panel.grid.major = element_blank(), # Removes major grid lines
        axis.text = element_text(size = 15), # Changes the size of text on both axis
        axis.title = element_text(size = 20), # Changes size of your axis labels
        legend.text = element_text(size = 15), # Changes the size of text within your
        legend.title = element_text(size = 20)) # Changes the size of the legend title
```

Have a play with the different text sizes until you think they are optimal. You may need to press the zoom button in the viewer panel to get a clearer idea of the scaling.

Hopefully you now have a nice clean, clear and aesthetically pleasing plot and have some awareness of the commands and functions used to make it.

## 8.6 Beautifying scatter plots

In Chapter 6.2.3 and 6.2.4 we made a very simple scatter plot for radiation exposure against cancer mortality (deaths per 100,000 person-years, 1959-1964) across nine counties in the US. Reopen your script from this workshop, recreate the accompanying data frame and rebuilt your `geom_point()` plot.

Use you previous chunks of code and knowledge of R to edit the labels and themes of this plot. Print it again and take a look.

Now there are some additional elements in this plot that could be adjusted. These include;

- Point shape - The shape of the individual points on the plot, as with colours there are lots of options numbered 0 - 25, descriptions of each one are listed here
- Point size - The size of each point
- Point colour - The colour of your points
- Line type (solid, dashed, etc) - Again thre are several options here, the notation and descriptions of which can be found here
- Line colour
- Line size - the weight of the line
- Line fill - Note that here this refers to the colour of the shaded area marking the standard error

So lets play with some of these. Try adding and adjusting the following arguments within `geom_point()`

- `shape = 1` - have a look at other point shapes you could use and play with this setting
- `size = 2` - again try playing with some point sizes
- `colour = "blue"` - or any other colour you fancy trying.

Once you are happy with your points we can take a look at your regression line. Try adding and adjusting the following arguments within `geom_smooth()`

- `colour = "cornflowerblue"`
- `fill = "lightblue"`
- `size = 1`
- `linetype = "dashed"`

As before please do experiment and play with the settings described by each of these arguments.

## 8.7 Explorting your plot to pdf

Once you are happy with your new and beautiful plots make sure you save it to to the figures folder using a function called `ggsave()`. This will allow you to export your plots as `.pdf` files which you can then download and put in your reports/publications.

You will need to have your plots saved as an object (using the `<-` syntax) before this will work. You can call these objects whatever makes most sense to you. Try running the code below for each of your beautified plots;

```
# Saving outputs
ggsave("figures/pecking_rates_boxplot.pdf", # Give R a path to save to and a file name
       plot = pecking_rates_box,# Tell R what to save - in this case your object
       width = 15, # Set .pdf width
       height = 10, # Set .pdf height
       units = "cm", # Specify units for .pdf width and height
       device = "pdf") # Tell R what file type to create, in this case a pdf
# Note, use trial and error to select a good width and height for your figure
```

Note that if you remove the `device =` line `ggsave` will automatically create a `.png` file. You can also use `width = , height = , units = "cm"` arguments to specify the size of your figure in cm. You will need to adjust these by trial and error until the proportional sizing of your plot to axis and legend labels is pleasing. This chunk will save your plot to the `figures` folder. From here you can download your figures by checking the box besides the file, clicking `More` and then `Export...` and `Download`.

## 8.8 Before you leave!

Make sure you save your script and download it if you would like to keep a local copy.

Please log out of posit Cloud!

## 8.9 References

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." Journal of Open Source Software 4 (43): 1686. https://doi.org/10.21105/joss.01686. Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2021. Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics. https://CRAN.R-project.org/package=ggplot2.

# Chapter 9

# BIO-7026A: Univariate Statistics Summative Assignment

Report Length: 2000 words (maximum)

## 9.1 The exercise

There are **two** separate parts to this assignment:

- **Part 1:** for each of the five questions below, describe (a) the test you selected (b) why that test is appropriate and (c) the procedures you followed to conduct and interpret each test.
- **Part 2:** present the results of your analyses in the style of the results section of a scientific paper.

## 9.2 The Problem

The kooki bird is a very rare (fictional) species, which now exists only on a few islands. The birds are largely frugivorous, and natural fruit yield on the islands can vary with weather conditions. Since 1980, a captive breeding programme has been in place and chick productivity has been measured for both captive birds and birds breeding in the wild. Weather data have also been measured at a local meteorological station. The investigators are interested in assessing how maximum daytime temperatures during September and October and rainfall during January and February might affect wild chick productivity. The length of the breeding season was also recorded as the (standardised) number of days during which the temperature reached 15°C. The data table below gives

the mean chick productivity for wild and captive birds, and the weather and breeding season length in each year.

In addition to this, on the largest island, a long-term experiment was set up to examine the effects of different levels of supplemental feeding with bananas and mangos. There are two levels of banana and mango supplementation (none vs the standard supplement of 100 g bird$^{-1}$). The experiment is a fully factorial design, with each of the four treatment combinations occurring in each year, and the arrangement of the experimental feeders on the island is fully randomized. As the birds are highly sedentary, they only experience the food supplementation at the feeder nearest them. The data table below gives the mean chick productivity for the pairs experiencing each treatment within each year.

Using whatever statistical analyses you feel appropriate, answer the following questions;

1) How does chick productivity of wild birds compare to that of the captive birds?
2) Is there any evidence for changes in productivity over the study period among the wild birds?

3) Is there evidence of the weather variables influencing chick productivity among the wild birds?
4) Does chick productivity of wild birds vary with the length of the breeding season?
5) What are the effects of the food supplementation treatments on kooki bird productivity?

**Hints & Tips**

- The data presented below will, in places, need reformatting before you will be able to conduct your analysis.
- For good examples of how results of analyses should be presented and described, consult Results sections of publications in a scientific journal, such as Journal of Animal Ecology or Journal of Applied Ecology. Part 2 of the assignment should be presented in this style.
- Don't be tempted to use multivariate techniques to analyse the effects of weather on productivity – use only the analytical approaches covered in this module.

Put your responses to **Part 1** and **Part 2** in a word document. You will be able to submit your work through the BIO-7026A page on Blackboard.

## 9.3 The Data

| Wild birds [1] | Captive-bred birds [1] | BO MO [2] | B1 MO [2] | B0 M1 [2] | B1 M1 [2] | Jan Rain (cm) [3] | Feb Rain (cm) [3] | Sept Temp (°C) [4] | Oct Temp (°C) [4] | Breeding Period (Standardised) [5] |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.54 | 0.04 | 0.54 | 0.73 | 0.56 | 0.73 | 2.30 | 2.30 | 15.05 | 18 | -0.52 |
| 0.55 | 0.02 | 0.53 | 0.64 | 0.54 | 0.68 | 2.40 | 2.10 | 16.1 | 17.55 | 0.13 |
| 0.58 | 0.04 | 0.61 | 0.71 | 0.61 | 0.72 | 1.90 | 1.10 | 17.4 | 17.5 | 0.39 |
| 0.50 | 0.02 | 0.49 | 0.64 | 0.51 | 0.71 | 2.20 | 3.50 | 16.55 | 18.35 | -0.02 |
| 0.49 | 0.02 | 0.45 | 0.52 | 0.48 | 0.60 | 2.20 | 1.90 | 13.05 | 17.5 | -0.17 |
| 0.41 | 0.06 | 0.32 | 0.34 | 0.33 | 0.38 | 1.50 | 1.40 | 13.1 | 14.85 | -1.72 |
| 0.32 | 0.03 | 0.45 | 0.46 | 0.48 | 0.63 | 1.10 | 1.70 | 14.65 | 17.4 | -0.37 |
| 0.33 | 0.50 | 0.44 | 0.52 | 0.51 | 0.57 | 1.10 | 0.80 | 16.85 | 18.05 | 0.08 |
| 0.54 | 0.33 | 0.54 | 0.62 | 0.57 | 0.63 | 1.50 | 0.80 | 16.45 | 17 | 0.29 |
| 0.35 | 0.62 | 0.36 | 0.46 | 0.40 | 0.52 | 1.10 | 2.50 | 15.65 | 18.15 | 0.42 |
| 0.26 | 0.10 | 0.29 | 0.35 | 0.33 | 0.42 | 0.60 | 0.40 | 14.45 | 18.1 | -1.57 |
| 0.38 | 0.01 | 0.35 | 0.38 | 0.45 | 0.46 | 0.90 | 0.70 | 14.6 | 17.95 | -0.02 |
| 0.22 | 0.02 | 0.41 | 0.50 | 0.47 | 0.60 | 0.20 | 4.20 | 16.75 | 17.1 | -0.77 |
| 0.46 | 0.08 | 0.31 | 0.58 | 0.34 | 0.60 | 1.80 | 1.80 | 16.65 | 16.3 | 0.34 |
| 0.43 | 0.13 | 0.43 | 0.56 | 0.46 | 0.63 | 0.50 | 2.00 | 14.65 | 17.9 | -0.97 |
| 0.57 | 0.81 | 0.41 | 0.52 | 0.42 | 0.68 | 2.10 | 1.10 | 14.9 | 18.45 | 0.03 |
| 0.40 | 0.06 | 0.50 | 0.64 | 0.52 | 0.62 | 1.50 | 1.00 | 18 | 17.45 | -0.12 |
| 0.53 | 0.09 | 0.51 | 0.60 | 0.59 | 0.80 | 0.80 | 1.60 | 16.1 | 18.35 | -1.17 |
| 0.36 | 0.09 | 0.36 | 0.37 | 0.58 | 0.46 | 1.10 | 0.30 | 16.4 | 17.1 | -1.72 |
| 0.38 | 0.03 | 0.35 | 0.48 | 0.42 | 0.46 | 0.50 | 1.40 | 14.5 | 17.35 | -1.67 |
| 0.45 | 0.16 | 0.59 | 0.55 | 0.59 | 0.67 | 1.30 | 1.60 | 14.6 | 18.65 | 0.49 |
| 0.51 | 0.01 | 0.54 | 0.68 | 0.59 | 0.71 | 2.10 | 2.80 | 15 | 16.2 | 0.89 |
| 0.65 | 0.05 | 0.45 | 0.77 | 0.41 | 0.67 | 2.30 | 3.50 | 13.5 | 20.3 | 0.54 |
| 0.63 | 0.03 | 0.55 | 0.84 | 0.58 | 0.97 | 2.70 | 1.60 | 16 | 18.2 | 1.44 |
| 0.58 | 0.01 | 0.59 | 0.57 | 0.71 | 0.77 | 1.20 | 1.20 | 17.8 | 16.4 | 1.74 |

| Wild birds [1] | Captive-bred birds [1] | BO MO [2] | B1 MO [2] | B0 M1 [2] | B1 M1 [2] | Jan Rain (cm) [3] | Feb Rain (cm) [3] | Sept Temp (°C) [4] | Oct Temp (°C) [4] | Breeding Period (Standardised) [5] |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.52 | 0.02 | 0.56 | 0.61 | 0.53 | 0.57 | 0.70 | 0.90 | 18.45 | 18.3 | 1.79 |
| 0.44 | 0.02 | 0.45 | 0.51 | 0.56 | 0.56 | 0.90 | 1.00 | 17.95 | 17.85 | 1.04 |
| 0.53 | 0.05 | 0.55 | 0.66 | 0.63 | 0.79 | 1.60 | 1.80 | 17.4 | 18.7 | 0.69 |
| 0.69 | 0.07 | 0.50 | 0.82 | 0.51 | 0.77 | 2.70 | 1.80 | 16.6 | 19.9 | 1.34 |

[1] Mean no. of chicks per pair of wild and captive birds each year (1980 onwards) [2] Chick no's per pair in each of the four food treatments (banana supplement (B1), mango supplement (M1) and no supplement (B0 and M0)) in each year [3] Total rainfall in January and February of each year [4] Mean temperature in September and October of each year [5] Standardised length of the breeding season (difference from long-term average)

# Chapter 10

# Introduction to Multivariate Statistics (BIO-7025A)

# Chapter 11

# Workshop 6/7 - Multivariate Linear Modelling

## 11.1 Introduction

Multivariate linear modelling techniques are used when entering more than one predictor (independent variable) into an analysis of a dependent variable (also called response variable). The process and much of the output are very similar to those of simple linear regression, in which the effect of just one predictor is assessed. The aim is to explain the variance in the dependent variable, just as in simple regression, but with multiple predictor variables. Each of the predictor variables may explain a unique part of the variance in the dependent variable, but they may also share some of the variance that they explain.

The form of a multiple regression equation is an extension of that in simple regression:

$$y = m_1x_1 + m_2x_2 + m_3x_3... + c$$

and the statistical analysis is also in the form of ANOVA and t-tests of each parameter.

1) Multiple regression can be greatly influenced by which predictors are included, so it is important to not just throw in any old predictor; there should be a biological hypothesis attached to each predictor that is included.
2) The default option for how predictions are entered into models is **Enter**: all predictors are entered simultaneously and the researcher can then

choose to exclude non-significant variables (this is often called a step-down method).

3) In general with multiple regression, the fewer predictors the better, and you should have a minimum of 10-15 data points per predictor (or your model is unlikely to have sufficient power to identify relationships, should they exist)
4) Assumptions: even distribution of residuals, no heteroscedasticity and avoid collinearity of predictors (particularly when the correlation coefficient $r > 0.7$).

In a linear model, you have three ways of entering a predictor variable:

- **Covariates:** continuous predictor variables are entered as covariates
- **Fixed factors:** these are categorical predictor variables for which you want to compare the different categories.
- **Random factors:** these are categorical variables that need to be included in the model but for which you don't want to compare the categories. Random factors are often included to account for some non-independence in your data (eg multiple datapoints from the same site or year).

**You MUST NOT include both fixed and random effects in the same linear model – for that you have to use a MIXED MODEL**

## 11.2 Practical 6 - Multiple Linear Regression

This week we will be analysing three separate data sets; Africa, chaffinch and beetle. You can find these in our new classroom LINK HERE. Try to pay special attention to the data interpretation steps and make sure you understand the logic behind each of the decisions made.

Once you have joined the classroom, spend some time setting up your workspace and script as covered in Chapter 2.5.3 and Chapter 2.5.4. You will need to install and load the packages `tidyverse` and `janitor` for this weeks sessions.

## 11.3 Data set 1: Africa

Here you will be analysing data from 34 sub-Saharan African nations, collected by the World Bank in 1985. The aim is to fit a model that can be used to predict per capita calorific intake rate and to examine, from a conservation perspective, how deforestation impacts on calorific intake.

The variables are:

Dependent Variable: Possible Predictor Variables:

- daycal: per capita daily calorific intake
- femlit: female illiteracy rate (for over 15s)
- under15: percentage of the population under 15

- deforest: annual rate of forest loss
- gnp: per capita Gross National Daily Product (US $)

Theories have been suggested that poor diet, as represented by low calorific intake, is more serious in countries with poor economies, lower levels of education, a young population, and high levels of deforestation. In this practical, try to assess whether such relationships exist.

### 11.3.1 Task 1 - Checking the data

Load the dataset `africa.csv` into your workspace under the object name `africa` and try performing some routine checks to make sure R has interpreted the variables correctly and all expected data is present and accounted for. You can use some of the functions explored in Chapter 3.2.3 if you need some help with this.

### 11.3.2 Task 2 - Exploring the data

Start by exploring the univariate relationship between per capita daily calorific intake and gross national product.

- Write down what question you are asking, and turn this into a word equation (i.e. y = mx + c, with symbols replaced by words).

Draw the scatterplot for the relationship between per capita daily calorific intake (daycal) and gross national product (gnp), revisit Chapter 6.2.3 if you're not sure how to do this.

- Does the graph suggest that the assumptions of linear regression (i.e. that all datapoints are contributing similarly to the slope) have been met?

### 11.3.3 Task 3 - Regression analysis

Try running a linear regression for daily calorific intake and gross national product, store your model under the object name `daycal_gnp_lm_1` revisit Chapter 6.2.4 if you're not sure how to do this.

- What does the regression model output tell you about the relationship between calorific intake and GNP?
- Write down what the slope, constant, r2 and p-value tell you about this relationship, and consider whether they accurately describe this relationship.

As we did in Chapter 6.2.4, try plotting the residuals against the predicted values.

- From this plot, have the assumptions of the linear regression been met?

We can also calculate leverage values, this measures the influence of each point on the fit of the regression and can range from 0: no influence to 1: complete influence. Try running the following;

```
# Checking the leverage
africa <- africa %>%
  mutate(leverage = hatvalues(daycal_gnp_lm_1))
```

- What do the leverage values are indicate?
- Are any countries not fitting the model well?

Try performing a log10 transformation on the gnp variable, revisit Chapter 7.2.3 if you unsure how to do this. Then re-run the regression model using the logged variable, and redraw the scatterplot

- Has the model fit improved? Compare the slope, intercept, r2 and p-value to the model with the unlogged GNP.

Have a look at the leverage values and the residuals for your new model. Country 11 (Gabon) still has something of a disproportionate effect on the fit of the regression line (large-ish leverage), even in the model with log (GNP). Use the following command to temporarily exclude this country from our scatter plot (note we are not deleting the data completely). This will allow you to explore the influence of Gabon.

```
africa %>%
  filter(name != "Gabon") %>%
  ggplot(data = ., aes(x = gnp, y = daycal)) +
  geom_point() +
  geom_smooth(method = lm)
```

Now use the same temporary exclusion method to re-run your regression model with Gabon temporarily excluded.

- How has the regression changed? What reasons may there be for omitting Gabon from the analysis? Why should you always be careful when dropping variables for this reason?

Re-include Gabon into subsequent analyses.

### 11.3.4 Task 4 - Multivariate linear regression analysis

We are now going to build a multivariate regression model using the Enter method, in which we begin by including all predictor variables in our regression. We then remove from our regression the variable that has the least (statistically non-significant) effect to produce a better model. We repeat this until we are left with a regression containing the best sub-set of predictors. At each stage

it is important to consider the impact of removing that predictor on the model. First we will draw a matrix plot to compare all of the relationships between `daycal` and all four predictor variables (use your logged gnp variable to reduce difficulties with Gabon). Try using the following command;

```
# Scatter matrix
pairs(africa[,c(3:7,9)])
# Note the numbers c(3:7,9) refer to column numbers for daycal, femlit, under15, deforest and log
```

- Write down what these graphs tell you about each of the relationships
- Do you see any signs of collinearity between predictor variables?

Now we can try fitting the full multiple regression by adding the other three predictors as independent variables with Enter as the data entry method. Try running the following;

```
# multiple regression
daycal_multi_lm_1 <- lm(daycal~gnp+deforest+under15+femlit, data = africa)
summary(daycal_multi_lm_1)
```

- Which is the most statistically significant variable?

Remove the non-significant variable with the highest p-value from the regression model and examine the change in the $r^2$.

- Has removing the variable altered the explanatory power of the regression ($r^2$) very much? What has happened to the coefficients (slopes) of the remaining variables?

By removing the least significant variable at each stage, you reduce the regression equation until it only contains significant variables (this is called the minimum model). At each stage, check that removing a predictor variable does not cause the $r^2$ of the model to drop too much or cause the sign of the slope coefficient or the significance of other variables in the model to change strongly.

- Look at your final model. How does the $r^2$ compare with the $r^2$ you obtained in exercise 1, using simple regression with log(GNP) as the sole predictor?
- Does your final model fit in with the original hypotheses about daily calorific intake?
- What are the agreements and disagreements between your model and the theory (look back at the suggestions listed at the start of the practical)? Looking at the coefficients may be useful here.
- Write a word equation for your final model.

- Use your final model to predict the expected per capita calorific intake rate for a country with a GNP of US $975, a female illiteracy rate of 45% and a deforestation rate of -2.0.

R has a function called `predict()` which can be used to predict values of y based on your model. Try runnign the following;

```
new <- data.frame(gnp =975, femlit =45, deforest= -2)
# Create a new data frame called `new` and specify your independent variable values

predict(enter_your_final_model_here, newdata=new, interval = 'confidence')
# Call the predict function to use your final multivariate linear model to predict the
# By including interval = 'confidence' the output will include 95% confidence interval
```

- Compare the daycal rate you predicted for a country with a GNP of US $975, a female illiteracy rate of 45% and a deforestation rate of -2.0 to that predicted by the `predict()` function in R.
- Does the 95% prediction interval lie above or below the minimum acceptable daily calorific intake rate of 2000 kcal? What does this mean for this hypothetical country?

You can also specify interactions to test using the `lm()` function by using a `*` between two variables with a potential interaction, as seen here;

```
daycal_multilog_lm_2 <- lm(daycal~log10_gnp+deforest*femlit, data = africa)
summary(daycal_multilog_lm_2)
```

- Try customising different models, in order to explore this technique.

## 11.4 Data set 2: Chaffinch

Now we will move onto a second data set. Food was presented in a feeder linked to a balance for measuring mass. Individual chaffinches that had been trapped, ringed, measured and sampled for parasite prevalence were recorded visiting the feeder and the mass of food each bird consumed was recorded. The night-time temperature was also recorded and presented as an anomaly from the mean for that season. The dataset therefore contains the following variables:

- Ring: Ring number of the bird
- Species: Chaffinch
- Age: Age class (BTO codes)
- Sex: Male or Female
- Wing: Wing length (to the nearest mm)

- Weight: Mass (to the nearest 0.1 g)
- Food: Mass of food consumed in a visit (to the nearest 0.1 g)
- Parasite: Parasite load
- Temp: Temperature anomaly (relative to the mean night time temperature for the season (0.5 degrees), closer to 1 = colder)

### 11.4.1 Task 1 - Checking the data

Load the dataset `chaffinch.csv` into your workspace under the object name `chaffinch` and try performing some routine checks to make sure R has interpreted the variables correctly and all expected data is present and accounted for. You can use some of the functions explored in Chapter 3.2.3 if you need some help with this.

- If the aim of your study was to examine the factors influencing consumption rate of food by wild birds, write down which of these variables you would consider using as dependent variables and which ones as predictor variables.
- Write down the direction you might predict for each relationship.
- Which variables are continuous and which are categorical?

### 11.4.2 Task 2 - Exporing potential relationships

As we did with the last data set, draw a matrix scatter plot of all the continuous variables

- Have a look at how each variable appears to relate to your dependent variable. Are these the same directions that you predicted?
- Do you see any problems with the data meeting the assumptions of regression?

The relationship between parasite load and amount of food consumed is non-linear. Given the spread of the data, a logarithmic transformation of parasite load may help to linearise this relationship. Transform the parasite variable and redraw the matrix plot.

### 11.4.3 Task 3 - Multiple regression analysis

You can now build a series of models to explore specific hypotheses. Each time you should do the following:

1) Consider whether each variable is a covariate or a fixed factor
2) Make sure you check that all assumptions are met
3) Explore any collinearity between predictor variables and think about what you could do about it (see below)
4) Construct the model that contains only the significant predictor variables

5) Explore how the model changes each time you remove a predictor variable
6) Explore whether different minimum models could have been constructed
7) Draw graphs to check that you understand the model output
8) Consider what the slope, intercept, $r^2$ and p-values are telling you

### 11.4.4 Task 4 - When good predictors go bad (multicollinearity)

An underlying assumption of mulivariate linear model is that your predictors are not correlated with each other. That is, you are assuming that each predictor brings in entirely new information to your model. If two predictors are correlated, then they are redundant. We call this multicollinear predictors, or multicollinearity. For example, if you wanted to predict the wood volume of a tree, and you had three predictors, height, diameter, and number of rings, it is very likely that diameter and number of rings will be highly correlated with each other.

There are three issues with multicollinearity;

1) Firstly, if two predictors are strongly correlated with each other (strongly multicollinear) and both are in the regression model, both can end up being non-significant. That is, multicollinear predictors can cancel each other out, even if both, individually, are significant in a regression.

2) Secondly, the 'cancelling out' effect means that when you are simplifying a model by removing predictors, you should remove predictors one at a time instead of doing the tempting thing and removing all non-significant predictors at once. And you should try alternative minimal models.

3) Finally, the real lesson of multicollinearity is that you have redundant predictive information. Did you expect multicollinearity? Were you trying to measure the same predictor in different ways? If so, then have a good think about which predictor is more appropriate for your purpose (e.g., which one is cheaper to measure, or more scientifically appropriate, or more accurately measured, or all three). Did you not expect multicollinearity? Maybe the fact that two predictors are correlated tells you something new about your data or your system.

How to detect multicollinearity? The simplest way is to run a matrix plot of your predictors, and to check correlation statistics.

So before you even start your regression modelling, you know that you are going to run the risk of a regression where putting two or more will lead to both being non-significant. However, this is only a risk. It is not certain. In fact, sometimes, you need both predictors in the model for either one to be significant, as we saw above. There is no way to predict which will happen, so you just have to try out different models, choosing your final model based on logic and scientific knowledge.

- Take a look at your final model, do you have collinear variables included?
- Can you justify their inclusion or do you need to adjust your model further?

## 11.5 Data set 3: Beetles

Finally we will consider our third data set. These data are from a study of what causes variation in the density of beetles among different patches of tall vegetation. Beetle densities were sampled on 20 sites, along with a range of other variables:

- beetle_density: Beetle densities in each site
- lansdscape_type: Surrounding landscape (1 = arable, 2 = grassland, 3 = heath, 4 = wood)
- patch_age: Estimated patch age in categories (from 1 = young to 3 = old)
- patch_area: Area (m2) of each scrub patch
- veg_height: Maximum grass height (cm) at each site
- veg_density: Density of grass vegetation (from 1 = sparse to 5 = dense)
- plant_species: Mean no. of plant species per m2
- soil_moisture: Mean soil moisture level
- soil)_penetrability: Soil penetrability
- min_temp: Mean minimum daily temperature
- max_temp: Mean maximum daily temperature

You therefore have 10 predictor variables, three of which are categorical (landscape_type, patch_age and veg_density) and the rest of which are continuous

### 11.5.1 Task 1 - Checking the data

Load the dataset `beetle.csv` into your workspace under the object name `beetle` and try performing some routine checks to make sure R has interpreted the variables correctly and all expected data is present and accounted for. You can use some of the functions explored in Chapter 3.2.3 if you need some help with this.

You should note that R has misidentified your categorical variables as continuous. You will need to convert these variables to factors in order to correctly build your models. Try running the following line;

```
beetle$landscape_type <- as.factor(beetle$landscape_type)
# Convert the variable landscape_type to a factor rather then a continuous variable
```

Try running the `as.factor()` function on your other categorical variables (`patch_age` and `veg_density`).

### 11.5.2 Task 2 - Multiple regression analysis

Construct a linear model with beetle_density as your dependent variable and include all other variables as your independent variables.

- Look at your model output – what has happened here?

The main problem is that you have only 20 datapoints, and your model is trying to use 10 different predictors to explain the variation among these 20 sites – that is **too many predictors**! There is no way that the model can partition the variation in beetle density among all these predictors meaningfully. Remember that you should ideally aim for 10-15 datapoints per predictor. With 20 datapoints, you can realistically construct a model with 2, or maybe 3, predictors.

Which should you choose? Your best bet at this point is to think about the data you have and what they are telling you. For example, take a look at your patch_age data, you may like to use the `group_by()` and `summarise()` functions to do this.

- How many datapoints do you have for each of the patch_age categories?

With only one example of age 1 and one example of age 3, you cannot possible understand the variation in beetle densities between these age categories – this is therefore a predictor that should be excluded, although you may wish to run the final model with and without the two sites of age 1 and 3, to see if their inclusion changes anything.

Categorical variables use up more degrees of freedom than continuous variables, so they can be a particular problem with small datasets.

Look at one of the other categorical variables, landscape_type. Draw a scatter matrix plot with landscape_type, beetle_density and all of the continuous variables.

- Do any of these variables differ clearly between the four landscape types?

If you suspect any differences of note between any of the variables and landscape type it may be worth investigating these more closely, try checking means and standard deviations across different landscapes (you can use the `group_by()` and `summarise()` functions to do this) alternatively you may like to visualise the differences with some box plots.

- What do you think now, do any of your continuous variables differ between the four landscape types?

Broadly, there doesn't seem to be much difference between the continuous variables measured and the four landscapes, so it is unlikely that the surrounding landscape type is an important variable to include. Rebuild a linear model with landscape_type excluded.

- How is your linear model looking now?

Let's think about the other variables.

Draw a matrix scatterplot of all of the remaining variables.

- Is there any evidence that any of the variables are strongly correlated? If so, why do you think they are correlated and do you think that including both in the model will help you to understand what influences beetle density?

As dry soils are typically harder, soil moisture and soil penetrability are likely to be two different ways of measuring the same thing, so you can exclude one of them. Try rebuilding your linear model with one of these variables excluded.

We are now left with seven predictor variables (and wishing we had collected data from more sites…). Think about each of these seven variables and how likely they are to help you understand the variation in beetle density:

- Is patch area likely to influence beetle density?
- Is the maximum height of the grass within a patch likely to influence beetle density?
- Is the density of grass in a patch likely to influence beetle density?
- Is soil wetness or softness likely to influence beetle density?
- Are both temperatures likely to influence beetle density?

From this point onwards, you have to use your knowledge of biology and your common sense to try and pick the variables that you think will be most relevant and useful to you. For example, if the aim of your study is to develop management proposals for these sites, you might focus on variables that can be managed (i.e. vegetation and soil moisture, but not temperature). You can also explore the effect of each variable on beetle density separately, but remember that the influence of variables can change when they are included along with others in a multivariate model, so don't use univariate exploration as the only basis for excluding variables.

- Have a go at building the best model that you can for understanding the variation in beetle densities among these 20 sites.
- Remember to draw graphs to make sure you check the model assumptions and understand the model output

## 11.6 Before you leave!

Make sure you save your script and download it if you would like to keep a local copy.

Please log out of posit Cloud!

# Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2023). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.34.