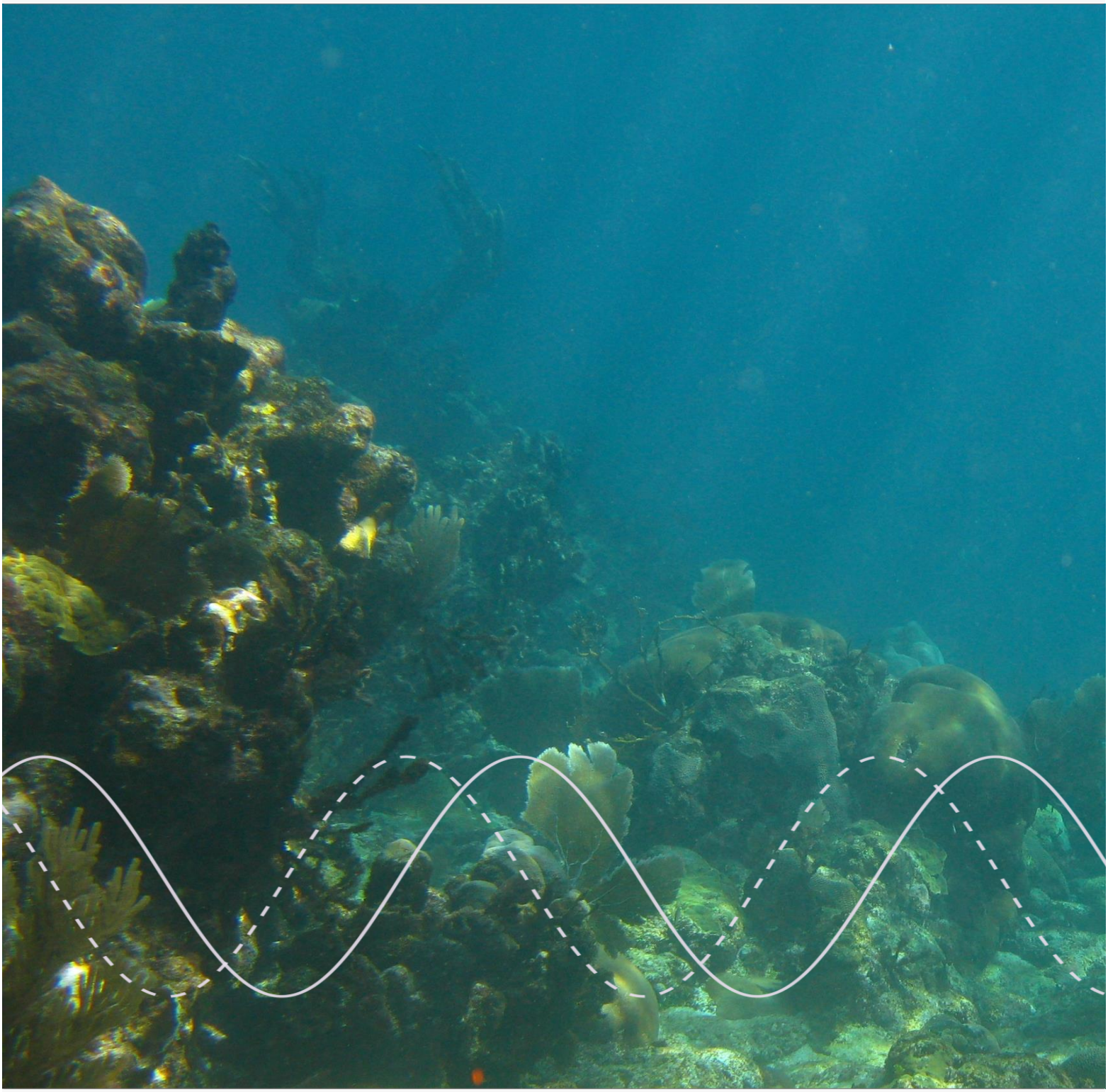
# H2 Data Science Practicals



Author: Tom Wilding
Revisions: Tim Szewczyk

SAMS Scottish Association for Marine Science
A partner of UHI

# Contents

# Overview

## 0.1 Practical sessions

In these practicals, you will practice applying the data science concepts you learn throughout the H2 Data Science course. The questions you answer by hand are useful for learning and revision, while producing appropriate graphics and correctly describing results are essential parts of the scientific process. Expertise in these core skills is essential to do well in future courses and scientific projects. H2 Data Science is, therefore, one of the most important courses you'll take!

In each practical, you will either generate your own data or process and analyse data sets that I have either obtained or generated to illustrate particular points. They are stored in labelled Excel spreadsheets under a variety of named worksheets.

Each session is 2.5h, during which you will work through a series of coding exercises and questions. These are not assessed or marked, but some of them appear in the assessments. You should complete all the material in each practical. My recommendation, particularly for those not comfortable with R, is that you read through the practical before the class.

Throughout the sessions, we will alternate between individual work and short re-caps as a class. If you find yourself racing ahead, all well and good. If you find yourself falling behind, then try some additional study ahead of the next session.

## 0.2 Assessments

During this time slot throughout the semester, there will be six practical sessions and three assessments (see the schedule on Brightspace). All three assessments are via Brightspace and consist of multiple choice, single fill-in-the-blank, and slightly more expansive answers. They are 'open Brightspace' but timed, so you will need to understand the material and be comfortable working in R to do well in these assessments. You are welcome to bring a hard copy of this booklet with annotations if you wish.

---

## 0.3 Getting started in R

R is a statistical language and computing platform that is widely used in the sciences. It is free and open source. We will be using it extensively in this course, so I recommend you review your notes and course material from last year to prepare for this course.

There are also fantastic resources available online. In addition to those listed in the module handbook, there are courses such as those from Software Carpentry or Swirl. You should access

these resources in your own time to complement, revise, and reinforce the concepts you'll be learning during this course.

Try to avoid copying and pasting code where possible during these practicals. R is best learned through your fingers, and working through errors, though frustrating, is an essential skill.

### 0.3.1 RStudio Project

Working in a Project within RStudio is the best way to avoid working directory complications.

Open RStudio, go to 'file' then 'New project'. Name and save your project (e.g., 'H2_DataScience.Rproj') in a convenient location. OneDrive works, though occasionally gives sync errors with hidden files if you use multiple machines.

Next, in the same folder as your .Rproj, create a folder called *data* and a folder called *code*.

Download the files in *Practicals* > *data* on Brightspace and move them into your *data* folder.

Your files should be visible under the 'Files' tab in R Studio, together with your new project.

The code in the practicals assumes this setup. I recommend creating a separate R script (.R) or R Markdown file (.Rmd) for each practical session, saved within the *code* folder.

Each time you open your .Rproj, the working directory will be set to the enclosing folder. The working directory is shown at the top of the Console pane, and you can check it with `getwd()`.

### 0.3.2 RStudio Settings

You can adjust many settings in RStudio via Tools > Global options. In the Appearance tab in the popup box, you can set the theme (e.g., if you prefer a dark theme), font size, etc. The Code tab has many nice features as well (e.g., rainbow parentheses under Display).

### 0.3.3 R packages and libraries

R packages are collections of functions, custom data structures, and datasets that are developed by the userbase. A new installation of R includes many useful default packages, visible on the 'Packages' tab in RStudio. There are many additional packages available from the official CRAN repository or less officially from GitHub. If you find yourself re-using custom functions across projects, you can even create your own personal package.

To install a package from CRAN, use the function `install.packages("packageName")`. This downloads the package files to your computer. Each time you open R, you will need to load that package to use it with `library(packageName)`.

Installing from other package sources is slightly more complicated, so see me if you have a need.

You can get an overview of a package with `?packageName`, and then see a list of all of the functions by scrolling to the bottom of the help page and clicking the "index" link.

The help for each function is available with `?functionName`, and you can see the underlying code with `functionName` without parentheses.

# Chapter 1

# Displaying and summarising data

Statistics can be divided into two broad categories - descriptive statistics and inferential statistics. In this practical session we will focus on descriptive stats and, in particular, how to appropriately display and summarise data. You should already be aware of several techniques for displaying data. These techniques are primarily graphical and include bar charts, histograms and graphs. When and how to use these different techniques is one focus of today's practical.

There are two purposes for visualizing data.

First, the best way to get a 'gut' feel for your dataset is to look at it graphically. Examining data graphically enables you to identify any outliers (i.e., suspicious observations which could be errors). It will also help you to select the most appropriate inferential statistical model (more on this through the course).

Second, visualizations are used to impart information as clearly as possible to 'the reader' (i.e., drawing the reader's attention to the most interesting aspects of your data). Graphics that are confusing, either through a lack of detail (e.g. no labels) or that contain too much information will fail in this central objective.

As you create graphics, keep in mind that they may be viewed on different machines or printed in grey scale. Importantly, some colour combinations may be difficult for colour-blind or visually impaired readers. Colour scales such as those available from ColorBrewer or viridis are designed with this in mind.

R has established best practices to make your meaning clear. Just like any language, you|can|write|with|your|own|system, but it's easier for everyone to use standard conventions. A full style guide is available if you're interested.

A few key points:

- Use `<-` to *assign* a value to an object. You may see `=`, which works, but is not preferred.
- Use spaces to make your code legible: `a <- 10`; `mean(x, na.rm=T)`; `c(1, 2, 3)`.
- Avoid spaces in column or file names (in general) as these are a pain to work with.
- Use names for objects that are short, but descriptive.
- Limit the length of a line of code to about 80 characters.
- Usually, variables should be nouns and functions should be verbs.
- Use `#` to write a comment which R will ignore.
- Run the line of code where your cursor is (or everything you've selected) with ctrl+r

## 1.1 Basic data exploration

We will mostly be working with dataframes. A dataframe is a 2D rectangular structure with columns and rows. In a tidy dataset, each row represents an 'observation' and each column represents a 'variable'. R (and often packages) contains several built-in dataframes.

The dataframe `cars` gives the max speeds and stopping distances for cars built in the early 20th century. We are going to use this dataset as a starting point to demonstrate a few basic concepts in relation to R programming and statistical analysis.

```
# functions for basic details of objects
str(cars)
class(cars)
names(cars)
head(cars)
```

```
head(cars, 2)
```

```
##   speed dist
## 1     4    2
## 2     4   10
```

```
tail(cars, 2)
```

```
##    speed dist
## 49    24  120
## 50    25   85
```

```
# what are the last 10 rows?
```

There are several ways to access subsets of a dataframe:

- Use `.$columnName` or `.[["columnName"]]` to extract a single column
- Use `.[rows,columns]` to extract a block

```
cars$speed
cars[["speed"]]
```

```
cars[1, 1] # row 1, column 1
```

```
## [1] 4
```

```
cars[1:5, 1]
```

```
## [1] 4 4 7 7 8
```

```
cars[1:3, ] # leaving the columns blank returns all columns
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
```

We can also change column names. We'll make a copy of the dataframe to do that. Rearranging and duplicating columns is also easy.

```
head(cars2, 2)
```

```
##   speed_mph dist_ft
## 1         4       2
## 2         4      10
```

```
cars2$speed_mph
```

```
##  [1]  4  4  7  7  8  9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15
## [26] 15 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24 25
```

```
cars2 <- cars2[, 2:1]
head(cars2, 2)
```

```
##   dist_ft speed_mph
## 1       2         4
## 2      10         4
```

```
cars3 <- cars2[, c(2, 1, 1)] # duplicate a column
head(cars3, 2)
```

```
##   speed_mph dist_ft dist_ft.1
## 1         4       2         2
## 2         4      10        10
```

```
cars3 <- cars3[, c(2, 1)] # remove the duplicated column
head(cars3, 2)
```

```
##   dist_ft speed_mph
## 1       2         4
## 2      10         4
```

```
cars3$dist_x_speed <- cars3$dist_ft * cars3$speed_mph # create a new column
head(cars3, 2)
```

```
##   dist_ft speed_mph dist_x_speed
## 1       2         4            8
## 2      10         4           40
```

```
rm(cars3) # remove the dataframe 'cars3'
```

You can also subset based on criteria. Say we only want rows where the speed is $> 20$ mph:

```
a <- which(cars2$speed_mph > 20)
str(a)
```

```
##  int [1:7] 44 45 46 47 48 49 50
```

```
cars_fast <- cars2[a, ] # or: cars2[which(cars2$speed_mph > 20), ]
class(cars_fast)
```

```
## [1] "data.frame"
```

```
ncol(cars_fast)  # and how many *rows* are there?
```

```
## [1] 2
```

```
head(cars_fast, 2)
```

```
##    dist_ft speed_mph
## 44      66        22
## 45      54        23
```

When you import data, you should check for missing values. These are represented as `NA`.

We can check each element of a vector using `is.na()`, which will return `FALSE` if an element *is not* `NA`, and `TRUE` if an element *is* `NA`.

```
is.na(cars2$speed_mph)
```

R converts a logical vector (i.e., `TRUE`/`FALSE`) to numeric (i.e., `1`/`0`) automatically. This is handy, but dangerous if you don't realize it.

```
sum(is.na(cars2$speed_mph))
```

```
## [1] 0
```

```
carsNA <- cars2
carsNA[c(2, 4, 5, 10), 1] <- NA
sum(is.na(carsNA$dist_ft))
```

```
## [1] 4
```

Another very useful check is `summary()`:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

```
summary(carsNA)
```

Once you have assured yourself that your dataframe looks sensible, that it contains the data you expect, and that you know what the data-types are, you can start to explore and summarise your data.

There are many graphical methods for data exploration and it is important to select the appropriate method. This will be determined by the nature of the data and what you wish to communicate to the reader.

---

## 1.2 Graphical methods for displaying data

Always keep in mind that the primary reason for data visualization is to impart information concisely and accurately to your reader.

Graphics must be clear, concise and easy to understand. Brightspace contains some examples of bad graphics ('Learning resources>Lecture support material>Introduction (Lectures 1-3)>Graphics').

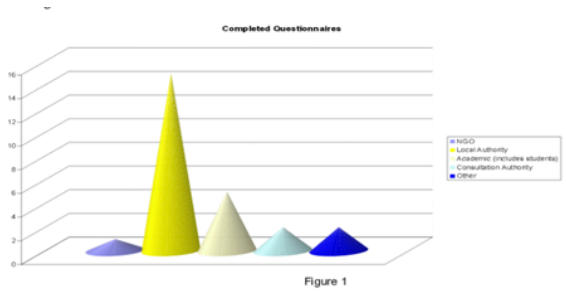



Figure 1.1: An example of a terrible graphic, as published in a Scottish government report.

In addition to poor design choices for effective communication (Fig. 1.1), graphics can also be deliberately misleading (Fig. 1.2).

### 1.2.1 Scatter plot

The scatter plot is used where you are plotting two continuous variables against each other. It can be used to check whether outliers are present in your data. This is the plot to use if you are doing a correlation analysis and you need to show the raw data to your reader.

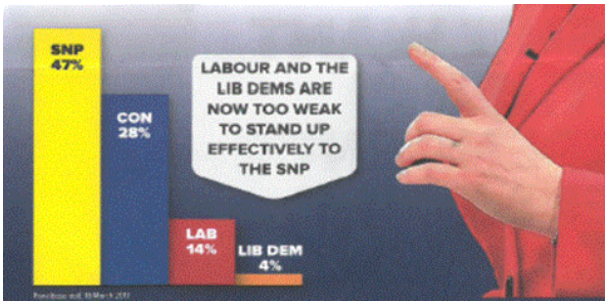
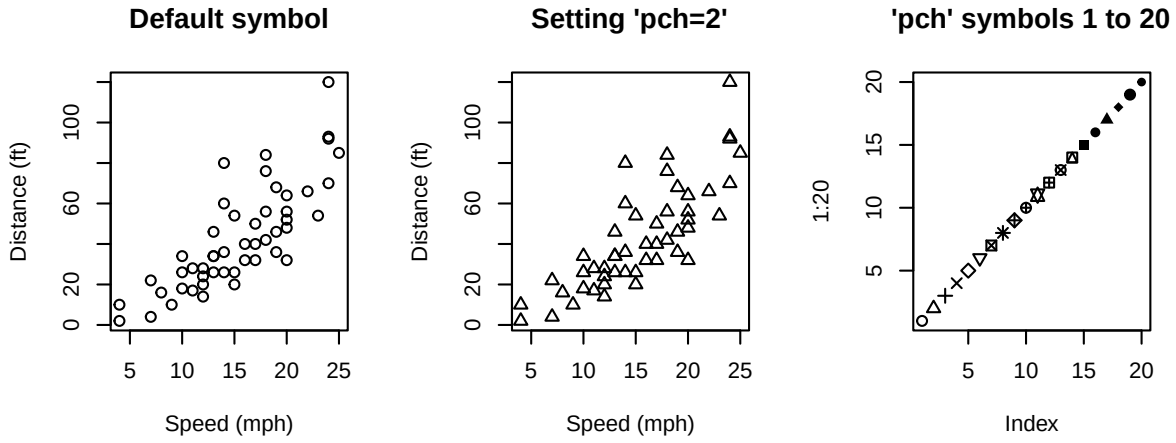



Figure 1.2: A misleading graphic. What type of plot is this and how is it misleading?

```
# plot(reponse ~ predictor, data = dataframe)
plot(dist_ft ~ speed_mph,
  data = cars2, xlab = "Speed (mph)", ylab = "Distance (ft)",
  main = "Default symbol")
plot(dist_ft ~ speed_mph,
  data = cars2, xlab = "Speed (mph)", ylab = "Distance (ft)",
  pch = 2, main = "Setting 'pch=2'")
# you can check out more symbols and their respective numbers using this plot:
plot(1:20, pch = c(1:20), main = "'pch' symbols 1 to 20")
```

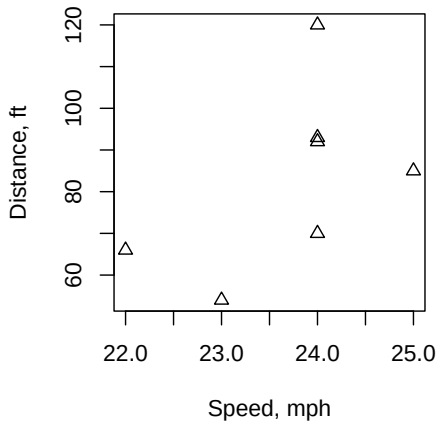



*Q1. Search the* `plot` *help page for 'title', then add an appropriate title to your plot.*

*Q2.* `?points` *opens the help page for points. Search the help page for 'pch' and change the symbol of your plot*

*Q3. In relation to the cars dataset, plot stopping distance by speed, using indexing to show only those cars where speed_mph>20*

```
plot(dist_ft ~ speed_mph, data = cars2[cars2$speed_mph > 20, ],
     xlab = "Speed, mph", ylab = "Distance, ft", pch = 2)
```




### 1.2.2 Boxplots

Boxplots are used to summarise a continuous variable by levels of a factor. We will use the `mtcars` dataset to illustrate this.

Explore the dataframe using the code you've already covered. Which variables are categorical, which are continuous (or might be)?

```
head(mtcars, 2)
```

```
##               mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```
boxplot(mpg ~ cyl, data = mtcars)
```

```
?boxplot
# See examples at bottom of the help page
# Produce a boxplot with axes labels and a title
```

Reproduce the plot shown in Fig. 1.3 (assume 1 gallon = 4.5 L). Remember you can learn about these data with `?mtcars`. You will need to generate a new variable (miles per litre) and label your boxplot appropriately. You can limit the extent of the y-axis by adding the argument `ylim = c(a, b)` where `a` and `b` are the limits you want (e.g., `ylim = c(0, 100)`).

*Q4. Use `?boxplot` to investigate what the box and whiskers in the boxplot acutally represent. Check you can reproduce the upper and lower adjacent values manually (see Chapter 7)*

### 1.2.3 Line plots

Line plots are most often seen in time-series plots with time (e.g. days or years etc) on the x-axis and the response on the y-axis. Line plots typically involve joining points with a line. The line can be straight or curved. Whatever the line is (which will be your choice), it indicates

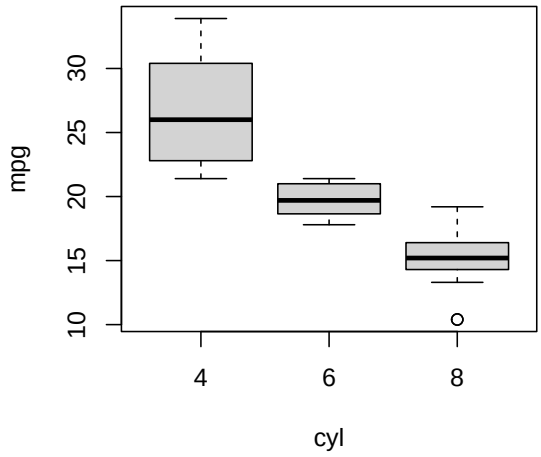



Figure 1.3: Boxplot showing miles per litre vs. number of carburetors

that you have made assumptions about the value of the response variable between successive measurements.

We will examine these plots using the dataset `lynx`, which consists of the number of Canadian lynx pelts sold per year between 1821 - 1934. It is a 'classic' dataset as it shows a cyclical 'boom-and-bust' lynx population (demonstrating predator-prey interactions).

First, we will create a variable `Year`.

```
str(lynx)
```

```
##  Time-Series [1:114] from 1821 to 1934: 269 321 585 871 1475 ...
```

```
class(lynx) # ts = time-series
lynx2 <- as.data.frame(lynx)
class(lynx2)
head(lynx2, 2)
lynx2$Year <- seq(from = 1821, to = 1934, by = 1)
```

In R, we use *functions* to perform actions on *objects* . Functions have arguments, taking the form `functionName(arg1=..., arg2=...)`. If you do not name the arguments, the function will assume that you are listing the arguments in order. See the help file with `?functionName` to see the argument order.

*Q5. Using `seq()`, write a piece of code which generates odd numbers between 1 and 20 with and without specifying `from`, `to`, and `by`.*

```
# change the name of the 1st column to 'Trappings'
names(lynx2)[1] <- "Trappings"
str(lynx2)
lynx2$Trappings <- as.numeric(lynx2$Trappings) # Time-Series is complicated.
str(lynx2)
```

Use `?plot` to investigate options for plotting. Find the `type=` argument for plotting both the points and a connecting line. This might be the best option in this case. Why? Remind yourself the purpose of graphics.

*Q6. Using the R plot function, produce a line plot of the Trappings data, as per Figure 1.4.*

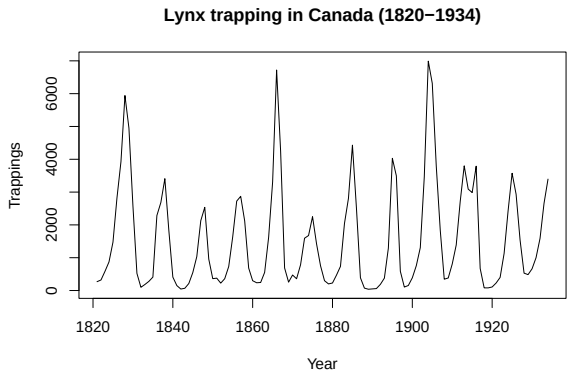



Figure 1.4: The number of lynx trapped in Canada (1820-1934)

Change the plot to show only the years up to 1900, then plot the Trappings on the log scale.

### 1.2.4 Histograms

Histograms are used to illustrate the distribution of continuous data. Histograms are often erroneously used to plot discrete data. In histograms the bars are adjacent (no gap) and this indicates that there is a continuum (i.e. that the data are not discrete).

```
# this gives very different information.
hist(lynx2$Trappings, main = "Lynx trapping", xlab = "Trapped lynx per year")
```

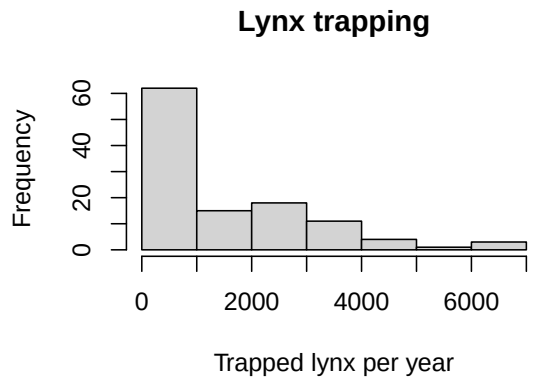



Figure 1.5: Lynx pelts per year with default settings.

Which range of values was most common across years?

Be aware that histograms can be quite sensitive to the bins that you use.

```
par(mfrow=c(1,2)) # panels for the plotting window
# R takes the number of breaks as a suggestion
hist(lynx2$Trappings, main = "Lynx trapping", xlab = "Trapped lynx per year",
     breaks = 5)
# this forces R to plot according to the defined breaks
hist(lynx2$Trappings,
     main = "Lynx trapping", xlab = "Trapped lynx per year",
     breaks = c(0, 500, 1000, 2000, 5000, 10000))
```

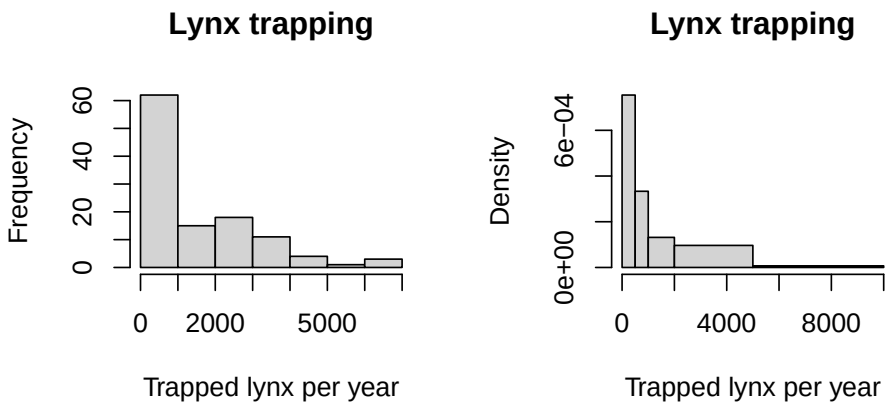
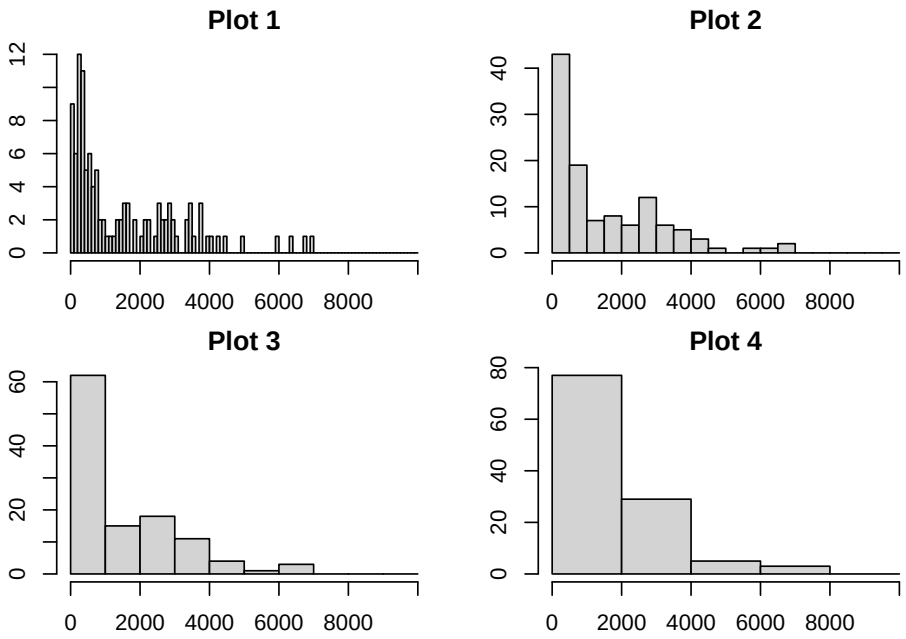



Figure 1.6: Lynx pelts per year with breaks=5 (left) and a vector of breaks (right).

```
par(mfrow = c(2, 2)) # plot panels (2 rows x 2 columns)
par(mar = rep(2, 4)) # change the plot margins
hist(lynx2$Trappings, main = "Plot 1", xlab = "Trapped lynx per year",
     breaks = seq(from = 0, to = 10000, by = 100))
hist(lynx2$Trappings, main = "Plot 2", xlab = "Trapped lynx per year",
     breaks = seq(from = 0, to = 10000, by = 500))
hist(lynx2$Trappings, main = "Plot 3", xlab = "Trapped lynx per year",
     breaks = seq(from = 0, to = 10000, by = 1000))
hist(lynx2$Trappings, main = "Plot 4", xlab = "Trapped lynx per year",
     breaks = seq(from = 0, to = 10000, by = 2000))
```




```
par(mfrow = c(1, 1)) # reset the par setting.
```

Which of these plots is the most useful? There is no definitive answer to this, but Plot 1 is very busy and Plot 4 fails to show relevant detail near 0; Plot 2 or 3 communicate the patterns in the data most clearly.

As a general guideline, 5-15 breaks usually work well in a histogram.

### 1.2.5 Bar graphs

When you create a `data.frame` it defaults to naming the rows 1…n, where n is the number of rows. **It is better practice to store relevant information in a column**, but you may occasionally come across a `data.frame` with row names. Converting between data types may lose this information.

Bar graphs are used to plot counts of categorical or discrete variables. We'll be using the `islands` dataset.

Working with data involves a lot of time spent tidying the datasets: cleaning, checking, and reshaping into useful formats. We will cover a more modern set of methods for this later in the course using the *tidyverse* package. For now, we'll stay with base R. First, we need to tidy the `islands` data.

```
str(islands)
class(islands) # this is a named numeric vector
head(islands)

# convert to a dataframe
islands.df <- as.data.frame(islands)
head(islands.df, 2)
```

```
# put the row names into a new column
islands.df$LandMass <- row.names(islands.df)
head(islands.df, 2)
```

```
##            islands   LandMass
## Africa       11506     Africa
## Antarctica    5500 Antarctica
```

```
# set row names to the row number
row.names(islands.df) <- 1:nrow(islands.df)
names(islands.df)[1] <- "Area"
head(islands.df, 2)
```

```
##    Area   LandMass
## 1 11506     Africa
## 2  5500 Antarctica
```

```
# reorder by area
islands.df <- islands.df[order(islands.df$Area, decreasing = TRUE), ]
head(islands.df, 3)
```

```
##     Area      LandMass
## 3  16988          Asia
## 1  11506        Africa
## 35  9390 North America
```

We can use the function `barplot()` to plot the vector of island areas.

```
par(mar = c(4, 0, 0, 0)) # change the margin sizes
barplot(islands.df$Area)
```

Figure 1.7: Island areas with barplot defaults

The whole dataset includes a lot of very small areas, so let's cut it down to just the 10 largest. Since the dataset is already sorted, we can take rows `1:10`.

```
barplot(islands.df$Area[1:10])
```

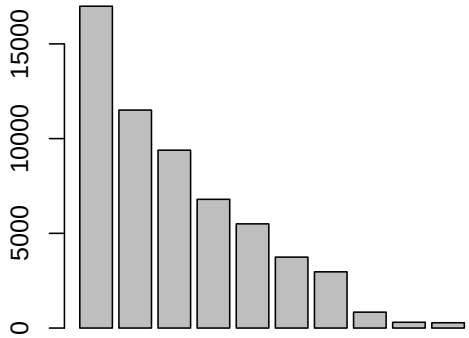



Figure 1.8: Top 10 island areas

And the next step is to add some names to the x-axis…

```
barplot(islands.df$Area[1:10], names = islands.df$LandMass[1:10])
```

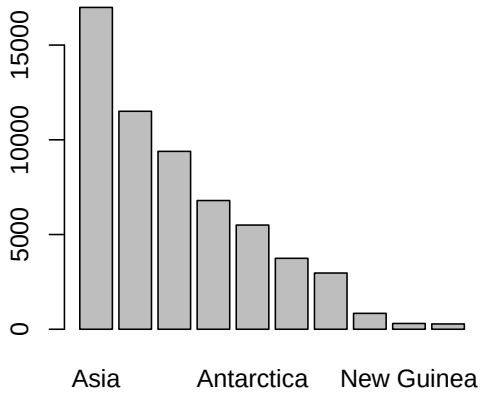



Figure 1.9: Top 10 island areas with names

Which of course are unreadable. The `las` argument (`?par`) controls how the axis labels relate to the axis line, so we can try adjusting that…

```
barplot(islands.df$Area[1:10], names = islands.df$LandMass[1:10], las=3)
```

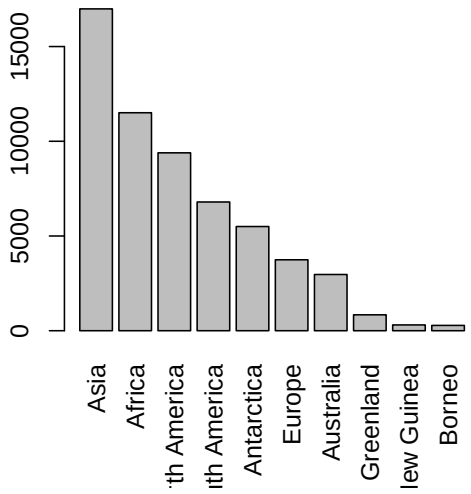



Figure 1.10: Top 10 island areas with names rotated

Maybe we just need to make the bars horizontal. To do this, we should adjust the margins again with `par(mar=...))`, set `horiz=TRUE`, and `las=1`, and use `[10:1]` so the largest is on top.

```
par(mar = c(4, 10, 0, 0))
barplot(islands.df$Area[10:1], names = islands.df$LandMass[10:1],
        horiz = TRUE, las = 1, xlab = "Area (km2)")
```

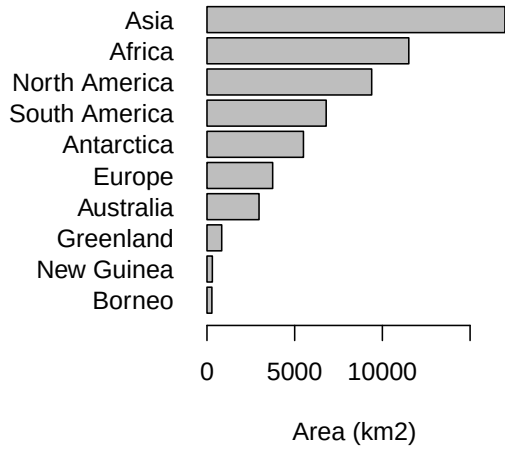



Figure 1.11: Finally! Did you know Antarctica is bigger than Europe?

As you may have noticed, visualization is an iterative process with lots of trial and error until you find a plot that communicates the message within the data well. There are several packages (e.g., `ggplot2`) that make these sort of adjustments and explorations less opaque than all of the options in `par()`.

## 1.3 Summary statistics

You will often need to summarise your data before you present it. Data summaries are usually contained in tables and they can replace graphics in certain situations, for example, where the data is relatively simple with a well understood distribution. There are numerous different types of summary statistics. Here we are concerned with central tendency and variability.

*Q8. What are the three main measures of central tendency?*

*Q9. What are three measures of variability?*

Different measures of central tendency and variability all have pros and cons and you need to be able to apply the most appropriate method to your data. Another summary statistic that you might include is sample size. R is very good at producing summary statistics, and there are myriad ways to produce them. We'll return to the `cars` dataset.

```
summary(cars2)
```

```
##     dist_ft         speed_mph
##  Min.   :  2.00   Min.   : 4.0
##  1st Qu.: 26.00   1st Qu.:12.0
##  Median : 36.00   Median :15.0
##  Mean   : 42.98   Mean   :15.4
##  3rd Qu.: 56.00   3rd Qu.:19.0
##  Max.   :120.00   Max.   :25.0
```

```
summary(cars2[cars2$speed_mph > 20, ])
```

```
# There are several ways to access a column in a dataframe
summary(cars2$speed_mph)
summary(cars2[, 2])
summary(cars2[, "speed_mph"])
summary(cars2[, c("speed_mph", "dist_ft")])
```

Often you'll wish to summarise your data across levels of certain factor. For example, levels of a certain treatment that you are applying. More complex summaries can be made using the `dplyr` package. We'll go into more detail later on some of the very powerful ways this package (and its friends in the *tidyverse*) can be used.

First, you'll need to install it. The *tidyverse* is a collection of packages. Install all of them with `install.packages("tidyverse")` (see Section @ref{R_intro}).

We'll use the built-in dataset `InsectSprays`. Viewing your raw data can be an important check as well. You can open a spreadsheet-styled viewer in R using `View(YourDataFrame)`.

```
str(InsectSprays)
```

```
## 'data.frame':    72 obs. of  2 variables:
##  $ count: num  10 7 20 14 14 12 10 23 17 20 ...
##  $ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
glimpse(InsectSprays) # glimpse() is loaded with tidyverse
```

```
## Rows: 72
## Columns: 2
## $ count <dbl> 10, 7, 20, 14, 14, 12, 10, 23, 17, 20, 14, 13, 11, 17, 21, 11, 1~
## $ spray <fct> A, A, A, A, A, A, A, A, A, A, A, A, B, B, B, B, B, B, B, B, B~
```

```
# spray is the categorical predictor; count is the response
View(InsectSprays)
```

To do more complex summaries, we're going to string together a series of functions. This can be done in a nested format (e.g., `fun1(fun2(fun3(dataset)))`), but this gets unwieldy very quickly.

So, let's introduce the *pipe* operator `|>`. This takes the output from one function and feeds it as the first input of the next (e.g., `dataset |> fun3() |> fun2() |> fun1()`), making code much more legible. Many functions in the *tidyverse* are built for piping.

```
?`|>`
```

```
# use group_by() with the grouping column name(s)
spray_summaries <- InsectSprays |>
  group_by(spray) |>
  summarise(count_mean = mean(count))
spray_summaries
```

```
# it is very easy to calculate any number of summary statistics
InsectSprays |>
  group_by(spray) |>
  summarise(mean = mean(count) |> round(2),
            median = median(count),
            max = max(count),
            sd = sd(count) |> round(2),
            N = n(),
            N_over_10 = sum(count > 10))
```

```
## # A tibble: 6 x 7
##   spray  mean median   max    sd     N N_over_10
##   <fct> <dbl>  <dbl> <dbl> <dbl> <int>     <int>
## 1 A     14.5    14      23  4.72    12         9
## 2 B     15.3    16.5    21  4.27    12        11
## 3 C      2.08    1.5     7  1.98    12         0
## 4 D      4.92    5      12  2.5     12         1
## 5 E      3.5     3       6  1.73    12         0
## 6 F     16.7    15      26  6.21    12        10
```

### 1.3.1 Which measure of central tendency to use

The choice of which measure of central tendency to use depends on the nature of the data and objectives of your research. However, there are some general rules. We will use datasets that you downloaded from Brightspace (Practicals > data). Remember to put these into the *data* folder in your working directory (or modify the file paths in the code accordingly).

```
library(readxl) # installed with tidyverse, but not loaded in library(tidyverse)
# this will load the 'Scallop %fat'data sheet from the xlsx spreadsheet.
scallop_df <- read_excel("data/practical_1.xlsx", sheet = "Scallop %fat")
str(scallop_df)
```

```
## tibble [49 x 1] (S3: tbl_df/tbl/data.frame)
##  $ Scallop % fat: num [1:49] 22.5 24.1 18.2 32.5 17.4 23.6 21.5 22.2 27.6 22.2 ...
```

```
# avoid spaces and symbols in column names. It's a pain.
names(scallop_df) <- "fat_pct"
```

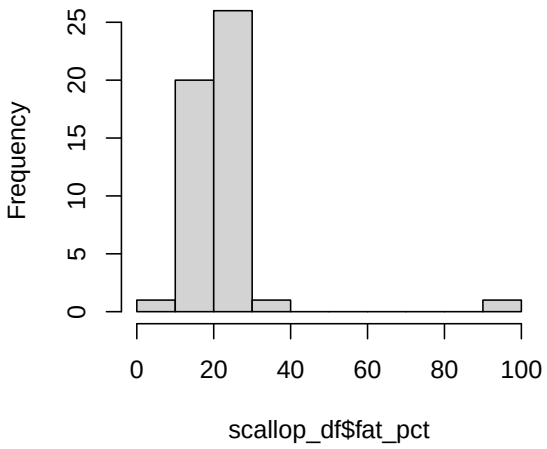
*Q10. Check the data using the methods above. Does it look OK to you?*

*Q11. Are these data likely to be continuous or discontinuous?*

*Q12. Create a plot to visualize the distribution of these data.*

*Q13. Do you spot any issues?*

```
hist(scallop_df$fat_pct, main = NULL) # (what does 'main = NULL' do?)
```




You should have spotted a potential outlier. Data entry errors are very common, and a check against the original data sheet shows that the decimal was typed in the wrong place. The following code helps you ID which data entry is in error. We can now search for the 'odd' observation i.e. determine in which row the outlier is located.

```
which(scallop_df$fat_pct > 50)
```

```
## [1] 36
```

```
scallop_df$fat_pct[35:37] # row 36 is 99, but should be 9.9
```

```
## [1] 22.8 99.0 12.9
```

```
scallop_df <- scallop_df[, c(1, 1)] # duplicate column
names(scallop_df) <- c("fat_pct_orig", "fat_pct_corr")
head(scallop_df, 2)
```

```
## # A tibble: 2 x 2
##   fat_pct_orig fat_pct_corr
##          <dbl>        <dbl>
## 1         22.5         22.5
## 2         24.1         24.1
```

```
# there are many ways to 'fix' the outlier in R.
# You need to correct the outlier in row 36 of column 'fat_pct_corr'
scallop_df$fat_pct_corr[36] <- 9.9
which(scallop_df$fat_pct_corr > 90)
# integer(0) - this means that no elements in fat_pct_corr contain values >90
```

Now summarise `scallop_df` using some of the methods above.

*Q14. Create a histogram for the corrected column. How does it differ from the original column with the error?*

*Q15. Calculate mean, variance, median, interquartile range, minimum, maximum and range for both fat_pct_orig and fat_pct_corr.*

*Q16. Suppose the outlier was even bigger (i.e. you typo was even worse). Adjust your data, multiplying the erroneous data item by 10; copy the '_orig' column and change row 36 in that column to 999.*

*Q17. Calculate the same summary statistics.*

*Q18. Which measures of central tendency and variability are most 'robust' against this outlier?*

Or look individually instead of calculating many metrics at once with `dplyr` functions:

```
summary(scallop_df$fat_pct_corr)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    8.50   16.90   20.60   19.56   22.50   32.50
```

```
var(scallop_df$fat_pct_corr)
```

```
## [1] 22.79836
```

```
IQR(scallop_df$fat_pct_corr)
```

```
## [1] 5.6
```

R is excellent at generating well formatted tables such as shown in Table 1.1. What is missing from from Table 1.1?

*Q19. How would the patterns seen in Table 1.1 influence your choice if you were required to summarise data that you thought might contain data that you weren't sure about? The three measure of central tendency are influenced to different extents by the 'shape' of the data they are used to describe.*

Table 1.1: Summary statistics with and without an outlier. Note which summary stats are most influenced by the outlier.

| Column | Mean | Median | Standard deviation | Range | Interquartile range |
|---|---|---|---|---|---|
| fat_pct_corr | 19.6 | 20.6 | 4.77 | 24.0 | 5.6 |
| fat_pct_orig | 21.4 | 20.6 | 12.20 | 90.5 | 5.3 |

Table 1.2: Summary of hake data.

| Year | Mean length (cm) |
|---|---|
| 1 | 201.8 |
| 2 | 497.0 |
| 3 | 988.9 |

```
hake_df <- read_excel("data/practical_1.xlsx", sheet = "Hake")
str(hake_df) # once again, column names made for excel
```

```
## tibble [499 x 2] (S3: tbl_df/tbl/data.frame)
##  $ Year            : num [1:499] 1 1 1 1 1 1 1 1 1 ...
##  $ Hake length (mm): num [1:499] 190 219 181 148 206 204 168 197 178 211 ...
```

*Q20. What type of variable is* `Length`*?*

*Q21. Select an appropriate graphical method and display these data.*

*Q22. In your own time, use the* `dplyr` *functions to summarise the hake data by year.*

```
hake_df$Year <- as.factor(hake_df$Year) # Treat as categorical, not numeric
names(hake_df) <- c("Year", "Length") # simplify the column names
```

Try to re-create Table 1.2.

The following 'settling velocity' data relates to the settling velocity of salmon faecal material. Shona Magill generated these data.

```
library(readxl)
fishPoo_df <- read_excel("data/practical_1.xlsx", sheet = "Settling velocity")
str(fishPoo_df)
```

```
## tibble [200 x 1] (S3: tbl_df/tbl/data.frame)
##  $ Settling velocity (mm s-1): num [1:200] 2.06 1.03 1.56 1.88 1.16 0.76 1.26 1.13 1.23 1.31 ...
```

*Q23. Produce a histogram of the settling velocity. Is it left or right skewed?*

*Q24. Which measures of central tendency and variability are most appropriate?*

*Q25. Sketch the distribution and indicate the relative positions of the mean and median.*

*Q26. Generate a new column of the log-transformed settling velocity data and plot these data.*

Table 1.3: Appropriate measures of central tendency and variability according to the underlying data distribution.

| Data distribution | Central tendency metric | Variability metric |
|---|---|---|
| Continuous, unimodal, symmetrical | Mean | Variance or sd |
| Continuous, skewed | Median | Interquartile range |
| Continuous, multimodal | None; state modes | None; summarise by group |
| Discontinuous | None; data-dependent | Range? |

*Q27. What measures of central tendency and variability could be applied to the log-transformed data? Selecting the preferable measure of central tendency and variability in a dataset is not necessarily straightforward.*

Table 1.3 gives some indication of what issues you might consider.

---

## 1.4 Conclusions

Visualizing and summarising data are the critical first steps in the data analysis and reporting workflow. We use graphical methods to firstly explore our own data. Once we have made sense of it we select the most appropriate method to convey that understanding to our readers. We may help that communication by summarising data in the most appropriate way taking into account the distribution of the data and the presence of outliers.

# Chapter 2

# Binomial & Poisson distributions

The role of statisticians is often to determine the probability of events, or a series of events, occurring in nature given our current understanding of the processes that are involved (normally on the basis of previous data). This requires a mathematical description of a theoretical relationship. We refer to this as a statistical model. Statistical models are like any model (e.g. a model aeroplane) in that they can be a useful simplification of reality. However, as George Box said, 'all models are wrong but some are useful'. We will come back to this concept again and again throughout this course.

Two models, the binomial and Poisson distributions, often provide excellent approximations of real-world events. This means that they can be used to determine the likelihood of events or series of events given certain 'reasonable' assumptions. In terms of planning, e.g. in the insurance industry, this is extremely useful.

Examples of the questions addressed using the binomial and Poisson models include:

- How likely is it that four hurricanes hit the US in a single season based on historic data?
- How likely is it that a '50-year' wave will hit in the next ten years? A 50 year wave is one of such size that it only occurs, on average, once in 50 years.
- Is a river flooding now more than it used to? Is there a change in flooding frequency?
- What proportion of clutches of fish eggs will contain all males?

This practical gives you the opportunity to practice using these distributions by hand and to use R to check that you get the answers correct. In the first part of this class you will conduct a number of short exercises to familiarise yourselves with the necessary commands in R to study probability using the binomial and Poisson distributions.

---

## 2.1 The Binomial distribution

### 2.1.1 Bernoulli trials

A Bernoulli trial is a single event with a *binary* outcome (i.e., a success or a failure). Binary outcomes include:

- Alive / dead

- Male / female
- Pregnant / not pregnant
- Guilty/ not guilty
- Heads/tails
- Win/lose

Things that can have more than two outcomes can be re-coded into two categories. This concept can be extended to numerous situations, such as:

- Income greater than £100,000 vs. less than £100,000
- Flower colour (blue vs. not blue; red vs. not red)

Each Bernoulli trial, by definition, is independent of all previous trials.

*Q28. If you tossed a fair coin 99 times and each time it landed heads (and you weren't cheating), what is the probability of obtaining a head on the next toss of the coin?*

This is a different question from asking whether 100 heads or (99 heads + 1 tails) is more likely in a throw of 100 coins. This is different because these events are independent.

### 2.1.2 The binomial distribution

The binomial distribution is a discrete probability distribution that applies to a series of Bernoulli trials. For example, if you had 10 eggs under a (big) chicken and they were all female, you might wonder how likely this was by chance. If this kept happening, you might question whether something was somehow making the eggs female. Here, the outcome can be female or male, and the trial size is 10 (each egg is a 'trial' with a binary outcome). You may question the assumption that the ratio female to male was 50:50 and favour an alternative hypothesis that the ratio was more female: less male. The binomial distribution allows you to quantify the probability of getting 10 females from 10 eggs (or any other number of females) for any given probability $P(female)$. That is, we are not restricted to 50:50.

You need to know two things to use the binomial distribution. These are:

- the number of trials ($n$, or sometimes $k$)
- probability of success ($p$)

From $p$, we can calculate the probability of failure as $q = (1 - p)$, since the two probabilities must sum to 1.

The distribution of a binomially distributed variable $y$ is specified $y \sim Binom(n, p)$. We denote $P(x)$ as the probability of getting $x$ successes where $x$ is an integer $0 : n$.

The **mean of a binomial distribution** is $n * p$. This gives you the *expected* outcome. For example, if $P(female) = 0.5$ and $n = 10$ eggs the expected number of females is $10 * 0.5 = 5$.

### 2.1.3 Binomial distributions by hand

Wongles always lay two eggs in a clutch but 50% of the eggs are infertile and don't hatch. We are interested in the proportions that we can expect to hatch from one clutch (two eggs)

*Q29. What is the event here?*

*Q30. What is the probability of success?*

*Q31. What is the number of trials?*

*Q32. How many possible outcomes are there and what are they?*

*Q33. Write down the model specification in the standard notation (with parameters)*

*Q34. Calculate the expected proportions of clutches of eggs that contain two fertile, two infertile and one of each using the quadratic equation approach (i.e. expand $(p+q)^2$).*

Often we are faced more than two events and the quadratic expansion approach you used above is no longer convenient. In such circumstances you use the binomial expression.

*Q35. Copy down the binomial probability mass function (from Chapter 7) and use it as an alternative method for calculating the fertility proportions for two eggs (as per the quadratic equation approach).*

Oozle birds are much more sensible than Wongles and always have broods of eight offspring and all of them hatch. We are interested in modelling the probabilities of the proportion of male and female offspring in broods of eight eggs.

*Q36. What is the Bernoulli event here (you are looking for something which has two mutually exclusive outcomes)?*

*Q37. What are the theoretical limits to your outcomes (i.e, max numbers of each).*

*Q38. What are the possible outcomes, and what is the total number of possible outcomes?*

*Q39. Write down, in the standard manner, the model for Oozle egg gender (starting with $y \sim Binom(n,p)$).*

We will assume that the probability ($p$) of any offspring being female is 0.5 and being male ($q$) is 0.5. For the extreme cases where all offspring are one sex, we can use simple probability theory: the probability of getting $n$ females in a brood of size $n$ is equal to $p^n$.

*Q40. Calculate the probability of obtaining eight male offspring.*

*Q41. What is the mean number of females you would expect in Oozle broods?*

It gets more complicated when you want to know the probability of getting, say, 1 male and 7 females from your clutch of eight eggs.

*Q42. Given that $p = q$, what shape would expect the distribution to be?*

*Q43. Use the Binomial expression (by hand) to calculate the probability of obtaining 0, 1, 2, 3, 4, 5, 6, 7 and 8 male offspring (note that the distribution is symmetrical).*

It is much easier, of course, to do this using R.

### 2.1.4 Binomial distributions in R

You can get probabilities for specific outcomes from a massive array of theoretical probability distributions from R. The binomial is just one of them and is implemented here.

```
num_female <- 4 # note that 4 is assigned to the variable called num_female
num_trials <- 8
p_female <- 0.5
# for a single probability: y~Binom(n=8, p=0.5) determine P(y_i=4)
paste0("P(", num_female, " female | ", num_trials, " eggs) = ",
       dbinom(num_female, num_trials, p_female))
```

```
## [1] "P(4 female | 8 eggs) = 0.2734375"
```

```
dbinom(4, 8, 0.5)
```

Often we want to know cumulative probabilities. This allows us to answer the question, for example, of what is the probability of obtaining $< 4$ females in a brood of 8 eggs. Here, $<4$ equates to the cumulative probability $\mathrm{P}(0) + \mathrm{P}(1) + \mathrm{P}(2) + \mathrm{P}(3)$.

```
# pbinom gives the cumulative probability
paste("The cumulative probability is",
      max(pbinom(0:num_female - 1, num_trials, p_female)))
```

```
## [1] "The cumulative probability is 0.36328125"
```

*Q44. Why do we parameterise* `pbinom()` *with* `num_female-1` *rather than* `num_female`*?*

*Q45. Would this change if the question was $P(\leq 4)$?*

*Q46. Take the 'max'out of the above line and run again. You should see 5 cumulative probabilities. Why is the first cumulative probability zero?*

*Q47. Calculate $P(< 4$ females $\mid 8$ eggs$)$ using* `dbinom()` *instead of* `pbinom()`*.*

```
# You can check what R is doing by running parts of code:
0:num_female - 1 # is this what you expected?
```

```
## [1] -1  0  1  2  3
```

```
0:(num_female - 1) # this is actually what we want.
```

```
## [1] 0 1 2 3
```

```
# Correct the code above. Why did this bug have no effect?
```

*Q48. What is the probability of getting 3 females?*

*Q49. What is the probability of getting 8 females?*

*Q50. What cumulative probabilities would you need to consider to answer the question 'What is the probability of getting fewer than three females?'*

*Q51. Write down the model that describes this random process (number of females per eight eggs). Your answer should be like this: $y \sim Binom(n, p)$.*

*Q52. What is the probability of getting $< 4$ females?*

*Q53. What is the probability of getting $\leq 4$ females?*

*Q54. What is the probability of getting $>$ than 8 females?*

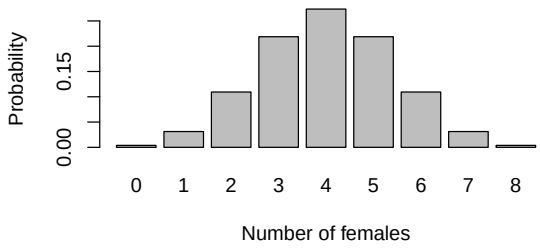
*Q55. What is the probability of getting $\geq$ than 2 females?*

Let's visualise these distributions in order to better understand them.

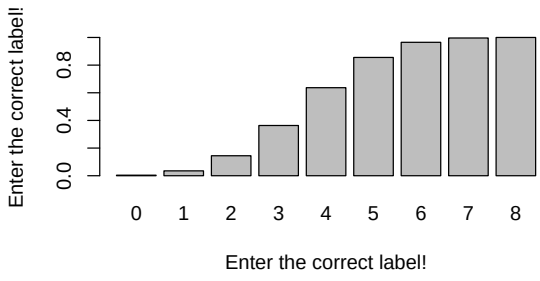
```
# Run this, then explore values of p_female
# Note: 'success' and 'failure' is arbitrary. Just make sure you're calculating
# what you think. How would you calculate the probabilities for males instead?
num_female <- seq(0, 8)
p_female <- 0.5 # what are the limits of p_female?
prFemale_df <- data.frame(num_female = num_female,
                          prob = dbinom(num_female, max(num_female), p_female))
prFemale_df
```

```
##   num_female       prob
## 1          0 0.00390625
## 2          1 0.03125000
## 3          2 0.10937500
## 4          3 0.21875000
## 5          4 0.27343750
## 6          5 0.21875000
## 7          6 0.10937500
## 8          7 0.03125000
## 9          8 0.00390625
```

```
barplot(prFemale_df$prob, names = prFemale_df$num_female,
        xlab = "Number of females", ylab = "Probability")
```




```
prFemale_df$cumul_prob <- cumsum(prFemale_df$prob)
barplot(prFemale_df$cumul_prob, names = prFemale_df$num_female,
        xlab = "Enter the correct label!", ylab = "Enter the correct label!")
```



Try re-plotting so that the two panels appear side by side (hint: `par(mfrow=c(..))`).

Note $P(y_i = x)$ is read as 'the probability that a random observation $y_i$ equals $x$'. Sometimes this includes conditions: $P(y_i = x|n, p)$, which is read as the probability that $y_i$ equals $x$ given $n$ and $p$. You may instead see $P()$, $Pr()$, $p()$, or $Prob()$. The interpretation of $P(y_i = 8 \mid n = 8,\ p = 0.5)$ is that the probability of 8 female offspring is 0.00391. So, if we have 500 broods, each with 8 eggs, we expect $500 * 0.00391 = 1.95 \approx 2$ broods to be all female.

*Q56. Why is a bargraph the appropriate plot here?*

*Q57. What do you notice about the shape of the distribution when $p = q = 0.5$?*

*Q58. Re-run the analysis with the probability of female as 0.8. Plot the results.*

*Q59. What is the shape of the distribution now?*

---

## 2.2 The Poisson distribution

The Poisson distribution is another probability distribution that describes discrete events that occur in space and/or time. The Poisson distribution is used to model (predict) the distribution of events that are rare, random, and independent. This can include events like earthquakes, storms, or the number of bends arriving at the SAMS recompression facility.

The Poisson distribution takes a single parameter: the mean. If a variable is Poisson distributed, its variance will equal its mean. This is a diagnostic feature of the distribution. The Poisson distribution is a discrete probability distribution, but its parameter, the mean, is continuous (similar to continuous $p$ for the discrete binomial distribution).

The Poisson formula is simple - copy it from Chapter 7 into here:

$\bar{y}$ is the mean, $x$ is the outcome of interest, $e$ is Euler's number, and ! is factorial.

### 2.2.1 Poisson distributions by hand

These problems usually start by asking you to calculate the mean number of observations made per unit of space or time. This is the Poisson parameter, and is referred to as the *rate* or as *lambda* ($\lambda$). The unit could be spatial (e.g., per $m^2$) or temporal (e.g., per hour or day).

Consider randomly throwing $1m^2$ quadrat- on a sandy beach covered in worm casts, as illustrated in Fig. 2.1. You then count the number of casts in each quadrat.

From these data you can calculate the mean number of observations per unit.

The code below generates a dataframe from which you can determine that the mean number of worms per quadrat is 1.41. We wish to predict the proportion of quadrats that would contain 0, 1, 2, 3, 4, and 5 worms, assuming that the worms are independently distributed across space (i.e., random: one worm's location has no effect on another worm's location, and there is no relevant environmental variation).

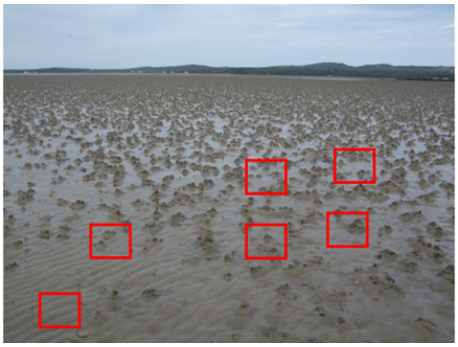


Figure 2.1: Worm casts on a sandy beach with 1x1 m quadrats.

We thus wish to determine the probability of observing our data assuming that the worms casts are randomly distributed. To do this, we need to know the probability of observing each count (zero to infinity), given the mean count per quadrat.

```
# num_worms is the number of worms per quadrat
# num_quadrats is numbers of quadrats observed containing that number of worms
# this dataset is summarised. Raw data might have columns: quadrat_id, num_worms
worm_df <- data.frame(num_worms = c(0, 1, 2, 3, 4, 5, 6),
                      num_quadrats = c(35, 28, 15, 10, 7, 5, 0))
worm_df # display dataframe
```

```
##   num_worms num_quadrats
## 1         0           35
## 2         1           28
## 3         2           15
## 4         3           10
## 5         4            7
## 6         5            5
## 7         6            0
```

As the number of worms per quadrat is low compared to the number of worms that could exist within a quadrat we consider these are rare events and hence can be (reasonably) described by a Poisson distribution (assuming they are independent).

From just the mean we are able to calculate the expected frequency of observing different number of worms in any quadrat (assuming the model assumptions are met - remind yourself what these assumptions are). The number of worms per quadrat ($y$) is discrete; it can only take integers greater or equal to zero.

Probability questions often includes terms such as $P(y_i \leq a)$. That is, the probability that an observation $i$ of the random variable $y$ is less than or equal to $a$. For example, in this context you might get asked for $P(y_i \leq 1)$ which asks what is the probability of a random quadrat being thrown ($y_i$) containing less or equal to 1 worm cast ($a$, an integer value). To be fully complete, we might even write $P(y_i \leq 1|\lambda)$, which acknowledges that we know the (sample) mean.

So, if you are interested in predicting the probability of obtaining 1 or fewer worms per quadrat, you would start your calculation by writing $P(y_i = 0) + P(y_i = 1) = ....$

*Q60. Use the Poisson formula to calculate the expected frequency of 0, 1 & 2 worms per quadrat.*

*Q61. What calculation would you need to conduct to determine $P(y_i \geq 1)$?*

Again, calculating Poisson distributed variables in R is easier than doing it by hand, but you must be sure of what you are actually asking (hence bothering with the hand calculations).

### 2.2.2 Poisson distributions in R

You can determine the expected probabilities for each worm count per quadrat once you have determined the mean count of worms per quadrat. Since we have a summarised dataset (i.e., the number of observations `num_quadrats` for each number of worms `num_worms`, rather than the raw data with a row for each quadrat), we need to do some basic calculations. The mean number of worms per quadrat = (total number of worms) / (total number of quadrats).

```
sum(worm_df$num_quadrats) # total number of quadrats
```

```
## [1] 100
```

```
lambda_worms <- with(worm_df, sum(num_worms * num_quadrats) / sum(num_quadrats))
paste("The mean is", lambda_worms, "worms per quadrat")
```

```
## [1] "The mean is 1.41 worms per quadrat"
```

Interpreting typical questions: taking $P(y_i \leq a)$, when $a = 5$ and you are asked to determine $P(y_i \leq 5)$, what you are being asked is "What is the probability of a random quadrat $(y_i)$ containing five or fewer worms?". In order to determine this, you need to know the mean number per quadrat. When the mean number is 1.41 worms per quadrat, $P(y_i \leq 5) = 0.997$ (3 sf). This means that, if your data are Poisson distributed, it is highly likely that there will be fewer than five worms in your quadrat if $\lambda = 1.41$.

*Q62. If the mean number of worms was 3 per quadrat, would you be more or less likely to get five worms in your quadrat?*

Needless to say, you can use R to calculate Poisson probabilities. Starting with a simple example:

```
# ?dpois
# for questions like 'determine P(y_i=a | lambda)'
a <- 5
lambda <- 1.41
dpois(a, lambda) # probability of observing 'a' worms per quadrat: dpois()
```

```
## [1] 0.01133859
```

```
ppois(a, lambda) # probability of observing >= 'a' worms: ppois()
```

```
## [1] 0.9966869
```

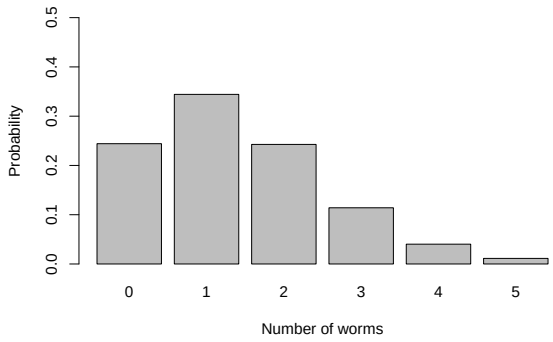
```
## ppois(0:a, lambda) # what does 0:a mean? What's another way to do this?
signif(ppois(0:a, lambda), 3) # round with ?signif
```

```
## [1] 0.244 0.588 0.831 0.945 0.985 0.997
```

Table 2.1: Worm cast observations and expected probabilities.

| num_worms | num_quadrats | prob | cumul_prob |
|---|---|---|---|
| 0 | 35 | 0.24414 | 0.24414 |
| 1 | 28 | 0.34424 | 0.58839 |
| 2 | 15 | 0.24269 | 0.83108 |
| 3 | 10 | 0.11406 | 0.94514 |
| 4 | 7 | 0.04021 | 0.98535 |
| 5 | 5 | 0.01134 | 0.99669 |
| 6 | 0 | 0.00266 | 0.99935 |

```
barplot(dpois(0:a, lambda),
        ylab = "Probability", xlab = "Number of worms",
        space = 0.2, ylim = c(0, 0.5), names.arg = 0:a)
```




Now change the plot so that the cumulative probabilities are plotted. You will need to change the code to use `ppois()` instead of `dpois()`, as well as the y-axis limits (`ylim`).

```
# create new columns in worm_df for the probabilities
worm_df$prob <- dpois(worm_df$num_worms, lambda)
worm_df$cumul_prob <- cumsum(worm_df$prob)
```

```
# Make Table 2.1. Use packageName::function() instead of loading with library()
knitr::kable(worm_df, digits=5,
             caption = 'Worm cast observations and expected probabilities.')
```

*Q64. Format the probabilities in the table to 3 significant figures.*

If individual probability values (not cumulative probability) are multiplied by the total number of quadrats thrown (100), we generate the expected frequency distribution for comparison with the observed results above.

*Q65. Write the appropriate code to add a column(called `num_quadrats_expected`) to 'worm_df' that is the expected number of quadrats (given that 100 quadrats was the total thrown).*

This expected number is the number of quadrats that you would expect to contain 0,1,…,5 worms, given that the mean density of worms is 1.41. The number of worms per quadrat is a discrete variable, yet you can have non-integer 'expectations' (i.e. means).

*Q66. Add another column that is the difference in the Expected number of quadrats and* `num_quadrats_expected`.

*Q67. Produce a bar graph of this difference.*

---

## 2.3 The Poisson approximation of the binomial model

Where the number of trials is large and the probability of success is small, the Poisson distribution can be used as an approximation of the binomial distribution. Under these circumstances you can calculate the mean of a variable that has a binomial distribution (number of trials x probability of success; $n * p$) and use that as an estimate of $\lambda$ in the Poisson model. The reason why you may wish to do this is because the Poisson model is much easier to use than the binomial model when $n$ is large.

There are several rules of thumb that apply here. Some say that when $p < 0.1$, the Poisson approximation may be preferable to the binomial, other texts state that $n$ should be $> 50$ and $p < 0.05$ (so $n * p = \lambda = 2.5$) whilst others state $n > 100$ and $n * p < 10$. The point is that as $n$ increases and $p$ decreases, the approximation gets better.

There are an infinite number of ways of multiplying two numbers together to get 5. Call our numbers $p$ and $k$. For example, when $p = 0.5$, $n = 10$ OR when $p = 0.05$, $n = 100$, $n * p = 5$. We can use this to illustrate the Poisson approximation of the binomial distribution.

```
# generate dataframe with probability for 0:16 'successes' from different
# distributions but where mean is 5.
# note that in y ~ Binom(10,0.5), probability of >10 successes is zero.
# We're going to leverage the power of the tidyverse here
library(tidyverse)
y_seq <- 0:16
binom_df <- tibble(y=rep(y_seq, times=3), # ?rep
                   n=rep(c(10, 20, 100), each=length(y_seq)),
                   p=rep(c(0.5, 0.25, 0.05), each=length(y_seq))) |>
    mutate(mean=n*p,
           prob=dbinom(y_seq, n, p),
           label=paste0("y ~ Binom(", n, ", ", p, ")"))
pois_df <- tibble(y=y_seq,
                  n=NA,
                  p=NA,
                  mean=5) |>
    mutate(prob=dpois(y_seq, mean),
           label=paste0("y ~ Pois(", mean, ")"))
PDF_df <- bind_rows(binom_df, pois_df) |>
  mutate(label=factor(label, levels=unique(label)))
```

```
head(binom_df, 2)
```

```
## # A tibble: 2 x 6
##       y     n     p  mean     prob label
##   <int> <dbl> <dbl> <dbl>    <dbl> <chr>
```

```
## 1     0    10   0.5     5 0.000977 y ~ Binom(10, 0.5)
## 2     1    10   0.5     5 0.00977  y ~ Binom(10, 0.5)
```

```
head(pois_df, 2)
```

```
## # A tibble: 2 x 6
##       y n     p      mean    prob label
##   <int> <lgl> <lgl> <dbl>   <dbl> <chr>
## 1     0 NA    NA        5 0.00674 y ~ Pois(5)
## 2     1 NA    NA        5 0.0337  y ~ Pois(5)
```

```
ggplot(PDF_df, aes(y, prob, fill=label)) +  # ggplot(data, aes(xVar, yVar))
  geom_bar(stat="identity", position="dodge", colour="grey30") +  # ?geom_bar
  scale_fill_brewer("Distribution", palette="Purples") + # from colorbrewer2.org
  labs(x="Number of successes with mean = 5", y="Probability") +
  theme_classic() + theme(legend.position=c(0.85, 0.85))
```

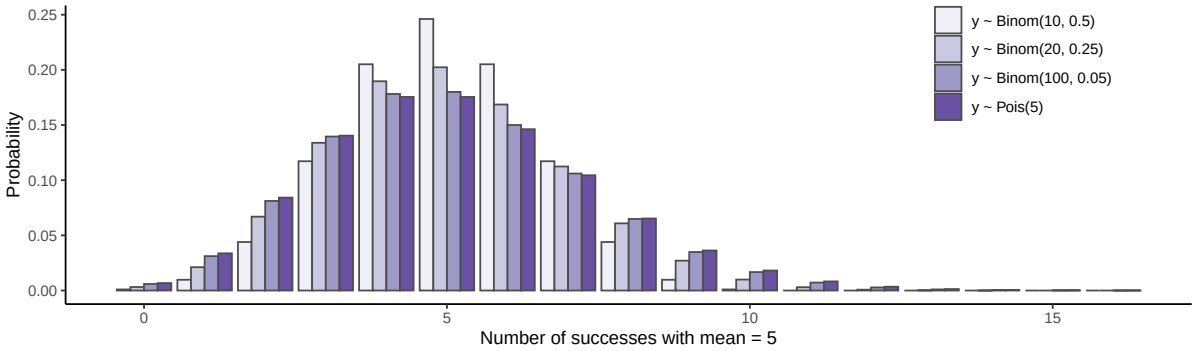



Figure 2.2: Binomial and Poisson distributions converge with larger numbers of events.

*Q68. What is the modal value in these distributions?*

Generate some random numbers from the distributions in Fig 2.2 and calculate their mean and variance. Here is $y \sim Pois(5)$:

```
y <- rpois(10000, 5)
paste0("Mean: ", signif(mean(y), 3), ", variance: ", signif(var(y), 3))
```

```
## [1] "Mean: 4.97, variance: 5.02"
```

*Q69. What do you notice about the mean and variance in the Poisson model?*

*Q70. What do you notice about the mean and variance in the binomial models as $n$ increases and $p$ decreases? When is it more similar to the Poisson? Is this what you expected?*

## 2.4 Conclusions

The binomial distribution is a discrete probability distribution that models situations where the outcome of an observation or experiment is binary (i.e., two possibilities) or can be coded as such. The binomial model enables us to predict the probability of making our observation or series of independent observations for any given probability of a success ($p$). This enables us to quantify how likely our observation is to have occurred, by chance. If the chance of our observation is very low, we can challenge the hypothesis with regard the probability of success ($p$) and suggest a different value.

The Poisson distribution is another discrete probability distribution that is used to predict the chance occurrence of observations that are rare, independent and randomly distributed with mean = variance = $\lambda$. The Poisson distribution can be used as an approximation of the binomial distribution where the number of trials ($n$) is large and the probability of success ($p$) is small. This approximation is useful as, unlike the Poisson distribution, the binomial calculation requires the handling of massive numbers (from large factorials).

# Chapter 3

# Normal distribution

Statistical inference is the process by which we infer from a sample statistic to the population. We need to infer from samples to populations because it is usually impossible to measure the entire populations. We call this this estimating population parameters from samples.

In order to infer from samples to populations we need to understand/predict how the statistics we generate from our samples 'behave'. To do this we use theoretical distributions.

*Q71. Which two theoretical distributions have we already covered in this course?*

The normal (a.k.a., Gaussian) distribution is another theoretical distribution and is central to inferential statistics. If your data (or, more accuarately, statistics derived from you data) are reasonably approximated by the normal distribution then you will be able to use a wide range of techniques to deal with it.

In this practical we will be examining the normal distribution, calculating Z scores both manually and in R, and interpreting those Z scores. We'll also use our knowledge of the normal distribution's behaviour to produce confidence intervals and assess whether data are reasonably assummed to be normally distributed, transforming the data where they are not. We'll apply the CLT and evaluate how well other distributions are approximated by the normal distribution.

---

## 3.1 Using the normal probability distributions

The length, in cm, of a catch of herring was measured. Five hundred individuals were studied. We will consider this group to be the entire population of interest. The population parameters are: mean length 37.6 cm, standard deviation 1.20 cm.

*Q72. What are the theoretical limits of the normal distribution?*

*Q73. Do you think normality a fair assumption for these data?*

*Q74. Assume length of herring is approximately normally distributed and write down the model which describes the fish length distribution in the standard notation.*

When we have population parameters such as we have here, we can calculate Z scores for individuals (or groups of individuals) from that population and calculate how unusual they are. For the moment, we are interested in determining what proportion of fish from this population expected to be <38cm.

*Q75. Sketch a normal distribution, showing mean of 37.6cm, note the sd on your sketch and sketch on the location of a fish at 38 cm.*

*Q76. Mark-out the bit of the sketch which the question asks you about*

*Q77. Mark out the bit of the sketch that our manually determined Z scores relate to.*

*Q78. Calculate the Z score for a fish of 38cm. How unusual is such an observation? What proportion of fish will be <38 cm, what proportion of fish >38 cm?.*

### 3.1.1 Using R to calculate areas under the normal curve

To solve this problem using R we use cumulative probability. Observations at the extreme low end are extremely unlikely, but the probability of observing data increases and reaches a maximum at the mean, after which it declines again. The normal distribution is symmetrical.

*Q79. What shape is the cumulative probability curve of a normally distributed variable?*

You can ask R to determine the probabilities of an eventuality occurring within a normal distribution. For example, you could ask what the probability is of observing a 38 cm or more fish, when the mean is 37.6 cm and the standard deviation is 1.20 cm. You need to specify the model, and the problem. Use `?dnorm` and look at your options.

```
# cumulative probability: which bit of the curve does this relate to?
pnorm(q = 38, mean = 37.6, sd = 1.2)
```

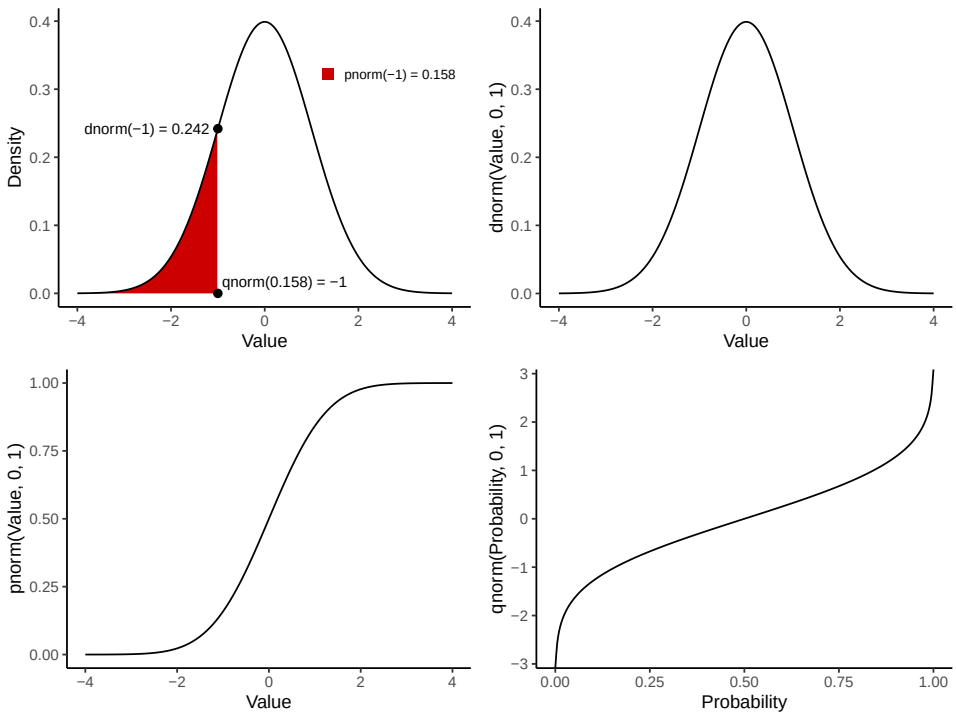



Figure 3.1: Functions for distributions in R illustrated with a standard normal.

We can plot any normal distribution we like. Play with the values for `mu` and `sigma` below, adjusting the values for `from` and `to` as needed to see the distribution.

```
mu <- 37.6  # population mean
sigma <- 1.2  # population sd
curve(dnorm(x, mean = mu, sd = sigma), from = 30, to = 45,
      main = "Normal density", ylab = "Density", xlab = "Fish length, cm")
```

I can define a population like this $y \sim Norm(37.6, 1.2)$ and ask questions about $P(y_i \leq a)$. E.g., what is the probability that a random fish drawn from this population is <38 cm ($P(y_i < 38)$). The answer to that is 0.6306 or approximately 63% (2 sf).

*Q80. How does your hand calculation compare? Is there a discrepancy? If so, why?*

---

## 3.2 Normal model adequacy

A separate fish population (long-eared wrasse) is described as $y \sim Norm(5.14cm, 25cm)$.

*Q81. What is the mean length and standard deviation of long-eared wrasse?*

*Q82. Calculate the proportion of fish that are less than zero cm in length.*

*Q84. What do your results indicate about the adequacy of the normal model to describe the length distribution of long-eared wrasse?*

*Q85. Recalculate your answers with 1/10$^{th}$ the standard deviation: $y \sim Norm(5.14, 2.5)$.*

---

## 3.3 Calculating quantiles in the normal distribution

A quantile is one of a series of values that divides a frequency distribution (i.e. a set of numbers) into equally represented groups (i.e., the same number of observations per group). For example, there are three values (Q1, Q2, Q3) that split a normal distribution into four groups (negative infinity to Q1, Q1 to Q2 (median), Q2 to Q3, and Q3 to positive infinity). Q3 - Q2 is the middle 50% of the data: the interquartile range.

There are 99 quantiles, called percentiles, that split your data into 100 groups. The 2.5 percentile is the value that splits your data into two groups corresponding to 2.5% along the distribution (from negative infinity for the normal distribution). Just as we can ask what proportion of a distribution is above or below a set value, we can also ask between what values will a given percentage of my data lie (e.g. what values correspond to the middle 95%?).

To solve this problem using R we use the cumulative probability function `qnorm()`.

```
qnorm(p = 0.975, mean = 37.6, sd = 1.2) # p is the cumulative probability
```

```
## [1] 39.95196
```

```
probs <- seq(from = 0.1, to = 0.9, by = 0.2) # vectorize for >1 probability
qnorm(p = probs, mean = 37.6, sd = 1.2)
```

```
## [1] 36.06214 36.97072 37.60000 38.22928 39.13786
```

```
probs_95 <- c(0.025, 0.975) # middle 95% (2.5% on either side)
qnorm(p = probs_95, mean = 37.6, sd = 1.2) # round as appropriate
```

```
## [1] 35.24804 39.95196
```

These values (`probs_95 = c(0.025, 0.975)`) identify proportions of the cumulative curve. That is, they identify the bottom 2.5% and the bottom 97.5% of the curve. Values between these two parameters will constitute the central 95% of your data.

For the next question, return to the herring data where the population was $y \sim Norm(37.6, 1.44)$.

*Q86. Find the values that capture the middle 95 and 90% of the herring data.*

We quote our mean and interval like this: "The mean fish length and 95% interval was 37.6 cm (35.2 cm, 40.0 cm)". Remember to use to same degree of precision (significant figures) for confidence intervals as was used to gather the data (or as specified in the question, defaulting to three).

*Q87. What value would you expect to correspond to 0.5 (i.e., what value would be found half-way along your distribution)?*

*Q88. Check your answer above using `qnorm()`*

You can easily visualise this distribution in R:

```
# your population is defined thus: y ~ Norm(mean=100, variance=225)
mu <- 100
sigma <- 15
# lower boundary and upper boundary of the region of interest
lb <- 80
ub <- 140

x <- seq(-4, 4, length.out = 100) * sigma + mu
hx <- dnorm(x, mu, sigma)
plot(x, hx,
  type = "n", xlab = "IQ Values", ylab = "",
  main = "Normal Distribution", axes = FALSE
)
i <- x >= lb & x <= ub
lines(x, hx)
polygon(c(lb, x[i], ub), c(0, hx[i], 0), col = "red") # create 100 polygons
area <- pnorm(ub, mu, sigma) - pnorm(lb, mu, sigma)
result <- paste(
  "P(", lb, "< IQ <", ub, ") =",
  signif(area, digits = 3)
)
mtext(result, 3)
axis(1, at = seq(floor(mu - 4*sigma), floor(mu + 4*sigma), sigma), pos = 0)
```

*Q89. Change the parameters in the above to your herring data where the population was distributed $y \sim Norm(37.6, 1.44)$ and check that the values you generated for the 95% interval correspond when plugged into `lb` and `ub` in the code.*

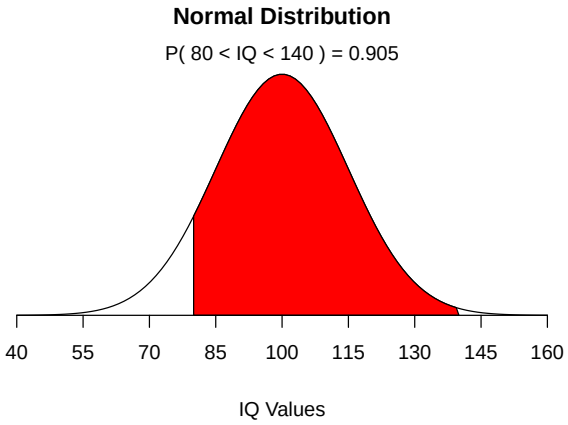



Figure 3.2: The normal distribution illustrating probability of a random individual from your population having an IQ between 80 and 140.

*Q90. What is the difference between $< x$ and $\leq x$ when applied to continuous data?*

*Q91. Does this also apply to discontinuous data?*

Think about how intervals change when the population standard deviation changes.

*Q92. With everything else equal, try doubling, quadrupling, and halving the standard deviation on the herring data then re-running the same code.*

*Q93. What effect does this have on your interval?*

---

## 3.4 Testing for normality

As we note in the lectures, many statistical tests require that data are reasonably approximated by a normal distribution and have homogeneous variance. R can be used to formally test the assumption that data are normally distributed, though you should have some idea of whether this is likely through consideration of the data source. This applies particularly where you have a small sample size and, consequently, evaluating the distribution is challenging.

The data you collect will be part of a population. The normality check assesses the viability of the assumption that the data you collected were drawn from a population that was normally distributed. You should note that populations are, in practice, *never* actually normally distributed. Your test is to assess how *reasonable* the assumption of normality is.

```
library(readxl)
MS <- read_xlsx("data/practical_3_4.xlsx", sheet = "Mood shrimp")
# check the data using head(), str() etc.
```

```
par(mfrow = c(1, 2))
qqnorm(MS$Shrimp1)
qqline(MS$Shrimp1, col = 2) # the data fall near the line
qqnorm(MS$Shrimp2)
qqline(MS$Shrimp1, col = 2) # the data deviate widely from the line
```

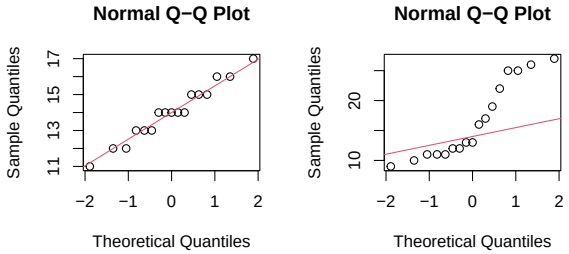



Figure 3.3: Normal (QQ) plots. The left indicates that the normality assumption might be reasonable, not so on the right.

The axes on the probability plots have been rescaled so that, if your data are perfectly normal, the data points would fall on a straight line (the red line on the graphs generated by `qqline()`). Any deviation from the red straight line in your data indicates a lack of normality. What we need to assess is how serious any deviation is and whether it is sufficient to indicate that the assumption of normality is not 'reasonable'. This is a subjective decision, and two different statisticians may tell you different answers about the same data.

As your sample size decreases, it will be increasingly difficult to see if your data are reasonably approximated by a normal distribution. There are many formal statistical methods for assessing normality, but as we already know it is impossible for your data to have been drawn from a population that is normal and this means such tests are largely redundant. You must assess the assumptions of your model (e.g. the general linear model, which includes ANOVA and linear regression), but you should be aware that all data fails the assumptions. However, this doesn't mean the outputs from such models aren't useful - provided the assumptions are 'reasonably' well met. There is no hard rule as to what constitutes 'reasonable'!

*Q94. Make histograms of both* `Shrimp1` *and* `Shrimp2`*. Comment on their apparent distributions.*

---

## 3.5 Data transformations

The assumption that our sample data are drawn from a normally distributed population is central to the use of many important inferential statistical techniques. However, frequently it is *not* reasonable to assume that data are likely to be normally distributed and they 'fail' normality tests (e.g. clearly do not fall on the red `qqline()`).

Non-normality often occurs because our measurements are near a logical zero. For example, chemical concentrations (e.g., of zinc in sediments) is limited by zero; you cannot have negative zinc concentrations. Similarly, you cannot have negative lengths, time, mass, etc. Where data is collected 'near' a logical zero they are often not normally distributed, since the normal distribution predicts a negative-value 'tail' which is impossible.

There are several simple mathematical transformation options that you can use in an attempt to convert your data to something that is reasonably approximated by a normal distribution even if it is not well approximated in the original measurement units.

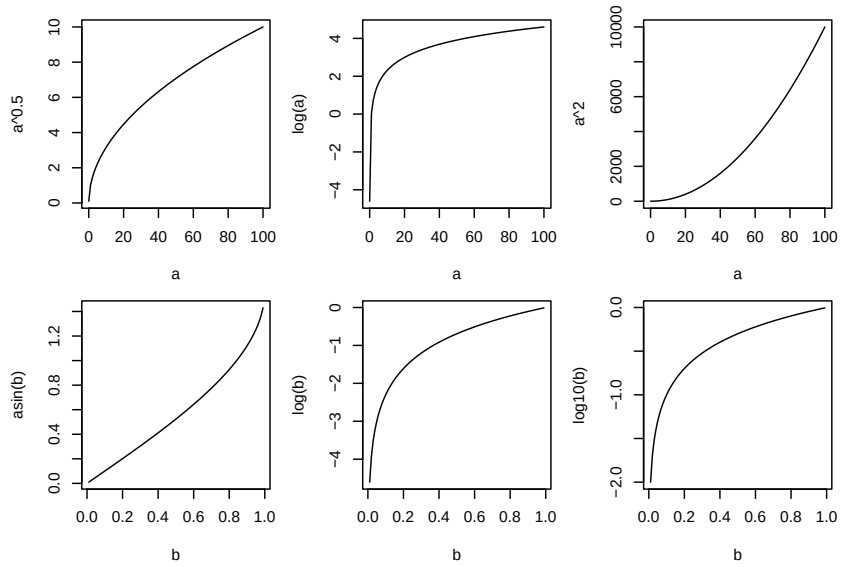
Here we will consider three common transformations. These are the log (base 10 or natural), the square-root, and the arcsine transformation. These transformations often have the additional benefit of correcting unequal variances (see Practical 5) in addition to non-normality so they are very useful. These three are often used in distinct circumstances.

- Log (`log()`): When the distribution is skewed right. Often 'cures' heteroscedasticity.
- Square-root (`sqrt()`): When the measurements are areas (e.g., leaf areas). Often used to 'down-weight' common species (e.g., Practical 6), which is unrelated to model assumptions.
- Arcsine (`asin()`): When the measurements are percentages or proportions. Tends to stretch out the tails (e.g., near 0 or 1 for proportions) and squash the middle (e.g., near 0.5).

Identifying the correct transformation can be led by an underlying comprehension of the nature of the data. However, this often doesn't work so expect some trial and error.

```
a <- seq(0.01, 100, length.out=100)
b <- seq(0.01, 0.99, length.out=100)

par(mfrow=c(2,3), mar=c(4,4,1,1))
plot(a, a^0.5, type="l")
plot(a, log(a), type="l")
plot(a, a^2, type="l")
plot(b, asin(b), type="l")
plot(b, log(b), type="l")
plot(b, log10(b), type="l")
```




*Q95. Using the Radon concentration worksheet, plot the data. Do they look normally distributed?*

*Q96. In your own time,use R to determine the log, square root, and reciprocals (and combinations of all of them: at least one converts the data to approximate normality).*

```
# get the Radon data into R; the units are parts per billion
readxl::excel_sheets("data/practical_3_4.xlsx")
```

```
##  [1] "Cod lengths"           "Shrimps"               "Caffeine"
##  [4] "Non-parametric shrimp" "Mood shrimp"           "Mood shrimp2"
```

```
##  [7] "Swimming activity"     "Clare's Al data"       "Distribution"
## [10] "Radon (ppb)"           "Zinc conc"             "Heart-beat,2012"
## [13] "Sheet1"
```

```
# load the dataset and try some transformations
```

*Q97. Generate a vector of numbers and then use* `sqrt()` *on them.*

---

## 3.6 Functions in R

Everything that *does* in R is a function. Writing your own functions has many benefits, including keeping your code legible and reducing work for yourself (since any changes need to be done in only one location instead of everywhere you've repeated the same code).

The `trim_values` function takes an argument (`x`), formats it to three digits, and returns the formatted output. Play around with the function to see how it works. We'll use this function to keep our outputted values 'clean'. In order to use the function, you must run the code defining it first. You will see your functions listed (alongside your dataframes, vectors, lists, etc) in R-Studio's 'Environment' pane.

```
trim_values <- function(x) {
  y <- format(x, digits = 3)
  return(y)
}

# After running the code above, we can use the function:
trim_values(1.23456)
```

```
## [1] "1.23"
```

```
long_numbers <- rnorm(3, 2, 1)
data.frame(orig=long_numbers,
           clean=trim_values(long_numbers))
```

```
##       orig clean
## 1 2.227048  2.23
## 2 1.242831  1.24
## 3 2.121569  2.12
```

Notice anything strange about the output? The function `format()` converts the vector from `numeric` to `character` to display each value with the same number of characters. The `digits=3` argument is also taken as a suggestion (kind of like `breaks` in `hist()`). Try some of the other options in `format()` or adjust the `digits` argument, re-running the function definition each time to see the changes.

---

## 3.7 The central limit theorem

The central limit theorem states that **the means of normally distributed data will, themselves, be normally distributed**. In addition, the theorem states that **the means of data which are NOT normally distributed will be normally distributed if the sample size is sufficiently large**. In this practical we are going to demonstrate this theorem using random data generated from various probability distributions.

### 3.7.1 The distribution of means from various data distributions

```
# Run this code, then adjust the values for obs_data as you like.
# Play with more distributions if you want: ?stats::distributions
obs_data <- c(rnorm(2000, 200, 20),
              rnorm(1500, 100, 20),
              rlnorm(100, 5, 0.5),
              rpois(1000, 15),
              rpois(500, 30))
xlims <- range(obs_data)

# Define size of each sample and the number of sampling repeats
sample_size <- 30
num_samples <- 10000

# Sample num_samples times from obs_data, with n = sample_size for each sample
sample_means <- numeric(length=num_samples) # initialize an empty vector
for(i in 1:num_samples) {
  sample_i <- sample(obs_data, sample_size, replace=T)
  sample_means[i] <- mean(sample_i)
}

# plot observed distribution
par(mfrow = c(2, 2))
par(mar = c(5, 2, 2, 2))
hist(obs_data,
     main = "Raw data distribution",
     sub = paste0("mean: ", trim_values(mean(obs_data)),
                  ", sd: ", trim_values(sd(obs_data))),
     breaks = 30, xlab = "Observed length (cm)", xlim = xlims
)
qqnorm(obs_data, main = "Raw data QQ")
qqline(obs_data)

# plot distribution of sample means
hist(sample_means,
     main = paste0("Sample distribution, N: ", sample_size),
     sub = paste0("mean: ", trim_values(mean(sample_means)),
                  ", sd: ", trim_values(sd(sample_means))),
     breaks = 30, xlab = "Mean length (cm)", xlim = xlims
)
```

```
qqnorm(sample_means, main="Sample mean QQ")
qqline(sample_means)
```

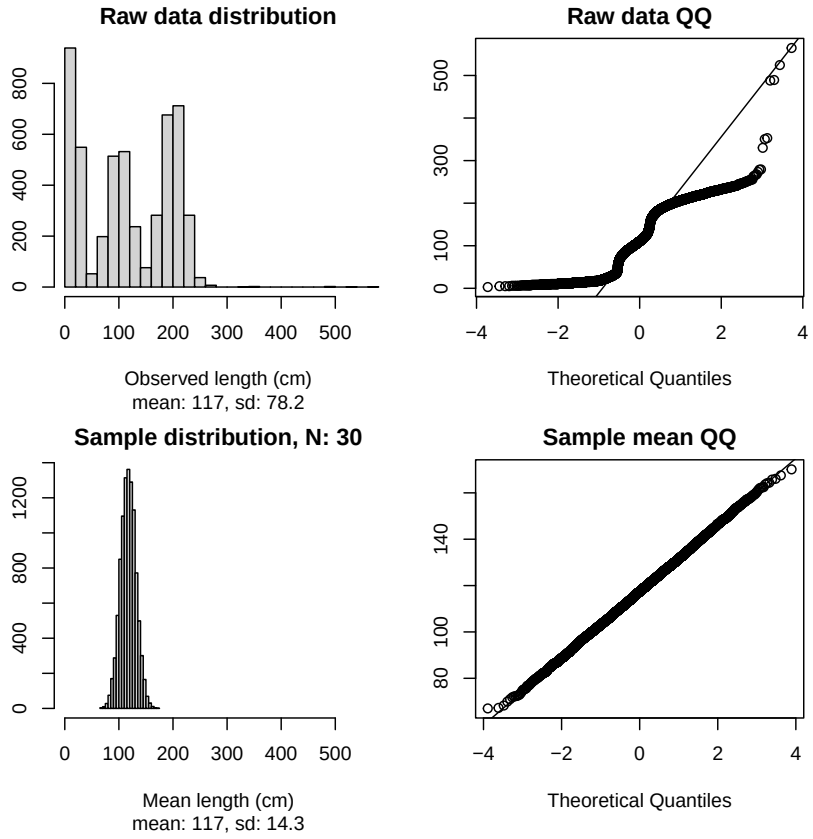



Figure 3.4: The central limit theorem in action (when n>30).

*Q98. What do you notice about the location of the mean as a function of N?*

*Q99. What do you notice about the spread around the mean in each group as a function of sample size – why did the pattern you have observed occur?*

You are witnessing the central limit theorem in action – the means of normally distributed numbers are, themselves, normally distributed, and the means of non-normally distributed numbers are *also* normally distributed provided the sample size is large enough. Note the relationship between the sample size and the range of values for the sample mean. How does the standard deviation of `sample_means` change with changes in `sample_size`?.

---

## 3.8 The standard error of the mean

The standard error of the mean is simply the standard deviation of sample means, of a given sample size, taken from a population. That is, it is the standard deviations calculated in the lower histogram in Fig. 3.4. It is calculated as $SE_{\bar{y}} = \frac{sd(y)}{\sqrt{n}}$.

```
library(tidyverse)
# Simulate a normally distributed population with mean mu and sd sigma
mu <- 10
sigma <- 2
sim_obs <- rnorm(10000, mu, sigma)

# Draw samples
sample_size <- c(2, 10, 30, 100) # size of each sample
num_samples <- 1000 # number of samples (=repeats) for each sample size

sample_mean_df <- tibble(N=rep(sample_size, each = num_samples),
                         id=rep(1:num_samples, times = length(sample_size))) |>
  mutate(sample_mean=0)

for(i in 1:nrow(sample_mean_df)) {
  sample_i <- sample(sim_obs, sample_mean_df$N[i])
  sample_mean_df$sample_mean[i] <- mean(sample_i)
}
```

```
sample_mean_df |>
  mutate(N=factor(N, levels=unique(N))) |>
  ggplot(aes(N, sample_mean)) +
  geom_hline(yintercept=mu, linetype=3) +
  geom_violin(scale="width", draw_quantiles=c(0.025, 0.5, 0.975), fill=NA) +
  geom_jitter(alpha=0.2, shape=1, width=0.05) +
  labs(x="Sample size", y=paste("Means of", num_samples, "simulated samples")) +
  theme_classic()
```

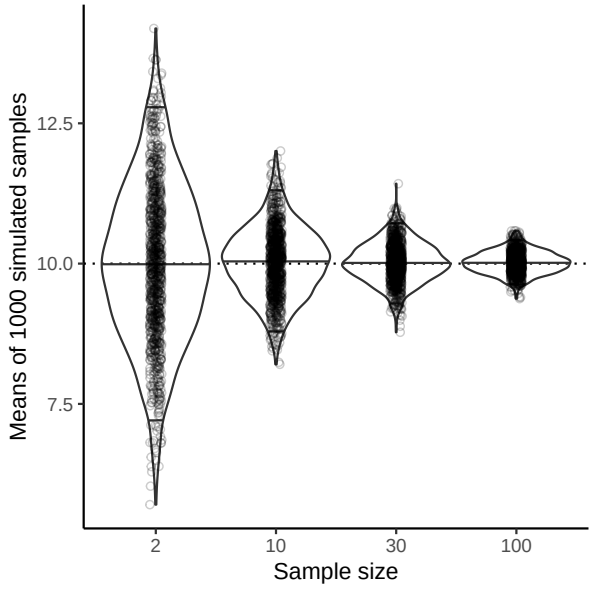



Figure 3.5: Simulated sample means.

*Q100. Calculate the theoretical standard error of the means for each* `sample_size` *given* `sigma` *and compare this with your actual standard error (=standard deviation) of sample means*

*Q101. Given what you know about the CLT, how does this change with non-normally distributed data? Try changing* `rnorm()` *to* `rpois()`

*Q102. Change* `sigma` *and* `sample_size` *and check that the standard error estimates are in line with their true values (i.e. as determined by using the standard error formula).*

---

## 3.9 Normal approximations for other distributions

In earlier exercises you saw that the Poisson distribution could be used to approximate the binomial distribution. In a conceptually similar way, the normal distribution can be used to approximate the Poisson model.

### 3.9.1 The Normal approximation of the Poisson distribution

Recall that the Poisson model takes only one parameter, the mean, and that where a variable is Poisson distributed the variance equals the mean. So if we have a variable $y \sim Pois(10)$, the mean (say density per quadrat) is 10 and so is the variance. We now have the two parameters that define the normal distribution (the mean and variance).

*Q103. For $y \sim Pois(10)$ write the equivalent normal distribution in standard notation.*

As a rough guideline, a normal approximation is reasonable if $\lambda \geq 30$. So $y \sim Pois(10)$ should not be approximated by the normal distribution while $y \sim Pois(30)$ could be. However, it is generally preferable to use the natural distribution for your data (e.g., a Poisson distribution for counts). The normal distribution is beneficial in some cases, but this is increasingly less so with modern methods.

*Q104. For a Poisson distribution $y \sim Pois(40)$ define the normal approximation in the standard way.*

*Q105. Using R and referring Practical 2, calculate $P(y_i \leq 35)$ from $y \sim Pois(40)$ (Ans = 0.242)*

*Q106. Using R, calculate $P(y_i < 35)$ from the above normal approximation of the Poisson distribution (Ans = 0.215).*

*Q107. What do you think of the normal approximation?*

*Q108. Evaluate $P(y_i < 3 \mid \lambda = 5)$ using the Poisson model and its normal approximation.*

### 3.9.2 The normal approximation of the binomial distribution

Whilst we accept the fundamental difference between continuous and discrete data, in making continuous-variable measurements we inevitably allocate each measurement to a single measurement category (we are limited by our ability to measure, eg. to the nearest millimeter if using a ruler). If we wish to treat a continuous variable as a continuous variable, the number of categories between the smallest and largest measurement should be a minimum of ~30 (the upper limit doesn't matter so much but there is no need to >300). For example, suppose we were measuring fish-lengths and the min was 20 cm and the max 40 cm. That means there are 20 x 1 cm measurement units between them. This falls under our 30 unit minimum – measuring to the nearest cm would not be adequate. There are, however, 40 x 0.5 cm units between them

so we could measure to the nearest 0.5 cm (we would probably use 0.1 cm though). But, even if we measure to the correct degree of precision, we are still categorising the lengths. For example, a fish of 20.5 cm actually means we think that fish was between 20.45 and 20.55 cm.

What is the point of this? Well, if you have a binomially distributed variable where $n$, the number of trials, is sufficiently large (meaning the number of outcomes is large too) then the binomial distribution begins to look like the normal distribution, particularly where $p$ is somewhere near 0.5. The general guideline is that where the number of trials multiplied by the smaller of the probability of success ($p$) or the probability of failure ($q = 1 - p$) is greater than 5 then we can reasonably use the normal distribution to model the data (i.e. both $n * p > 5$ and $n * (1 - p) > 5$).

Turtles lay clutches of 40 eggs. We might be interested in predicting the number of male and female offspring. Assume the proportion that are male is 0.5. Our investigation stems from the observation that several clutches of eggs, on a particular island, contained only 10 males. We might wonder how unlikely this was by chance, assuming $P(male) = 0.5$.

*Q109. Referring to Practical 2 if necessary, use the binomial model, calculate how unusual (i.e. the probability) it is to get* $\leq 10$ *males.*

*Q110. Plot your probability (i.e. from 0 – 40 males) as a bar graph. Comment on its shape.*

The variance of a binomially distributed variable is simply $n * p * q$.

*Q111. Calculate the mean and variance of this population.*

*Q112. Are* $n * p$ *and* $n * (p - 1)$ *both >5?*

*Q113. Given these two variables, specify the equivalent normal distribution.*

We have the 'real' distribution $y \sim Binom(40, 0.5)$ and its normal approximation. We ask what is $P(y_i \leq 10)$ given $y \sim Binom(40, 0.5)$ and given $y \sim Norm(20, 10)$.

*Q114. Calculate the Z score for 10 males then find the probability of making this observation.*

*Q115. How do the values from the different models compare?*

---

## 3.10 Conclusions

The normal distribution is central to statistics. A huge variety of observations are reasonably approximated by the normal distribution (or can be made to be normally distributed through transformation).

The normal distribution can be used to assess the likelihood of a given observation, if we know the population mean and standard deviation from which it came. Furthermore, given our knowledge of the population parameter (mean and variance) we can determine values which define intervals on that population. Frequently scientists determine the values that bound 95% of their data.

The central limit theorem says that sample means taken from a normally distributed population will themselves be normally distributed and, in addition, means of sufficiently large samples will also be normally distributed even where the original data are not normally distributed (the sample size required depends on the extent of the skew in the original data).

The normal distribution is a good approximation of the Poisson distribution when lambda is large and the binomial model where the number of trials is large and the probability of success around 0.5 (i.e. the distribution of values is not too skewed).

# Chapter 4

# t-tests & confidence intervals

The goal of science is to understand nature (i.e. everything!). In order to do that, we want to know the values of population parameters (e.g. the mean size of barnacles on the back-beach, the variance in fail-rate of a machine component, the maximum satelite signal strength per satelite transect, the mean size of a fisher's catch). However, we are usually limited in our capacity to measure entire populations due to logistical/time/money constraints so we take a (random) sample, and infer from that sample to our population of interest. This is statistical inference and it's based on our statistical models of the world.

When we take a random sample we can never know how well it reflects the population. For example, our random sample of barnacles might contain (by chance alone) mostly big ones, or barnacles that varied considerably (or negligibly) in size. The t-distribution is similar to the normal distribution, but it accounts for this added uncertainty. This enables us to estimate the probability that a given sample came from a population with any given mean (with caveats). The t-distribution also enables us to put confidence intervals around the mean of our sample, and gives us some idea of the range of values of the mean that are likely.

The t-distribution models the chances of making observations from a given population where the population parameters are estimated from your random sample. The t statistic is calculated in the same way as the Z score for samples, but the expected scores follow the t-distribution (instead of the normal distribution) which accounts for sampling uncertainty. As the sample size gets smaller we get increasingly less confident about the reliability of the population parameter estimates and the t-distribution becomes less like the normal distribution (Fig. 4.1). The shape of the t-distribution depends on the *degrees of freedom* ($df = \nu = N - 1$). There is more guidance on the t-distribution, and links to other sources on Brightspace.

*Q116. Would you feel as confident about basing an estimate of the heights of the lab population on a sample of 2 compared with a sample of 50?*

*Q117. How does sample size affect the reliability of our population parametr estimates?*

## 4.1 Single sample t-tests

This is analogous to the calculation of Z scores and enables us to determine how unlikely our sample mean is, given any hypothesised mean. However, before using any parametric tests such as t-tests, we need to assure ourselves that the model assumptions are reasonably met. Note that t-tests, as illustrated here, are not testing null-hypotheses.

Imagine we are fisheries inspectors and have sampled the cod landed by a fishing boat called the 'The Evening Star'. We know that the mean size of the cod landed should be greater than

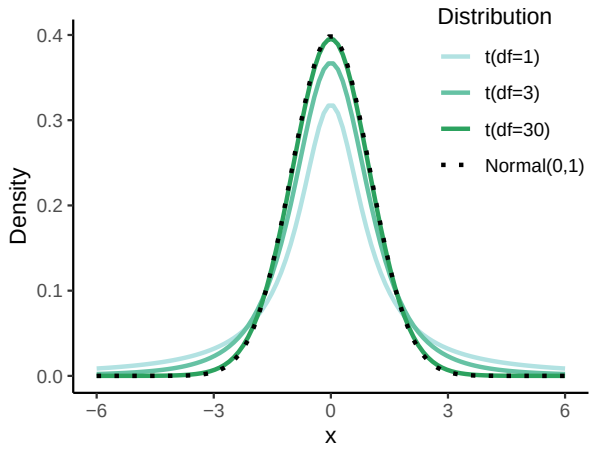



Figure 4.1: Comparison of different t-distributions (note that t with 30 df is nearly identical to the normal distribution).

36.6 cm. We need to assess how likely it is that our sampled cod come from a population (of landed fish) where the mean actually is $\geq$ 36.6 cm. We are testing the hypothesis that there is one 'population' of legally landed cod, and the Evening Star's cod are a part of that population. We use the t-distribution to assess the probability the Evening Star cod are drawn from the legally-landed cod population. If this probability is low then we might speculate that the cod are, in fact, drawn from a different population (i.e., that the boat is using illegal gear).

We do not know the population mean or standard deviation of the population from which the cod were caught and hence cannot use a normal distribution to model the likelihood of observing any particular value.

Before starting problems like this always state your hypotheses. You should (unless instructed otherwise) state both the null and alternative hypothesis. Here we are wondering whether the mean cod size on the Evening Star is $<$ 36.6 cm. The hypothesis should be worded thus:

**$H_0$ (the null hypothesis):** The true value of the mean of the Evening Star cod is $\geq$ 36.6 cm ($\mu \geq$ 36.6 cm).

**$H_1$ (the alternative hypothesis):** The true value of the mean Evening Star cod is less than 36.6 cm ($\mu <$ 36.6 cm).

We use the t-test to determine the probability of drawing the Evening Star sample from a population where the true mean is 36.6 cm or greater.

*Q118. Given the hypothesis, is this one or two tailed test?*

We collect a sample of 20 fish (found in the worksheet 'Cod lengths'). The sample size is $<$ 30 so we can't assume that the means will be normally distributed under the CLT. We can check the normality assumption by plotting the data using a 'normality' plot or 'QQ-plot' (Fig. 4.2).

```
library(readxl)
cod_df <- read_excel("data/practical_3_4.xlsx", sheet = "Cod lengths")
str(cod_df)
```

```
## tibble [20 x 1] (S3: tbl_df/tbl/data.frame)
##  $ CodLength_cm: num [1:20] 34.1 35.6 36.1 34.6 38.3 35 36.8 38 34.4 35.5 ...
```

```
qqnorm(cod_df$CodLength_cm, main = NULL)
qqline(cod_df$CodLength_cm)
```

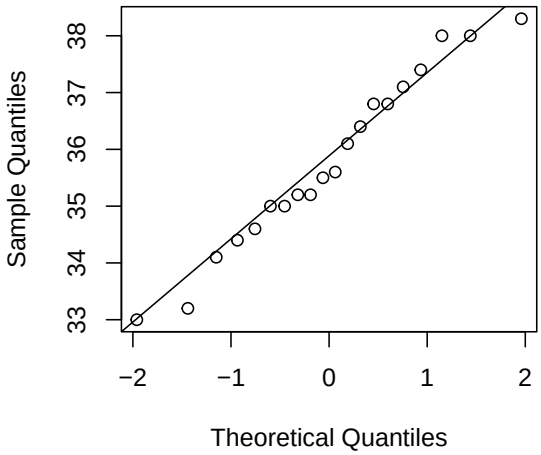



Figure 4.2: QQ-plot for the sample of code from the Evening Star.

*Q119. Do you think the normality assumption reasonable?*

*Q120. What parameter are we actually trying to understand/model? How does the distribution of this parameter change with sample size (think CLT)?*

Now we wish to assess how likely our sample is to have been drawn from a population where the mean actually is 36.6 cm. If the ES mean is less than the 'legal' mean and it is 'unlikely' to have been drawn from the legal population we might wonder if the mean of true Evening Star landed cod population is <36.6 cm and decide to prosecute the skipper.

*Q121. Use R to calculate summary statistics (mean and standard deviation/standard error of the mean) for the cod sample data (as per results below).*

```
cat("Mean:", mean(cod_df$CodLength_cm))
cat("SD:", sd(cod_df$CodLength_cm))
cat("Var:", var(cod_df$CodLength_cm))
cat("N:", length(cod_df$CodLength_cm))
```

```
## Mean: 35.785SD: 1.549966Var: 2.402395N: 20
```

*Q122. Determine the standard error of the mean (see Chapter 7 if necessary)*

```
sd(cod_df$CodLength_cm) / sqrt(length(cod_df$CodLength_cm))
```

```
## [1] 0.3465829
```

*Q123. Now manually calculate the t statistic for this sample and determine the probablity of observing your data assuming that the mean of the population was actually 36.6 cm.*

```
T_stat <- (mean(cod_df$CodLength_cm) - 36.6) /
  (sd(cod_df$CodLength_cm) / sqrt(length(cod_df$CodLength_cm)))
```

*Q124. How does this value compare to the expectation under the null hypothesis? Use* `pt()`.

Remember you are conducting a single-tailed test here, so you need to look-up your t-statistic at an $\alpha$ rate of double your P value of interest (so if you are setting a 95% CI you need to look-up $\alpha$ value of 0.100).

*Q125. Check you answer against that given by R.*

```
t.test(cod_df$CodLength_cm, mu = 36.6, alternative = "less",
       conf.level = 0.95, var.equal = TRUE)
```

```
##
##  One Sample t-test
##
## data:  cod_df$CodLength_cm
## t = -2.3515, df = 19, p-value = 0.01482
## alternative hypothesis: true mean is less than 36.6
## 95 percent confidence interval:
##      -Inf 36.38429
## sample estimates:
## mean of x
##    35.785
```

Hopefully your manually calculated t statistic and the one generated by R match. The p-value given by R is exact i.e. there is a probability of 0.014819 that a sample of 20 cod with mean of 35.785 cm would be drawn from a legally landed cod population where the true mean was 36.6 cm or more (assuming model assumptions are met).

*Q126. Can we now confidently send the skipper to jail?*

Remember the confidence interval relates to future (often hypothetical) observations, not an observation that has been made. Confidence intervals are notoriously difficult to define and and are often incorrectly used. BrightSpace has numerous resources to help you.

Evaluating evidence is a central part of statistcal analysis/modelling. In this example, assume you are evaluating whether a fishers catch is 'surprisingly' small (e.g. that the fisher is using an illegal net). If you don't believe the fisher (i.e that the fish sample was not drawn from a population of fish with a mean of 36.6 cm) then she goes to jail. If you do believe her, but she was fishing illegally, she avoids jail. In these circumstances you should set-out your P-value thresholds ahead of getting the data. For the moment, assume we set the P-value ($\alpha$ value) threshold at 0.05. In this scenario, where we have two clear competing hypotheses, we are in the realm of 'Neyman-Pearson's' decision theory (not in Fisher's hypothesis significance testing approach; see P-value lecture).

*Q127. Given the P value, do you reject the null hypothesis?*

*Q128. If you had set alpha at 0.01 would you reject the null hypothesis?*

*Q129. If you set alpha at 0.01, rather than 0.05, what type of error are you reducing and what type of error are you increasing?*

Now to play around with some random data that you generate yourself. We generate 100 random numbers drawn from X, where $X \sim Norm(100, 10)$.

*Q130. What is your standard deviation in this model?*

*Q131. What does the symbol '~' mean?*

We calculate summary statistics for this randomly generated dataset. Note that in this case, we *know* the population parameters.

```
num_iter <- 3
sample_size <- 3
mu <- 100
sigma <- 10

paste0("True mean: ", mu, ", true sd: ", sigma)
for (i in 1:num_iter) {
  sample_i <- rnorm(n = sample_size, mean = mu, sd = sigma)
  sample_mean <- signif(mean(sample_i), 4)
  sample_sd <- signif(sd(sample_i), 4)
  print(paste0("Sample ", i, " mean:", sample_mean, ", sd: ", sample_sd))
}
```

```
## [1] "True mean: 100, true sd: 10"
## [1] "Sample 1 mean:100.1, sd: 14.63"
## [1] "Sample 2 mean:107.9, sd: 5.05"
## [1] "Sample 3 mean:100.9, sd: 13.9"
```

Note: these are random samples, so the values will be different each time you run the code. However, R uses pseudo-random number generation. Use `set.seed()` for fully replicable code.

*Q132. Is there a discrepancy between the population parameters you defined and the actual mean and variance that are estimated from the samples?*

Your answer to the above should be yes. You know there is a discrepancy because you know the true parameters. In most real life situations you do not know the true population mean and variance. You can only sample them. If your sample is very large (and representative) then you can generate a very good estimate of those population parameters. However, as your sample size is reduced, the reliability of your estimate decreases. Look at the random numbers you've generated. Get a sense for where most of the numbers lie with `sigma=10`. To output your sample, just run `sample_i`.

The t-distribution is the distribution of values you get when you subtract sample means from the true mean and standardise by the sample standard error (i.e., $\frac{\bar{y}-\mu}{SE_{\bar{y}}}$). Think about this and relate it to the formula for determining single-sample T-statistics and what the critical values actually are.

The code below simulates repeated samples from a population with $y \sim Norm(\mu, \sigma)$. Each sample takes `sample_size` individuals, with `num_samples` unique samples. For each sample `i`, the t-statistic is calculated and stored in `T_sample[i]`. Fig. 4.3 shows the distribution of `T_sample` (the histogram) with the corresponding t-distribution (df=`sample_size - 1`) as the solid line and a standard normal distribution as the dotted line. Play with the values for `sample_size` below to see how the shapes change.

```
mu <- 10 # population mean
sigma <- 0.5 # population sd
num_samples <- 1e5 # number of samples
```

```
sample_size <- 3 # size of each sample
T_sample <- numeric(sample_size) # sample t statistic

for (i in 1:num_samples) {
  sample_i <- rnorm(sample_size, mu, sigma)
  sample_SEM <- sd(sample_i) / sqrt(sample_size)
  T_sample[i] <- (mean(sample_i) - mu) / (sample_SEM)
}

curve(dnorm(x, 0, 1), from = -6, to = 6, lty = 2,
      xlab = "Simulated t-statistics", ylab = "Density",
      main = paste("t-statistics of", format(1e5, big.mark=",", scientific=F),
                   "samples, each with N =", sample_size))
hist(T_sample, freq = F, add = T, col = rgb(1, 0, 0, 0.25),
     breaks = seq(floor(min(T_sample)), ceiling(max(T_sample)), by=0.2))
curve(dnorm(x, 0, 1), from = -6, to = 6, add = T, lty = 2)
curve(dt(x, sample_size - 1), from = -6, to = 6, add = T)
legend("topright", lty = c(2, 1, 1), col = c(1, 1, 2), bty = "n",
       c("Normal(0,1)", paste0("t(df=", sample_size-1, ")"), "t-stat (sim)"))
```

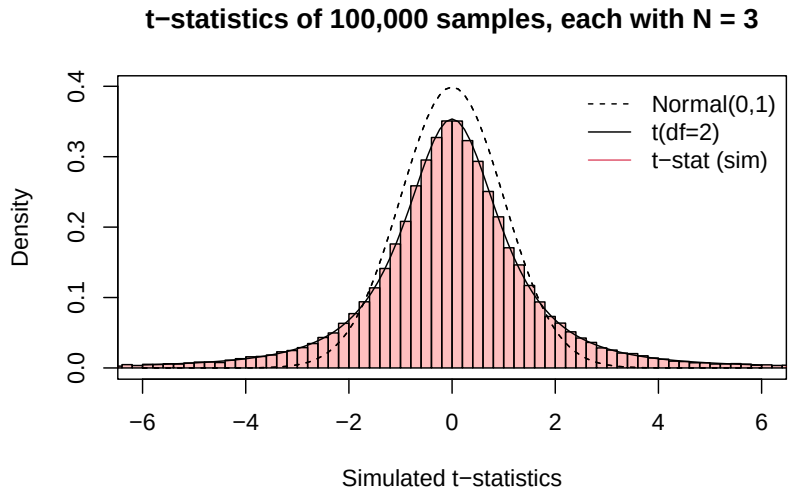



Figure 4.3: Histogram of 50,000 t-statistics calculated from 100,000 samples, along with the corresponding theoretical t-distribution (solid line) and a standard normal (dotted line).

Notice how the histogram and the solid lines are nearly identical? These simulations illustrate that the t-distribution *is* the distribution of t-statistics for a given sample size.

## 4.2 Confidence Intervals

Say you are interested in knowing the mean of a population (e.g. barnacle mass on the back beach). You cannot afford to determine the mass of each barnacle, so you take a random sample. You don't know how 'good' (i.e. representative) your sample is. It might have included lots of small barnacles, or big ones, or a wide- or narrow-range of sizes, you can never know (unless you sample everything). When you calculate the mean of this sample you don't know how close

it is to the population mean, but you do know the probability associated with that estimate. Confidence intervals capture this uncertainty, and you use the t-distribution to determine them.

We'll invent a population of barnacle diameters, called `barnacle_diam`, and then create a histogram of that population and superimpose values on that. Again, these are random numbers so your values will be slightly different from mine.

```
pop_size <- 100000  # number of barnacles in the population
meta_mu <- 200
meta_sigma <- 25
barnacle_diam <- rnorm(pop_size, mean = meta_mu, sd = meta_sigma)
mu <- mean(barnacle_diam)
sigma <- sd(barnacle_diam)
hist(barnacle_diam, main = NULL)
Q95 <- quantile(barnacle_diam, c(0.025, 0.975))
abline(v = Q95, col = "green", lwd = 3)
text(x=Q95[2], y=pop_size/7,
     labels=paste0("mu: ", round(mu, 1), "\nsigma:", round(sigma, 1)))
```

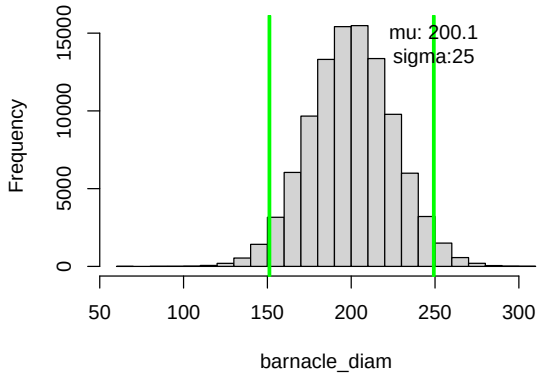



Figure 4.4: Histogram of a simulated barnacle population with 2.5% and 97.5% quantiles.

Now we can take samples from that population: this is the reality, you take samples (usually) from populations where you don't know the true mean and standard deviation. Let's take 4 samples, each with size `sample_size`.

```
hist(barnacle_diam, main = NULL)
sample_size <- 5
num_samples <- 4
abline(v = mu, col = "blue", lwd = 4)

for (i in 1:num_samples) {
  sample_i <- sample(barnacle_diam, size = sample_size)
  print(sample_i)
  abline(v = mean(sample_i), col = "red", lwd = 0.5)
}
```

```
## [1] 127.0598 204.3132 212.1038 155.2723 202.1684
```

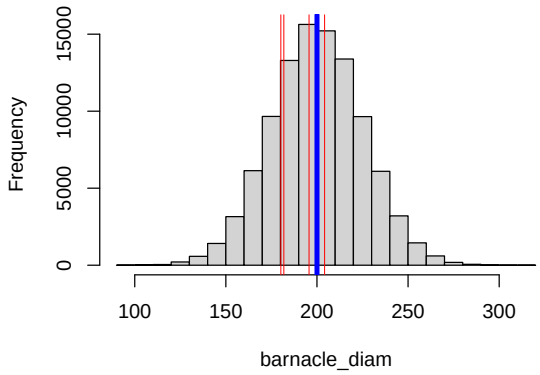



Figure 4.5: Histogram of the barnacle population showing location of 4 sample means (red lines), each with N = 5. The blue line shows the true population mean mu.

```
## [1] 188.9613 173.8530 202.7288 209.3079 203.4615
## [1] 173.6354 170.4280 162.4823 199.2663 203.2909
## [1] 181.7948 218.1346 179.5933 236.8487 204.5922
```

Our sample means inevitably differ from the true population mean ($\mu$), even if only a bit. Likewise, the sample standard deviations will differ from the true population standard deviation ($\sigma$). If you keep repeating this sampling you can generate a distribution of sample standard deviations. This distribution is not normal, but is instead related to the chi-square distribution (don't worry too much about this). The point is that if your sample size is small, your estimate of the standard deviation is often very poor.

```
par(mfrow=c(2,2))
# sim_df will hold the sample sizes N, and the median and mean sample sd's
num_samples <- 1e4
sim_df <- data.frame(N=c(2, 4, 10, 30),
                     sd_median=NA, sd_mean=NA,
                     sd_q025=NA, sd_q25=NA, sd_q75=NA, sd_q975=NA,
                     mn_median=NA, mn_mean=NA,
                     mn_q025=NA, mn_q25=NA, mn_q75=NA, mn_q975=NA)

# for each sample size N, draw a sample and store its mean and sd
# repeat this num_samples times
# plot a histogram of the sample sd's, then store the mean and median
for (i in 1:nrow(sim_df)) {
  samp_sd_i <- numeric(num_samples)
  samp_mn_i <- numeric(num_samples)
  for (j in 1:num_samples) {
    sample_ij <- sample(barnacle_diam, size = sim_df$N[i])
    samp_sd_i[j] <- sd(sample_ij)
    samp_mn_i[j] <- mean(sample_ij)
  }
  hist(samp_sd_i, main = paste(num_samples, "sample SDs for N =", sim_df$N[i]),
       breaks = 20, xlim = c(0, 100))
  abline(v = sigma, col = "blue", lwd = 2)
```

```
  sim_df[i, 2:13] <- c(median(samp_sd_i), mean(samp_sd_i),
                       quantile(samp_sd_i, probs = c(0.025, 0.25, 0.75, 0.975)),
                       median(samp_mn_i), mean(samp_mn_i),
                       quantile(samp_mn_i, probs = c(0.025, 0.25, 0.75, 0.975)))
}
```

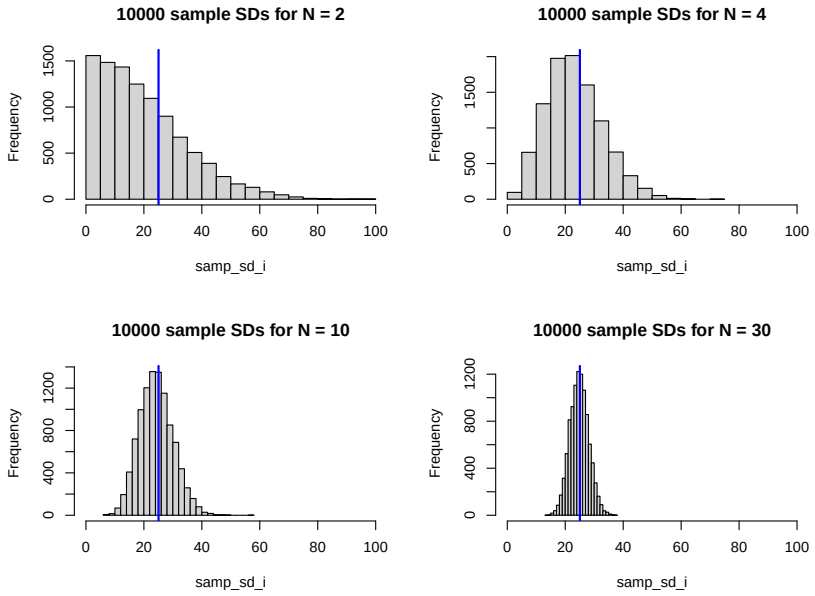



Figure 4.6: Histograms of sample standard deviations from repeated samples of the same population. The true population standard deviation is shown in blue.

```
par(mfrow=c(1,2))
plot(sim_df$N, sim_df$sd_median,
  xlim = c(0, 30), ylim = range(c(sim_df[,2:7], sigma)),
  type = "b", xlab = "Sample size", ylab = "Standard deviation"
)
segments(sim_df$N, sim_df$sd_q25, sim_df$N, sim_df$sd_q75, lwd=2)
segments(sim_df$N, sim_df$sd_q025, sim_df$N, sim_df$sd_q975)
lines(sim_df$N, sim_df$sd_mean, type = "b", col = "dodgerblue")
abline(h = sigma, lty = 2)
legend("topright", c("Mean sample SD", "Median sample SD", "True SD"),
  col = c("black", "dodgerblue", "black"),
  lty = c(1, 1, 2), pch = c(1, 1, NA), bty = "n"
)
plot(sim_df$N, sim_df$mn_median,
  xlim = c(0, 30), ylim = range(c(sim_df[,8:13], mu)),
  type = "b", xlab = "Sample size", ylab = "Mean"
)
segments(sim_df$N, sim_df$mn_q25, sim_df$N, sim_df$mn_q75, lwd=2)
segments(sim_df$N, sim_df$mn_q025, sim_df$N, sim_df$mn_q975)
lines(sim_df$N, sim_df$mn_mean, type = "b", col = "dodgerblue")
abline(h = mu, lty = 2)
legend("topright", c("Mean sample mean", "Median sample mean", "True mean"),
  col = c("black", "dodgerblue", "black"),
```

```
  lty = c(1, 1, 2), pch = c(1, 1, NA), bty = "n"
)
```

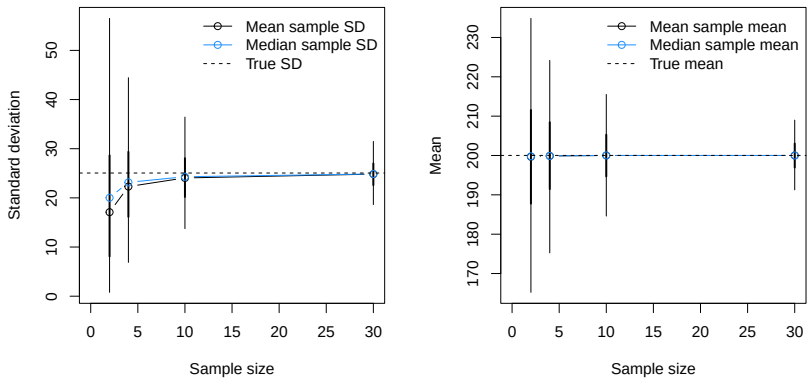



Figure 4.7: Mean (black), median (blue), and 50% and 95% quantiles (vertical lines) for (left) sample standard deviations at each sample size compared to the true population standard deviation (dotted line) or for the (right) sample means.

*Q133. What is the most common standard deviation for your samples by sample size?*

The t-distribution allows for the fact that the standard deviation of small samples is, usually, less than that of the population as seen in Fig. 4.6.

The take home message here is that when you take a sample from a population with unknown $\mu$ and $\sigma$, you won't know how 'accurate' you sample is but you do know how your random samples 'behave' - they are modelled using the t-distribution. From this knowledge you can build a 95% confidence interval which is described as an interval which, if repeated for 100 samples, would include $\mu$ within its boundaries in 95 of those samples (on average). Read that again. You don't have knowledge of the true value of the mean or sd (as you did for Z score calucations) and the t-distribution accounts for this uncertainty.

```
num_samples <- 5
sample_size <- 3

# plot population
hist(barnacle_diam, xlim = c(mu-6*sigma, mu+6*sigma),
     main = NULL, ylim = c(0, length(barnacle_diam)/6),
     col = "grey90", border = "grey50", xlab = "Barnacle diameter")
abline(v = mu, col = "blue", lwd = 2)
y_pos <- seq(0, length(barnacle_diam)/6, length.out=num_samples)

# draw samples, calculate mean and 95% CIs, and plot them
for (i in 1:num_samples) {
  sample_i <- sample(barnacle_diam, size = sample_size)
  points(x = mean(sample_i), y = y_pos[i], col = "red", pch = 16, cex = 0.75)
  sample_ci <- c(
    mean(sample_i) + qt(0.025, (sample_size - 1)) * (sd(sample_i) / sample_size^0.5),
    mean(sample_i) + qt(0.975, (sample_size - 1)) * (sd(sample_i) / sample_size^0.5)
  )
```

```
  arrows(sample_ci[1], y_pos[i], sample_ci[2], y_pos[i],
         col = "red", code = 3, angle = 90, length=0.05)
}
```

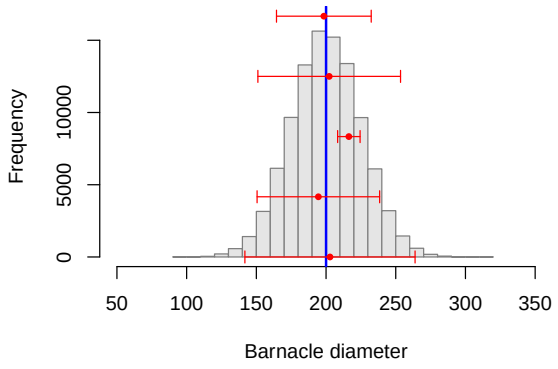



Figure 4.8: Histogram illustrating the barnacle population with population mean (blue) and sample means with 95% CIs (red) repeated across 20 samples.

*Q134. Try different values for* `sample_size`*. How does this influence the width of your CIs?*

*Q135. What proportion of your 95% CIs would you expect to include the true value of the mean? Does the sample size have an impact on this?*

*Q136. Keep repeating the above code until you get an example where your 95% CI misses the true value of the mean.*

*Q137. See if you can find the relevant bit of the code, and determine 99% and 90% or 75% CIs (pick any value you fancy, but note that some values are likely to put the CIs some distance from the mean so get ready to adjust your axis limits which are set to 6 $\sigma$ on either side of $\mu$).*

Let's explore the influence of sample size on the width of the confidence interval a little more.

We will plot the resultant mean estimate and 95% CI on Fig. 4.9. NOTE: the example shown in Fig. 4.9 is a random example, not linked to the output below. Repeat this process, but this time determine the mean and sd (or variance) for samples of 2, 5 and 10. Do this simply by changing `n=…` in `rnorm()` Repeat this 5 times for each sample size and sketch your results (where they fit) onto Fig. 4.9.

```
a <- rnorm(n = 10, mean = 100, sd = 10)
signif(a, 3)
```

```
##  [1]  97.9 102.0  78.1  93.2 102.0 110.0  94.1 102.0  98.4 102.0
```

```
t.test(a, mu = 100)
```

```
##
##  One Sample t-test
##
## data:  a
```

```
## t = -0.77679, df = 9, p-value = 0.4572
## alternative hypothesis: true mean is not equal to 100
## 95 percent confidence interval:
##   91.89388 103.96207
## sample estimates:
## mean of x
##  97.92798
```

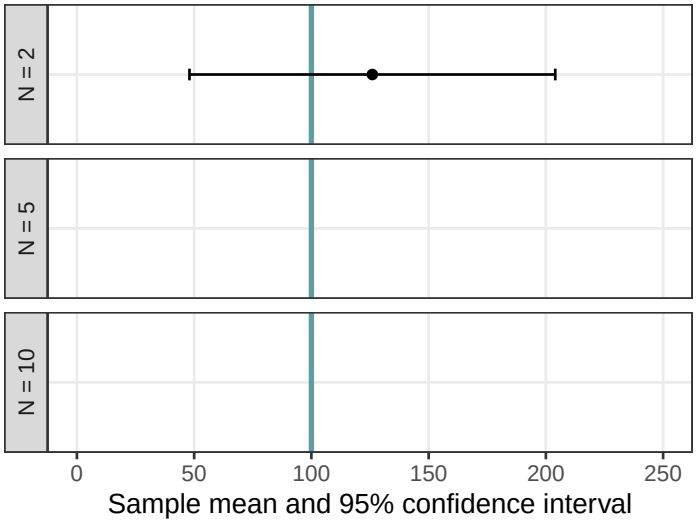



Figure 4.9: Confidence interval size vs. sample size.

*Q138. Look at your 95% confidence intervals – are they getting broader or narrower as your sample size decreases?*

*Q139. What proportion of 95% CIs would you expect to include the true value of the mean?*

## 4.3 Comparing means (two-sample t tests)

The two-sample t-test is one of the most widely used inferential statistical tests. The two-sample t-test is a special case of analysis of variance (ANOVA) where you are only comparing two means. The results are identical and so we do not focus on two-sample t-tests. You need to know of the existence of the two-sample t-test because it is so commonly used and cited, but you will be comparing means using ANOVA in Practical 5.

## 4.4 Non-parametric Tests

Non-parametric tests are often used to compare samples where the data are non-continuous or fail the assumptions of parametric general linear models, typically converting data to ranks rather than using the actual values. Non-parametric test include 'classics' such as the 'Mood' and 'Wilcoxon' tests. However, we make you aware of the GLM family which will usually supply you with a much more elegant solution to model data that doesn't fit the simple linear model. You should be aware of the existence of 'non-parametric' tests because they are prevalent in the literature. Remind yourself of the disadvantages of non-parametric tests.

## 4.5 Conclusions

The t-test is a 'classic' statistical test which doesn't assume knowledge of population parameters. The strength of the t-test (its ability to quantify differences between samples) is proportional to the sample size. The larger the sample size, the better the estimate of the population parameters and the more precisely we are to be able to detect differences between the means.

The central limit theorem tells us that the means of non-normally distributed data will be normally distributed if the sample size is sufficiently large. If your sample size is $> 30$ it is likely that the means of that sample will be normally distributed regardless of the distribution of the original data.

Parametric tests including the t-test are quite 'robust' against deviations from normality, particularly as sample sizes increase. However, parametric test are less robust against heteroscedasticity, regardless of sample size. Always check this assumption and be prepared to transform the data if the assumption of homoscedasticity is not tenable (more of this in Practical 5).

The t-test is in the 'general linear model' family (and this is a subset of the generalised linear modelling family). General linear models are usually used to model continuous data. If you have count data, you should start with a different member of the GLM family (you might not be able to transform count data to something approximating a normal distribution). Non-parametric tests are frequently adopted when data do not conform to the assumptions of normality but they are invariably used for NHST with all the inherent problems with that approach.

A final reminder with regard to many statistical tests, including all in the GLM family: they make the assumption that data are independent. You must always ensure that your experimental design lends itself to making independent observations IN RELATION TO THE QUESTION YOU ARE ASKING. This is the most critical and fundamental of the assumptions of parametric and non-parametric tests. Non-independence (e.g. measuring the same 'subject' (e.g. an urchin) over time) can be modelled using more complex 'mixed' models. Application of mixed modelling is beyond this course but you should be aware of the limitations of the techniques that you are learning and know where to go next.

# Chapter 5

# General Linear Models

General linear models are a key member of the generalised linear modelling family and they are among the most widely used and reported models in the marine science literature, particularly biology. This course focuses on two subsets of linear models: ANOVA and regression. ANOVA and regression are typically used for different data modelling scenarios: ANOVA when you've got a categorical predictor variable(s) and regression when your predictor is continuous.

## 5.1 Analysis of variance (ANOVA)

ANOVA is a widely used modelling approach that enables you to compare means and put confidence intervals on the differences between those means. For a predictor with only two categories, ANOVA is identical to the 2-sample t-test, so we'll just use ANOVA.

When you see "analysis of variance", think "analysis of means". In an ANOVA, we analyse the variance in the data in order to compare the means of different groups. In an ANOVA, we compare means by determining the ratio of [the variance between treatments and the overall mean (large black arrows in Fig. 5.1)] and [the variance within treatments (sum of the dotted arrows in Fig. 5.1)]. The black line under the red dots in Fig. 5.1 shows the actual data distribution and the actual parameters for mean values (50 and 150 cm for A and B respectively). You then take samples from A and B (n=4 in this example) and, from these, derive your parameter estimates for the mean of each group and the overall mean.

In Fig. 5.1 below you can see that the solid black arrows are much larger than the dotted ones leading you to think that the chance that these two samples are drawn from the same population (with a value of the overall mean) as very unlikely. Make sure you understand Fig. 5.1 (more detail in the ANOVA lecture).

In most circumstances you know that the means that you are comparing with ANOVA are different (i.e. that testing a null hypothesis of no difference isn't useful). ANOVA allows you to put confidence intervals around differences between means, or groups of means.

### 5.1.1 ANOVA in R

There are numerous variations on the theme of ANOVA. We cover one-way ANOVA and we mention two-way ANOVA with and without replication. The objective of ANOVA is to establish the size of the difference (called the 'effect size') between different groups (e.g., treatments or locations) and put a confidence interval on those differences.

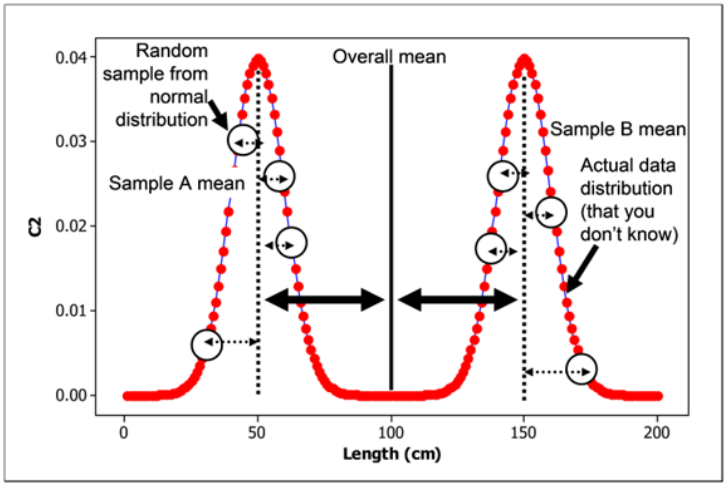



Figure 5.1: Sources of variation (within group and between groups) as quantified by ANOVA.

### 5.1.2 One-way ANOVA

One-way ANOVA is a procedure we use to estimate the magnitude of differences between means of $\geq 2$ groups. We also use it to put confidence intervals on those differences.

The first example data is the yield in $\mu g$ C $ml^{-1}$ of a species of microalgae (*Isochrysis galbana*) in laboratory culture exposed to three light levels (low, medium and high). We are interested in these particular light levels because they represent the means of winter, spring, and summer Scottish sun intensity. The data are in worksheet 'Microalgae'.

Note: the function to conduct an ANOVA is `aov()`. The function `anova()` converts various statistical model outputs to the standard ANOVA-table output (including any from the GLM family).

```
library(readxl)
algae_wide_df <- read_excel("data/practical_5.xlsx", sheet = "Microalgae")
# check these data as usual
```

*Q158. What is your objective in this type of experiment? What are you interested in estimating?*

*Q159. What assumptions should be met prior to undertaking parametric ANOVA?*

*Q160. Under which circumstances could you begin to relax the assumption that the data are normally distributed (think central limit theorem)?*

*Q161. Given your sample size can we assume normality of means?*

*Q162. Are the data normally distributed (be careful how you word your answer to this question, see the following question)?*

*Q163. Is it reasonable to assume that these data are drawn from a population that is normally distributed?*

We can check the homoscedasticity assumption using Bartlett's test. Before we can use this test we need to rearrange the data so that the data is in a single indexed column. We'll use the *tidyverse* as before.

```
library(tidyverse)
head(algae_wide_df, 2)
```

```
## # A tibble: 2 x 3
##     low medium  high
##   <dbl>  <dbl> <dbl>
## 1  13.1   12    14.2
## 2  11.5   11.5  13.1
```

```
algae_df <- algae_wide_df |>
  pivot_longer(everything(), names_to = "Treatment", values_to = "Yield")
glimpse(algae_df, width=80)
```

```
## Rows: 15
## Columns: 2
## $ Treatment <chr> "low", "medium", "high", "low", "medium", "high", "low", "me~
## $ Yield     <dbl> 13.07599, 12.00000, 14.20000, 11.53923, 11.50000, 13.10000, ~
```

```
bartlett.test(Yield ~ Treatment, data = algae_df) #?bartlett.test
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Yield by Treatment
## Bartlett's K-squared = 2.8413, df = 2, p-value = 0.2416
```

*Q164. With regard to Bartlett's test, is the assumption of homogeneity reasonable?*

A more elegant (and much better) way of checking model assumptions is to check residual patterns. A 'residual' is the difference between an actual data value and that predicted by the model. Here we have randomly assigned five cultures each of the same species to three specific treatments (light levels).

*Q165. What type of experiment is this (how many factors, are they fixed or random)?*

Next to conduct the analysis:

```
algae_aov <- aov(Yield ~ Treatment, data = algae_df) # ?aov
```

Before we look at the output, let's assess the assumptions using the residuals. The default residual plots created by R are shown in Fig. 5.2 and enables us to rapidly assess whether the model assumptions are reasonable.

```
par(mfrow = c(2, 2), mar=c(4,4,1,1))
plot(algae_aov)
```

**Interpretation of residual patterns:**

- **Upper left:** Residuals v. fitted. This is the residual values against the fitted values. The fitted values are the means of the three groups (remember that ANOVA is about comparing means). The spread for the lower values (low and medium light) is higher than for the high light so this might make us consider the homoscedasticity assumption.

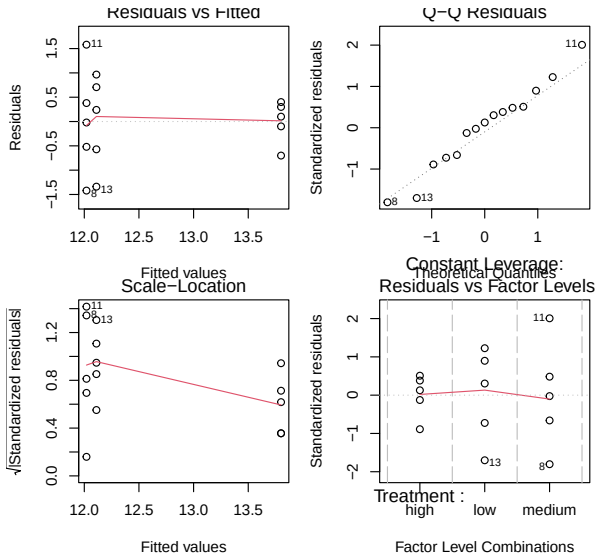



Figure 5.2: Residual plots from one-way ANOVA.

- **Upper right:** Normal Q-Q plot. This assesses the normality assumption. The points (each point is an observation) lie around the straight line so this assumption is reasonable. Note that general linear models assume that the means of groups are normally distributed, and this always applies when the means are based on large sample sizes (roughly $n > 30$)). When $n < 30$, you should check that the distribution of the residuals is reasonably 'normal'.

- **Lower left:** Scale-location. This specifically looks to assess whether residuals increase with fitted values, which is a common issue in these types of analysis. In this case, the scale decreases with fitted value. This is similar to the Upper Left plot, but with `sqrt(abs(standardized_residuals))` on the y-axis instead of just `residuals` to focus just on the magnitude of the residuals.

- **Lower right:** Constant leverage, residuals vs. factor levels. This indicates how each treatment is fitted (i.e. the residuals associated with each treatment). You might be concerned if one particular treatment was associated with extremely high residuals (outliers). R automatically identifies potential outliers (8, 11, and 13 in this case) for you to further assess. In this case there is nothing in particular to worry about.

The residual plots allow you to investigate different aspects of the data and the how their assumptions are met. The interpretation of the plots overlaps in the sense that the same issue might be apparent in several of the plots.

*Q166. What are the 'fitted values' for an ANOVA?*

*Q167. Are your effects fixed or random?*

*Q168. Assuming you have chosen $\alpha = 0.05$, what might you be interested in going on to test next? Hint: you are testing to see whether the means of three populations are different.*

Everything looks ok, so we can then look at the results of the ANOVA.

```
# anova(algae_aov) # outputs an anova-type table, but unnecessary with aov()
summary(algae_aov)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Treatment    2 10.050   5.025   6.487 0.0123 *
## Residuals   12  9.296   0.775
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Reporting that there are 'significant' differences between means is not enough. What your readers should be interested in is what the differences between the means actually are, and how confident you are in your assessment. This can be provided by the Tukey test in R.

```
algae_grp_diffs <- TukeyHSD(x = algae_aov, conf.level = 0.95)
# HSD stands for 'honestly significant difference'
algae_grp_diffs
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Yield ~ Treatment, data = algae_df)
##
## $Treatment
##                     diff       lwr        upr     p adj
## low-high    -1.68922461 -3.174304 -0.2041449 0.0260570
## medium-high -1.78000000 -3.265080 -0.2949203 0.0194474
## medium-low  -0.09077539 -1.575855  1.3943043 0.9854641
```

You can see that the mean yield at the high light level is higher than at both the low and medium: the 95% confidence interval of the difference in comparing high and low light levels are 1.69 (0.205, 3.17) $\mu g$ C $ml^{-1}$. (I've inverted the results so the the difference is seen as positive (high - low rather than low - high)). This confidence interval is much more important than any P-values and you should report both (CI because it is useful, P value by convention).

### 5.1.3 Two-way ANOVA

You've had a look at one-way ANOVA (i.e., one predictor), which is a good starting point. However, in nature you often find numerous factors combine to influence an outcome. This is called an 'interaction'. Two-way ANOVA allows you to investigate the nature of this interaction term. You can also get three-way ANOVA and more, but these get logistically challenging because you need to replicate across each level. The interpretation also gets increasingly difficult. You need to be aware of the existence of two-way ANOVA and what it offers, and how to interpret simple graphics (below), but we do not cover implementation of two-way ANOVA.

In Fig. 5.3 we have the outcome of an experiment. Each dot on the plot represents the mean Response (e.g. growth) of a number of replicates, subject to the combination of Temperature (cold and hot) and Nutrient (N and P). We are interested in the main effects (Temp and Nutrient) and their interaction (Temp * Nutrient) In panel A, there is no effect of Temp on the Response (the lines between Cold and Hot are horizontal), but there is a main effect of Nutrient (P is higher than N). In B, there is an effect of Temp (Hot is, on average higher than Cold) but there is also an interaction as the effect of Hot is more for P than for N (where it has no effect in this example). In C, the interaction is stronger compared with B. In D, there are no treatment main effects because mean Cold = mean Hot and mean N = mean P, but there is a very strong interaction effect; the effect of Temp is reversed by Nutrient, so that the level of Nutrient (N

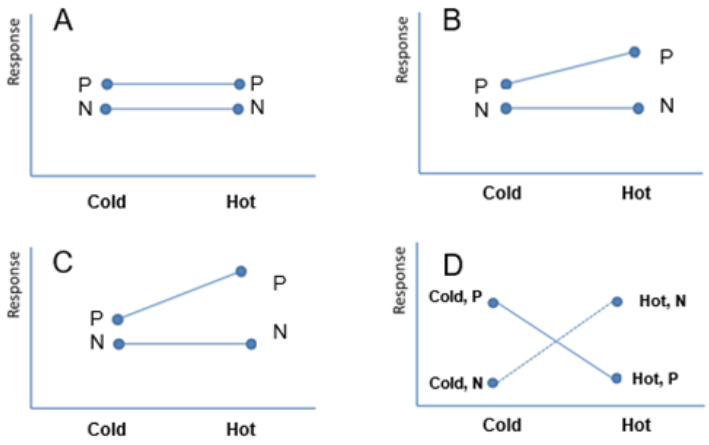



Figure 5.3: Graphic illustrating differing levels of interaction on the response from two treatments A and B.

or P) determines the effect of Temp. When it is Cold the Response is high for Nutrient P and low for Nutrient N, when it is Hot, the Response is low for Nutrient P and high for Nutrient N (but the average for hot=cold, and average for N=P)

---

## 5.2 Regression

Correlation and regression are used to examine the strength of association between two variables. In correlation, both variables are measured (and therefore associated with measurement error). In regression, one variable is fixed (by the experimenter) and is asssumed to have no 'error' associated with it and the other, called the 'response variable', is measured (so has measurement error). You must be able to distinguish which of correlation or regression analyses are most appropriate.

Correlation analysis is used to measure association, where you are not attempting to formally link cause-and-effect. Regression analysis is generally used where you have experimentally manipulated the fixed factor and are looking at the response in another factor. Causation is implicit in inferential regression analysis (correlation analysis is often used in 'exploratory' data analysis where any link between cause-and-effect is inherently more speculative).

The media often misreport science because it is difficult to resist the impulse to attribute causation. An overwhelming number of spurious correlations (i.e., those *clearly* having no causal relationship) are documented on tylervigen.com.

### 5.2.1 Overview

Regression is at the heart of linear models. ANOVA and t-tests are, basically, special cases of linear regression models. The regression coefficient is a measure of the strength of the relationship between the dependent variable (the one you measure) and the independent variable (the one you fix like a fixed factor in ANOVA). The regression coefficient is denoted by $R^2$ compared with $r$ in correlation. The regression coefficient $R^2$ ranges from 0 to 1 (unlike $r$ which ranges from -1 to 1). A value $R^2 = 0$ indicates no relationship to the independent variable while

$R^2 = 1$ indicates that the independent variable is entirely responsible for the variability in the measured variable.

As usual, null hypothesis significance testing is often applied to regression statistics. As usual, the null hypothesis being tested is usually "there is no functional relationship between the response and the predictor" and this is usually conceptually nonsense. In conducting regression analysis, your objective is to quantify to the most appropriate precision and accuracy possible the relationship between X (the aspect you control, the predictor, plotted on the X axis) and Y (the variable you measure, the response, plotted on the Y axis). Your objective is to quanitfy this relationship, put confidence intervals on it, and then interpret your findings in relation to the objectives of the study and in relation to other research.

*Q169. Sketch a graph demonstrating the null hypothesis (of no relationship) in regression analysis.*

Let us now consider an example in which cause and effect does exist. The data in worksheet 'Regression1' shows the weight loss in *Tribolium confusum*, the confused flour beetle, at different relative humidities (data from Sokal and Rohlf, 1995). The relative humidity (RH) to which the beetles are exposed can be fixed and the weight loss (via evaporative losses) of the beetles then assessed. There is no way that the null hypothesis can be true in this case: humidity will obviously influence weight loss in beetles.

*Q170. In this case, what is your response variable (what are you measuring) and your predictor (i.e. what is it that you are manipulating to determine the extent of the response)?*

*Q171. Plot the data in R and check your prediction. In this case, the predictor must be displayed on the x-axis and the response must be on the y-axis.*

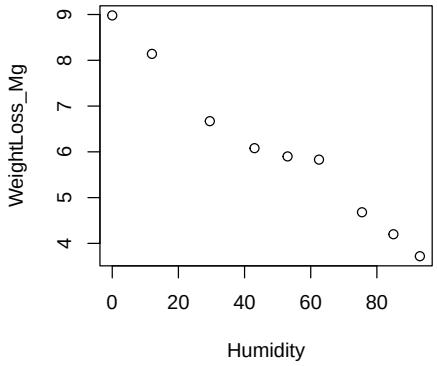
We are interested in whether the whole data set can be usefully represented by a linear regression relationship. We wish to estimate the relationship, and put a confidence interval on our estimate. Common sense tells us that there *is* some sort of relationship (testing a null hypothesis is not very useful) but it might go in either direction (positive or negative) and we don't know the strength (i.e. slope) of that relationship.

### 5.2.2 Linear regression in R

In R we can use a variety of techniques to conduct linear regression. The easiest is to use `lm()`. It is worth noting that `lm()` would also work for all your other general linear models (e.g. ANOVA). They are, in fact, the same model, it is just the default output (and necessary input formatting) that differs. Try reproducing the ANOVAs above with `anova(lm(...))`.

Import data and begin:

```
beetle_df <- read_excel("data/practical_6.xlsx", sheet = "Beetles")
# inspect the dataframe, then make a scatter plot
par(mfrow=c(1,1))
plot(WeightLoss_Mg ~ Humidity, data = beetle_df)
```



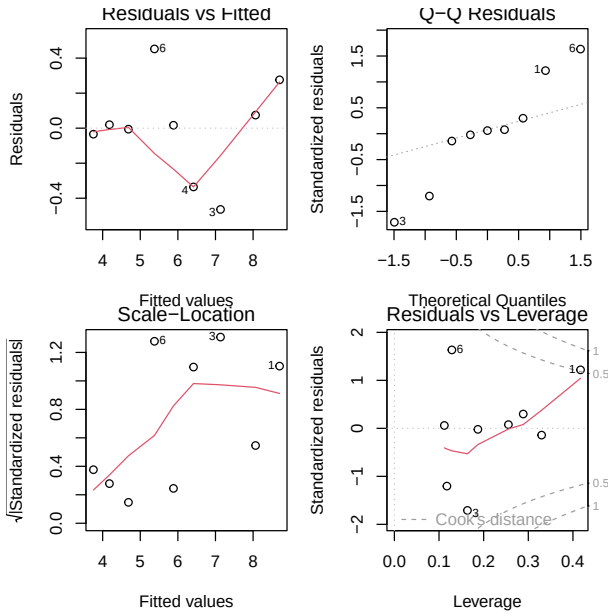

An aside on plotting: you can provide `plot()` with either a vector for the x-axis and a vector for the y-axis (i.e., `plot(x_var, y_var)`) *or* you can use a formula, specifying the dataframe (i.e., `plot(y ~ x, data=data_df)`). Just be aware of which variable is on which axis.

Now we have explored and plotted the data we can conduct the regression analysis.

```
# weightloss is modelled as (~) a function of humidity
beetle_lm <- lm(WeightLoss_Mg ~ Humidity, data = beetle_df)
# beetle_lm
# str(beetle_lm) # lm outputs are complex structures
```

Before we go on and interpret the model output we need to assess the model assumptions. This is done in the same way as for ANOVA with the same commands.

```
par(mfrow = c(2, 2), mar=c(4,4,1,1)) # set up 4 in 1 plot.
plot(beetle_lm) # plot the regresssion residuals.
```




The small sample size here ($n = 9$) makes a proper analysis of the residuals difficult. The plot should be assessed in the same way as for the ANOVA residuals. Basically, any pattern is

bad. The upper left (Residuals v Fitted) doesn't cause any major concern, thought upper right (Normal QQ) indicates a possible problem. Scale-Location (lower left) is difficult to interpret but no obvious pattern is present. The Residuals v. leverage (lower right) indicates a potential issue as well. A point with a large residual (i.e. where it is very different to that expected by the model) and with a high leverage (i.e. at the extreme ends of the predictors range) has a large Cook's distance and has a disproportionate effect on the slope and intercept. These points should be examined in more detail.

*Q172. Which point has the largest Cook's distance?*

We will now proceed to looking at the linear regression analysis results on the basis that the residuals do not raise any concerns.

```
summary(beetle_lm)
```

```
##
## Call:
## lm(formula = WeightLoss_Mg ~ Humidity, data = beetle_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46397 -0.03437  0.01675  0.07464  0.45236
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.704027   0.191565   45.44 6.54e-10 ***
## Humidity    -0.053222   0.003256  -16.35 7.82e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2967 on 7 degrees of freedom
## Multiple R-squared:  0.9745, Adjusted R-squared:  0.9708
## F-statistic: 267.2 on 1 and 7 DF,  p-value: 7.816e-07
```

The regression equation of the form $y = a + bx$ can be determined. The regression equation is:

$WeightLoss = 8.70 - 0.05322 * humidity$

Common-sense check: the coefficient is negative. As the humidity increases, the weight loss decreases (as expected and shown in the scatter plot).

*Q173. What is the effect on weight loss of increasing the relative humidity by 10%?*

*Q174. What is the weight loss, predicted by the model, when relative humidity is 0%?*

*Q175. What does the model suggest the weight loss will be when relative humidity is -50% and +150%? Are these values sensible? What does this tell you about extrapolating beyond the data range in using regression analysis in predictions?*

The residual error is the variance in y around the line. The $R^2$ is the proportion of this variance that is explained by the regression line. In the current case $R^2 = 0.97$. This is an extremely high value and indicates that the regression model is extraordinarily good at accounting for the variance in weight loss based on the relative humidity.

The P values allow us to assess if the slope and the intercept are likely different from zero.

*Q176. Given the very high $R^2$ (and looking at your plot) would you expect the regression model to be significantly better than the null model in explaining the variance in weight loss?*

*Q177. With $\alpha = 0.05$, do you reject or accept the null hypothesis? What would you wish to report in relation to the slope coefficient if you were reporting the results from this analysis?*

```
confint(beetle_lm)
```

```
##                   2.5 %      97.5 %
## (Intercept)  8.25104923  9.15700538
## Humidity    -0.06092143 -0.04552287
```

The confidence intervals are, again, 'clunky' to describe. What the above table indicates is that, if alternate yous (like in a multiverse) repeated your experiment with the same sample size, you would expect the intercept of the line to lie between 8.25 and 9.16 (3 sf) in 95% of those replicates, and you would expect the slope to vary between -0.0455 and -0.0609. The confidence interval is the scientifically interesting bit rather than the P value.

### 5.2.3 Plotting the regression line and confidence intervals

A regression model (i.e. the linear relationship between the predictor and response variables) allows us to predict values for any value of the predictor, along with confidence levels. We can plot this regression line without too much effort.

```
beetle_pred <- predict(beetle_lm, interval = "confidence", level = 0.95)

par(mfrow=c(1,1))
plot(WeightLoss_Mg ~ Humidity, data = beetle_df,
     xlab = "Relative humidity (%)", ylab = "Weight loss (mg)")
lines(beetle_df$Humidity, beetle_pred[, "fit"])
lines(beetle_df$Humidity, beetle_pred[, "lwr"], lty = 2)
lines(beetle_df$Humidity, beetle_pred[, "upr"], lty = 2)
```

Try generating 90% confidence intervals and add them to the plot.

*Q178. Which will have the wider interval, a 99.99% interval or a 50% interval and why?*

*Q179. Do the confidence intervals in Fig. 5.4 run parallel to the regression line?*

*Q180. If not, what does this suggest about the degree of confidence you have in values predicted at various points along the line?*

*Q181. At what value of relative humidity are your predictions of weight loss likely most accurate?*

We can make predictions based on our regression line, and put confidence intervals on those predictions. Say we had a relative humidity of 50% in the above example. You could ask for the model-predicted weight loss and you'd want confidence intervals on that prediction.

```
# predict() needs a data.frame with the same predictors used in beetle_lm
predict(beetle_lm,
        newdata = data.frame(Humidity = 50),
        interval = "predict",
        level = 0.95)
```

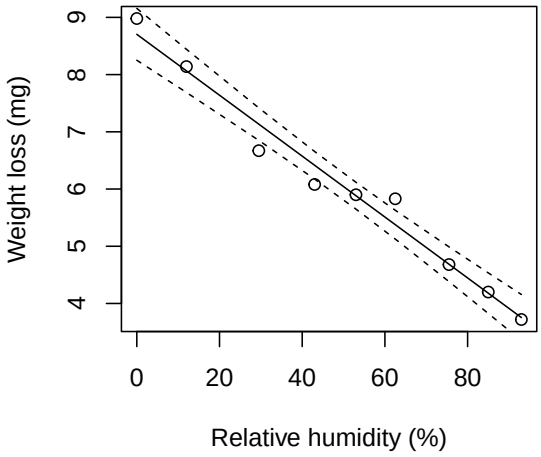



Figure 5.4: Regression line (solid) with upper and lower 95% CI (dashed)

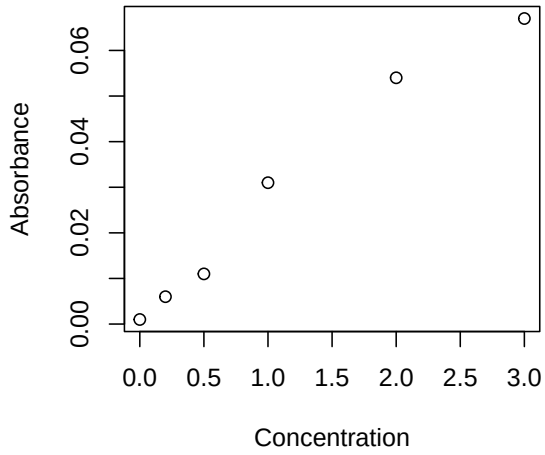
```
##       fit      lwr      upr
## 1 6.04292 5.303471 6.782368
```

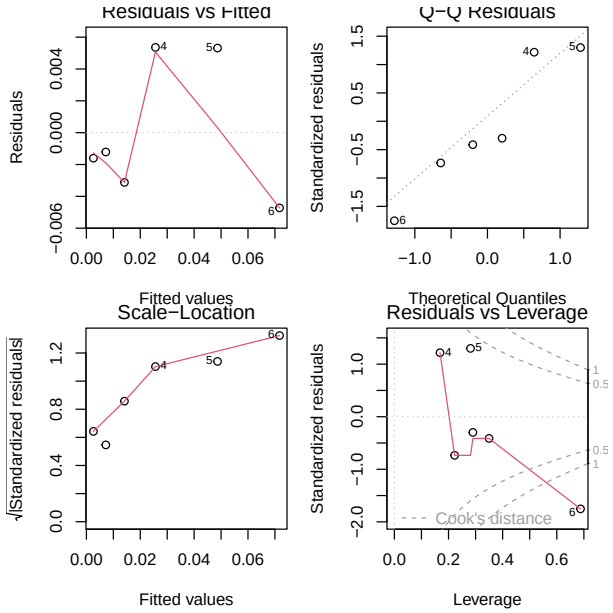
```
# or more fully:
predict(beetle_lm,
        newdata = data.frame(Humidity = seq(0, 100, by=25)),
        interval = "predict",
        level = 0.95)
```

```
##        fit      lwr      upr
## 1 8.704027 7.868990 9.539064
## 2 7.373474 6.608630 8.138317
## 3 6.042920 5.303471 6.782368
## 4 4.712366 3.949031 5.475701
## 5 3.381812 2.549540 4.214084
```

These are prediction intervals and they are broader than confidence intervals. The confidence intervals express your confidence about where the mean regression line would lie (if you went back in time and were somehow able to repeatedly sample from the same population). The prediction interval expresses your confidence about where the *values* would lie if you went back and repeatedly took an individual of the population at a given value of the predictor (e.g. 50% humidity). For more kicks, import PhosphateCalibration (1st year practical data) into R and duplicate the following plots and confidence intervals.

```
phosphate_df <- read_xlsx("data/Practical_6.xlsx", "PhosphateCalibration")
```




```
##
## Call:
## lm(formula = Absorbance ~ Concentration, data = phosphate_df)
##
## Residuals:
##         1         2         3         4         5         6
## -0.001605 -0.001213 -0.003125  0.005355  0.005314 -0.004726
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.002605   0.002853   0.913 0.412856
## Concentration 0.023040   0.001849  12.464 0.000238 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.004823 on 4 degrees of freedom
```

```
## Multiple R-squared:  0.9749, Adjusted R-squared:  0.9686
## F-statistic: 155.3 on 1 and 4 DF,  p-value: 0.0002383
```

*Q182. Are you happy with your model assumptions?*

*Q183. Write down the regression equation.*

*Q184. Determine the confidence interval for the regression line.*

*Q185. For a concentration of 0.75 units, what values would you expect (95 times in 100) to see from your experimental set-up?*

You should get:

```
##          fit         lwr        upr
## 1 0.01988519 0.005298129 0.03447225
```

---

## 5.3 Conclusions

Correlation is a measure of association between two variables. It is appropriate to use correlation to measure this association when one cannot or does not wish to assume that any relationship is causative. Pearson correlation coefficients should only be used where it is fair to assume (by looking at scatter plot) that the relationship is approximatley linear. Where linearity does not apply, attempt to transform one or both of the variables. Where there are outliers (that cannot be removed) or where one is uncertain about some of the data, then non-parametric ranked based correlation coefficients, such as the Spearman coefficient, should be used. As with GLMs, correlation analysis assumes that all points are independent of each other.

Linear regression is one of the most widely used statistical techniques. It is used to examine causal relationships, often where experimental manipulations are conducted. Regression is a general linear model and it lies within the generalised linear model family (GLMs). GLMs allow you to model data that is not normally distributed, including proportions (bounded by 0 and 1), or counts (bounded by 0). Using a GLM is a much better way of analysing these data compared with transforming the response variable or using non-parametric techniques. All members of the GLM family make the assumptions that measurements are independent of each other. Where this assumption fails you can use generalised linear mixed models (GLMMs). Extensions of simple linear regression include multiple regression which examines the influence of two or more continuous variables on a response variable.

# Chapter 6

# Multivariate analysis

So far, every model we've used has had a single response variable. Multivariate analyses expand this to allow the analysis of, for example, counts of each species within a community. Statistical routines for multivariate analysis are relatively new and have co-evolved with computational capacity over the last 50 years or so. Multivariate data typically take the form of a matrix of samples (each row is a sample) v 'features' (columns). Features may include things like faunal counts, chemical concentrations, or environmental conditions.

Multivariate analysis typically includes three stages: 1) data transformation or standardisation, 2) generation of a dissimilarity matrix and, 3) ordination (display) of that matrix. Each of these elements has numerous options, with advantages and disadvantages to each. Multivariate analysis are less inferential than most univariate approaches and implementation can feel like a 'black-art'! Statisticians are still arguing about the best way to approach multivariate analyses.

Multivariate data, such as species inventories, have been recorded for centuries and form a critical resource. Multivariate time series data are particularly important for assessing relative change that might be attributable to man's activities, for example in relation to climate or land-use change. The 'internet of things', the generation of multiple data-streams from the same station (e.g. a glider with CTD), and the development of bioinformatics (e.g. metabarcoding) all lend themselves to multivariate analyses.

More detail on the the material we cover here is given in Clarke et al. 2014. Change in marine communities: An approach to statistical analysis and interpretation.. If that link does not work, you can try navigating to the PRIMER-e downloads page, fill out your info to download the Manuals, then select "Change in Marine Communities". Unfortunately, the PRIMER software developed by the authors is not free. Instead, we'll use our gloriously free and open source R.

## 6.1 Multivariate analysis in R

For the non-metric NMDS workflow, we'll start by simulating some data. Then, we'll transform the data ('log' and fourth-root) and generate dissimilarity matrices from this transformed data. Finally, we'll plot it using NMDS. We'll generate diversity indices with our simulated data, then have a look at some real data (available via the package *vegan*).

For the PCA workflow, we'll have a look at some environmental data using PCA. Throughout this practical, we'll illustrate some coding techniques which will help you practice wrangling your data to make it suitable for analysis. As we've discussed, data wrangling is a time-consuming and crucial part of data analysis so familiarity with this is essential.

Recall from Section @ref{normal} that the ^ operator raises each element in a vector to a power. For example, say we create a vector `obs_values <- c(1, 10, 35)`. We can calculate a fourth-root transformation of the whole vector with `obs_values^(1/4)`. Remember from previous courses that this is identical to `sqrt(sqrt(obs_values))`.

```
library(vegan)
# these toy data duplicate those in the multivariate lecture.
worm_df <- data.frame(
  row.names = c("CE1", "CE2", "Ref1", "Ref2"),
  Capitella = c(1000, 1500, 10, 50),
  Malacoeros = c(500, 2000, 50, 25),
  Mysella = c(1, 0, 25, 30),
  Nucella = c(1, 0, 20, 15)
)
worm_df
```

```
##      Capitella Malacoeros Mysella Nucella
## CE1       1000        500       1       1
## CE2       1500       2000       0       0
## Ref1        10         50      25      20
## Ref2        50         25      30      15
```

```
# alt-log transformation: ifelse(x==0, 0, log(x))
worm_df_log <- decostand(worm_df, method = "log", logbase = 10)
worm_df_4rt <- worm_df^0.25
```

Take a look at `worm_df`, `worm_df_log`, and `worm_df_4rt` to be sure they make sense.

Next we use `vegdist()` to generate different distance matrices for the raw and transformed data. See `?vegdist` for more information. We'll use Bray-Curtis for the raw and $4^{\text{th}}$-root transformed data, and alternative Gower for the alt-log transformed data. See the multivariate lecture area on Brightspace for details on these algorithms. Feel free to investigate other combinations of data transformation and dissimilarity matrices.

```
vegdist(worm_df, method = "bray")
```

```
##            CE1       CE2      Ref1
## CE2  0.4002399
## Ref1 0.9228376 0.9667129
## Ref2 0.9050555 0.9585635 0.3333333
```

```
vegdist(worm_df_4rt, method = "bray")
```

```
##             CE1        CE2       Ref1
## CE2  0.18044721
## Ref1 0.39098221 0.59100112
## Ref2 0.36024122 0.55728021 0.08642723
```

```
vegdist(worm_df_log, method = "altGower")
```

```
##            CE1       CE2      Ref1
## CE2  0.6945378
## Ref1 1.4247425 2.1192803
## Ref2 1.3138181 2.0083559 0.3010300
```

*Q140. How many combinations of transformation and dissimilarity matrix do we have already?*

*Q141. Examine one or more of the dissimilarity matrices. Which sites are closer (smaller numbers) and which are futher apart (bigger numbers)? Does this align with an 'eyeballing' of the data? How has the data transformation changed the resultant dissimilarity matrix?*

Next we want to plot our dissimilarities, so we can easily visualise which sites are more similar/dissimilar to each other. We will use functions from *vegan* to plot the ordinations. Remember that you can run `?packageName` to learn more about any R package, typically with helpful examples and vignettes of the most useful applications of the package.

We have numerous options in relation to displaying the dissimiliarty matrices. We'll explore non-metric multiple dimensional scaling, abbreviated to NMDS, nMDS, nmMDS, or just MDS.

*Q142. Use `?vegdist` to see what alternative distance measures are available.*

The `metaMDS()` function calculates a dissimilarity matrix (as we did above) and produces an R object with all the information needed for plotting. It expects that samples are in rows and species (or features) are in columns. We can also specify the distance metric, the number of axes, whether to autotransform the data, and many other options. See `?metaMDS` for the default values and other available arguments.

```
ord_raw <- metaMDS(worm_df, distance = "bray", k = 2,
                   autotransform = FALSE, trace = FALSE)
ordiplot(ord_raw, choices = c(1, 2), display = "sites",
         type = "text", main = "NMDS: raw data, Bray-Curtis")
```

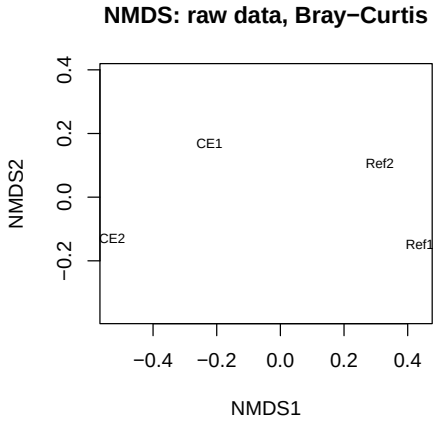



Figure 6.1: Simple pattern in example worm data.

Compare this with the $4^{\text{th}}$-root transformed data using the same distance metric.

```
ord_4rt <- metaMDS(worm_df_4rt, distance = "bray", k = 2,
                   autotransform = FALSE, trace = FALSE)
```

*Q143. Plot* `ord_4rt`*. How has the 4th root changed your data intepretation?*

*Q144. Examine your ordination(s) and interpret, cross referencing to the raw and transformed data. Are the patterns that you see in the data apparent on the ordination?*

You can include on your plot the species 'locations' (as determined by their correlation with the axes). This shows where the main associations are occurring.

```
# you can also plot the 'species' on the ordination,
ordiplot(ord_4rt, choices = c(1, 2), display = c("sites", "species"),
         type = "text", main = "NMDS, 4th-rt trans., Bray-Curtis")
```

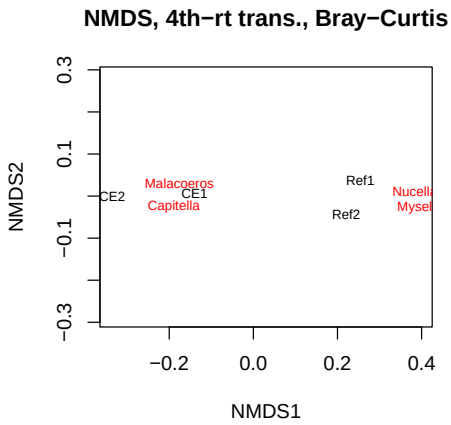



Figure 6.2: Species superimposed onto ordination, indicating species-site associations.

*Q145. Replot the untransformed data, but this time include the species. How has the transformation changed your interpretation of the data?*

Let's have a look at another simulated dataset.

```
comm_df <- data.frame(
  row.names = c("Dunst", "Creran", "Lismore", "Charl"),
  SpA = c(1, 20, 30, 40),
  SpB = c(11, 22, 50, 1),
  SpC = c(500, 40, 30, 20),
  SpD = c(10, 25, 35, 50),
  SpE = c(4, 3, 2, 1),
  SpF = c(40, 250, 1, 9)
)
comm_df
```

```
##         SpA SpB SpC SpD SpE SpF
## Dunst     1  11 500  10   4  40
## Creran   20  22  40  25   3 250
## Lismore  30  50  30  35   2   1
## Charl    40   1  20  50   1   9
```

```
ord_comm <- metaMDS(comm_df, distance = "bray", k = 2,
                    autotransform = FALSE, trace = FALSE)
ordiplot(ord_comm, choices = c(1, 2),
         display = c("sites", "species"), type = "text")
```

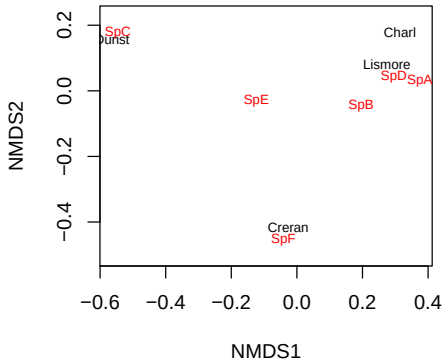



Figure 6.3: Ordination for a simulated dataset showing sites and species.

*Q146. Have a look at these raw data. What are the main trends? Which sites are more similar?*

These data are still much simpler than most 'real' data sets but it is still difficult to summarise the similarities and differences between stations. However, multivariate analyses help you in this process.

*Q147. Now forth-root transform these data, generate the new dissimilarity matrix and plot it.*

*Q148. What has the transformation done to your interpretation of differences between the Sites and Site x Species associations?*

*Q149. Add a title to your graph.*

---

## 6.2 Diversity indices

Prior to the development of the multivariate techniques you'll be using today, univariate indices were derived from multivariate data. A classic example of such a univariate measure in ecology is the Shannon-Wiener diversity index (a.k.a., Shannon's H). This index balances the *number* of species in a sample and the *relative abundance* of each species (where 'species' can once again be any sort of feature). Univariate measures of 'evenness' can also be derived from multivariate data and, when reporting species data, you may also wish to include species richness, which is just the number of species present regardless of their abundances.

We can use the `comm_df` dataset we invented to explore some diversity concepts.

*Q150. Looking at the `comm_df` data (by eye) and given the description above, which of the sites is assocaited with the lowest and highest diversity?*

```
shannon_H <- diversity(comm_df, "shannon", base = exp(1))
richness <- specnumber(comm_df)
barplot(shannon_H, names.arg = comm_df$Site, main = NULL, ylab = "Shannon's H")
```

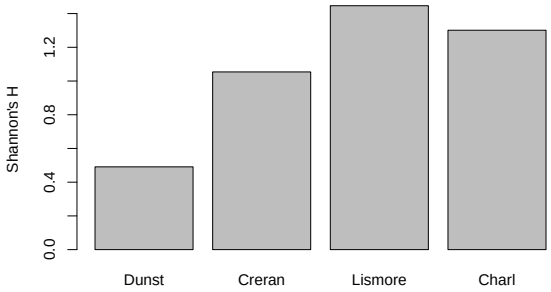



Figure 6.4: Shannon diversity

Try plotting `richness`.

*Q151. Do the plots correspond to what you expected?*

*Q152. What does `specnumber` do?*

*Q153. Change `comm_df` to introduce some very evenly distributed species across sites and some with extremes or absenses and see how this effects diversity and richness.*

*Q154. How could you 'counter' any extremes (as in superabundant taxa) in the raw count data that you've generated? Try your idea.*

*Q155. Which description (diversity or richness) is 'best' for describing your multivariate data? How does this compare to NMDS?*

## 6.3 NMDS on real data

Now you've been introduced to NMDS and diversity indices, with 'fake' data, you are in a better position to interpret real data. Using simulated (or at least simple) data to learn new statistical techniques is usually the best approach because it gives you the opportunity to get a better sense for how the algorithms work (and to be sure your code is free from bugs!).

Have a look at the `varespec` and `varechem` datasets included in the *vegan* package. I've reproduced the examples below:

```
data(varespec) # ?varespec -- percent cover of 44 spp in lichen pastures
data(varechem) # ?varechem -- associated soil charecteristics
ord_vare <- metaMDS(varespec^0.25, distance = "bray",
                    trace = FALSE, autotransform = FALSE)
```

```
par(mfrow=c(1,3), mar=c(4,4,1,1))
ordiplot(ord_vare, choices = c(1, 2), display = "sites", type = "text")
ordiplot(ord_vare, choices = c(1, 2), display = "species", type = "text")
ordiplot(ord_vare, choices = c(1, 2), display = c("sites", "species"),
```

```
  type = "text")
# You can superimpose environmental variables onto NMDS-ordinations.
ef <- envfit(ord_vare, varechem) #
plot(ef, p.max = 0.1, col = "green") # overlay environmental variables
plot(ef, p.max = 0.01, col = "blue") # subset based on p
```

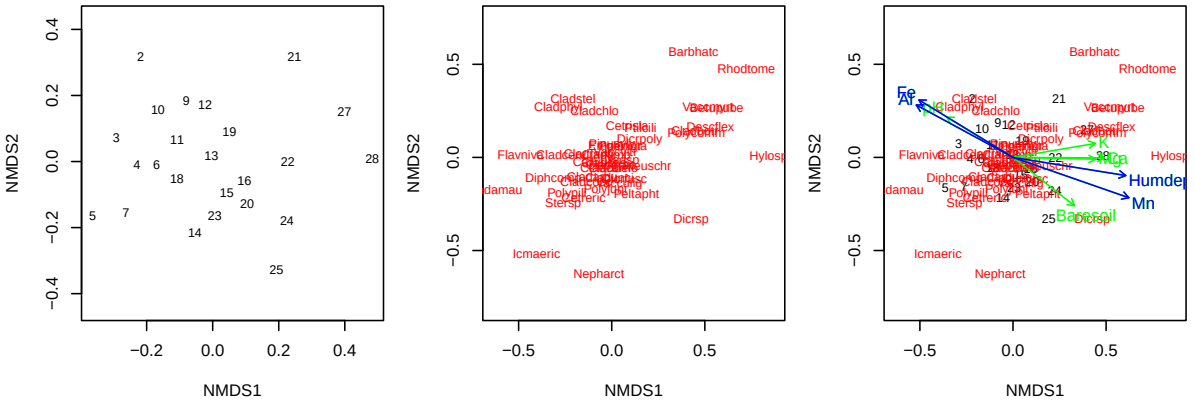



Figure 6.5: vare- data from vegan showing sites only, species only, and both plus environmental overlays.

See how the multivariate analysis has taken all those data, both species and environmental, and 'communicated' them in one single figure? You can see numerous relationships (both positive and negative) in this figure, and species-site-environment associations. It also illustrates a potential challenge with multivariate ordinations. It is very easy to get cluttered and overloaded. There is no easy way around this, though there is further help in these packages.

You are in charge of the analysis. You can change the emphasis and elements of the message depending on your data transformation, dissimilarity metric, and ordination technique (hence 'black art'). There is no absolutely 'correct' way to go about multivariate stats, so different statisticians will have their favoured approaches and methods.

Note: Some functions (e.g. `metaMDS()`) default to autotransform your data if the function thinks it is necessary. This can be useful but, in scientific reports, you must specify what transformations you used. Here you don't know what the function applies as it depends on the data, but it could be the square or fourth-root and/or Wisconsin transformation (which is a double standardisation). My advice is only to use transformations that you specify.

---

## 6.4 Principal components analysis (PCA)

PCA is a long-established multivariate technique that is often applied to 'environmental' data rather than 'count' data. Environmental data is, usually, quite different from species count data in that most environmental parameters (e.g. metal concentrations) are present, at least to some degree. This contrasts to species data where many species are often absent (zeros). These zero counts would lead to problems if analysed using PCA, since PCA would 'think' that sites that shared lots of 'absences' were more similar, which is not necessarily desirable.

With environmental data, such as temperature, light, and concentration, zeros tend to be less prevalent. However, environmental data (e.g. that describing your sampling location in space

and time) might include all manner of different variables on different scales (e.g., radient flux in lumens, temperature in C, nutrient/contaminant concentrations in mg/l). What you wouldn't wish to see is your arbitrary choice of measurement unit (e.g., C or K) having any influence on your analysis such that a variable is given more weight simply because the numbers are larger.

Instead, we want all of our variables to be treated 'equally'. You can do all this by setting `pcomp(..., scale=TRUE)`. Scaling means that each measurement is expressed in units of standard deviation (a Z-score!!). Usually it is desirable to center the data as well by subtracting the mean. Centering and scaling means that each of the environmental variables is of 'equal importance' regardless of the magnitude of the raw values.

A basic, and very friendly, introduction to PCA is given in Chapter 4 of Clarke et al. 2014.

The data set we'll look at here is from SAMS Professor Tom Wilding's PhD thesis. He set up an experiment to examine the relative leach_dfing of trace metals from concrete, granite, and a control (artificial seawater). Concrete contains cement which is enriched in vanadium and molybdenum, and these elements could leach_df out in dangerous amounts. Granite, the main constituent of this concrete, might also leach_df some trace elements. He suspended concrete and granite powder in artificial water, constantly agitated it, and measured the leach_dfate concentrations over 100 days Wilding and Sayer 2002.

```
library(readxl)
library(tidyverse)
leach_df <- read_excel("data/practical_5.xlsx", sheet = "Leaching") |>
  mutate(Treat_abbr = factor(Treat, # abbreviate for cleaner plotting
                             levels = c("concrete", "control", "granite"),
                             labels = c("conc", "ctrl", "gran")),
         Treat_day = paste(Treat_abbr, Day, sep="_"), # treat + days in exprmnt
         Conc = signif(Conc)) |>
  select(Treat_day, Element, Conc) # remove columns that aren't of use,
#summary(leach_df)

# not dominated by zeros; try other values (e.g. <10)
table(leach_df$Conc == 0)
```

```
##
## FALSE  TRUE
##   143     4
```

```
mean(leach_df$Conc == 0) # recall that R treats T/F as 1/0
```

```
## [1] 0.02721088
```

```
par(mfrow=c(1,1))
hist(leach_df$Conc, main = NULL)
```

Next, we need to re-organise the data into a wider format so that each element is a column and each row is a sample.

```
leach_df_wide <- leach_df |>
  pivot_wider(names_from="Element", values_from="Conc")
```

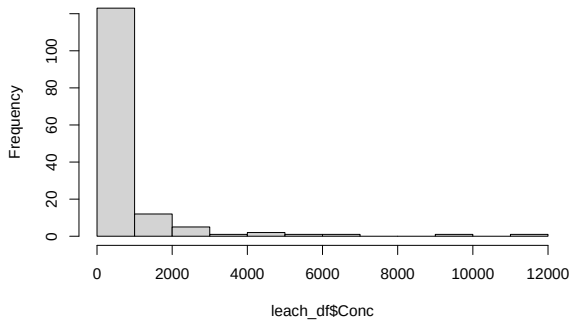



Figure 6.6: Histogram of leaching data (raw) illustrating the wide range of values and skew.

The column `Treat_day` is coded as `trt_day`, where `trt` is the 4 letter code indicating treatment type and `day` is the number of days elapsed in the experiment (one of 1, 4, 7, 17, 32, 50 or 100). So `gran_32` means the granite treatment sampled at day 32.

We can calculate the principal components using `prcomp()`, subsetting the dataframe to give only the columns with element concentrations (i.e., removing `Treat_day`, which is the first column). We'll also set the arguments for centering and scaling to `TRUE`.

```
PCA_leach <- prcomp(leach_df_wide[, -1], center = TRUE, scale = TRUE)
screeplot(PCA_leach, main = NULL, ylab = "Relative variation explained")
```

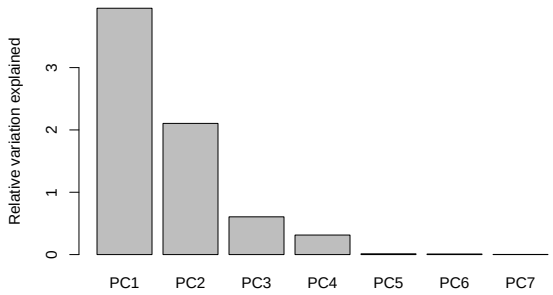



Figure 6.7: Scree plot showing the variance explained by each principal component.

```
# take a look at PCA_leach
PCA_leach
str(PCA_leach)
```

You can see from the screeplot that the amount variance explained declines with principal component as expected and that there is very little variation left afer 3 principal components. That is, nearly all of the variation in the dataset is captured by PC1, PC2, and PC3. In this case, PCA has essentially solved the 'curse of dimensionality' by successfully reducing 7 dimensional data to about three.

Data transformations are critical to PCA analysis, as they are with NMDS. In most PCAs you center and standardize your data so that each column is on the same scale.

We can plot our results using `biplot()` which has some helpful defaults including labels for the samples (`Treat_day`) and the correlation strength of each element with PC1 and PC2:

```
biplot(PCA_leach, xlabs = leach_df_wide$Treat_day, cex = 0.75)
```

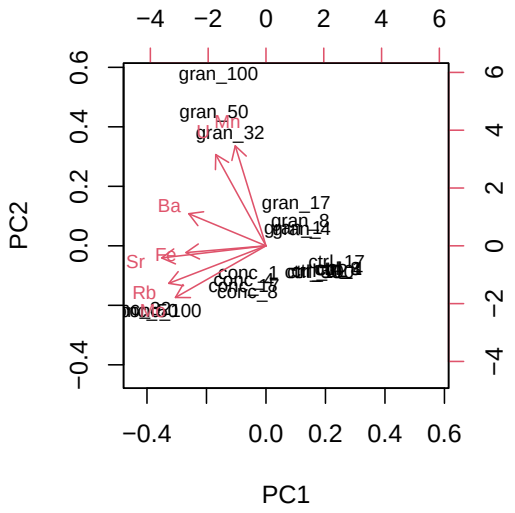



Figure 6.8: PCA of the complete dataset, illustrating the common challenge of overplotting.

To plot more than 2 dimensions you could use a 3D plot, but these are frankly difficult to interpret since it is reduced *back* to 2D on a page or computer screen. Another option is to use colour for the 3rd axis. Here's an example using `ggplot2`. Adding arrows showing the loadings (see below) is possible, but more work.

```
PCA_leach$x |>
  as_tibble() |>
  mutate(Treat_day = leach_df_wide$Treat_day) |>
  ggplot(aes(PC1, PC2, colour = PC3, label = Treat_day)) +
  geom_label(size = 3, fill = NA) +
  scale_colour_gradient2(mid = "cornsilk2") +
  theme_bw()
```

The ordination plots the relative positions (in terms of similiarity) of the samples. There are numerous label overlaps making the interpretation of the ordination difficult. If you were producing this for publication you would need to sort this out.

*Q156. Which elements are positivley associated with granite and concrete, particulalry after longer periods of leaching?*

As is typical, overlapping points make interpretation more difficult. There are elegant solutions to this (in terms of labelling) but for now, we'll split the data and analyse it separately.

We'll need more data wrangling to split it efficiently and we'll use `grepl()`, which identifies a pattern within a character using a regular expression (a.k.a., regex), returning a `TRUE` or `FALSE` for each element in the character vector. Here, we'll filter the dataframe to only include rows where `Treat_day` contains `conc` or `gran`. Remember the 'or' operator `|`?

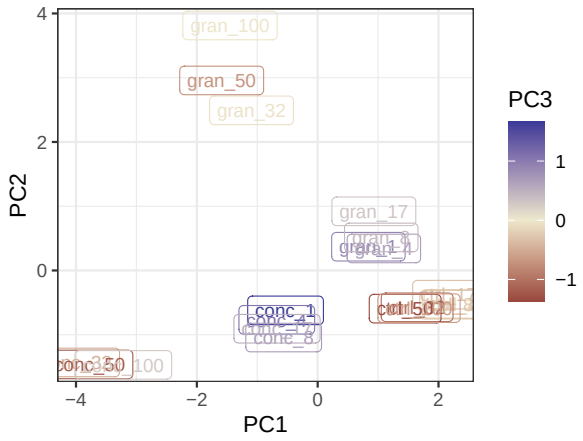



Figure 6.9: PCA of the complete dataset using ggplot2.

```
#?grepl
#cbind(leach_df_wide$Treat_day, grepl("conc|gran", leach_df_wide$Treat_day))
leach_df_trts <- leach_df_wide |>
  filter(grepl("conc|gran", Treat_day))
PCA_trts <- prcomp(leach_df_trts[, -1], scale = TRUE, center = TRUE)
biplot(PCA_trts, xlabs = leach_df_trts$Treat_day)
```

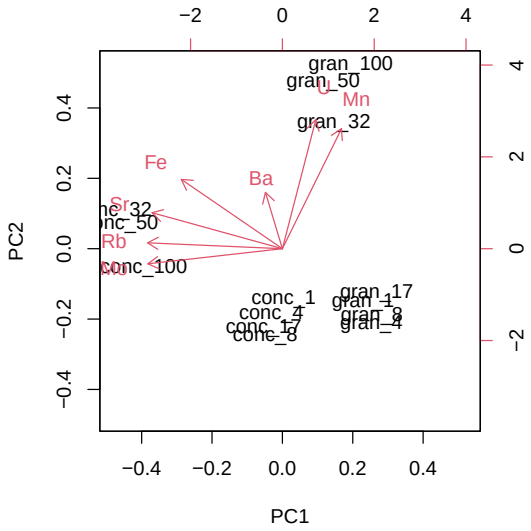



Figure 6.10: PCA of the concrete and granite, illustrating temporal and treatment differences.

*Q157. Repeat the analysis, but set* `scale = FALSE`*. Which element now seems to dominate the analysis? Explain what you see.*

A PCA is normally reported with the proportion of the variation explained by each of the principal components (and the cumulative proportion). If the cumulative proportion for the 1st two PCs is high, then your 2D (PC1 and PC2) ordination is a good representation of the similarities between samples. In that sense a high cumulative proportion is analogous to a low stress (for NMDS).

Let's have a look at a summary of the principal components.

```
PCA_leach_summary <- summary(PCA_leach)
#PCA_leach_summary
#str(PCA_leach_summary)
PCA_leach_summary$importance[, 1:4] # extract only PC1-4
```

```
##                             PC1      PC2       PC3       PC4
## Standard deviation     1.988091 1.450775 0.7782675 0.5599535
## Proportion of Variance 0.564640 0.300680 0.0865300 0.0447900
## Cumulative Proportion  0.564640 0.865320 0.9518500 0.9966400
```

As you can see, PC1, PC2, and PC3 capture more than 95% of the variance in our data. Adding PC4 brings that up above 99%.

Finally, we can look at factor loadings. This is a measure of how each feature (metal concentration in this case) relates to the principal components. In PCA, the principal components are sequentially 'less' influential since they describe increasingly smaller amounts of variation. By centering and standardising your response variables, you can assess the relative importance of each in driving the patterns you observe in your ordination. The magnitude is what we're interested in rather than the sign.

In an object created by `prcomp()`, the loadings are stored as `.$rotation`.

```
# ?prcomp
# str(PCA_leach)
signif(PCA_leach$rotation[, 1:5], 3) # factor loadings for PC1-5
```

```
##       PC1     PC2     PC3    PC4      PC5
## Ba -0.367  0.2090  0.6510 -0.619 -0.00405
## Fe -0.380 -0.0440 -0.7370 -0.553  0.01420
## Mn -0.146  0.6500 -0.0963  0.211  0.67300
## Mo -0.430 -0.3360  0.0795  0.283 -0.09810
## Rb -0.462 -0.2440  0.0327  0.305  0.21600
## Sr -0.495 -0.0788  0.0663  0.168  0.01510
## U  -0.238  0.5940 -0.1110  0.254 -0.70000
```

Here you can see that Sr, Rb, Mo are relatively 'important' in driving the multivariate pattern you've observed (i.e., high absoluate values on PC1) while Mn and U have high values for PC2 (you could say that PC2 accounts for Mn and U).

---

## 6.5 Conclusions

I will let you write your own summary and conclusions from this practical as this will help you consolidate your understanding. You may wish to consider the following: the nature of multivariate vs. univariate data, types and characteristics of multivariate data (e.g. counts vs. environmental data), the analytical methods introduced here for dealing with data of different types, the purpose and effect of transformations and standardisations and where they are used, the use of summary metrics (e.g. diversity metrics) and their advantages and disadvantages. You should also be familiar with the interpretation of simple ordinations (both PCA and NMDS) and the concept of stress and 'proportion explained' (for NMDS and PCA respectively).

# Chapter 7

# Appendix

## 7.1 Probability

### 7.1.1 Probability for equally likely outcomes

The probability of event $A$ occuring is $P(A) = \frac{r}{n}$, where $n$ is the number of trials and $r$ is the number of trials during which $A$ occurred.

### 7.1.2 Multiplication and addition

The general case of the multiplication rule is that, for two events $A$ and $B$:

$P(A\&B) = P(A) * P(B|A)$, where $P(B|A)$ is the probability that $B$ occurs given than $A$ has already occurred.

The Addition Law states that, for two events $A$ and $B$:

$P(A \textit{ or } B \textit{ or } A\&B) = P(A) + P(B) - P(A\&B)$

and

$P(A \textit{ or } B \textit{ but not } A\&B) = P(A) + P(B) - 2 * P(A\&B)$

and when $A$ and $B$ are mutually exclusive, then:

$P(A \textit{ or } B) = P(A) + P(B)$

### 7.1.3 Bayes' theorem

$P(A|B) = \frac{P(B|A)*P(A)}{P(B)}$, where $P(B) = P(A) * P(B|A) + P(A') * P(B|A')$.

## 7.2 Univariate statistics

### 7.2.1 Mean

`mean()`

The mean is calculated as $\bar{y} = \frac{\sum y}{n}$.

For measures of central tendency and dispersion, we use greek letters to refer to population values and latin letters to refer to samples:

| | Population | Sample |
|---|---|---|
| Mean | $\mu$ | $\bar{y}$ |
| Variance | $\sigma^2$ | $s^2$ |
| Standard deviation | $\sigma$ | $s$ |

### 7.2.2 Median, quartiles and adjacent values

`median()`, `quantile()`, `IQR()`

With data ordered by rank, the median value is the middle value or the $(\frac{n+1}{2})^{th}$ value, the lower quartile, $Q1$, is the $(\frac{n+1}{4})^{th}$ value, the upper quartile, $Q3$, is the $(\frac{3*(n+1)}{2})^{th}$ value. The inter-quartile range is $IQR = Q3–Q1$. The upper adjacent value is the upper value that is less than $Q3 + (1.5 * IQR)$. The lower adjacent value is the lower value that is more than the $Q1–(1.5 * IQR)$. Outliers are defined as values that lie outside the range $Q1 - 1.5 * IQR$ or $Q3 + 1.5 * IQR$.

### 7.2.3 The sum of squares

The sum of squares is given by $SS = \sum y^2 - \frac{(\sum y)^2}{n}$ where $n$ is the number of observations.

### 7.2.4 Measures of dispersion

`var()`, `sd()`, `length()`

When dealing with **populations** the following formulae are used:

**Variance:** $\sigma^2 = \frac{SS}{n}$

**Standard deviation:** $\sigma = \sqrt{\frac{SS}{n}} = \sqrt{\sigma^2}$

When dealing with **samples** the following formulae are used:

**Variance:** $s^2 = \frac{SS}{n-1}$

**Standard deviation:** $s = \sqrt{\frac{SS}{n-1}}$

## 7.3 The Binomial distribution

`dbinom()`, `pbinom()`, `qbinom()`, `rbinom()`

The binomial distribution describes the expected distribution for two mutually exclusive outcomes. The formula is given by:
$P(x) = \frac{n!}{x!(n-x)!} p^x q^{n-x}$

where $P(x)$ is the probability of $x$ 'successes' occurring, $n$ is the number of trials, $p$ is the probability of $x$ in a single trial, and $q$ is the probability of *not* $x$ in a single trial with $p + q = 1$.

A binomially distributed variable has **mean** $= n * p$ and **variance** $= n * p * q$.

## 7.4 The Poisson distribution

`dpois()`, `ppois()`, `qpois()`, `rpois()`

The Poisson distribution for a sample is defined as follows:

$P(x) = \frac{\bar{y}^x e^{-\bar{y}}}{x!}$

where $\bar{x}$ is the sample mean and $x$ is the value (count) of interest.

## 7.5 Z scores

`scale()`

The *standard normal distribution* is $Norm(\mu = 0, \sigma = 1)$. Z scores are used to convert any normal distribution to the standard normal distribution:

$z = \frac{y-\mu}{\sigma}$

where $y$ is the value, $\mu$ is the mean of the population and $\sigma$ is the standard deviation of the population.

## 7.6 Samples taken from a population

`sd()`, `sqrt()`, `length()`, `mean()`, `sqrt()`

### 7.6.1 Standard error of the mean

The *standard error* is the standard deviation of sample means:

$SEM = \sigma_{\bar{y}} = \frac{\sigma}{\sqrt{n}}$

where $\sigma$ is the population standard deviation and $n$ is the sample size. If we are testing a sample taken from a population *of known population mean and population standard deviation* we use the formula:

$z = \frac{\bar{y}-\mu}{\sigma_{\bar{y}}}$

where $\bar{y}$ is a particular sample mean, $\mu$ is the population mean, and $\sigma_{\bar{y}}$ is the standard error of the mean.

## 7.7 The t distribution

`dt()`, `pt()`, `qt()`, `rt()`

The t-distribution describes the distribution of $T$ statistics expected under the null hypothesis.

### 7.7.1 The one sample t-test

`t.test()`

The T statistic is calculated as:

$T = \frac{\bar{y}-\mu}{s_{\bar{y}}}$

Where $\bar{y}$ is the sample mean, $\mu$ is the hypothesized population mean, and $s_{\bar{y}}$ is the standard error such that $s_{\bar{y}} = \frac{s}{\sqrt{n}}$ with sample standard deviation $s$.

Use `t.test()`, or find the p-value associated with your T with `pt(T, df-1)`.

### 7.7.2 Confidence intervals for sample means

`t.test()`, `qt()`, `mean()`, `sd()`, `length()`, `sqrt()`

The 95% CI for a mean is calculated as:

$\bar{y} \pm t_{\alpha/2, df=n-1} * s_{\bar{y}}$

## 7.8 ANOVA

`anova(lm())`, `aov()`

An ANOVA table contains the following:

| Source of variation | Sum of Squares | Degrees of Freedom | Mean Square | F |
|---|---|---|---|---|
| Between groups | $SS_{groups}$ | $df_{groups}$ | $\frac{SS_{groups}}{df_{groups}}$ | $\frac{MS_{groups}}{MS_{error}}$ |
| Error | $SS_{error}$ | $df_{error}$ | $\frac{SS_{error}}{df_{error}}$ | |
| Total | $SS_{total}$ | $df_{total}$ | | |

with:

$SS_{groups} = \sum n_j(\bar{y_j} - \bar{\bar{y}})^2$

$SS_{error} = \sum (y_{ij} - \bar{y_j})^2$

$SS_{total} = SS_{groups} + SS_{error} = \sum (y_{ij} - \bar{\bar{y}})^2$

where $y_{ij}$ is the $i^{th}$ observation in group $j$, $\bar{y_j}$ is the mean for group $j$, and $\bar{\bar{y}}$ is the grand mean. With $N$ total observations, $n_j$ observations per group, and $k$ groups, the degrees of freedom are:

$df_{groups} = k - 1$

$df_{error} = N - k = k(n_j - 1)$

$df_{total} = df_{groups} + df_{error} = N - 1$

The critical value (CV) for samples of equal size is given as:

$CV = q\sqrt{\frac{MS_{error}}{n}}$

where $q$ either comes from a table online (the *studentized range distribution*), or from `qtukey(1-alpha, k, df_error)`.

## 7.9 Regression & correlation

`lm()`, `cor()`, `confint()`, `predict()`

We do not bother calculating regression coefficients by hand.