H2 Data Science Practicals

Tim Szewczyk (v1: Tom Wilding)

2025-09-01

Table of contents

Preface

Practical sessions

In these practicals, you will apply concepts you learn throughout the H2 Data Science course. The questions throughout 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 truly one of the most important courses you'll take!

You will use datasets provided on Brightspace, datasets included in R, and datasets you simulate yourself.

Please read through the practical before the class. Each session is scheduled for 3.5h with a recommended break in the middle. You will work through a series of coding exercises and questions which are not assessed or marked, but may appear in the assessments. You should complete all the material in each practical.

The practical booklet is provided online and as a pdf on Brightspace. The web version has additional features that are not possible with pdfs and is the recommended interface. The underlying files (lightly edited) are available on Brightspace.

R setup

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. There are many resources available, including:

- Your notes and course material from H1 Maths and Data Science (also on Brightspace > Practicals)
- Courses such as those from Software Carpentry or Swirl
- Online videos such as IQUIT R

Use these resources as needed to complement, revise, and reinforce the concepts you'll learn during this course.

During the practicals, use 'copy-and-paste' thoughtfully. R is best learned through your fingers, and working through errors, though frustrating, is an essential skill.

RStudio Projects

Working in a 'Project' within RStudio is the best way to avoid working directory complications. This is a very common source of frustration and errors. It also helps organize work to ensure that necessary scripts, documents, data, and output are all contained within the same folder. Other benefits include tracking your history of R commands and integrating cleanly with more sophisticated version control (e.g., git – more to come later).

Create an R Project

We will create an R Project for the semester. During each practical, you should work within this project. To set up a new project as we need it:

1. Sign into OneDrive, then open RStudio
2. Select *File* > *New Project…*
3. Choose *New Directory* > *New Project*
4. Set the *Directory name* as **H2_DataScience** and use the *Browse* button by *Create project as a subdirectory of:* to set a convenient location in your OneDrive folder. Click 'Create Project'.
5. In the *Files* panel in RStudio, click the *New Folder* button and create a folder called **data**.
6. On Brightspace, download H2DS_practicalData.xlsx in *Practicals* and move it into your newly created **data** folder.
7. On Brightspace, download the files in *Practicals* > *Weekly qmd* and move them into the **H2_DataScience** folder.
8. Close RStudio (to learn how to open appropriately)

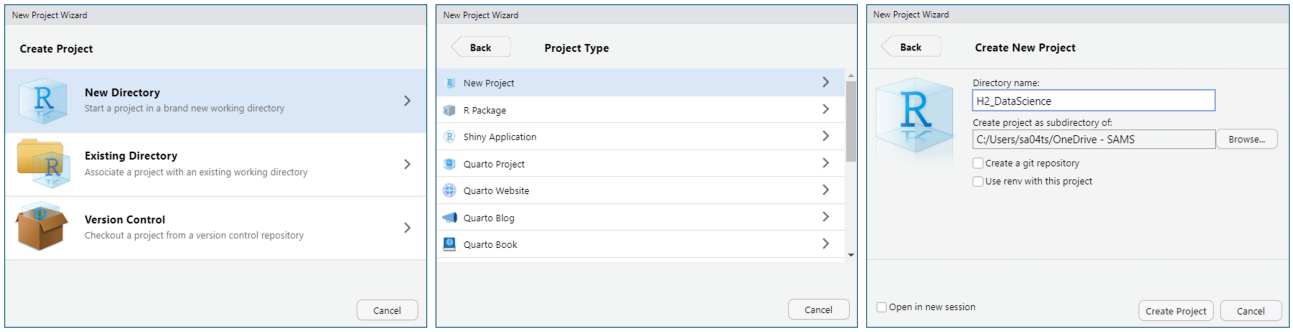

Figure 1: Setting up a new RStudio project.

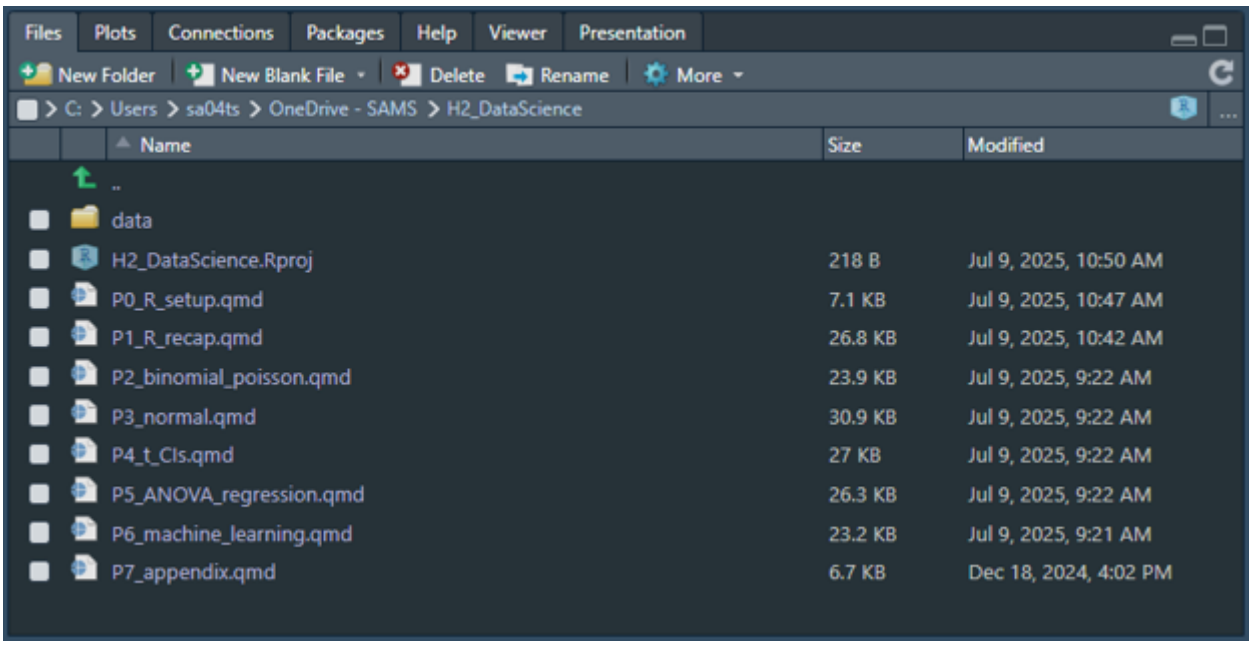

Figure 2: H2_DataScience directory structure.

The code in the practicals assumes this organization. If you choose to put your data files elsewhere, you will need to update the scripts accordingly. You're now prepared and organized for a semester in R!

Opening R

When using a project, you should open R via the .proj file – *not* the script you plan to work on.

> I repeat: **Open H2_DataScience.Rproj** instead of the .R or .qmd file you plan to work on!

This is the start-up process you should use for the practicals:

1. Open Windows Explorer (or Finder on Mac) and find your **H2_DataScience** folder.
2. Double click on **H2_DataScience.Rproj**. This opens your R project with the working directory set to that folder. The working directory is shown at the top of the Console pane, and you can check it with `getwd()`. This is where R is 'situated' when loading or saving files.
3. In the *Files* panel, open the .qmd file for the week (or your .R file if you prefer).

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).

R packages and libraries

R packages are collections of functions, custom data structures, and datasets that are developed by the user base. A new installation of R includes many useful 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.

View 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 by running `functionName` without parentheses.

R scripts (.R) vs Quarto documents (.qmd)

In H1, you used R scripts (.R). These are just text files. The extension tells your computer to associate them with R, and also lets you run lines of code with 'ctrl + enter' and other convenient things in R and RStudio.

Quarto documents (.qmd) are also text files. However, RStudio interprets the text differently, allowing you to intersperse written prose, figures, references, R code, and output in a single document (similar to a jupyter notebook or a live script in Matlab). Code is marked as "chunks" and you can specify options for how the code and output are displayed. The text uses markdown formatting, which allows all sorts of formatting (headers, bold, italic, equations, hyperlinks, tables, images…). RStudio can render a .qmd file into many other formats (e.g., pdf, docx, html, epub…).

Code chunks can be added with the green button with a '(+)c' on the top right of a .qmd document in RStudio. Code chunks look like this:

````
```{r}
# This is a code chunk
a <- 1:3
a
```
````

```
[1] 1 2 3
```

Run the full block with the green 'play' button at the top right of the block. The output from the code is shown just below the block.

For code-heavy work, Quarto documents are a handy way to produce nicely formatted output without the hassle of copying and pasting code, output, and figures into, e.g., a word document. There are many guides and tutorials online.

This manual is written as a Quarto book. **Versions of the .qmd files for each practical are available on Brightspace** to make things easier for you. You should now have these downloaded and saved in your project directory.

Note that the project will be submitted as a .qmd file with a version rendered to html.

Writing R code

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. See the full style guide for more.

A few key points:

- Use `<-` to *assign* a value to an object. You may see `=`, which works, but is not preferred.
- Use `#` to write a comment which R will ignore.
- Use spaces to make your code legible: `a <- c(1, 2, 3)`.
- Avoid spaces in column names or file names 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.
- Run the line of code where your cursor is (or everything you've selected) with ctrl + enter

Chapter 1

R recap

Statistics can be divided into two broad categories: **descriptive statistics**, which describe or summarise data, and **inferential statistics**, which allow us to infer something about a population from a sample.

This session focuses on how to appropriately display and summarise data (i.e., descriptive stats). You should already be aware of several techniques for displaying data (e.g., bar charts, histograms, scatter plots). When and how to use these techniques is one focus of today.

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 (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', drawing 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, in grey scale, or by colour-blind or visually impaired readers. Colour scales such as those available from ColorBrewer or viridis are designed with this in mind.

It's best practice to load necessary packages at the top of your document. Today we'll use the *tidyverse* package, which is actually a collection of packages. First, you'll need to install it with `install.packages("tidyverse")`. Installation needs to be done once per machine, but loading is needed each time you re-open R.

```
library(tidyverse)
library(readxl) # installed with tidyverse, but not loaded in library(tidyverse)
set.seed(2025) # for fully reproducible code
```

1.1 Basic data exploration

1.1.1 Object structure

We will mostly work with dataframes. A `data.frame` is a 2D rectangular object 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 `data.frame` `cars` gives the max speed and stopping distance for cars built in the early 20th century. It is already available in your R session. We will use `cars` to demonstrate a few basic programming and statistical concepts.

```
# functions for basic details of objects
str(cars) # structure overview
class(cars) # object class
names(cars) # column names
head(cars) # first few rows
```

```
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?
```

1.1.2 Subsetting, renaming, and rearranging

There are several ways to access subsets of a `data.frame`:

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

```
cars$speed # whole column
cars[["speed"]] # whole column
```

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

```
[1] 4
```

```
cars[1:5, 1] # rows 1-5, column 1
```

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

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

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

We can also change column names. For illustration, let's make a copy of the `data.frame` to do that.

```
cars2 <- cars
names(cars2)
```

```
[1] "speed" "dist"
```

```
names(cars2)[1] <- "speed_mph" # change first column name
names(cars2)
```

```
[1] "speed_mph" "dist"
```

```
names(cars2) <- c("speed_mph", "dist_ft") # change both column names
```

Rearranging and duplicating columns is also easy.

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

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

```
head(cars2[, 2:1], 2) # rearrange columns
```

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

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

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

```
cars3 <- cars3[, 1:2] # 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' from your R environment
```

> **i** Recall that you must *assign* the results of an operation (`<-`) to save it. For example, running `cars2[, 2:1]` will display the results, but `cars2 <- cars2[, 2:1]` will *overwrite* `cars2` in your R environment.

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

```
cars_fast <- cars2[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)
```

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

1.1.3 NAs and summary

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 `TRUE` if an element *is* `NA`, and `FALSE` if an element *is not* `NA`.

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

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

```
sum(is.na(cars2$speed_mph)) # how many are NA?
```

```
[1] 0
```

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

```
[1] 0
```

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 are confident that your `data.frame` 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. The appropriate method depends on 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').

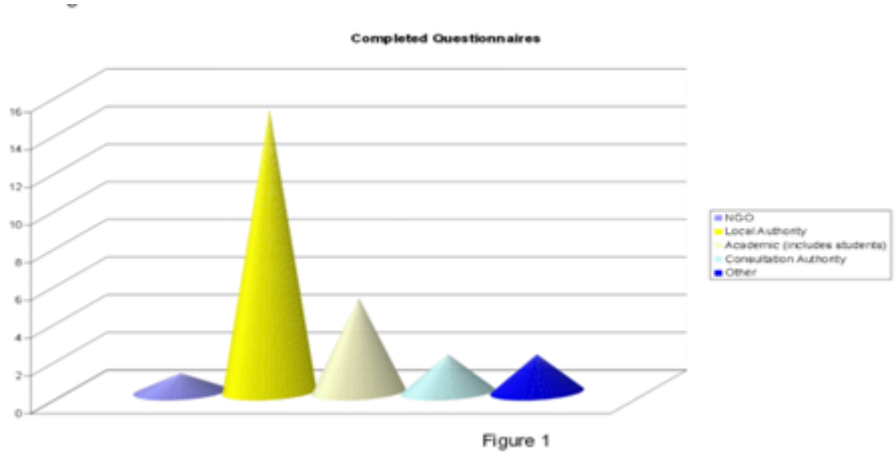

Figure 1.1: An example of a terrible graphic, as published in a Scottish government report.

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

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

1.2.1 Scatter plots

The scatter plot is used to plot two continuous variables against each other. It is commonly used for analyses like correlation or linear regression. The `plot()` function in R will create a scatter plot if given two numeric variables. There are two options for specifying the variables.

Using `data_df`, the same plot can be created with either syntax:

```
data_df <- data.frame(x=1:3, y=4:6)
plot(data_df$x, data_df$y) # two vectors: x, y
plot(y ~ x, data=data_df) # columns in dataframe: formula y ~ x
```

There are many options for modifying the output of `plot()`.

```
par(mfrow=c(1,3)) # set the plot window to show 1 row, 3 columns

# plot(response ~ 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=1:20, main="'pch' symbols 1 to 20")
```

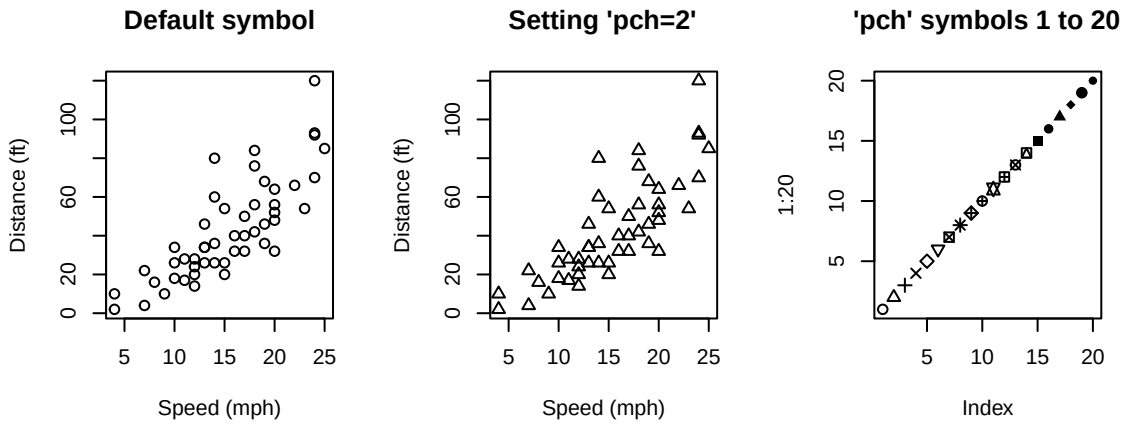

Figure 1.3: Symbol options.

```
par(mfrow=c(1,1)) # reset to a single panel
```

Q. 1.1

With the `cars` dataset, plot stopping distance by speed for only those cars with a speed less than or equal to 15 mph.

Q. 1.2

Use the `plot` help page to add an appropriate title to your plot.

Q. 1.3

`?points` opens the help page for points. Search it for 'pch' and change the symbol in your plot.

View solution

```
plot(dist ~ speed, data=cars[cars$speed <= 15, ], pch=2,
     xlab="Speed (mph)", ylab="Distance (ft)",
     main="Cars with max speed <= 15 mph")
```

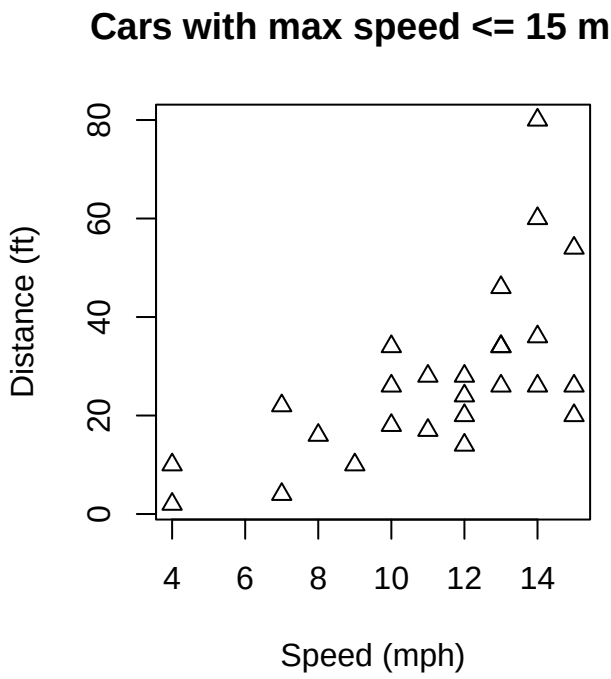

Figure 1.4: Stopping distance by speed.

1.2.2 Boxplots

Box plots are used to summarise a continuous variable by levels of a factor. We will use the `mtcars` dataset to illustrate this. You can learn about these data with `?mtcars`.

Explore the `data.frame` using the strategies covered above. Which variables are categorical? Which are continuous?

```
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
# See examples at bottom of the help page
```

Q. 1.4

Reproduce the plot shown in Figure 1.5 (assume 1 gallon = 4.5 L). You will need to generate a new variable (miles per litre) and label your box plot 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)`).

View solution

```
mtcars2 <- mtcars
mtcars2$mpl <- mtcars2$mpg / 4.5

boxplot(mpl ~ cyl, data=mtcars2, xlab="Cylinders", ylab="Miles per litre")
```

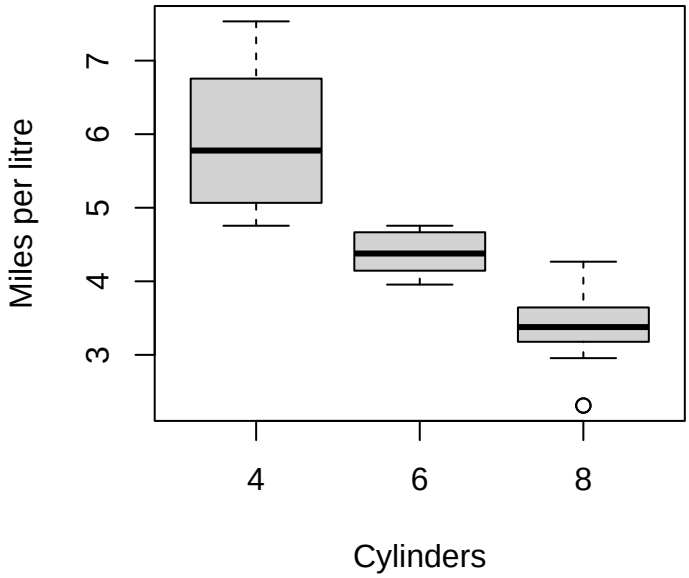

Figure 1.5: Boxplot showing miles per litre vs. number of carburetors

Q. 1.5

Use `?boxplot` to investigate what the box and whiskers actually represent.

Note that box plots are not the most visually intuitive. Packages like `ggplot2` (and extensions) make alternatives like those in Figure 1.6 simple to produce. We will cover some of these later.

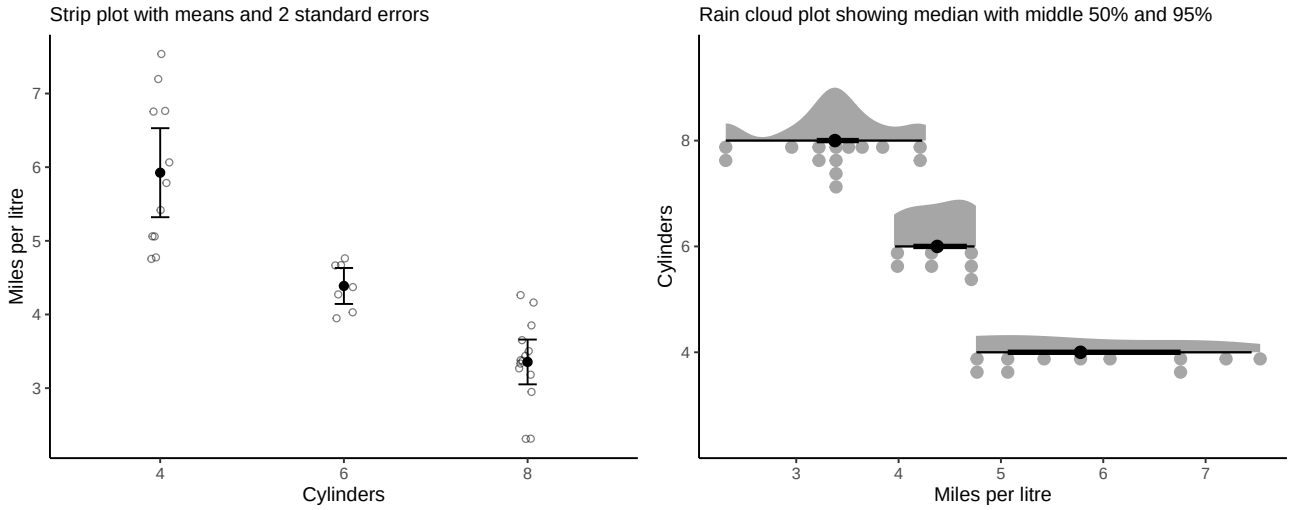

Figure 1.6: Alternatives to boxplots

1.2.3 Line plots

Line plots are most often seen in timeseries plots with time on the x-axis and the response on the y-axis. Line plots involve joining points with a line, which indicates 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 ...
```

```
str(lynx)
lynx2 <- as.data.frame(lynx)
str(lynx2)
lynx2$Year <- seq(from=1821, to=1934, by=1)
lynx2$Trappings <- as.numeric(lynx2$x) # Time-Series is complicated.
str(lynx2)
head(lynx2, 2)
```

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 for a function with `?` to see the argument order (e.g., `?seq`).

Q. 1.6

Using `seq()`, write a piece of code which generates odd numbers between 1 and 20. Try with and without naming the arguments.

View solution

```
seq(from=1, to=20, by=2)
seq(1, 20, 2)
```

Use `?plot` to investigate options for plotting. Find the `type=` argument for plotting both the points and a connecting line. Why might this be the best option here?

Q. 1.7

Using `plot()`, produce a line plot similar to Figure 1.7.

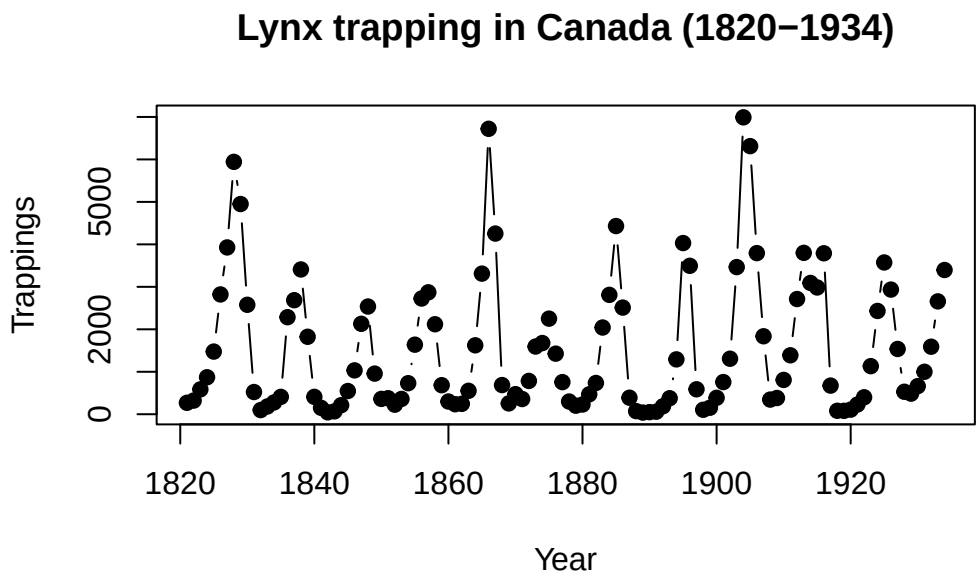

Figure 1.7: The number of lynx trapped in Canada (1820-1934)

View solution

```
plot(Trappings ~ Year, data=lynx2, pch=19,
     main="Lynx trapping in Canada (1820-1934)", type="b")
```

Q. 1.8

Create a plot that shows the log number of trappings from 1850 to 1900.

View solution

```
plot(log(Trappings) ~ Year,
     data=lynx2[lynx2$Year >= 1850 & lynx2$Year <= 1900, ],
     pch=19,
     main="Lynx trapping in Canada (1850-1900)", type="b")
```

1.2.4 Histograms

Histograms illustrate the distribution of **continuous** data. In histograms the bars are adjacent (no gap). This indicates that the underlying values are continuous rather than discrete.

```
hist(lynx2$Trappings, main="Lynx trapping", xlab="Trapped lynx per year")
```

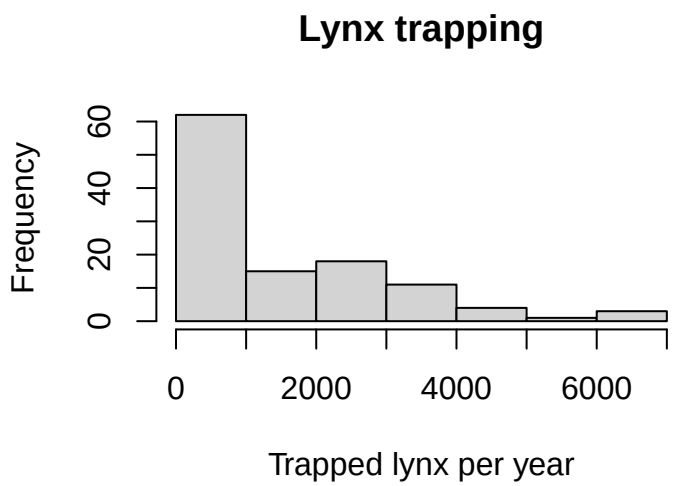

Figure 1.8: Lynx pelts per year with default settings.

Q. 1.9

What conclusions do you draw from this plot? Which range of values was most common across years?

Be aware that histograms can be quite sensitive to the number of bins, and you should explore different options. You can set the number or values of break points with `breaks=...`.

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

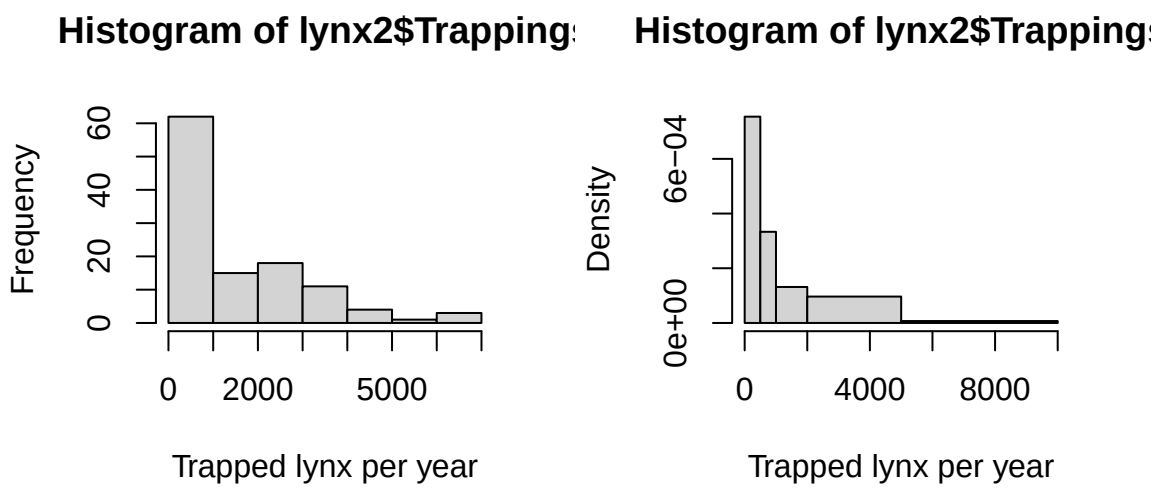

Figure 1.9: Lynx pelts per year with breaks=5 (left) and a vector of breaks (right).

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

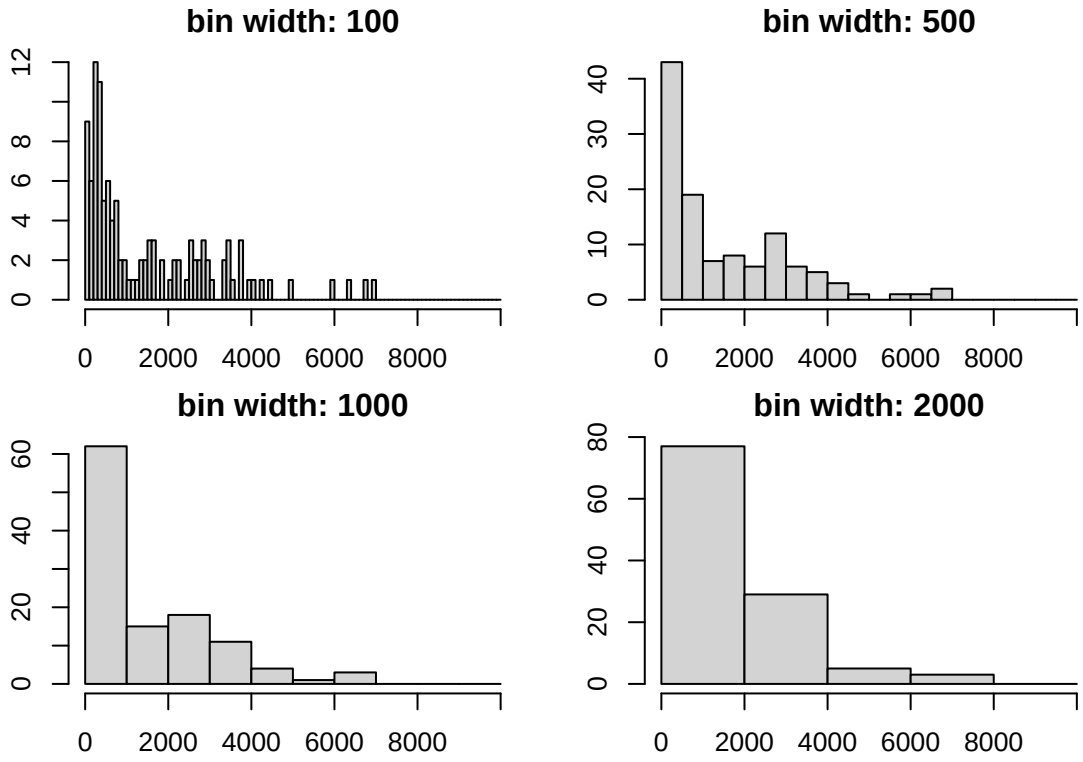

Figure 1.10: Histograms of lynx pelts per year with different breaks

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

Which of these plots is the most useful? There is no definitive answer, but the first is very busy and the last fails to show relevant detail near 0. Bin widths of 500-1000 communicate the patterns most clearly.

Generally, 5-15 breaks usually work well.

1.2.5 Bar graphs

Bar graphs are used to plot counts of categorical or discrete variables. We'll be using the `islands` dataset which is a named vector of island areas.

> **i** Many objects in R can have row names. However, converting between data types may lose this information. Consequently, **it is better practice to store relevant information in a column**. Nevertheless, there are occasions where this is useful and you may come across datasets with data stored as row names.

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)
str(islands_df) # rownames are not shown!
```

```
# 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 1.11: 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])
```

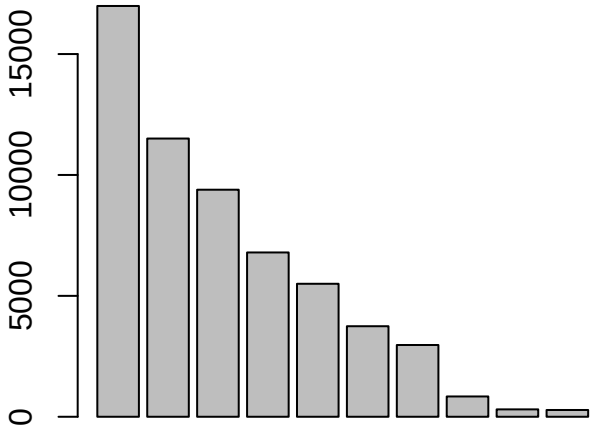

Figure 1.12: Top 10 island areas

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])
```

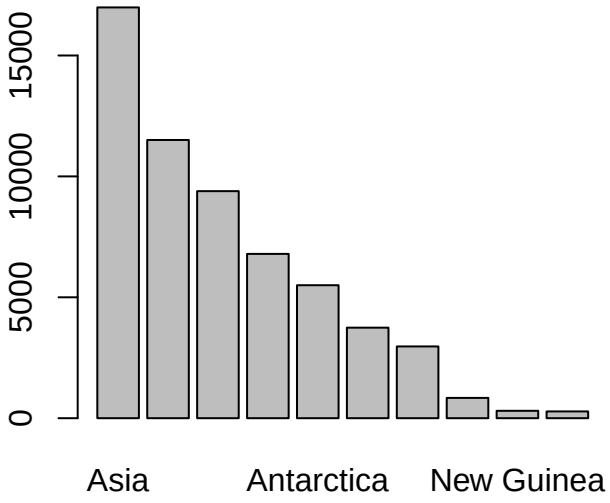

Figure 1.13: Top 10 island areas with names

Which of course are unreadable. There are many options here (e.g., see `las` in `?par`) but we will rotate the plot. To do this, we need to re-adjust the margins, 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)")
```

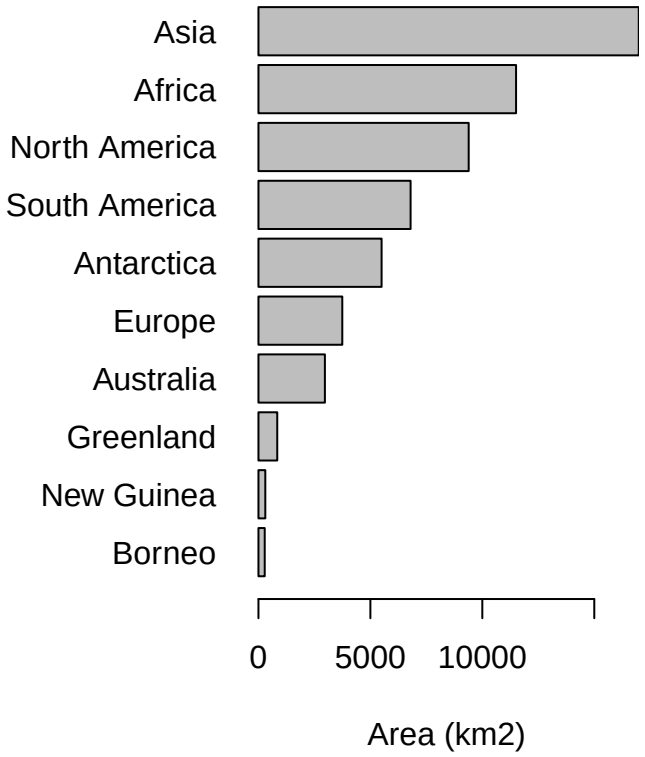

Figure 1.14: Finally! Did you know Antarctica is bigger than Europe?

Data visualization is an iterative process with lots of trial and error to 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()`.

> **i** We will cover *ggplot2* and other the *tidyverse* packages in more detail soon. You are welcome to use whichever system you prefer.

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 sometimes replace graphics (e.g., where the data is relatively simple or where individual precise values are important). There are many types of summary statistics. Here we are concerned with central tendency and variability.

Q. 1.10

What are the three main measures of central tendency?

Q. 1.11

What are three measures of variability?

The most appropriate metrics of central tendency or variability will depend on 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 `cars2` dataset.

```
summary(cars2)
```

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

```
# Recall the options to access a column in a dataframe
summary(cars2$speed_mph)
summary(cars2[, 1])
summary(cars2[, "speed_mph"])
```

Often you'll wish to summarise your data across levels of a certain factor, such as levels of a certain treatment. 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 others in the *tidyverse*) can be used.

We'll use the built-in dataset `InsectSprays`. Viewing your raw data can be an important check as well. You can open a spreadsheet-style 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, B~
```

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

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

So, let's use the *pipe* operator `|>`. This takes the output from one function and feeds it as the first input of the next (e.g., `dataset |> fun1() |> fun2() |> fun3()`), 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) |> signif(3),
            median=median(count),
            max=max(count),
            sd=sd(count) |> signif(3),
            N=n(),
            N_over_10=sum(count > 10),
            Pr_over_5=mean(count > 5))
```

```
# A tibble: 6 x 8
  spray  mean median   max    sd     N N_over_10 Pr_over_5
  <fct> <dbl>  <dbl> <dbl> <dbl> <int>     <int>     <dbl>
1 A     14.5    14      23  4.72    12         9    1
2 B     15.3    16.5    21  4.27    12        11    1
3 C      2.08    1.5     7  1.98    12         0    0.0833
4 D      4.92    5      12  2.5     12         1    0.167
5 E      3.5     3       6  1.73    12         0    0.167
6 F     16.7    15      26  6.21    12        10    1
```

1.3.1 Choosing a central tendency metric

The choice of central tendency metric depends on the nature of the data and objectives of your research. 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).

```
# this will load the 'Scallop %fat' data sheet from the xlsx spreadsheet.
scallop_df <- read_excel("data/H2DS_practicalData.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"
```

Q. 1.12

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

Q. 1.13

Are these data likely to be continuous or discontinuous?

Q. 1.14

Create a plot to visualize the distribution of these data.

Q. 1.15

Do you spot any issues?

View solution

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

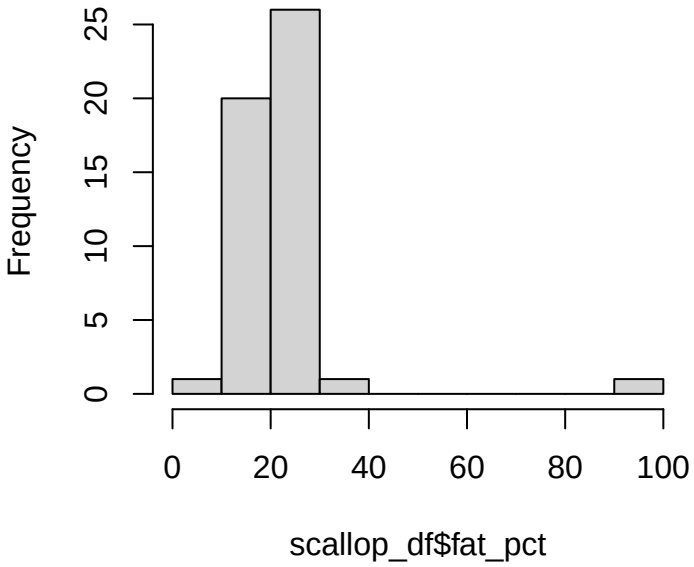

Figure 1.15: Histogram of fat percentage.

You should have spotted a potential outlier. Data entry errors are common, and a check against the original data sheet shows that the decimal was typed in the wrong place. The following code helps you locate the error.

```
which(scallop_df$fat_pct > 50) # which() returns the indexes
```

```
[1] 36
```

```
scallop_df$fat_pct[35:37] # row 36 is 99.0, but should be 9.90
```

```
[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.

Q. 1.16

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

Q. 1.17

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

Q. 1.18

Suppose the outlier was even bigger (i.e. your typo was even worse). Adjust your data, multiplying the erroneous data item by 10; copy the `fat_pct_orig` column and change row 36 to 999.

Q. 1.19

Calculate the same summary statistics.

Q. 1.20

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

Or look individually instead of calculating many metrics at once with `tidyverse` 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 this table?

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

Q. 1.21

How would the patterns seen in Table 1.1 influence your choice if you were required to summarise data that you thought might contain values that could be erroneous? Consider how each metric is influenced by the data distribution and by outliers.

Let's load a dataset that gives the length of hake across three years of sampling.

```
hake_df <- read_excel("data/H2DS_practicalData.xlsx", sheet="Hake")
str(hake_df) # once again, column names made for excel rather than R
```

```
tibble [499 x 2] (S3: tbl_df/tbl/data.frame)
 $ Year            : num [1:499] 1 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 ...
```

Q. 1.22

What type of variable is `length`?

Q. 1.23

Select an appropriate graphical method and display these data.

Q. 1.24

In your own time, summarise the hake data by year using the *tidyverse* functions you used above with the `InsectSprays` data.

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

Table 1.2: Summary of hake data.

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

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

```
fishPoo_df <- read_excel("data/H2DS_practicalData.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 ...
```

Q. 1.25

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

Q. 1.26

Which measures of central tendency and variability are most appropriate?

Q. 1.27

Considering the distribution, is the mean or median larger?

Q. 1.28

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

Q. 1.29

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

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

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, symmetric	Mean	Variance or sd
Continuous, skewed	Median	Interquartile range
Continuous, multimodal	None; state modes	None; summarise by group
Discontinuous	None; data-dependent	Range?

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 and Poisson distributions

We often wish to determine the probability of events occurring given our current understanding of the processes that are involved. This requires a mathematical description of a theoretical relationship. We refer to this as a statistical model. Models are simplified representations of reality. As George Box said, 'All models are wrong but some are useful'.

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.

For example, the binomial or Poisson distributions can be used to answer:

- How likely is it that four hurricanes hit the US in one season based on historic data?
- How likely is it that a '50-year' wave will hit in the next ten years?
- Is a river flooding more frequently now than it used to?
- What proportion of egg clutches in a fish population will be all males?

This practical gives you the opportunity to practice using these distributions.

```
library(tidyverse)
```

2.1 The Binomial distribution

2.1.1 Bernoulli trials

A Bernoulli trial is a single event with a *binary* outcome (i.e., two categories that are mutually exclusive). Binary outcomes include:

- Alive vs. dead
- Reproductive vs. not reproductive
- Present vs. absent

Other variables can be re-coded into a binary outcome. For example:

- Flower colour (blue vs. not blue)
- Income ($\geq$ £100,000 vs. $<$ £100,000)

By definition, each Bernoulli trial is independent of all previous trials.

Q. 2.1

A fair coin has been heads in 99 consecutive flips. What is the probability of heads on the next flip?

This is different to asking whether (100 heads) or (99 heads + 1 tails) is more likely in a throw of 100 coins. This is different because each flip is 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 a chicken laid 4 eggs and they were all female, you might wonder how likely this was by chance. Here, the outcome can be female or male, and the number of trials is 4: each egg is a 'trial' or 'event' with a binary outcome. You may question the assumption that the ratio of female:male was 50:50 and favour an alternative hypothesis that females are more likely. The binomial distribution allows you to quantify the probability of getting x females from n eggs for any given probability $p = P(female)$. 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)
- The 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 as $y \sim Binom(n, p)$. We denote $P(x)$ as the probability of getting x successes where x is an integer from 0 to n.

The **mean of a binomial distribution** is $n * p$. This gives you the *expected* outcome. For example, if $P(female) = 0.5$ and $n = 4$ eggs the expected number of females is $4 * 0.5 = 2$.

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 proportion that we expect to hatch from one clutch (two eggs).

Q. 2.2

What is the event?

Q. 2.3

What is p?

Q. 2.4

What is the number of trials?

Q. 2.5

How many possible outcomes are there for a clutch? What are they?

Q. 2.6

Write down the model specification with parameters (i.e., $y \sim Binom(...)$).

Q. 2.7

Calculate the expected proportions of clutches that contain (a) two fertile, (b) two infertile, and (c) one of each. Use a probability tree if needed.

Q. 2.8

Using the binomial probability mass function from Chapter 7, calculate the expected proportions for two eggs.

Unlike Wongles, Oozles always have broods of eight offspring and all of them hatch. We are interested in modelling the probabilities of the number of male and female offspring in these broods of eight eggs.

Q. 2.9

What is the Bernoulli event?

Q. 2.10

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

Q. 2.11

What are the possible outcomes? What is the number of possible outcomes?

Q. 2.12

Write down the model for Oozle egg sex: $y \sim Binom(n, p)$.

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.

Q. 2.13

Calculate the probability of obtaining eight male offspring.

Q. 2.14

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.

Q. 2.15

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

Q. 2.16

Use the binomial expression to calculate the probability of obtaining 0, 1, 2, 3, 4, 5, 6, 7 and 8 male offspring (note that the distribution is symmetric).

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

2.1.4 Binomial distributions in R

R can calculate probabilities for specific outcomes from a massive array of theoretical probability distributions. The binomial is just one of them.

⚠ CHANGE THIS FOR NEXT YEAR! Very confusing and instead should just demonstrate the best method. Obviously.

```
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)
dbinom(num_female, num_trials, p_female) # dbinom(4, 8, 0.5)
```

```
# formatted output just because:
paste0("P(", num_female, " female | ", num_trials, " eggs) = ",
       dbinom(num_female, num_trials, p_female))
```

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

The function `dbinom()` gives the probability of a single outcome.

Often we want to know cumulative probabilities instead. This allows us to answer questions like *"What is the probability of obtaining < 4 females in a brood of 8 eggs?"* Here, <4 equates to the cumulative probability P(0) + P(1) + P(2) + P(3).

For these calculations, we can use `pbinom(...)` instead of `sum(dbinom(...))`.

```
# 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"
```

Q. 2.17

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

Q. 2.18

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

Q. 2.19

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

Note that R has vectorized the calculation, returning the probability for each value in vector from `0:num_female - 1`.

Q. 2.20

Calculate P(< 4 females | 8 eggs) using `dbinom()` instead of `pbinom()`.

The above code has a bug in it. You can check what R is doing by running parts of the code:

```
0:num_female - 1 # Oops! 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
```

Q. 2.21

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

Q. 2.22

What is the probability of getting 3 females?

Q. 2.23

What is the probability of getting 8 females?

Q. 2.24

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

Q. 2.25

Review your model that describes this random process (number of females per eight eggs) that you wrote above: $y \sim Binom(n, p)$.

Q. 2.26

What is the probability of getting < 4 females?

Q. 2.27

What is the probability of getting ≤ 4 females?

Q. 2.28

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

Q. 2.29

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

Let's visualize 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 <- 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")
```

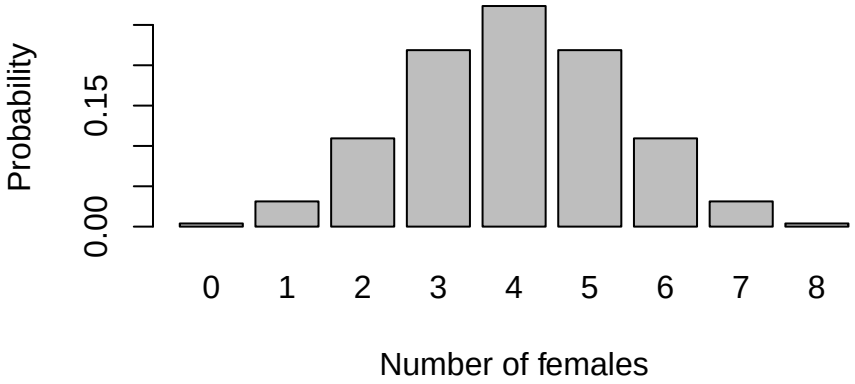

Figure 2.1: Binomial probability distribution

```
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!")
```

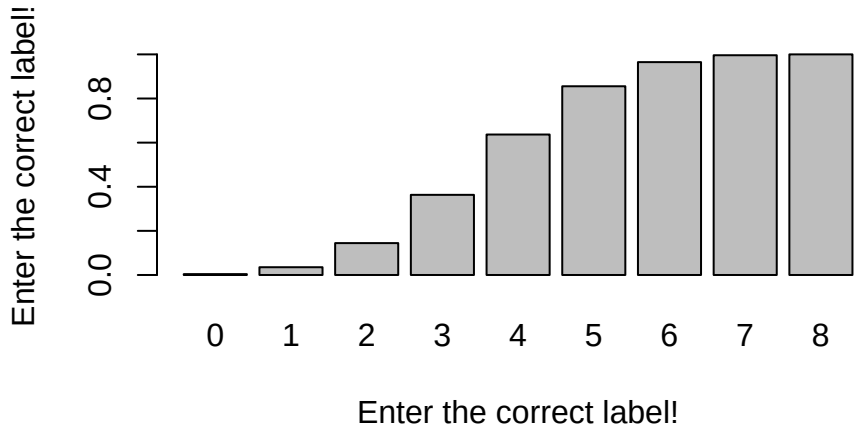

Figure 2.2: Cumulative binomial probability distribution

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 sub 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 see $P()$, $Pr()$, $p()$, or $Prob()$, which all mean the same thing.

The interpretation of $P(y_i = 8 \mid n = 8,\ p = 0.5) = 0.00391$ is that the probability of 8 female offspring in a clutch of 8 eggs where each egg has 50% probability of being female is 0.00391.

In other words, if we have 500 broods, each with 8 eggs, we expect $500 * 0.00391 = 1.95 \approx 2$ broods to be all female. Is there something strange about our Oozle or is it just one of the 2/500 by chance?

Q. 2.30

Why is a bar graph the appropriate plot here?

Q. 2.31

What do you notice about the shape of the distribution when $p = q = 0.5$

Q. 2.32

Re-run the analysis with the probability of female as 0.8 and plot the results. How has the shape of the distribution changed?

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 whales spotted on a cruise.

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).

Find the formula for the Poisson distribution in Chapter 7.

Here, $\bar{y}$ is the mean, x is the outcome of interest, e is Euler's number, and ! is factorial. Note that here, we use $\bar{y}$ as our stand-in for the population parameter $\lambda = \mu = \sigma$ which defines the Poisson distribution.

Q. 2.33

Translate the Poisson formula into an R function by completing the following code.

```
# hint: e^2 = exp(2)
# hint: 3! = factorial(3)
calc_poisson_prob <- function(x, y_bar) {
  # Translate the formula here using x and y_bar
}
```

2.2.1 Poisson distributions by hand

The first step is to calculate the mean number of observations per unit. This is the Poisson parameter, and is referred to as the *rate* or as *lambda* (λ). The unit could be spatial (e.g., per m^2) or temporal (e.g., per hour).

Last weekend, I randomly threw a $1m^2$ quadrat repeatedly on a sandy beach covered in worm casts Figure 2.3. In each quadrat, I counted the number of casts.

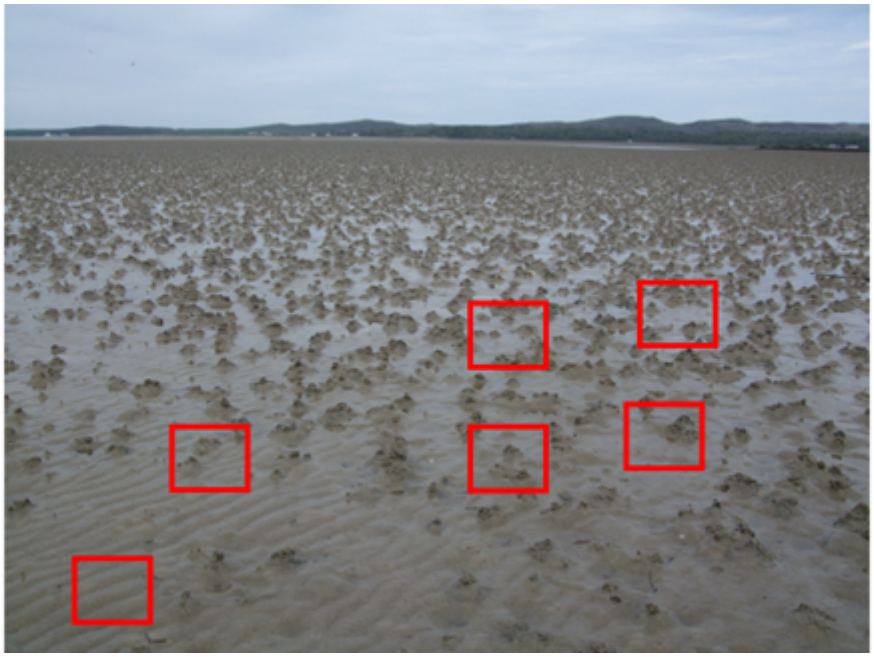

Figure 2.3: Worm casts on a sandy beach with 1x1 m quadrats.

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).

We will then compare our observations to the expectations from the theoretical Poisson distribution. To do this, we need to know the probability of observing each count, given the mean count per quadrat.

```
# num_worms is the number of worms per quadrat
# num_quadrats is number of quadrats that contained each 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
```

```
  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 relatively small, the occurrences are rare enough to be reasonably described by a Poisson distribution if worms occur independently.

From the mean, we can calculate the expected frequency of observing different numbers of worms in any quadrat (assuming the model assumptions are met: What are the assumptions?). The number of worms per quadrat (y) is discrete; it can only take integers greater or equal to zero.

We are often interested in probabilities 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, you may need to calculate $P(y_i \leq 1)$, which is the probability that a random quadrat (y_i) contains 1 or fewer worm casts (a, an integer value). To be fully complete, we might even write $P(y_i \leq 1|\bar{y})$, which acknowledges that we know the (sample) mean.

So, to calculate the probability of obtaining 1 or fewer worms per quadrat, you could start by writing $P(y_i = 0) + P(y_i = 1) = ...$

Q. 2.34

Use your `calc_poisson_prob()` function to calculate the expected frequency of 0, 1 & 2 worms per quadrat.

Q. 2.35

What calculation would you need to conduct to determine $P(y_i \geq 1)$? What is the theoretical upper limit of the Poisson distribution?

R has built-in functions for calculating these, but it is important to know what you are asking them to calculate.

2.2.2 Poisson distributions using *pois() functions

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 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
```

```
sum(worm_df$num_quadrats * worm_df$num_worms) # total number of worms
```

```
[1] 141
```

```
lambda_worms <- with(worm_df, sum(num_worms * num_quadrats) / sum(num_quadrats))
lambda_worms # the mean number of worms per quadrat
```

```
[1] 1.41
```

Given this, we can find $P(y_i \leq 5)$: the probability of a random quadrat containing five or fewer worms. With a mean of 1.41 worms per quadrat, $P(y_i \leq 5) = 0.997$ (3 sf). This means that if your data are Poisson distributed with $\lambda = 1.41$, it is highly unlikely to find more than five worms in a quadrat.

Q. 2.36

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

To do these calculation in R, we can use `dpois()` and `ppois()`:

```
# ?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
# for questions like 'determine P(y_i >= a | lambda)
1-ppois(a-1, lambda) # probability of observing >= 'a' worms: ppois()
```

```
[1] 0.01465169
```

Q. 2.37

Why is `a-1` used in the `ppois()` function above? When would you use `a` instead?

```
ppois(0:a, lambda) # what does 0:a mean? What's another way to make this vector?
```

```
[1] 0.2441433 0.5883853 0.8310759 0.9451405 0.9853483 0.9966869
```

```
signif(ppois(0:a, lambda), 3) # round with ?signif
```

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

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

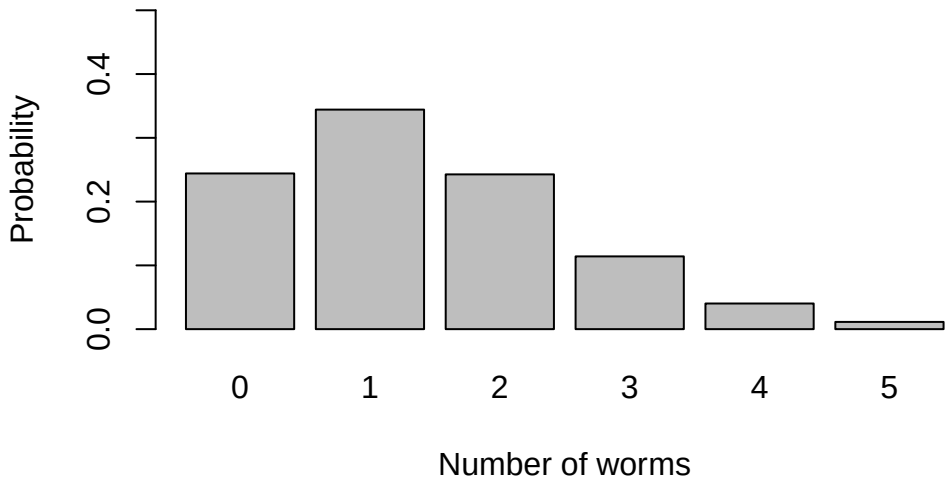

Figure 2.4: Poisson probability distribution.

Q. 2.38

Change the code to plot the cumulative probabilities. You will need to use `ppois()` instead of `dpois()` and adjust 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)
```

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

Q. 2.39

Format the probabilities in Table 2.1 to 3 decimal places.

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

Q. 2.40

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

This 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 per m^2. Note that the number of worms per quadrat is a discrete variable, but you can have non-integer 'expectations' (i.e. means).

Q. 2.41

Add another column that is the difference in the observed number of quadrats and `num_quadrats_expected`.

Q. 2.42

Produce a bar graph of this difference.

2.3 The Poisson approximation of the binomial model

When the number of trials n is large and the probability of success p 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 $(n * p)$ and use that to approximate $y \sim Pois(\lambda = np)$. Using the Poisson distribution is computationally more efficient in these cases. There is no settled threshold, but as n increases and p decreases, the approximation gets better.

If we set the mean $\lambda = np = 5$, we can visualize the distributions with different combinations of n and p (e.g., $p = 0.5, n = 10$; $p = 0.05, n = 100$, etc). We can use this to illustrate how the binomial distribution converges to the Poisson distribution as n gets larger.

```
# 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.
y_seq <- 0:13
binom_df <- tibble(y=rep(y_seq, times=4), # ?rep
                   n=rep(c(10, 20, 100, 500), each=length(y_seq)),
                   p=rep(c(0.5, 0.25, 0.05, 0.01), 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, ")"))
distr_df <- bind_rows(binom_df, pois_df) |>
  mutate(label=factor(label, levels=unique(label)))
```

```
ggplot(distr_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="PuBu") + # from colorbrewer2.org
  labs(x="Number of successes with mean = 5", y="Probability") +
  scale_x_continuous(breaks=y_seq) +
  theme_classic() +
  theme(legend.position=c(0.85, 0.85))
```

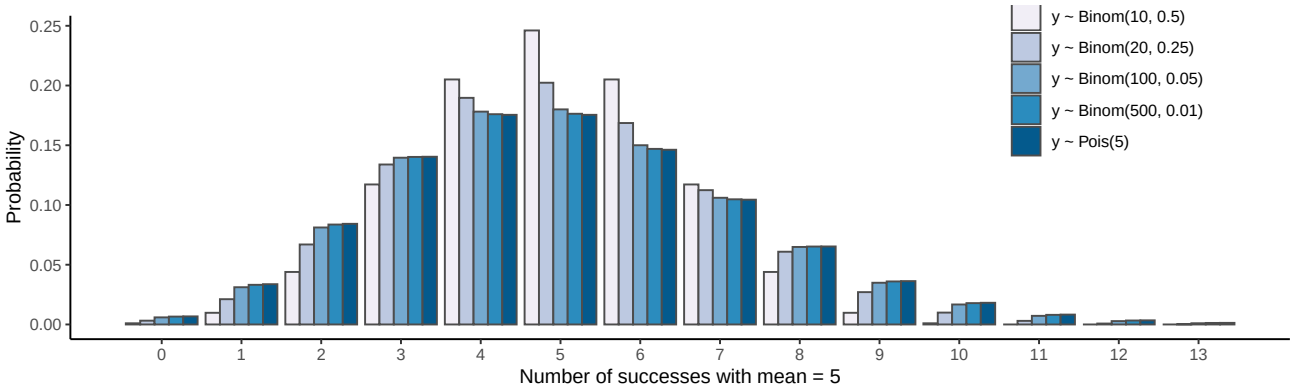

Figure 2.5: Binomial and Poisson distributions converge with larger numbers of events, depending on p.

Q. 2.43

What is the modal value in each of these distributions?

Q. 2.44

Generate some random numbers from the distributions in Figure 2.5 and calculate their mean and variance.

Here is $y \sim Pois(\lambda = 5)$:

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

```
[1] "Mean: 4.98, variance: 5"
```

Q. 2.45

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

Q. 2.46

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 is 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) in a known number of trials. 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 to the probability of success (p) and suggest a different value.

The Poisson distribution is another discrete probability distribution that is used to predict the probability of counts that are rare, independent and randomly distributed with mean = variance = λ. 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 population. Inference is the process of estimating population parameters from a sample.

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

Q. 3.1

Which two theoretical distributions have we already covered?

The normal (a.k.a., Gaussian) distribution is a theoretical distribution that is central to inferential statistics. If your data (or, more accurately, 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 examine the normal distribution, calculate Z scores, and interpret those Z scores. We'll assess whether data are reasonably assumed 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.

```
library(tidyverse)
library(readxl)
set.seed(2025)
```

3.1 Using the normal distribution

The length 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: $\mu = 37.6cm$ and $\sigma = 1.20cm$.

Q. 3.2

What are the theoretical limits of the normal distribution?

Q. 3.3

Do you think normality a fair assumption for these data?

Q. 3.4

Assume herring length is approximately normally distributed and write down the model which describes the fish length distribution: $y_i \sim Norm(\mu, \sigma)$.

If we know the population parameters, 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 are expected to be < 38 cm.

```
mu <- 37.6
sigma <- 1.2
y <- 38

y_lt38_df <- tibble(len=seq(mu-3*sigma, mu+3*sigma, length.out=1e3),
                    density=dnorm(len, mu, sigma),
                    shade=len < 38)
ggplot(y_lt38_df, aes(len, ymin=0, ymax=density, fill=shade)) +
  geom_ribbon(colour="grey30") +
  scale_fill_manual(values=c("white", "red3"), guide="none") +
  labs(x="Herring length (cm)", y="Probability density") +
  theme_classic()
```

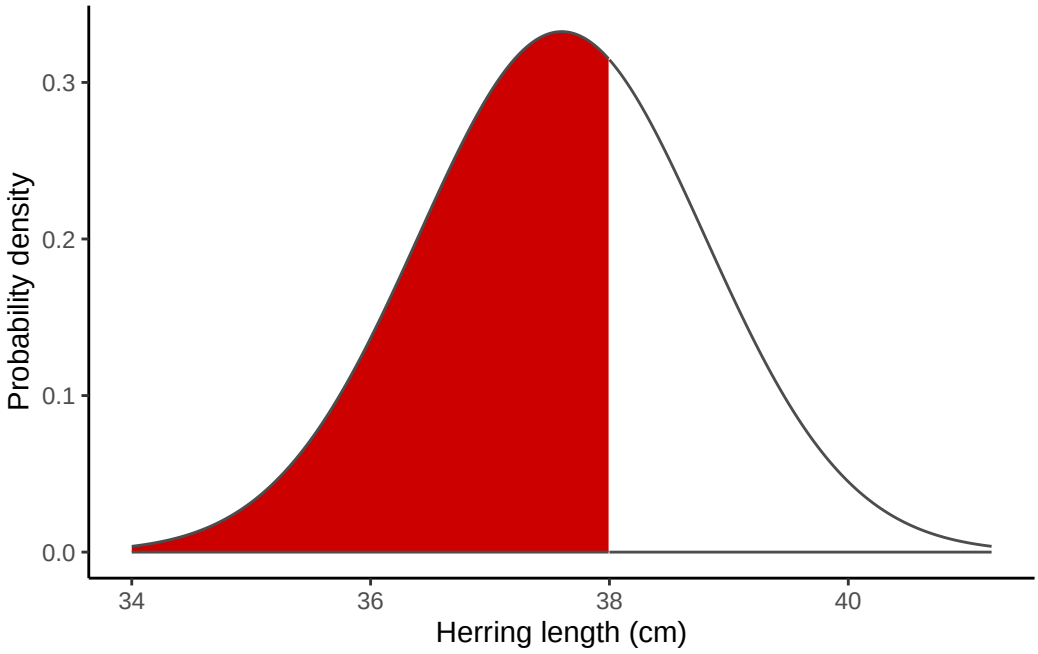

Figure 3.1: Normal distribution of herring lengths.

3.1.1 Using R to calculate areas under the normal curve

To solve this problem using R we use the 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.

Q. 3.5

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

The R functions for the normal distribution take the same form as those for the binomial and Poisson distributions. To calculate the probability is of observing a fish less than 38 cm, you need to specify the model and select the appropriate distribution function. Use `?dnorm` or `?pnorm` and look at your options (Figure 3.2).

```
library(cowplot)
norm_df <- tibble(x=seq(-4, 4, length.out=1e3),
                  x2=seq(0, 1, length.out=1e3),
                  lo=0) |>
  mutate(d=dnorm(x),
         q=qnorm(x2),
         p=pnorm(x),
         above_m1=x > -1)
labels_df <- tribble(~x, ~y, ~label,
```

```
                    -1, dnorm(-1), "dnorm(-1) = 0.242  ",
                    -1, 0, " qnorm(0.158) = -1")
p1 <- ggplot(norm_df, aes(x)) +
  geom_ribbon(aes(ymin=lo, ymax=d, fill=above_m1)) +
  geom_line(aes(y=d)) +
  geom_point(data=labels_df, aes(y=y), size=2) +
  geom_text(data=labels_df[1,], aes(y=y, label=label), hjust=1, size=3) +
  geom_text(data=labels_df[2,], aes(y=y, label=label), size=3,
            hjust=0, vjust=0, nudge_y=0.01) +
  scale_fill_manual("", values=c("red3", "white"),
                    labels=c("pnorm(-1) = 0.158", "")) +
  labs(x="Value", y="Density") +
  theme_classic() +
  theme(legend.position=c(0.8, 0.8),
        legend.text=element_text(size=8),
        legend.key.size=unit(0.3, "cm"))
p2 <- ggplot(norm_df, aes(x, d)) + geom_line() +
  labs(x="Value", y="dnorm(Value, 0, 1)") +
  theme_classic()
p3 <- ggplot(norm_df, aes(x, p)) + geom_line() +
  labs(x="Value", y="pnorm(Value, 0, 1)") +
  theme_classic()
p4 <- ggplot(norm_df, aes(x2, q)) + geom_line() +
  labs(x="Probability", y="qnorm(Probability, 0, 1)") +
  theme_classic()

plot_grid(p1, p2, p3, p4, align="hv", axis="tblr")
```

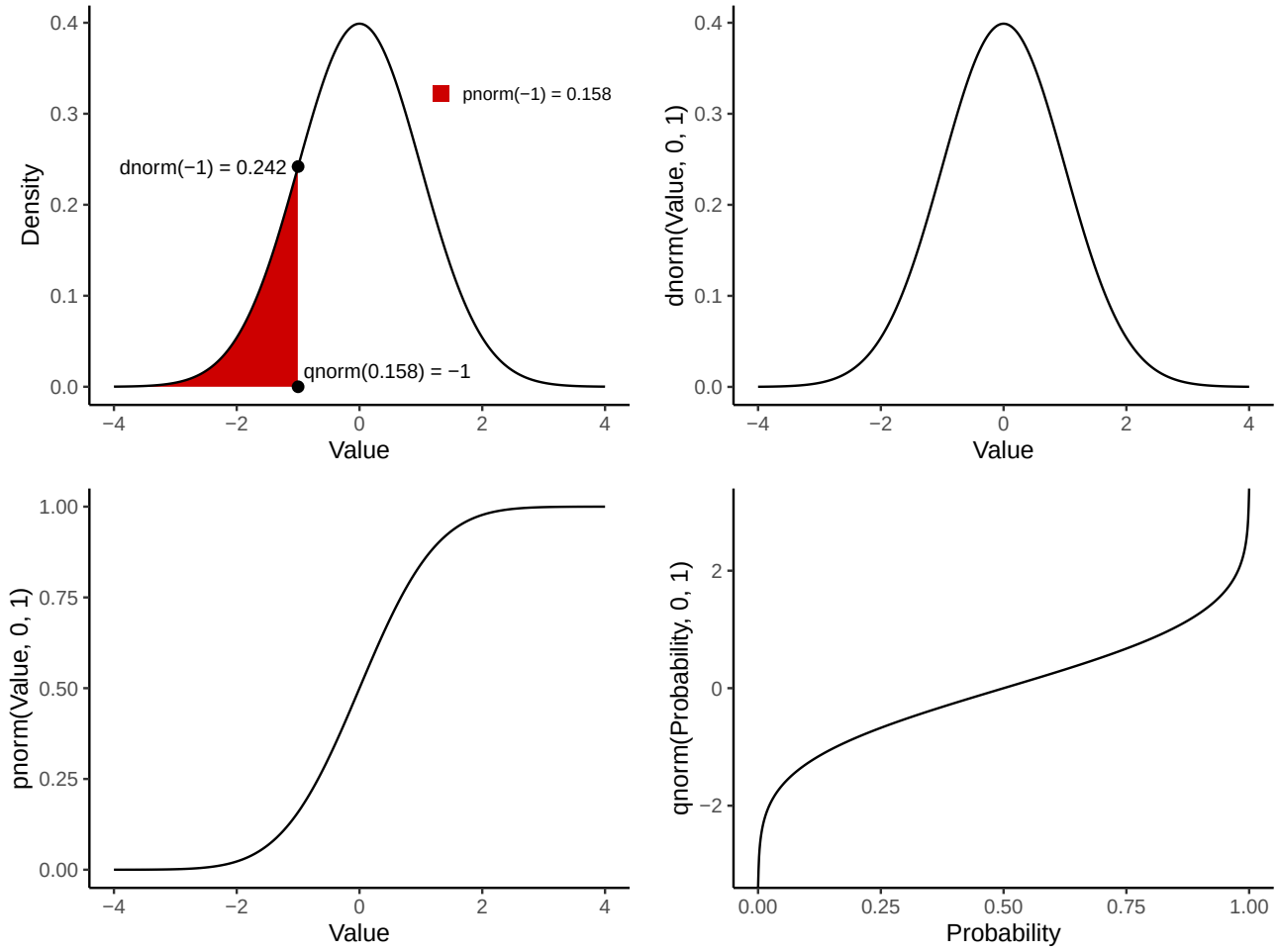

Figure 3.2: Functions for distributions in R illustrated with a standard normal.

To calculate $P(y_i < 38|\mu = 37.6, \sigma = 1.2)$, we use `pnorm()`, which gives the cumulative probability from `-Inf` to the value we choose.

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

Q. 3.6

What is the probability of a randomly selected fish from this population being less than 35 cm?

Q. 3.7

What proportion of individuals are greater than 39 cm?

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")
```

3.2 Normal model adequacy

In a population of long-eared wrasse, we calculate μ and σ and find $y \sim Norm(15.2cm, 25.1cm)$.

Q. 3.8

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

Q. 3.9

Calculate the proportion of fish that are expected to be less than zero cm in length.

Q. 3.10

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

Q. 3.11

Would your conclusions change if σ were smaller? For example: $y \sim Norm(15.2, 2.51)$.

3.3 Quantiles

A quantile is a value 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 - Q1 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 which 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 quantile function `qnorm()`, speciying the cumulative probability as an argument. To find the value that splits the upper 2.5%:

```
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 multiple values
rbind(probs, quantiles=qnorm(p = probs, mean = 37.6, sd = 1.2))
```

```
              [,1]     [,2] [,3]     [,4]     [,5]
probs      0.10000  0.30000  0.5  0.70000  0.90000
quantiles 36.06214 36.97072 37.6 38.22928 39.13786
```

```
qnorm(p = c(0.025, 0.975), mean = 37.6, sd = 1.2) # middle 95%
```

```
[1] 35.24804 39.95196
```

These values (`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 constitute the central 95% of your data.

We quote our mean and interval like this:

> The mean fish length and 95% interval was 37.6 cm (35.2 – 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).

Q. 3.12

Find the values that capture the middle 90% of the herring data, where $y \sim Norm(37.6, 1.2)$.

Q. 3.13

What value would you expect to correspond to `p=0.5`?

Q. 3.14

Check your answer above using `qnorm()`.

To visualize regions of the normal distribution, let's use *ggplot*.

```
mu <- 37.6
sigma <- 1.2
lb <- 36 # lower boundary
ub <- 40 # upper boundary

norm_df <- tibble(z=seq(-4, 4, length.out=1000),
                  x=z*sigma + mu,
                  densNorm=dnorm(x, mu, sigma))
ggplot(norm_df, aes(x, densNorm)) +
  geom_line() +
  geom_ribbon(data=norm_df |> filter(x > lb & x < ub),
              aes(ymin=0, ymax=densNorm), fill="steelblue") +
  labs(x="Herring length (cm)",
       y="Probability density",
       subtitle=paste0("P(", lb, "< y <", ub, ") = ",
                       signif(pnorm(ub, mu, sigma) - pnorm(lb, mu, sigma), digits=3))) +
  theme_classic() # changes how the plot looks: ?theme
```

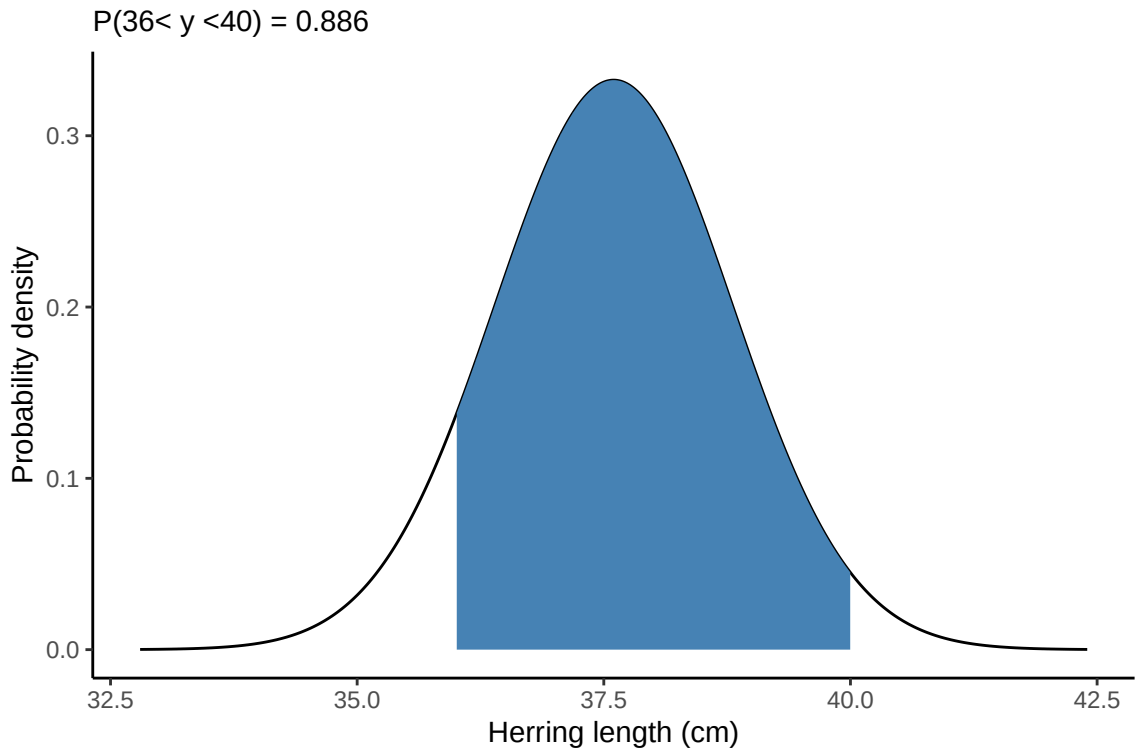

Figure 3.3: The normal distribution illustrating probability of a random herring from your population having a length between 36 and 40 cm.

Q. 3.15

Change the parameters in code the above to check that the values you generated for the 95% interval correspond when plugged into `lb` and `ub` in the code.

Q. 3.16

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

Q. 3.17

Does this also apply to discontinuous data?

How would you expect these intervals to change when the population standard deviation σ changes?

Q. 3.18

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

Q. 3.19

Does the change in the 95% interval match your expectations?

A z score gives the number of standard deviations away from the mean for any value. The z score is the value from the standard normal distribution ($\mu = 0, \sigma = 1$) that corresponds with the same quantile of the original distribution. The values for any normal distribution can be converted to Z scores by subtracting μ and dividing by σ, such that $z_i = \frac{y_i - \mu}{\sigma}$

Q. 3.20

With $y \sim Norm(37.6, 1.2)$, what is the z score for an individual of length 34?

3.4 Testing for normality

Many statistical tests assume 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 which makes evaluating the distribution 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. Note that populations are, in practice, *never* actually normally distributed. Your test is to assess how *reasonable* the assumption of normality is.

```
MS <- read_xlsx("data/H2DS_practicalData.xlsx", sheet = "Mood shrimp")
# check the data using head(), str() etc.
```

We'll learn more advanced methods in Chapter 5 for the analyses we cover, but we can plot how well our observations follow the expectations of a normal distribution with `qqnorm()` and add a line of perfect fit with `qqline()`.

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

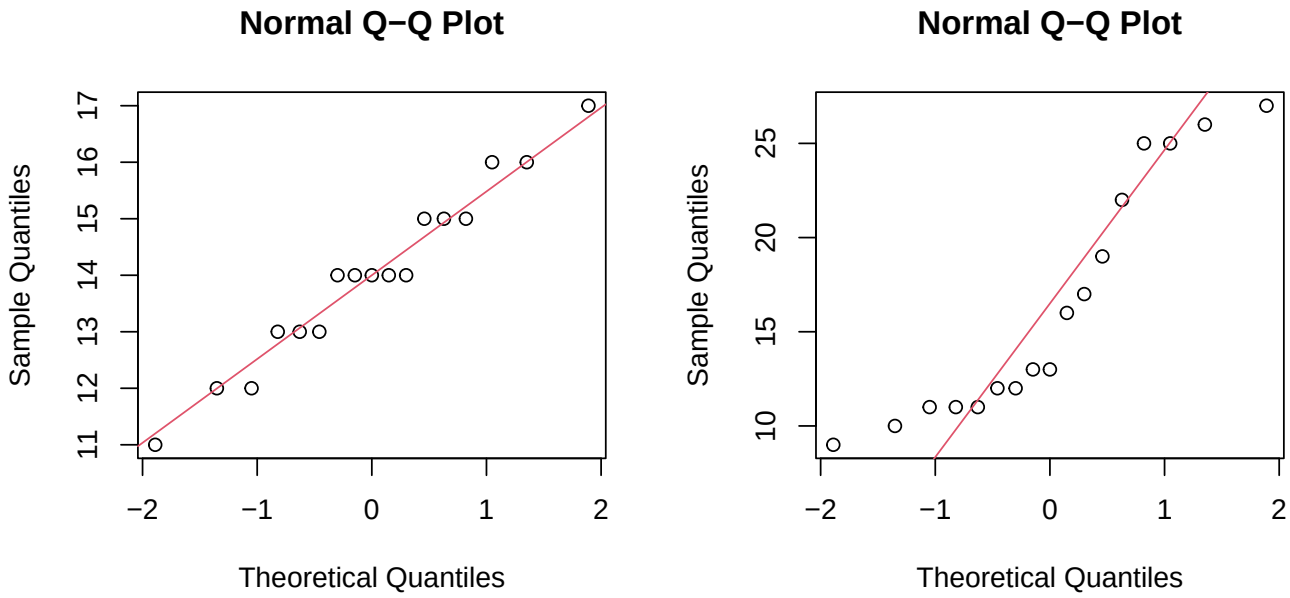

Figure 3.4: Normal (QQ) plots. The left indicates that the normality assumption might be reasonable, not so on the right.

The axes are automatically scaled so that perfectly normally distributed data would fall on a diagonal line. Any deviation from the red straight line 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 sample size decreases, it becomes 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 a population to be distributed perfectly normally and this means such tests are largely

redundant. You must assess the assumptions of your model, but you should be aware that all data fails the assumptions. The question is whether the assumptions are *reasonably* well met such that the model results can be useful. There is sadly no hard rule as to what constitutes 'reasonable'!

Q. 3.21

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 approximately normally distributed.

A common issue is that our measurements are logically bounded. For example, chemical concentrations (e.g., zinc in sediments) cannot be negative. Similarly, you cannot have negative lengths, time, mass, etc. Proportions must be between 0 and 1. Where data is collected 'near' a logical boundary they are often not normally distributed, since the normal distribution predicts 'tails' which are impossible.

One solution is to use a mathematical transformation 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. Often transformations will also correct unequal variances (see Chapter 5) in addition to non-normality so they are very useful. The appropriate transformation depends on the data.

Common transformations include:

- Log (`log(x)`, inverse: `exp(x)`): When the distribution is skewed right and all values are >0. Often 'cures' heteroscedasticity. Commonly used for observations spanning orders of magnitude such as body size.
- Square-root (`sqrt(x)`, inverse: `x^2`): When the measurements are areas (e.g., leaf areas). Often used to 'down-weight' common species (e.g., Chapter 6), which is unrelated to model assumptions. Values must be >0.
- Arcsine (`asin(x)`, inverse: `sin(x)^2`): When the measurements are proportions. Tends to stretch out the tails (e.g., near 0 or 1 for proportions) and squash the middle (e.g., near 0.5).
- Logit (`boot::logit(x)`, inverse: `boot::inv.logit(x)`): Used for proportions excluding 0 and 1. Also commonly used in models with a binary (Bernoulli) response variable (e.g., survival probability predicted by temperature, where the response variable is 'alive'/'dead').
- Reciprocal (`1/x`, inverse: `1/x`): When the distribution is skewed right. Generally 'stronger' than a log transformation and often has logically interpretable units ($m\ s^{-1}$, $s\ m^{-1}$).

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.

We can visualize the relationship between data on the original scale and the transformed values. For values bounded by 0:

```
positive_df <- tibble(orig=seq(0.01, 10, length.out=1e3)) |>
  mutate(sqrt=sqrt(orig),
         ln=log(orig),
         reciprocal=1/orig,
         squared=orig^2) |>
  pivot_longer(cols=2:5, names_to="transformation", values_to="new_value")

ggplot(positive_df, aes(orig, new_value)) +
  geom_line() +
  facet_wrap(~transformation, scales="free", nrow=1) +
  labs(main="Positive values", x="Original value", y="Transformed value")
```

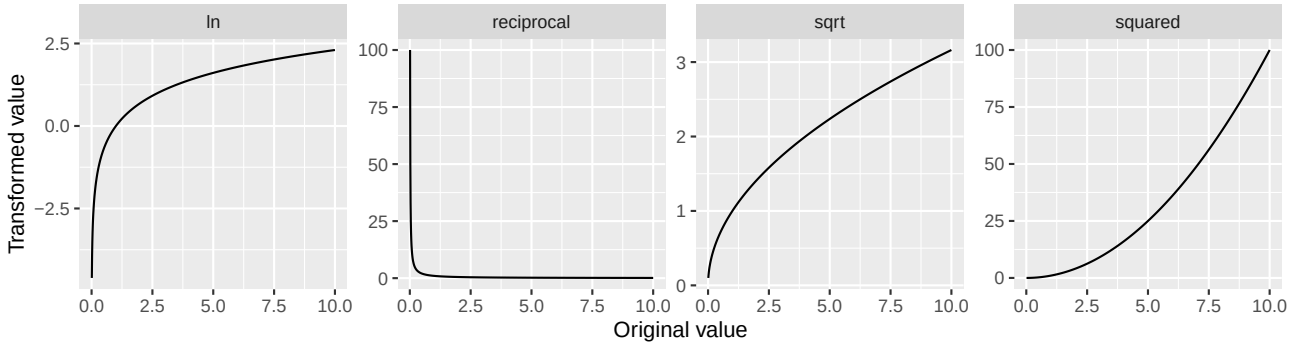

Figure 3.5: Effect of common transformations on positive data.

And for proportions:

```
proportion_df <- tibble(orig=seq(0.01, 0.99, length.out=1e3)) |>
  mutate(sqrt=sqrt(orig),
         ln=log(orig),
         asin=asin(orig),
         logit=boot::logit(orig)) |>
  pivot_longer(cols=2:5, names_to="transformation", values_to="new_value")

ggplot(proportion_df, aes(orig, new_value)) +
  geom_line() +
  facet_wrap(~transformation, scales="free", nrow=1) +
  labs(main="Proportions", x="Original value", y="Transformed value")
```

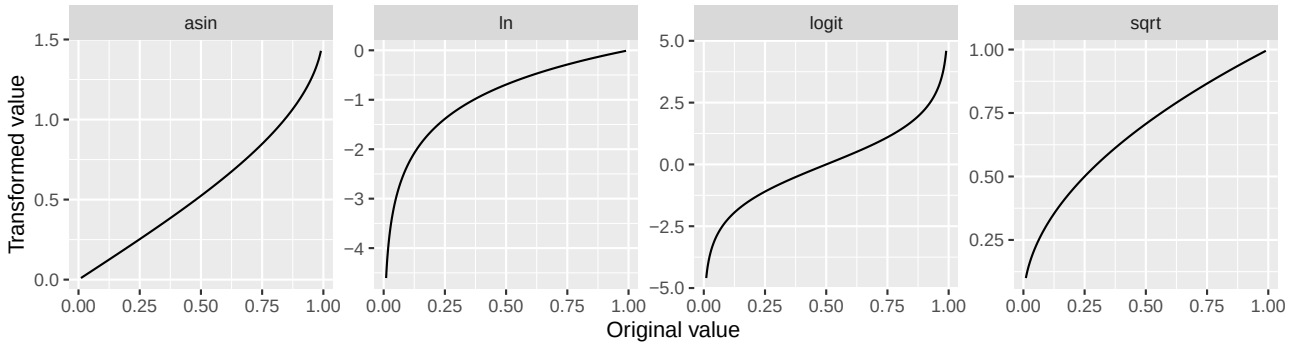

Figure 3.6: Effect of common transformations on proportions.

Q. 3.22

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

```
# get the Radon data into R; the units are parts per billion
readxl::excel_sheets("data/H2DS_practicalData.xlsx")
```

Q. 3.23

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).

3.6 The Central Limit Theorem

The central limit theorem (CLT) is about how *sample means* are distributed.

The CLT 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.

3.6.1 The distribution of means from non-normal data

The following code chunk:

1. Generates a non-normally distributed dataset (`obs_data`)
2. Repeatedly samples from it `num_samples` times, each with `sample_size` observations
3. Calculates and stores the sample mean for each `num_samples` repeat
4. Plots histograms and QQ-plots for the raw data and for the sample means

Run the following code and then experiment with the `sample_size`.

```
# Non-normal data: a mixture of several distributions
obs_data <- c(rnorm(2000, 200, 20),
              rnorm(1500, 100, 50),
              rnorm(1000, 400, 80),
              rnorm(800, 300, 100))
xlims <- range(obs_data)

# Define size of each sample and the number of sampling repeats
sample_size <- 30
num_samples <- 3000

# 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_means[i] <- sample(x=obs_data, size=sample_size, replace=T) |> mean()
}

# plot observed distribution
par(mfrow = c(2, 2),
    mar = c(5, 2, 2, 2))
hist(obs_data,
     main = "Raw data: obs_data",
     sub = paste0("mean: ", signif(mean(obs_data), 3),
                  ", sd: ", signif(sd(obs_data), 3)),
     breaks = 30, xlab = "Observations", xlim = xlims
)
qqnorm(obs_data, main = "Raw data QQ")
qqline(obs_data)

# plot distribution of sample means
hist(sample_means,
     main = paste0("Sample means, N: ", sample_size),
     sub = paste0("mean: ", signif(mean(sample_means), 3),
                  ", sd: ", signif(sd(sample_means), 3)),
     breaks = 30, xlab = "Sample means", xlim = xlims
)
qqnorm(sample_means, main="Sample mean QQ")
qqline(sample_means)
```

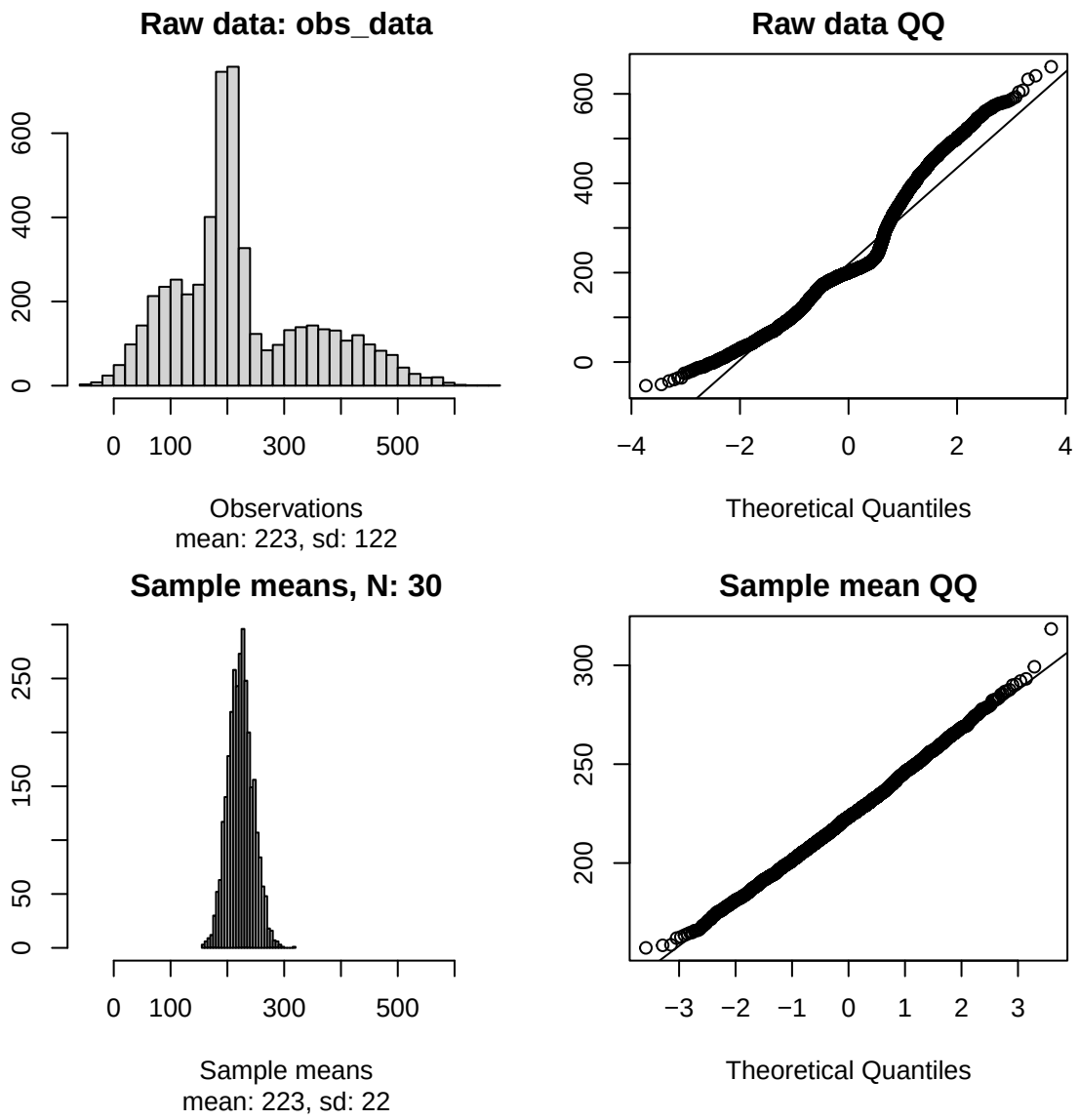

Figure 3.7: The central limit theorem in action.

Q. 3.24

What do you notice about the location of the mean as a function of `sample_size` (n)?

Q. 3.25

What do you notice about the spread around the mean of sample means as a function of `sample_size`? Why did the pattern you have observed occur?

The CLT states that `sample_means` will be normally distributed if (a) the underlying data are normally distributed, OR (b) the `sample_size` is large enough. With R, we can see this in action.

Note also 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`?.

3.7 The standard error of the mean

The standard error of the mean is the standard deviation of sample means, of a given sample size, taken from a population. It describes the dispersion of sample means ($\bar{y}$'s) we would expect if we were to repeatedly sample

the same population over and over again. It is the standard deviation of the distribution shown in the lower histogram in Figure 3.7 : `sd(sample_means)`.

Usually we only have a single sample rather than 10,000 (=`num_samples`) as above. In that case, we estimate it from a single sample as $SE_{\bar{y}} = \frac{sd(y)}{\sqrt{n}}$, where y is a vector of n observations.

We can explore this similarly to the simulated sampling we did for the CLT.

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

# Define sample size and number
sample_size <- c(3, 10, 30, 100) # size of each sample
num_samples <- 3000 # number of samples (=repeats) for each sample size

# Create a dataframe to store results
sample_mean_df <- tibble(N=rep(sample_size, each = num_samples),
                         id=rep(1:num_samples, times = length(sample_size))) |>
  rowwise() |> # enforces new sample() call for each row
  mutate(sample_mean=mean(sample(x=sim_obs, size=N))) |>
  ungroup()
```

```
sample_mean_df |>
  mutate(N=factor(N, levels=unique(N), labels=paste("n:", unique(N)))) |>
  ggplot(aes(sample_mean, colour=N)) +
  geom_vline(xintercept=mu, linetype=3) +
  geom_density(linewidth=0.9) +
  scale_colour_viridis_d("Size of each sample", option="mako", end=0.85) +
  labs(x=paste("Means of", num_samples, "simulated samples")) +
  theme_classic()
```

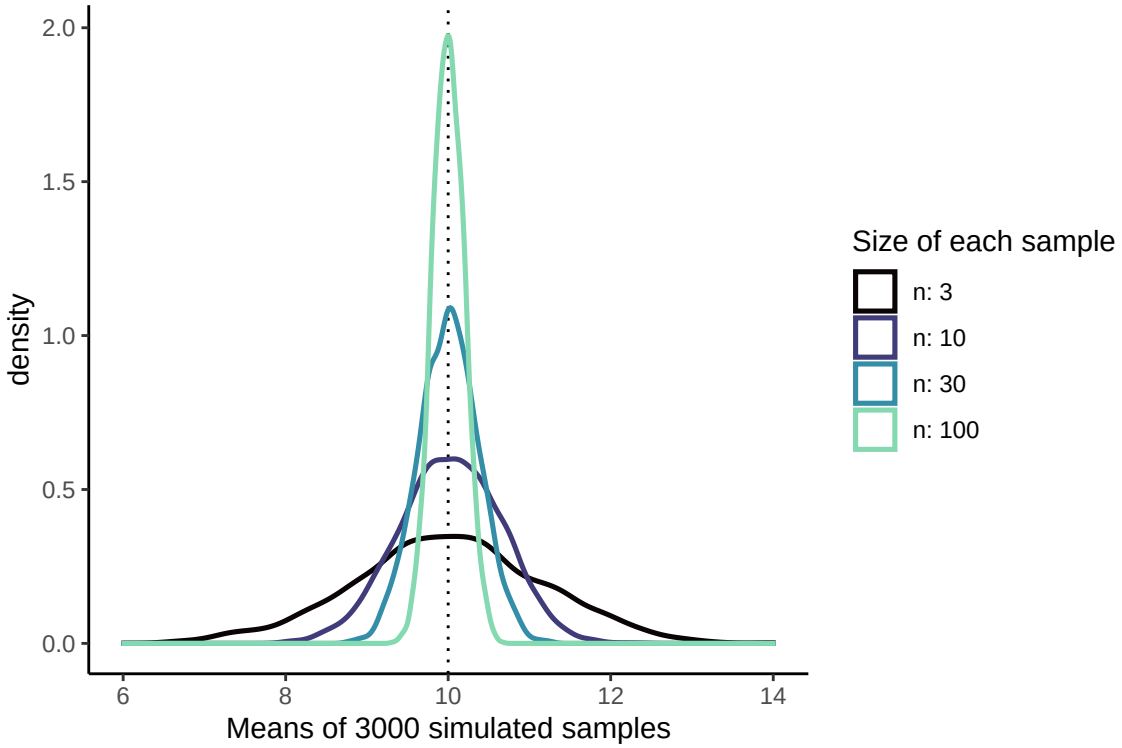

Figure 3.8: Simulated sample means. Each curve shows the sampling distribution for means of samples with size n. The standard deviation of each curve is the standard error of the mean.

Q. 3.26

Calculate the theoretical standard error of the mean for each `sample_size` given `sigma` (i.e., from the equation) and compare this with the standard error from the simulated sample means in `sample_mean_df`.

Hint:

```
sample_mean_df |>
  group_by(N) |>
  summarise() # what goes here?
```

View solution

```
sample_mean_df |>
  group_by(N) |>
  summarise(se_bySim=sd(sample_mean)) |>
  ungroup() |>
  mutate(se_byFormula=sigma/sqrt(N))
```

```
# A tibble: 4 x 3
      N se_bySim se_byFormula
  <dbl>    <dbl>        <dbl>
1     3    1.16         1.15
2    10    0.631        0.632
3    30    0.371        0.365
4   100    0.194        0.2
```

Q. 3.27

Given what you know about the CLT, how does this change with non-normally distributed data? Try creating `sim_obs` using `rpois()`.

Q. 3.28

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).

3.8 Normal approximations

In Chapter 2 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 both the Poisson distribution and the binomial distribution under certain conditions.

While the Poisson and binomial distributions are discrete and the normal distribution continuous, in practice all of our observations are discontinuous, subject to our measurement precision. A recorded mass of 437 g really indicates a mass between 436.5 and 437.5 g. The assumption that a variable is continuous is typically reasonable if there at least ~30 possible values between the smallest and largest observation. This constraint similarly applies to the approximation of binomial and Poisson distributions with the normal distribution.

All models are wrong, but some are useful! Normality is always approximate in practice.

3.8.1 The Normal approximation of the Poisson distribution

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

Q. 3.29

For $y \sim Pois(10)$ write the equivalent normal distribution: $y \sim Norm(...)$.

A normal approximation may be 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. As λ gets larger, the Poisson distribution becomes more and more computationally demanding compared to the normal distribution. Many standard statistical analyses also assume normal distributions.

However, when feasible, it is generally preferable to use the natural distribution for your data (e.g., a Poisson distribution for counts) through the appropriate GLM. The normal distribution is beneficial in some cases, but this is increasingly less so with modern methods.

Q. 3.30

Calculate $P(y_i \leq 35)$ from $y \sim Pois(40)$ and compare it to the same probability under the normal approximation of this distribution.

Q. 3.31

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

3.8.2 The Normal approximation of the binomial distribution

When n is large, the binomial distribution tends toward a normal distribution, particularly if p is near 0.5. Roughly speaking, if $np > 5$ and $n(1-p) > 5$, then the normal distribution may be a reasonable approximation. The mean of the binomial distribution is np and the variance is npq.

We might be interested in predicting the number of male and female offspring in turtle clutches. Assume the proportion that are male is 0.5 and clutch size is 40. We've observed several clutches of eggs on a particular island with only 10 males. We might wonder how unlikely this was by chance, assuming $P(male) = 0.5$.

Q. 3.32

What is the probability of observing 10 or fewer males in this scenario?

Q. 3.33

Plot the probability (i.e. from 0 – 40 males) as a bar graph and comment on its shape.

Q. 3.34

Calculate the mean and variance of this population.

Q. 3.35

Are np and $n(1-p)$ both > 5?

Q. 3.36

Specify the normal distribution that approximates this binomial distribution.

Q. 3.37

What is $P(y_i \leq 10)$ for the normal approximation?

3.9 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 parameters (μ, σ) 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

The t-distribution and confidence intervals

The goal of science is to understand the world (universe!). 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 satellite signal strength per satellite transect, the mean size of a fisher's catch, …). However, we usually cannot measure entire populations due to logistical/time/money constraints. Instead, we take a (random) sample, and infer from that sample to our population of interest based on our statistical models of the world. This is statistical inference.

Unfortunately, we can never know how well our sample 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 values of the mean that are likely.

The t-distribution models the probability of making a given observation from a population whose parameters are estimated from your 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. For small sample sizes, we are less confident about the population parameter estimates, and the t-distribution consequently becomes shorter than the normal distribution and with fatter tails (Figure 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.

```
library(tidyverse)
library(readxl)
set.seed(2025)
```

```
t_Norm.df <- tibble(
  x = seq(-6, 6, length.out = 100),
  Norm = dnorm(x),
  t01 = dt(x, 1),
  t03 = dt(x, 3),
  t30 = dt(x, 30)
) |>
  pivot_longer(2:5, names_to = "distr", values_to = "Density") |>
  mutate(Distribution = factor(distr,
    levels = c("t01", "t03", "t30", "Norm"),
    labels = c("t(df=1)", "t(df=3)", "t(df=30)", "Normal(0,1)")
  ))

ggplot(t_Norm.df, aes(x, Density, colour = Distribution, linetype = Distribution)) +
  geom_line(linewidth = 1) +
  scale_colour_manual(values = c(RColorBrewer::brewer.pal(5, "BuGn")[2:4], "black")) +
  scale_linetype_manual(values = c(1, 1, 1, 3)) +
```

```
  theme_classic() +
  theme(legend.position = c(0.85, 0.8))
```

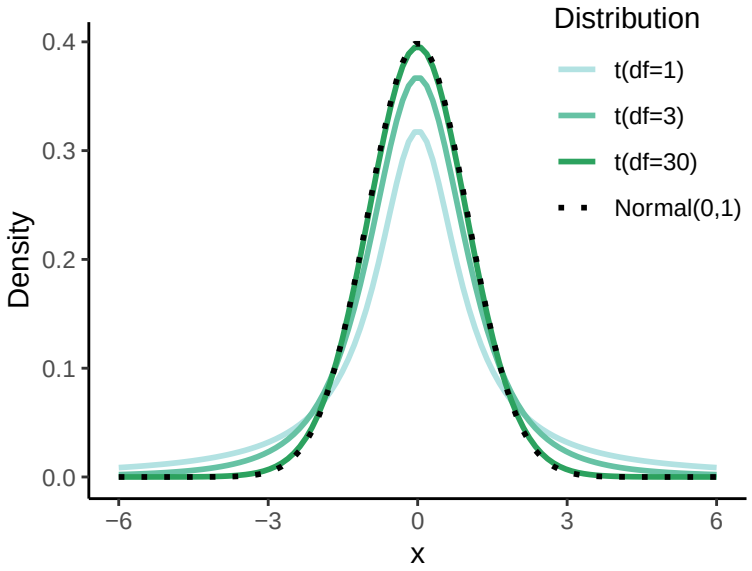

Figure 4.1: Comparison of different t-distributions (note that at df=30, the distribution is nearly identical to the normal distribution).

Q. 4.1

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?

Q. 4.2

How does sample size affect the reliability of our population parameter estimates?

4.1 Single sample t-tests

The single sample t-test is analogous to the calculation of z scores. It enables us to determine how unlikely our sample mean is, given any hypothesized mean. However, before using any parametric tests such as t-tests, we need to assure ourselves that the model assumptions are reasonably met.

Imagine we are fisheries inspectors and have sampled the cod landed by a fishing boat. We know that the mean size of the landed cod should be greater than 36.6 cm. We need to assess how likely it is that our sampled cod come from a population where $\mu \geq 36.6cm$. We are testing the hypothesis that there is one 'population' of legally landed cod, and these cod are a part of that population. We use the t-distribution to assess the probability that our sample was drawn from a 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 mean μ or standard deviation σ of the population and hence cannot use a normal distribution to model the likelihood of observing any particular value.

First, we should clearly state our hypotheses:

H_0 (the null hypothesis): The population mean of the cod on this boat is greater than or equal to 36.6 cm ($\mu \geq 36.6cm$).

H_1 (the alternative hypothesis): The population mean of the cod on this boat is less than 36.6 cm ($\mu < 36.6cm$).

We use a t-test to calculate the probability of drawing our sample from a population where the mean is 36.6 cm or greater.

Q. 4.3

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' (Figure 4.2).

```
cod_df <- read_excel("data/H2DS_practicalData.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)
```

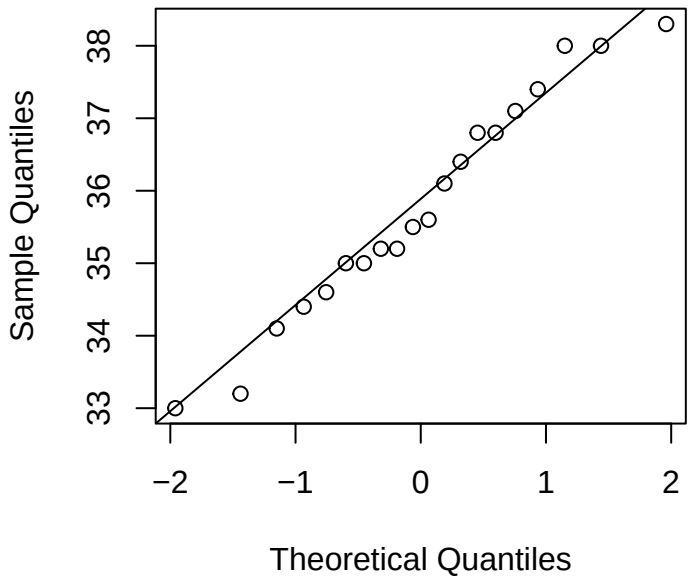

Figure 4.2: QQ-plot for the sample of cod.

Q. 4.4

Do you think the normality assumption is reasonable?

Q. 4.5

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

We wish to assess how likely our sample is to have been drawn from a population where μ is at least 36.6 cm. If our sample mean $\bar{y}$ is less than the 'legal' mean and it is 'unlikely' to have been drawn from a legal population, we might wonder if the mean of the landed cod on this boat is <36.6 cm and recommend legal action.

Q. 4.6

Calculate the sample mean, standard deviation, and standard error of the mean (Chapter 7).

View solution

```
cod_df |>
  summarise(mean=mean(CodLength_cm),
            sd=sd(CodLength_cm),
            N=n(),
            SEM=sd(CodLength_cm)/sqrt(N))
```

```
# A tibble: 1 x 4
   mean    sd     N   SEM
  <dbl> <dbl> <int> <dbl>
1  35.8  1.55    20 0.347
```

Q. 4.7

Now manually calculate the t-statistic for this sample and determine the probability of observing your data assuming that $\mu = 36.6cm$.

View solution

```
ybar <- mean(cod_df$CodLength_cm)
SEM <- sd(cod_df$CodLength_cm) / sqrt(length(cod_df$CodLength_cm))
T_stat <- (ybar - 36.6) / SEM
round(T_stat, 2)
```

```
[1] -2.35
```

Q. 4.8

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

View solution

```
pt(T_stat, df=19) |> round(5)
```

```
[1] 0.01482
```

Q. 4.9

Check you answer against that given by the `t.test()` function. See `?t.test`.

View solution

```
t.test(cod_df$CodLength_cm, mu = 36.6, alternative = "less")
```

```
    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: 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).

Q. 4.10

What is your next step as the regulatory agent?

Evaluating evidence is a central part of statistical analysis. In this example, your conclusion about the population mean μ based on your sample determines whether legal action is taken. When a binary decision is necessary, it is best to set the decision threshold before collecting data. This is called the α value. It is often set to 0.05, but this is a subjective choice. If $P < \alpha$, the null hypothesis is rejected; the sample is considered too unlikely if $\mu \geq 36.6cm$. In this scenario, we have two clear competing hypotheses, so we are in the realm of 'Neyman-Pearson's' decision theory (not Fisher's hypothesis significance testing approach).

Q. 4.11

Given the P value, do you reject the null hypothesis?

Q. 4.12

If you had set α at 0.01 would you reject the null hypothesis?

Q. 4.13

If you set α at 0.01 rather than 0.05, what type of error are you reducing and what type of error are you increasing? Which wrong conclusion becomes more likely and which becomes less likely?

Now let's use simulation to explore variability among samples. We will repeatedly sample from $y \sim Norm(100, 10)$.

Q. 4.14

What is the standard deviation in this model?

Q. 4.15

What does the symbol '~' mean?

We will repeat our sampling `num_samples` times. Each sample will be `sample_size` values drawn from a normally distributed population with mean `mu` and standard deviation `sigma`. For each sample, we will calculate the sample mean $\bar{y}$ and sample standard deviation s. Note that in this case, we *know* the population parameters.

```
# set simulation details and population parameters
num_samples <- 5
sample_size <- 3
mu <- 100
sigma <- 10

# initialize plot
set.seed(1)
xlims <- c(mu-4.5*sigma, mu+3*sigma)
plot(NA, NA, xlim=xlims, ylim=c(0,num_samples),
     xlab="Observations", ylab="Sample number")
abline(v=mu, lty=3)
points(mu, 0, pch=4, col="steelblue")
segments(mu-sigma, 0, mu+sigma, 0, col="steelblue")
text(x=xlims[1], y=0, adj=c(0,0.5), col="steelblue",
     labels=paste0("mu: ", mu, ", sigma: ", sigma))

# repeatedly sample
for (i in 1:num_samples) {
```

```
  sample_i <- rnorm(n = sample_size, mean = mu, sd = sigma)
  sample_mean <- signif(mean(sample_i), 3)
  sample_sd <- signif(sd(sample_i), 3)
  # add to plot
  points(sample_i, rep(i, sample_size))
  points(sample_mean, i, pch=4)
  segments(sample_mean-sample_sd, i, sample_mean+sample_sd, i)
  text(x=xlims[1], y=i, adj=c(0,0.5),
       paste0("ybar: ", sample_mean, ", s: ", sample_sd))
}
```

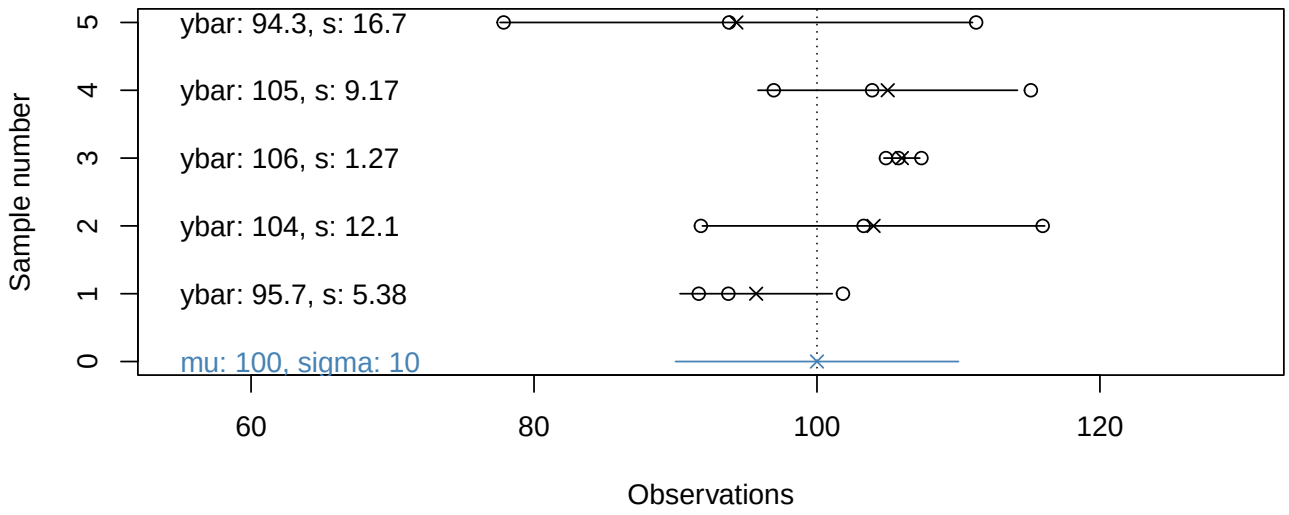

Figure 4.3: Repeated samples from a normal distribution. Open points are observations in the sample, with 'x' giving the mean and lines showing 1 sd.

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 reproducible code.

Q. 4.16

Is there a discrepancy between the population parameters and the sample statistics? Does the discrepency seem greater for the mean or the standard deviation?

The discrepancy (μ vs. $\bar{y}$, σ vs. s) is called *sampling error*. In most situations, we do not know μ and σ, but must estimate them from our sample.

If the sample is very large (and representative) then the estimate of the population parameters is likely very good. However, as the sample size is reduced, the reliability of the estimate decreases. Try adjusting `sample_size` in the code above and see how it affects the results.

The t-distribution is the distribution of values you get when you subtract sample means from the population mean and standardize by the sample standard error (i.e., $\frac{\bar{y}-\mu}{SE_{\bar{y}}}$).

Q. 4.17

How does this distribution relate to the T-statistic calculated above? The P-value?

We can extend the simulated sampling above to calculate a T-statistic for each sample in addition to $\bar{y}$ and s. We will set `num_samples` very large to better represent the distribution of sample T-statistics. The red histogram in Figure 4.4 shows the distribution of these sample T-statistics. The solid curve is the theoretical t-distribution (df=`sample_size - 1`) and the dotted line is a standard normal distribution.

Q. 4.18

Explore different values for `sample_size` below to see how the shapes change.

```
mu <- 10 # population mean
sigma <- 10 # population sd
num_samples <- 1e5 # number of samples
sample_size <- 3 # size of each sample
T_sample <- numeric(num_samples) # sample t statistics

# for each repeat: draw a sample, calculate SE and T, and store T in T_sample
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(num_samples, 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)"))
```

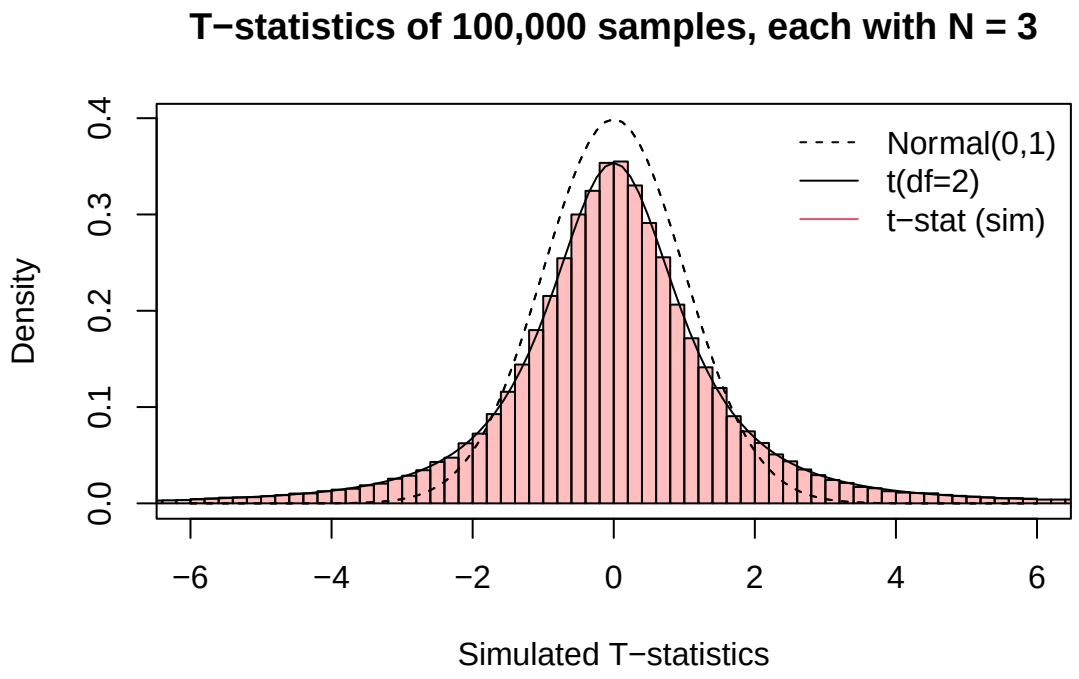

Figure 4.4: Histogram of 100,000 T-statistics calculated from 100,000 samples, along with the corresponding theoretical t-distribution (solid line) and a standard normal (dotted line).

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.

> **i** When we perform a t-test, we compare the T-statistic from our sample to this distribution: the distribution of T-statistics we would expect under the hypothesized μ. The P-value gives the probability of our T-statistic (or one more extreme) under the hypothesized μ. If it is very small, it is very unlikely that our sample would occur with that μ, and we might conclude the value of μ is something different.

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`. We'll 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.

```
# barnacle population meta-details
pop_size <- 100000  # number of barnacles in the population
barnacle_mu <- 200
barnacle_sigma <- 25

# the full population:
barnacle_diam <- rnorm(pop_size, mean = barnacle_mu, sd = barnacle_sigma)
mu <- mean(barnacle_diam)
sigma <- sd(barnacle_diam)

# histogram of the population with middle 95% shown
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)))
```

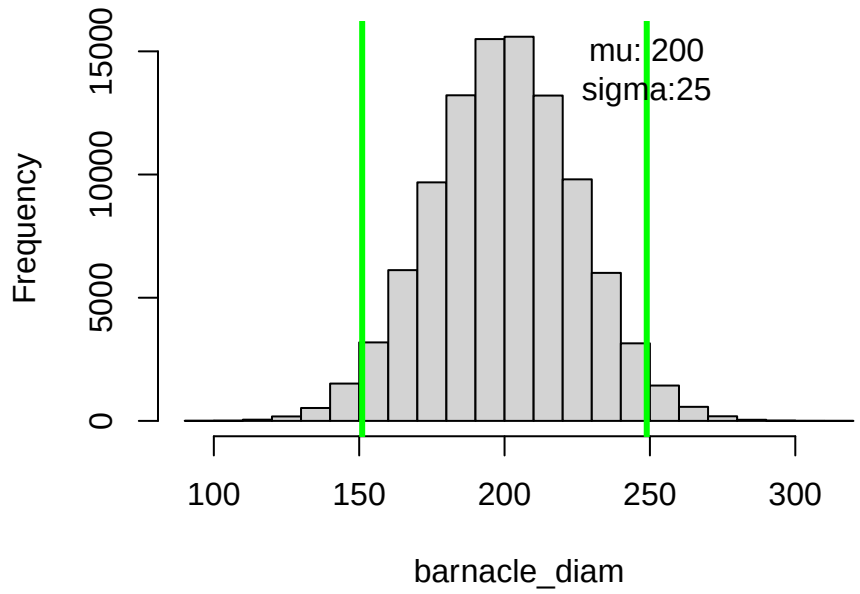

Figure 4.5: Histogram of a simulated barnacle population with 2.5% and 97.5% quantiles.

Now we can take samples from that population. This is reality: you take samples from populations where you don't know the population 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] 252.2560 209.5652 194.1364 180.0893 199.9900
[1] 181.4844 212.2622 212.0780 205.8330 209.9440
[1] 214.3341 150.0059 206.0774 194.1674 191.1762
[1] 168.1111 194.2603 180.3917 178.7273 193.4117
```

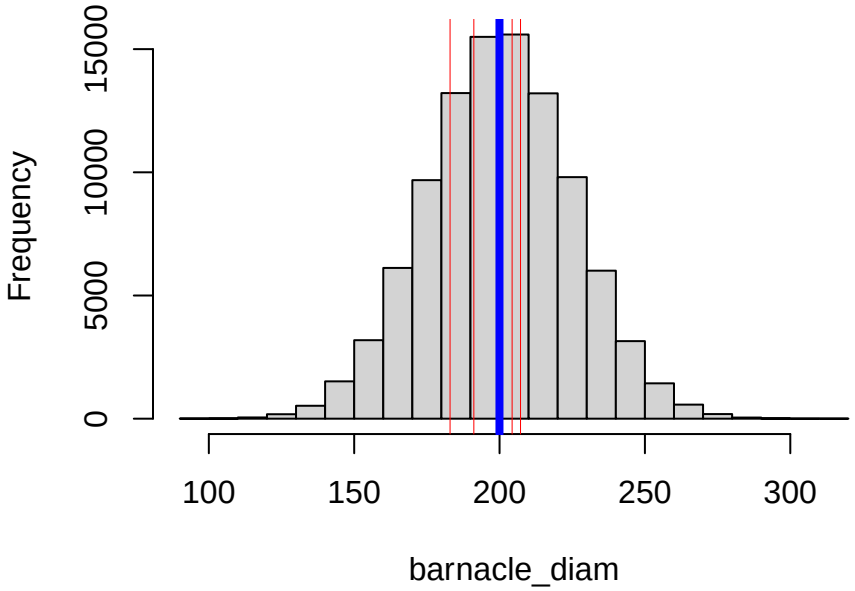

Figure 4.6: Histogram of the barnacle population showing location of 4 sample means (red lines), each with N = 5. The blue line shows the true population mean mu.

Our sample means inevitably differ from the true population mean (μ), even if only a bit. Likewise, the sample standard deviations will differ from the true population standard deviation (σ).

If we repeat this sampling enough times, we can generate a distribution of sample standard deviations (Figure 4.7). 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)))
}
```

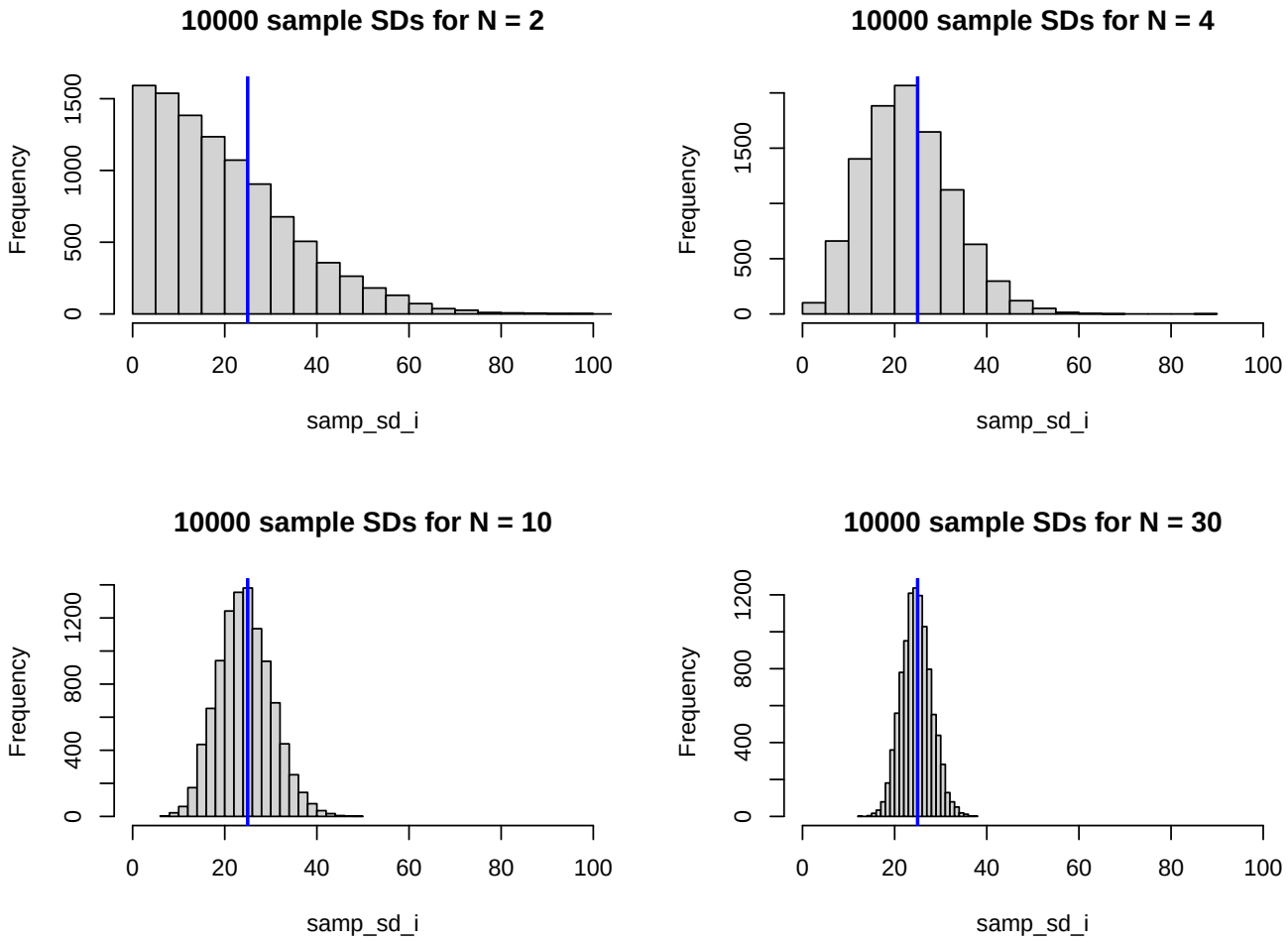

Figure 4.7: Histograms of sample standard deviations from repeated samples of the same barnacle population. The true population standard deviation is shown in blue.

```
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"
)
```

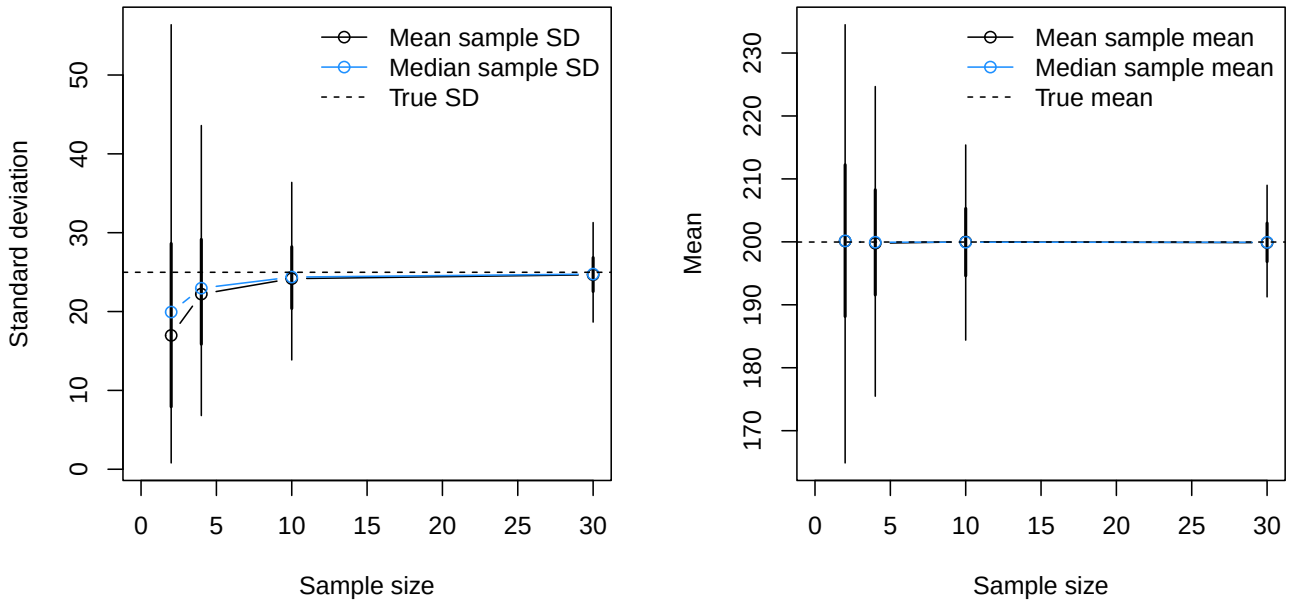

Figure 4.8: Mean (black), median (blue), and 50% and 95% quantiles (vertical lines) for (left) sample standard deviations at each sample size compared to the true population standard deviation (dotted line) or for the (right) sample means.

Q. 4.19

Does the difference between the mean (black) and median (blue) in Figure 4.8 match your expectations based on the shape of the distributions in Figure 4.7?

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 Figure 4.7 and Figure 4.8.

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

A 95% confidence interval is the interval that, when calculated on infinite repeated samples, contains the population mean for 95% of those samples (and thus misses the population mean in 5% of the samples).

We can modify Figure 4.6 by adding 95% confidence intervals for each sample from our barnacle population.

```
num_samples <- 5
sample_size <- 3

# plot population
hist(barnacle_diam,
     xlim = c(barnacle_mu-6*barnacle_sigma, barnacle_mu+6*barnacle_sigma),
     ylim = c(0, length(barnacle_diam)/6),
     main = NULL,
     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) / sqrt(sample_size)),
    mean(sample_i) + qt(0.975, (sample_size - 1)) * (sd(sample_i) / sqrt(sample_size))
  )
  arrows(sample_ci[1], y_pos[i], sample_ci[2], y_pos[i],
         col = "red", code = 3, angle = 90, length=0.05)
}
```

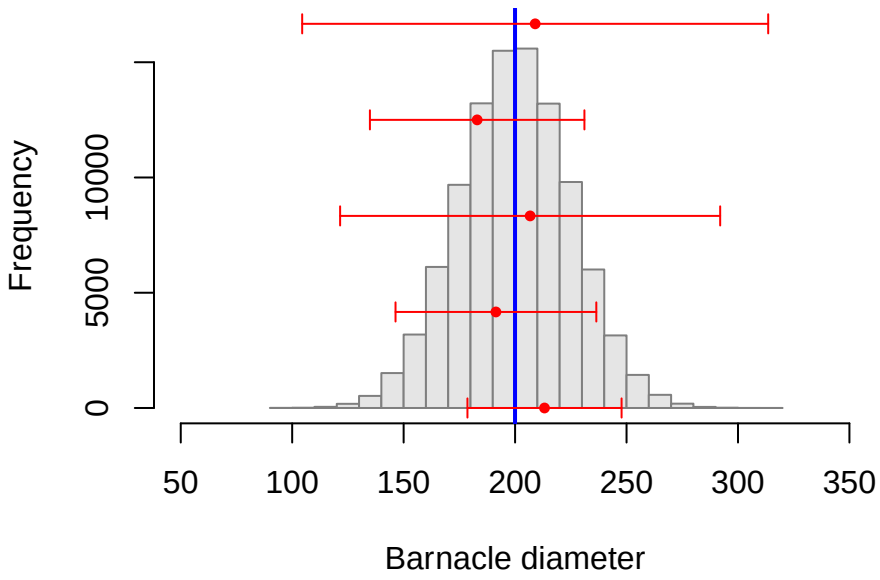

Figure 4.9: Histogram illustrating the barnacle population with population mean (blue) and sample means with 95% CIs (red) repeated across 5 samples.

Q. 4.20

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

Q. 4.21

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

Q. 4.22

What proportion of your 95% CIs would you expect to include the true value of the mean? Does that change for different values of `sample_size`?

Q. 4.23

Find the relevant bit of the code and determine 99% CIs, then 79% CIs. Note that you may need to adjust the x-axis limits which are set to 6 σ on either side of μ in `hist()`).

In the above code, we calculated CIs manually using `qt()`. We can instead simply use `t.test()` and specify whatever confidence level we like:

```
barnacle_sample <- sample(barnacle_diam, size=10)
examp_ttest <- t.test(barnacle_sample, conf.level=0.89) # 89% CIs
examp_ttest
```

```
    One Sample t-test

data:  barnacle_sample
t = 35.493, df = 9, p-value = 5.533e-11
alternative hypothesis: true mean is not equal to 0
89 percent confidence interval:
 196.7245 217.4111
sample estimates:
mean of x
 207.0678
```

```
# Or just the confidence intervals:
examp_ttest$conf.int
```

```
[1] 196.7245 217.4111
attr(,"conf.level")
[1] 0.89
```

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

For sample sizes of 2, 5, and 10, we will collect a sample and calculate the 95% CIs with `t.test()`. As above, we will repeat our sampling a few times to assess the variability among samples of the same size.

```
num_samples <- 10
sample_sizes <- c(2, 5, 10)

CI_df <- expand_grid(N=sample_sizes,
                     sample_id=1:num_samples) |>
  rowwise() |>
  mutate(obs=list(sample(barnacle_diam, N)),
         mn=mean(obs),
         ci_lo=t.test(obs)$conf.int[1],
         ci_hi=t.test(obs)$conf.int[2]) |>
  ungroup() |>
  mutate(N=factor(N, levels=unique(N), labels=paste("N =", sample_sizes)))
ggplot(CI_df, aes(mn, sample_id, xmin=ci_lo, xmax=ci_hi, colour=N)) +
  geom_vline(xintercept=barnacle_mu, colour="cadetblue") +
  geom_point(size=1.5) +
  geom_errorbarh(height=0.15) +
  facet_wrap(~N, ncol=1, scales="free_y", strip.position="left") +
  labs(x="Sample mean and 95% confidence interval") +
  theme_bw() +
  theme(axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y=element_blank(),
        legend.position="none",
```

```
        panel.grid.minor=element_blank())
```

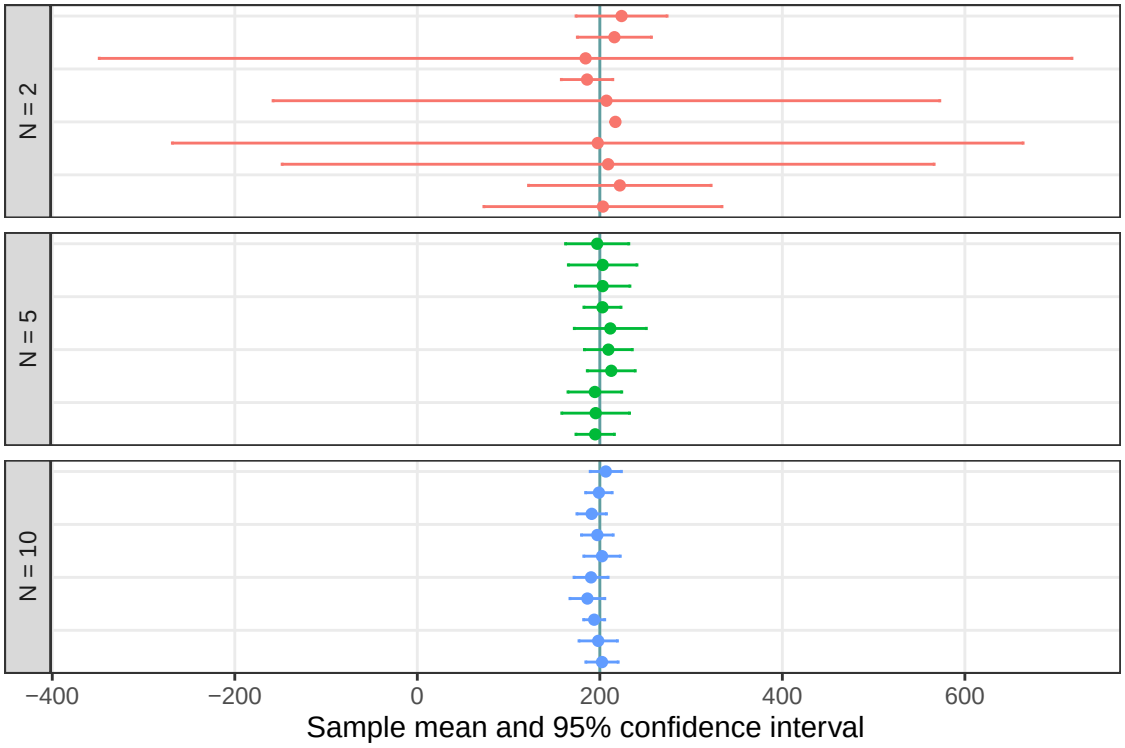

Figure 4.10: Confidence interval size vs. sample size.

Q. 4.24

What do you notice about the average width of CIs in Figure 4.10 as sample size changes? What about the variability in width?

Q. 4.25

What proportion of 95% CIs would you expect to include μ for N=2? For N=100?

4.3 Comparing means (two-sample t tests)

The two-sample t-test is a widely used inferential statistical test. The two-sample t-test is a special case of analysis of variance (ANOVA) where there are only two groups. The results are identical and so we only mention its existence. It is good to be aware of the two-sample t-test because it is so commonly used, but you will be comparing means using ANOVA in Chapter 5.

4.4 Non-parametric Tests

Parametric tests are so named because they estimate population parameters. 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 for analysis 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 Chapter 5).

The t-test is in the 'general linear model' family (which is a subset of the generalized linear modelling family). General linear models are usually used to model continuous data where residual error is approximately normal. If you have count data, you should start with a different member of the GLM family before trying transformations to acheive approximate normality. 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 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

ANOVA and regression

General linear models are a key member of the generalized linear modelling (GLM) family and they are among the most widely used 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 the predictor is categorical and regression when the predictor is continuous.

5.1 Analysis of variance (ANOVA)

ANOVA is a method to compare means among groups and put confidence intervals on the differences between those means. For a predictor with only two categories, ANOVA is identical to the two-sample t-test, so we'll just use ANOVA.

> **i** When you see "analysis of variance", think "analysis of means". The variance in the data is partitioned in different ways to draw conclusions about the means.

In an ANOVA, we compare the means of different groups by analyzing the variance (Figure 5.1). The variance is partitioned into 1) the mean variance between treatment means and the overall mean (thick arrows), and 2) the mean variance within treatments (thin arrows). By comparing the magnitude of these two quantities, we draw conclusions about whether the difference between group means is plausibly due to noise or not.

```
library(tidyverse)
theme_set(theme_classic())
# simulate two populations A and B
sigma <- 1.5
mu_df <- data.frame(Group=c("A", "B"),
                    Lab=c("mu[A]", "mu[B]"),
                    mu=c(12, 17)) |>
  mutate(density=dnorm(mu, mu, sigma))

pop_df <- data.frame(value=seq(min(mu_df$mu) - 3*sigma,
                               max(mu_df$mu) + 3*sigma,
                               length.out=1e3)) |>
  mutate(A=dnorm(value, mu_df$mu[1], sigma),
         B=dnorm(value, mu_df$mu[2], sigma)) |>
  pivot_longer(2:3, names_to="Group", values_to="density")
x_lim <- xlim(min(pop_df$value), max(pop_df$value))
y_lim <- ylim(-max(pop_df$density)*0.1, max(pop_df$density)*1.1)

# simulate sampling
N <- 10
set.seed(1)
samp_df <- data.frame(Group=rep(c("A", "B"), each=N),
                      y=c(rnorm(N, mu_df$mu[1], sigma),
                          rnorm(N, mu_df$mu[2], sigma))) |>
  mutate(ypos=seq(max(mu_df$density)*0.05, max(mu_df$density)*0.6, length.out=n())) |>
```

```
  group_by(Group) |>
  mutate(y_bar=mean(y),
         y_bar_pos=mean(ypos)) |>
  ungroup() |>
  mutate(y_bar_lab=max(ypos)*1.1,
         y_bar_bar=mean(y),
         Lab=paste0("bar(y)[", Group, "]"))

# plot
ggplot(samp_df, aes(y, colour=Group)) +
  # population in background
  geom_hline(yintercept=0, linewidth=0.2, colour="grey80") +
  geom_line(data=pop_df, aes(value, density), linewidth=1, alpha=0.25) +
  geom_text(data=mu_df,
            aes(x=mu, y=-density*0.05, label=Lab),
            vjust=1, parse=T, size=6, alpha=0.5) +
  geom_segment(data=mu_df,
               aes(x=mu, xend=mu, y=-density*0.05, yend=0),
               linewidth=2, linetype=1, alpha=0.5) +
  # observations
  geom_point(aes(y, ypos), size=2.5, shape=1, stroke=1) +
  # group means
  geom_segment(aes(x=y_bar, xend=y_bar, yend=0, y=y_bar_lab), linewidth=0.75, linetype=2) +
  geom_text(aes(x=y_bar, y=y_bar_lab, label=Lab), nudge_y=0.02, parse=T, size=6) +
  # grand mean
  geom_segment(aes(x=y_bar_bar, xend=y_bar_bar, yend=0, y=y_bar_lab), colour="black",
               linewidth=0.75, linetype=2) +
  annotate("text", x=samp_df$y_bar_bar[1], y=samp_df$y_bar_lab[1]+0.02, label="bar(bar(y))",
           parse=T, size=6, colour="black") +
  # within-group variation
  geom_segment(aes(x=y_bar, xend=y, y=ypos, yend=ypos),
               arrow=arrow(ends="both", length=unit(0.05, "inches")), colour="black") +
  # among-group variation
  geom_segment(aes(x=y_bar, xend=y_bar_bar, y=y_bar_lab*0.9, yend=y_bar_lab*0.9),
               arrow=arrow(ends="both", length=unit(0.1, "inches")),
               colour="black", linewidth=1) +
  scale_colour_manual(values=c("steelblue2", "firebrick"), guide="none") +
  x_lim + y_lim +
  labs(x="Response variable", y="")
```

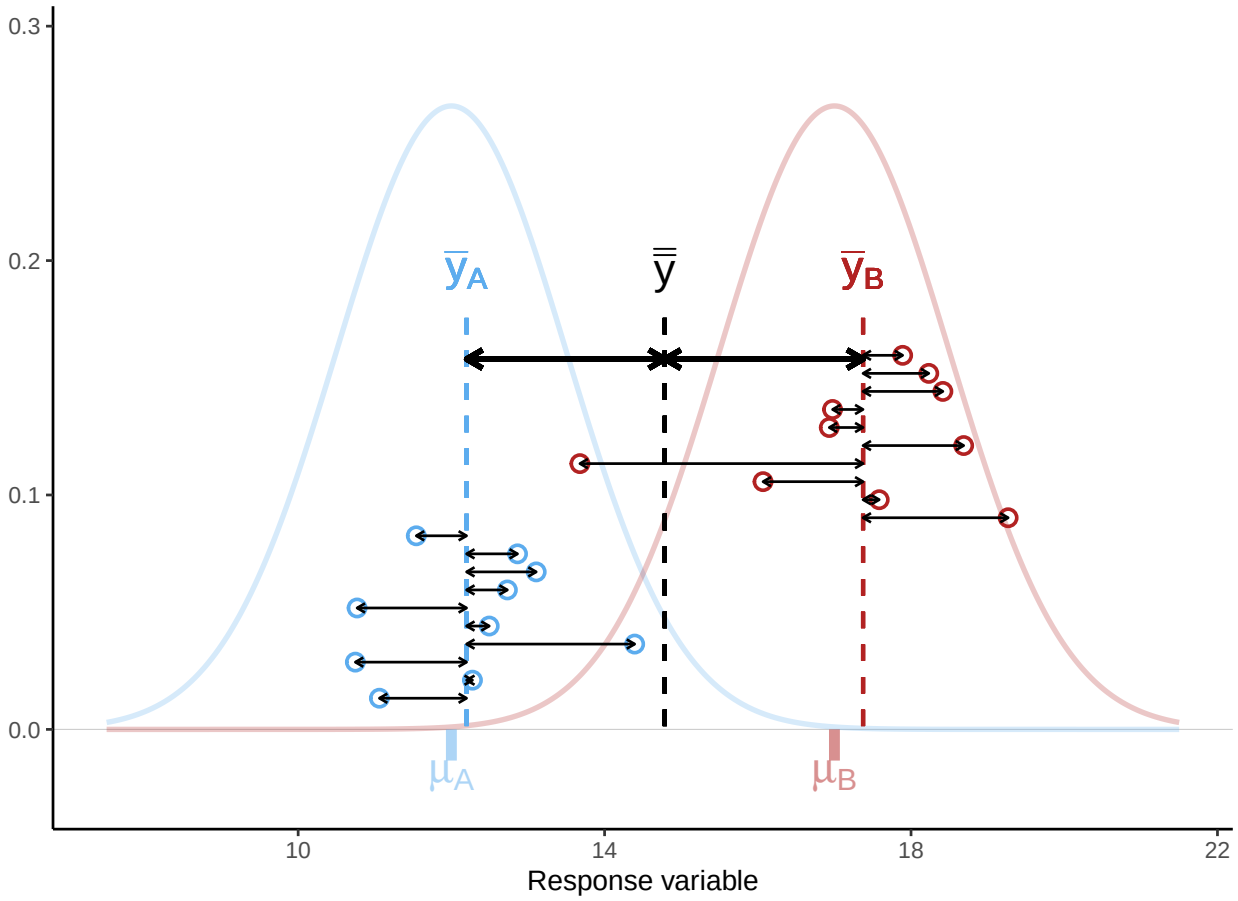

Figure 5.1: Sources of variation (within group vs. among groups) as quantified by ANOVA. Points are observations, dashed vertical lines give sample means for A (blue), B (red), and overall (black). Thick arrows show differences between group means and the overall mean, and thin arrows show differences between observations and group means. The population means (mu[A], mu[B]) and the effect size (mu[B] - mu[A]) are unknown and must be inferred from the sample with confidence intervals.

The null hypothesis in an ANOVA is that there is no difference in group means: $\mu_A = \mu_B = ... = \mu_k$. Under the null hypothesis, the data come from a single population and our groupings are meaningless with regard to the response variable.

In some cases, this may be of interest. What is almost always more interesting, however, is estimating means, differences between means, and our confidence in those.

```
library(tidyverse)
library(readxl)
library(emmeans)
```

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. 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.

5.1.2 One-way ANOVA

One-way ANOVA is a procedure we use to estimate the magnitude of differences between means of two or more 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, 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 the worksheet 'Microalgae'.

```
algae_wide_df <- read_excel("data/H2DS_practicalData.xlsx", sheet = "Microalgae")
```

Check these data as usual.

Q. 5.1

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

Q. 5.2

What assumptions should be met prior to undertaking parametric ANOVA?

Q. 5.3

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

Q. 5.4

What is the sample size in this case? Can we assume sample means will be normally distributed?

Q. 5.5

Are the data normally distributed? Be careful how you word your answer to this question.

Q. 5.6

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

To work with our data, we need to rearrange the `data.frame` to a tidy format with each variable corresponding to one column, and each observation corresponding to one row. We'll use the *tidyverse* as before.

```
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)
```

```
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, ~
```

Q. 5.7

What do you notice about the variable types? Prepare `algae_df$Treatment` for analysis and plotting.

View solution

The column should be a `factor` with the levels in a logical order. Remember that unless you specify the levels, R will order them alphabetically. In this case, that would be `"high", "low", "medium"`.

```
algae_df$Treatment <- factor(algae_df$Treatment,
                             levels=c("low", "medium", "high"))
glimpse(algae_df)
```

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

```
ggplot(algae_df, aes(Treatment, Yield)) +
  geom_boxplot() +
  geom_point(shape=1)
```

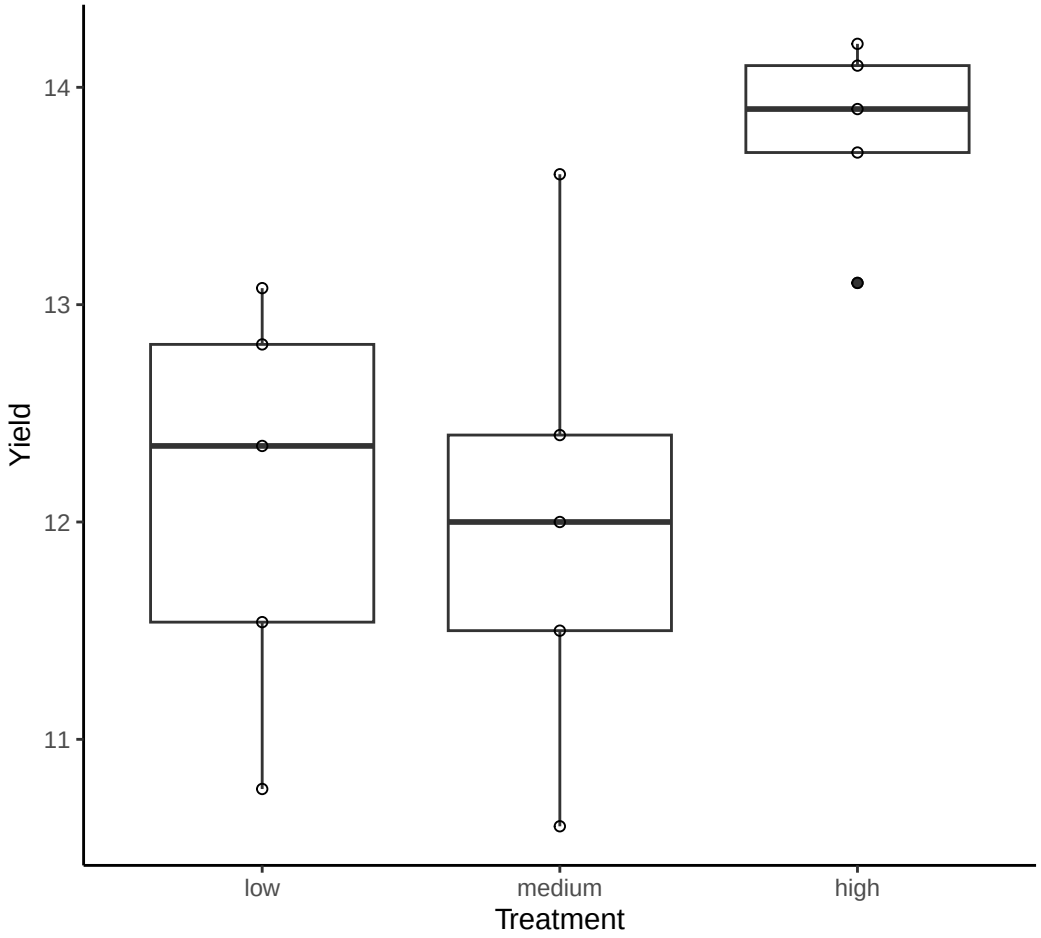

Figure 5.2: Boxplot of algal yield at different light levels.

While there are formal tests to evaluate model assumptions (e.g., the Shapiro-Wilkes test or the Bartlett test), a better way of checking model assumptions is to investigate the residual error. **Residual error** is the difference between the actual data values and the values predicted by the model. Here we have randomly assigned five cultures each of the same species to three specific treatments (light levels).

Residual error is the leftover noise: the difference between your model prediction and each observation.
Sampling error is the error due to the sampling process: the difference between your model prediction and the true population value.

Next to conduct the analysis. This is a one-way ANOVA (one predictor) with fixed effects (we are interested in the differences between specific groups rather than characterizing the variation among groups generally).

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

> 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. An ANOVA table can be produced with `aov(...)` or `anova(lm(...))`, but post-hoc analysis requires `aov()`.

Before we look at the output, let's assess the assumptions using the residuals. Rather than using `qqnorm()` as in Chapter 3, it is better to use `plot(aov_object)`. These default residual plots (Figure 5.3) enable us to rapidly assess whether the model assumptions are reasonable. Remember, the assumptions to assess here are:

1. The *residual error* is normally distributed (rather than variable per se, which is what `qqnorm()` gives).
2. The *residual error* is homoscedastic, with constant variance across groups.

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

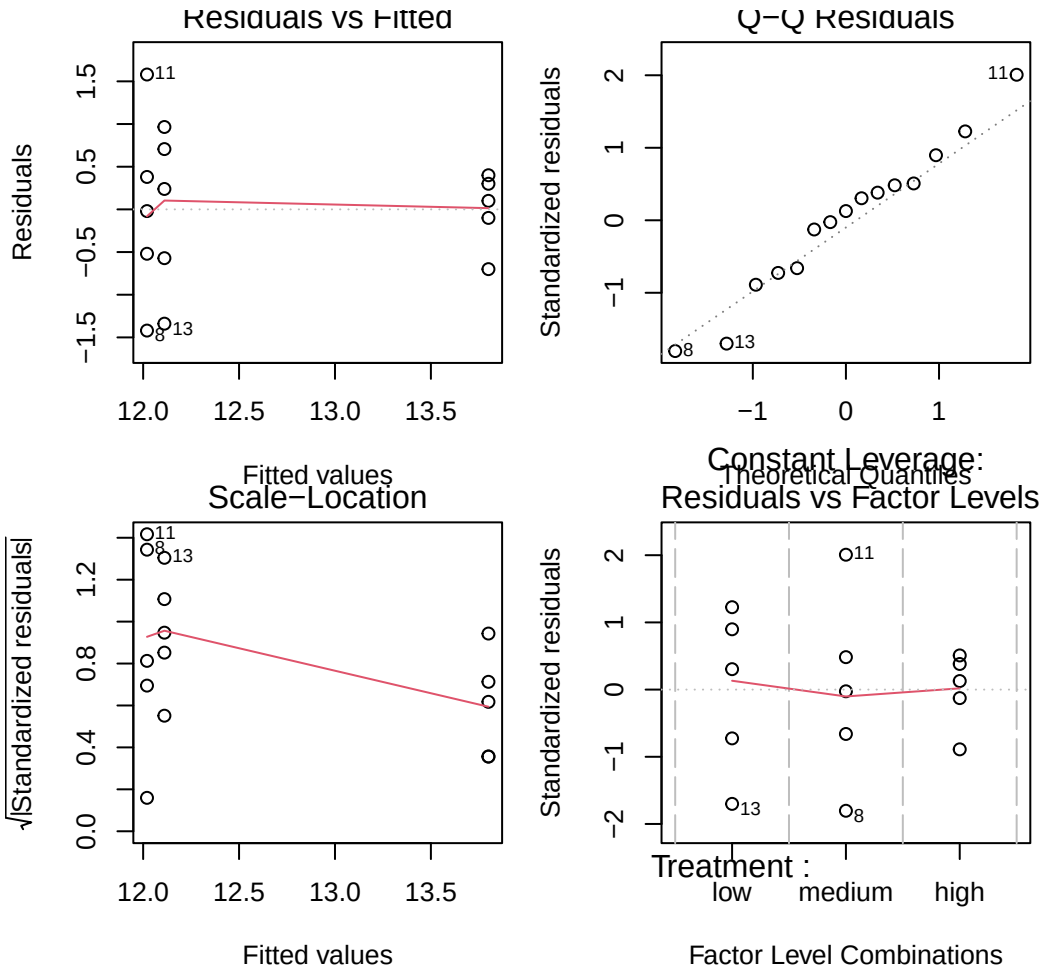

Figure 5.3: Residual plots from one-way ANOVA.

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.

- **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 residual magnitude increases with fitted values, which is a common issue in these types of analyses. 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.

Q. 5.8

What are the 'fitted values' for an ANOVA?

Q. 5.9

Are your effects fixed or random?

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
```

Q. 5.10

Assuming you have chosen $\alpha = 0.05$, what do you conclude? What might you be interested in going on to test next?

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 HSD test.

```
# HSD stands for 'honestly significant difference'
algae_grp_diffs <- TukeyHSD(x = algae_aov, conf.level = 0.95)
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
medium-low  -0.09077539 -1.5758551 1.394304 0.9854641
high-low     1.68922461  0.2041449 3.174304 0.0260570
high-medium  1.78000000  0.2949203 3.265080 0.0194474
```

```
plot(algae_grp_diffs)
```

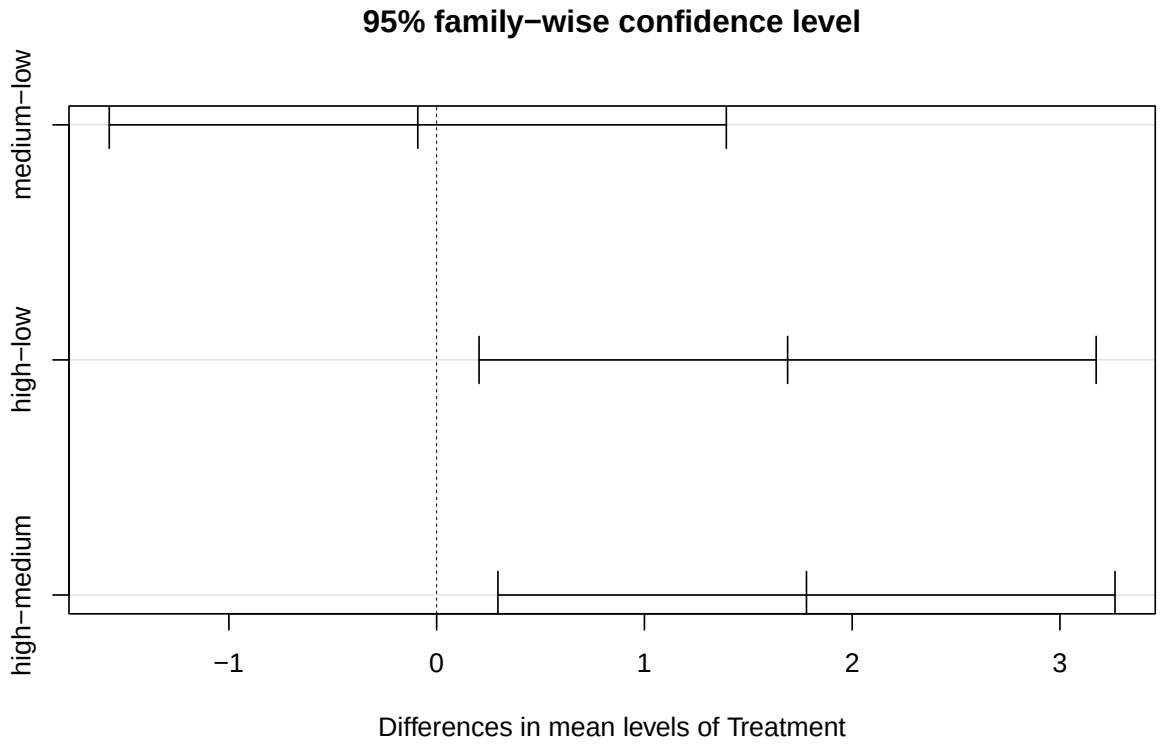

Figure 5.4: Tukey HSD plot from a one-way ANOVA.

The mean yield under high light is significantly higher than under both low light (mean difference [95% CI]: 1.69 [0.204-3.17] $\mu g\ C\ ml^{-1}$; p=0.03) and medium light (1.78 [0.295-3.27] $\mu g\ C\ ml^{-1}$, p=0.02). There was no significant difference between low and medium light levels (p=0.99).

Q. 5.11

In the output from `TukeyHSD()`, why is the first value in `diff` negative?

We of course also want to report the group means with confidence intervals. While we could calculate these individually as in Chapter 4, the simplest way is to use the *emmeans* package. This has the added benefit of using our model's assumption of homoscedasticity in calculating the standard error.

The `emmeans()` function can correctly calculate means and confidence intervals for complex models (`?emmeans`). Our model is simple, so we only need to provide our `aov` object and the name of the predictor variable (`specs=...`). The confidence level is specified with `level=...`, which is set to `0.95` by default.

```
algae_emm <- emmeans(algae_aov, specs="Treatment", level=0.89)
algae_emm
```

```
 Treatment emmean    SE df lower.CL upper.CL
 low         12.1 0.394 12     11.4     12.8
 medium      12.0 0.394 12     11.3     12.7
 high        13.8 0.394 12     13.1     14.5

Confidence level used: 0.89
```

5.1.3 One-way ANOVA (reprise)

Import the 'LimpetDist' sheet from "H2DS_practicalData.xlsx", which gives travel distance of limpets on three different surfaces. Perform a one-way ANOVA of distance predicted by surface type. Make sure you perform appropriate data quality checks and assumption checks. Report the ANOVA table and the pairwise effect sizes with confidence intervals.

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 assumed 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 whether correlation or regression analyses are most appropriate for a given research question and design.

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 quantify 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.

Q. 5.12

What does the plot look like when there is no relationship between the predictor and the response?

Let us now consider an example in which cause and effect does exist. The data in worksheet 'Beetles' in *practical_6.xlsx* 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.

Q. 5.13

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)?

Q. 5.14

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/H2DS_practicalData.xlsx", sheet = "Beetles")
# inspect the dataframe, then make a scatter plot
par(mfrow=c(1,1))
plot(WeightLoss_Mg ~ Humidity, data = beetle_df)
```

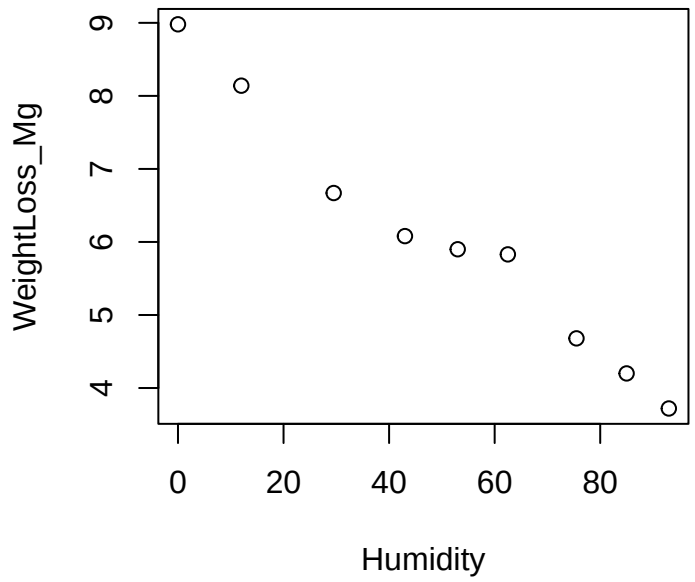

Figure 5.8: Beetle weight loss as a function of relative humidity.

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.

```
# weight loss 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 regression residuals.
```

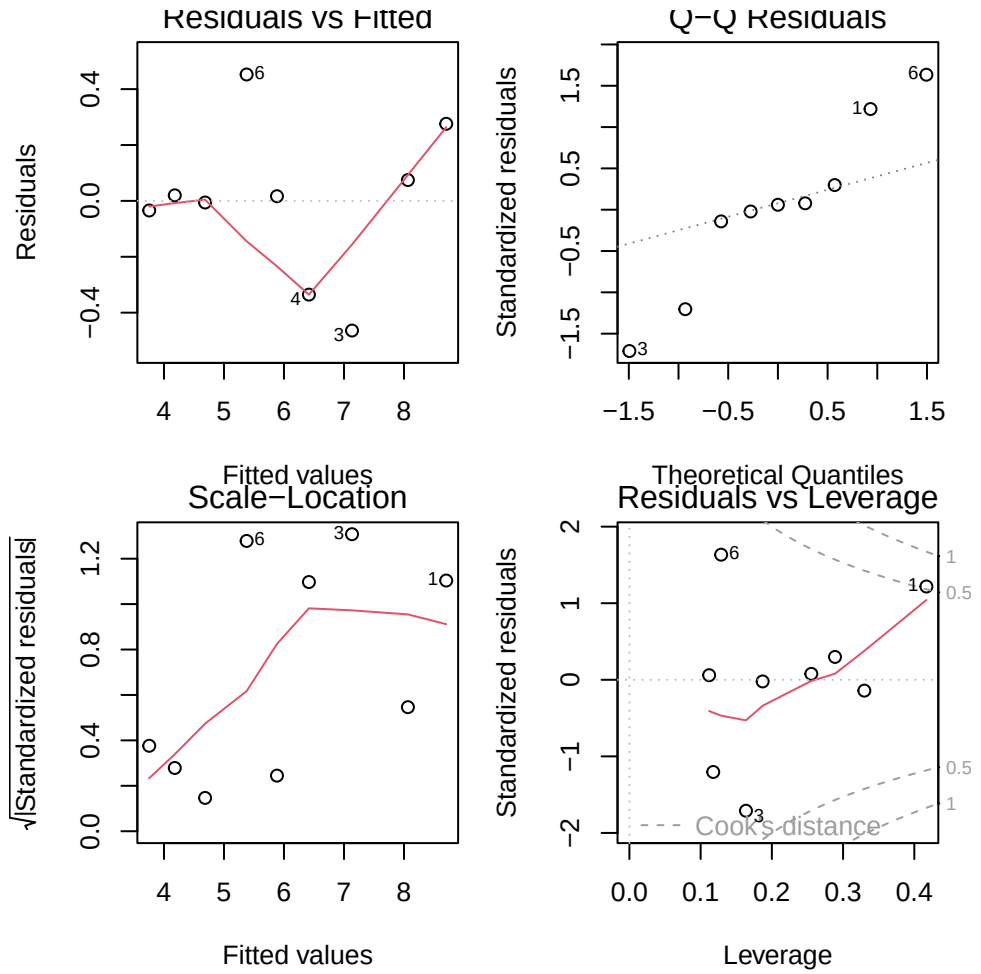

Figure 5.9: Regression diagnostics

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, though the 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.

Q. 5.15

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).

Q. 5.16

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

Q. 5.17

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

Q. 5.18

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.

Q. 5.19

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?

Q. 5.20

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.

If we imagine there were many alternate you's (like in a multiverse) repeating the same experiment on the same population with the same sample size (but independent *samples*), and each 'you' calculated 95% CIs with `confint()`, then 95% of you would have intervals that include the true population intercept and slope. While you do not know if you are in the unlucky 5% that failed to capture the population values, the 95% confidence interval serves as our best estimate for likely values (but see Bayesian statistics for more intuitive intervals!).

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

par(mfrow=c(1,1))
plot(WeightLoss_Mg ~ Humidity, data = beetle_df, ylim=c(3, 10),
     xlab = "Relative humidity (%)", ylab = "Weight loss (mg)")
lines(beetle_df$Humidity, beetle_pred_line[, "fit"])
lines(beetle_df$Humidity, beetle_pred_line[, "lwr"], lty = 2)
lines(beetle_df$Humidity, beetle_pred_line[, "upr"], lty = 2)
```

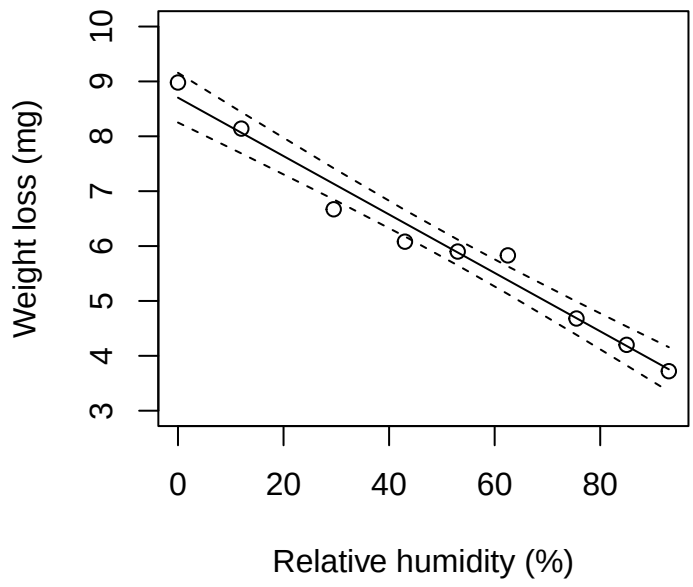

Figure 5.10: Regression line (solid) with upper and lower 95% confidence intervals on the regression line (dashed).

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

Q. 5.21

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

Q. 5.22

Do the confidence intervals in Figure 5.10 run parallel to the regression line?

Q. 5.23

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

Q. 5.24

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)
```

```
      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
```

And we can plot these intervals too:

```
new_humidity_df <- data.frame(Humidity = 0:100)
beetle_pred_line <- predict(beetle_lm,
                            newdata = new_humidity_df,
                            interval = "confidence",
                            level = 0.95)
beetle_pred_obs <- predict(beetle_lm,
                           newdata = new_humidity_df,
                           interval = "prediction",
                           level = 0.95)

par(mfrow=c(1,1))
plot(WeightLoss_Mg ~ Humidity, data = beetle_df, ylim=c(3, 10),
     xlab = "Relative humidity (%)", ylab = "Weight loss (mg)")
lines(new_humidity_df$Humidity, beetle_pred_line[, "fit"])
lines(new_humidity_df$Humidity, beetle_pred_line[, "lwr"], lty = 2)
lines(new_humidity_df$Humidity, beetle_pred_line[, "upr"], lty = 2)
lines(new_humidity_df$Humidity, beetle_pred_obs[, "lwr"], lty = 3)
lines(new_humidity_df$Humidity, beetle_pred_obs[, "upr"], lty = 3)
```

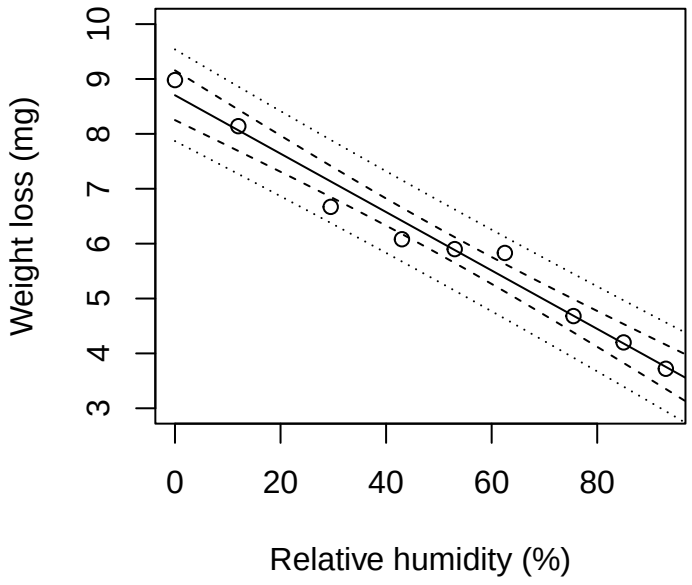

Figure 5.11: Regression line (solid) with upper and lower 95% confidence intervals on the regression line (dashed) and 95% prediction intervals (dotted).

These are prediction intervals and they are broader than confidence intervals. The confidence intervals express your confidence about the *regression line* for the population. The prediction interval expresses your confidence about the distribution of the *observations* for the population.

5.2.4 Regression (reprise)

Import the 'PhosphateCalibration' sheet from "H2DS_practicalData.xlsx" (1st year practical data) into R and perform a linear regression with absorbance predicted by concentration. Perform all appropriate checks for data quality and assumptions. Report relevant parameter estimates with 92% confidence intervals.

This is the workflow for performing a linear regression.

> Most often, researchers report 95% CIs, but occasionally 80%, 90%, or 95%. It would be a bit cheeky to report the 92% CIs, but keep in mind that there is nothing magic about 95% (just like 0.05).

Q. 5.25

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 approximately 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 generalized 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 analyzing 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 generalized 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

Machine Learning

6.1 Multivariate analysis

So far, every model we've used has had a single response variable. **Multivariate analyses** are techniques that allow the analysis of multiple response variables, such as counts of each species within a community. Statistical routines for multivariate analysis are relatively new and have co-evolved with computational capacity. Multivariate data are typically stored as a matrix of samples (rows) 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) Calculation of a dissimilarity matrix
3) Ordination (display) of the dissimilarity matrix

Each of these steps has numerous options, with advantages and disadvantages to each. Multivariate analysis are less inferential than most univariate approaches and implementation can feel subjective. Statisticians are still arguing about the best way to approach multivariate analyses.

Multivariate data occur in a number of common situations, including species inventories, multiple data streams from the same station (e.g., a glider with CTD), and in bioinformatics (e.g., metabarcoding). Many are critical resources in understanding relative change through time.

More detail on the the material we cover here is accessible in Clarke et al. (2014) on Brightspace, entitled *Change in marine communities: An approach to statistical analysis and interpretation.*

```
library(tidyverse)
library(readxl)
library(vegan)
library(plotly)
set.seed(2025)
```

6.2 Non-Metric Multidimensional Scaling (NMDS)

For the NMDS workflow, we'll work through the above steps (`data |> transform() |> dissimilarity() |> ordination()`) first with two simulated datasets and then with a more complex (real) dataset included in the package *vegan*.

Recall that the `^` operator raises each element in a vector to a power. For example, for the 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))`.

6.2.1 1: Data transformation

```
# 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.

6.2.2 2: Dissimilarity matrix

A dissimilarity matrix summarises the dissimilarity between each pair of samples. There are many methods of summarising the dissimilarity (or distance), particularly when that dissimilarity occurs across multiple dimensions (here, genera).

The `vegdist()` function can generate distance matrices using many different methods. See `?vegdist` for more information. We'll use Bray-Curtis for the raw and 4^{th}-root transformed data, and alternative Gower for the alt-log transformed data. See the help page and Brightspace material for more information on these. 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
```

Q. 6.1

Based on the dissimilarity matrices, which sites are more similar (smaller numbers) and which are more dissimilar (bigger numbers)? Does this align with an 'eyeballing' of the data? How has the data transformation changed the resultant dissimilarity matrix?

6.2.3 3: Ordination

Next we want to plot the dissimilarities to visualize which sites are more similar or dissimilar to one another. We will use functions from *vegan*. 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 dissimilarity matrices. We'll explore non-metric multiple dimensional scaling, abbreviated to NMDS, nMDS, nmMDS, or just MDS.

As you might suspect, there are R functions which combine the three steps, though often it is preferable to separate them. 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 (features) are in columns. We can also specify the distance metric, the number of axes to project to, 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)
```

```
Warning in metaMDS(worm_df, distance = "bray", k = 2, autotransform = FALSE, :
stress is (nearly) zero: you may have insufficient data
```

```
Warning in postMDS(out$points, dis, plot = max(0, plot - 1), ...): skipping
half-change scaling: too few points below threshold
```

```
ordiplot(ord_raw, choices = c(1, 2), display = "sites",
         type = "text", main = "NMDS: raw data, Bray-Curtis")
```

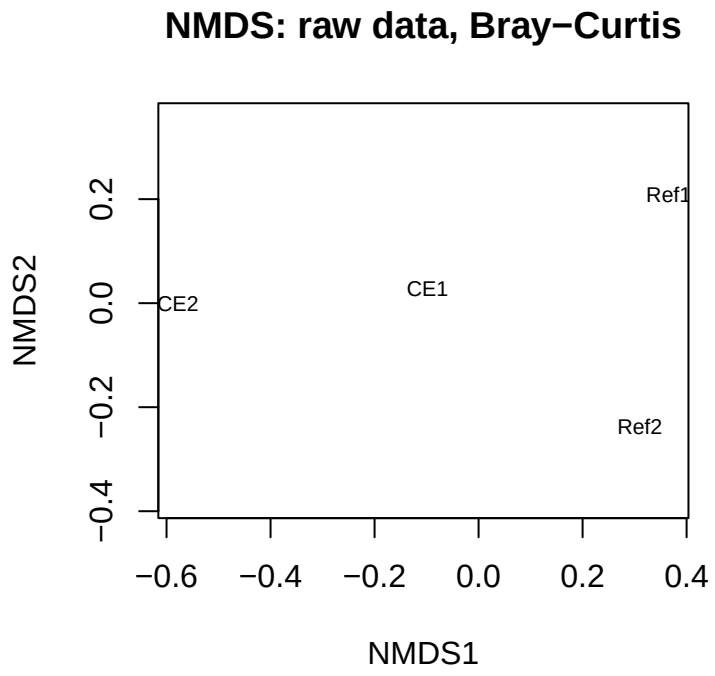

Figure 6.1: Simple pattern in example worm data.

Note that `metaMDS()` involves some randomization, and your plot will change each time you run the code. For fully reproducible code, use `set.seed()`.

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

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

```
Warning in metaMDS(worm_df_4rt, distance = "bray", k = 2, autotransform =
FALSE, : stress is (nearly) zero: you may have insufficient data
```

```
Warning in postMDS(out$points, dis, plot = max(0, plot - 1), ...): skipping
half-change scaling: too few points below threshold
```

Q. 6.2

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

Q. 6.3

Interpret the ordinations, 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")
```

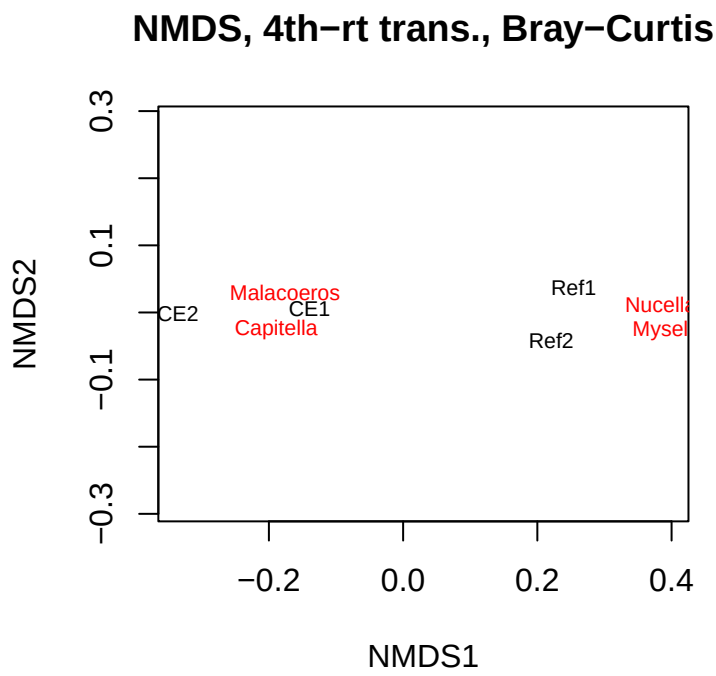

Figure 6.2: Species superimposed onto ordination, indicating species-site associations.

Q. 6.4

Re-plot 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)
```

```
Warning in metaMDS(comm_df, distance = "bray", k = 2, autotransform = FALSE, :
stress is (nearly) zero: you may have insufficient data

Warning in postMDS(out$points, dis, plot = max(0, plot - 1), ...): skipping
half-change scaling: too few points below threshold
```

```
ordiplot(ord_comm, choices = c(1, 2),
         display = c("sites", "species"), type = "text")
```

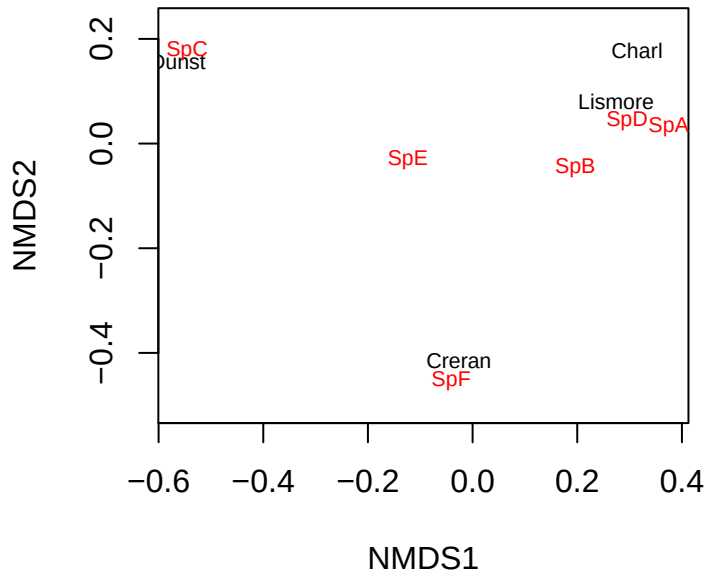

Figure 6.3: Ordination for a simulated dataset showing sites and species.

Q. 6.5

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.

Q. 6.6

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

View solution

```
comm_4rt_df <- comm_df^0.25
ord_comm_4rt <- metaMDS(comm_4rt_df, distance = "bray", k = 2,
                        autotransform = FALSE, trace = FALSE)
```

```
Warning in metaMDS(comm_4rt_df, distance = "bray", k = 2, autotransform =
FALSE, : stress is (nearly) zero: you may have insufficient data
Warning in postMDS(out$points, dis, plot = max(0, plot - 1), ...): skipping
half-change scaling: too few points below threshold
```

```
ordiplot(ord_comm_4rt, choices = c(1, 2),
         display = c("sites", "species"), type = "text")
```

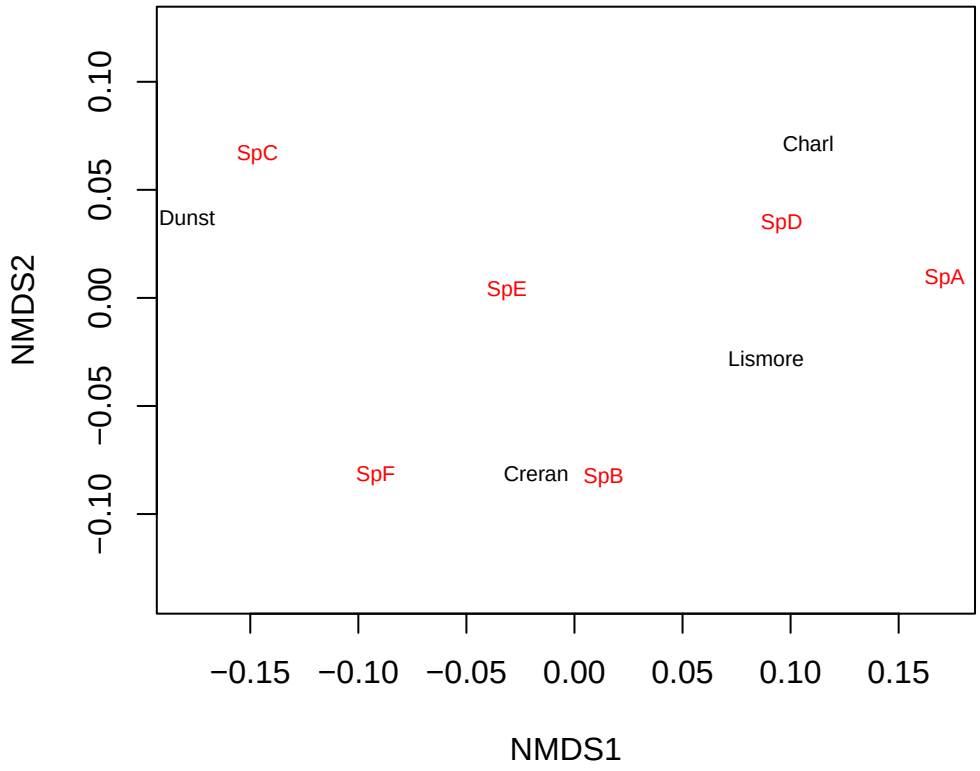

Figure 6.4: Ordination for the 4th root transformed data.

Q. 6.7

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

6.3 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.

Q. 6.8

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

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

Figure 6.5: Shannon diversity

Try plotting `richness`.

Q. 6.9

Do the plots correspond to what you expected?

Q. 6.10

What does `specnumber()` do?

Q. 6.11

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

Q. 6.12

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

6.4 NMDS on 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!). Now we will progress to real observations.

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 characteristics
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
```

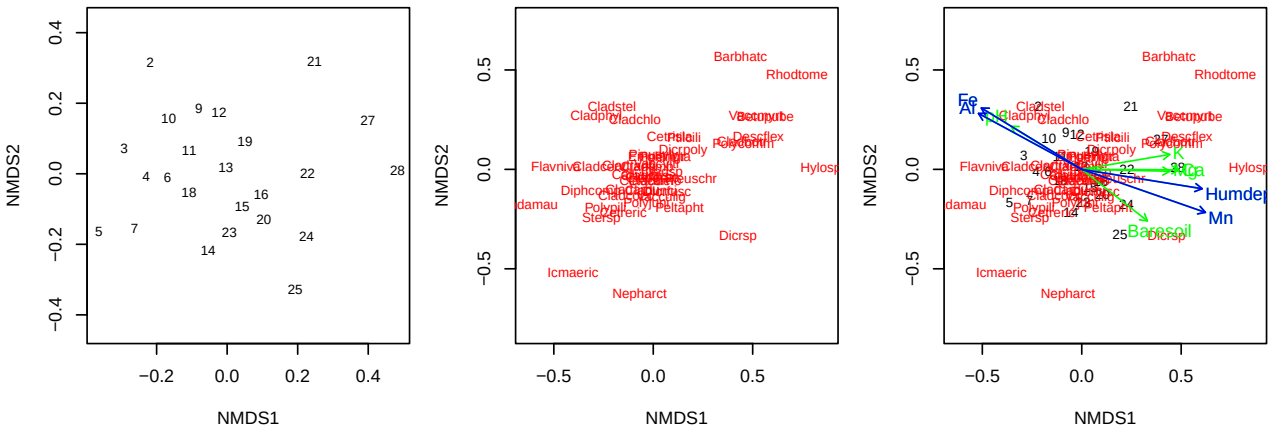

Figure 6.6: vare- data from vegan showing sites only, species only, and both plus environmental overlays.

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 quickly gets 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. 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 any of several or a combination. My advice is only to use transformations that you specify.

6.5 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 consider sites with many shared absences to be more similar, which is often not desirable.

In PCA, it is important that all variables are of a roughly similar magnitude. Environmental data might include all manner of different variables on different scales (e.g., radiant flux in lumens, temperature in C, nutrient/contaminant concentrations in mg/l, coordinates in easting/northing). If used in their raw form, the variables with larger values would have a greater influence on the outcome of the PCA.

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 use is from SAMS Professor Tom Wilding's PhD thesis. He set up an experiment to examine the relative leaching 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 out in dangerous amounts. Granite, the main constituent of this concrete, might also leach some trace elements. He suspended concrete and granite powder in artificial water, constantly agitated it, and measured the leachate concentrations over 100 days Wilding and Sayer 2002.

```
leach_df <- read_excel("data/H2DS_practicalData.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
```

Q. 6.13

Do some exploratory analysis of this dataset. How are the concentrations distributed? Do similar treatments show qualitatively similar distributions? Elements?

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")
```

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")
```

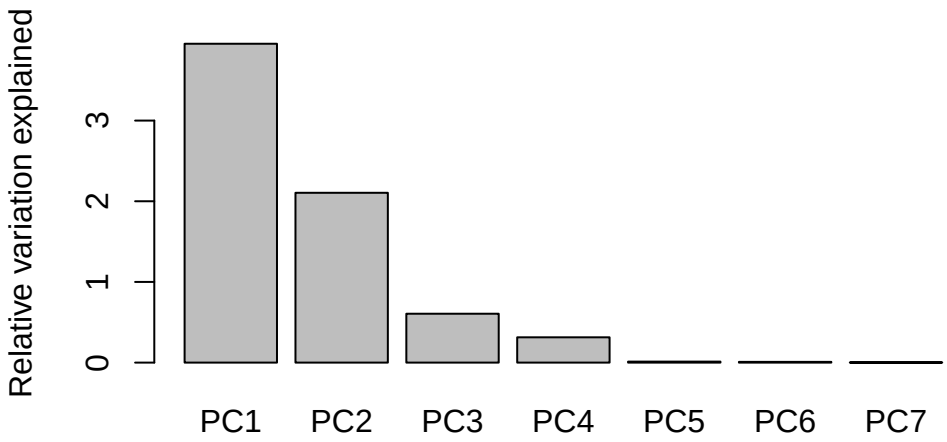

Figure 6.7: Scree plot showing the variance explained by each principal component.

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

You can see from the scree plot that the amount of variance explained declines with principal component as expected, and that there is very little variation left after 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 7D data to about 3D.

Data transformations are critical to PCA analysis, as they are with NMDS. In most PCAs the data are standardized 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)
```

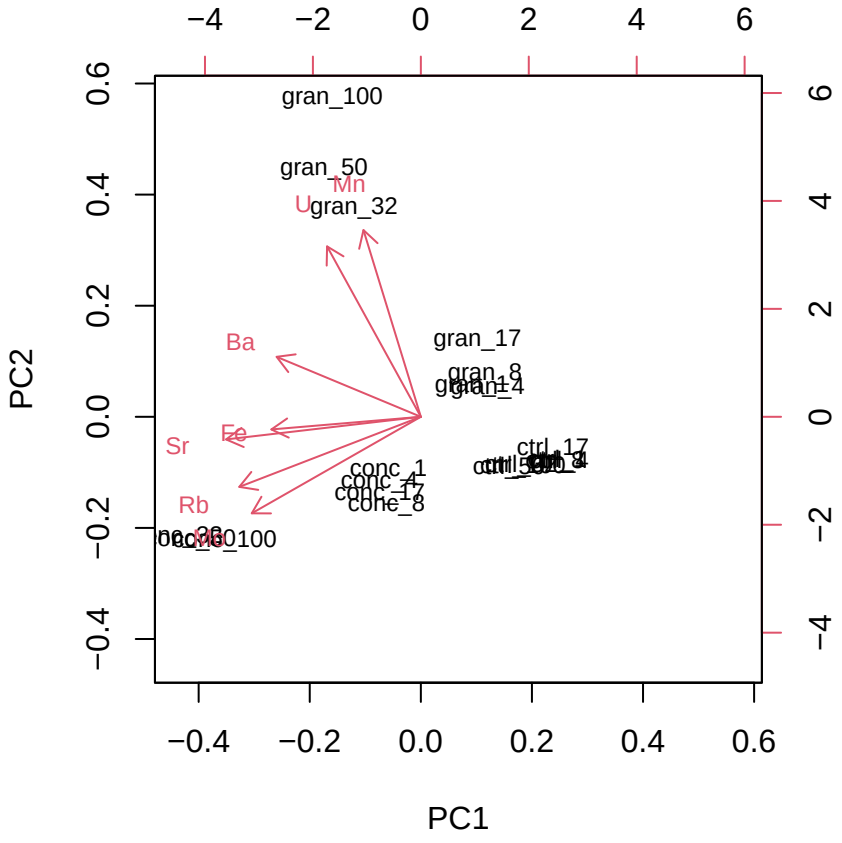

Figure 6.8: PCA of the complete dataset, illustrating the common challenge of overplotting.

To plot more than 2 dimensions you could use a static 3D plot, but these are often difficult to interpret since it is reduced *back* to 2D on a page. Another option is to use colour for the 3rd axis, or, for digital distribution, a package like *plotly* can produce an interactive 3D plot.

```
if(knitr::pandoc_to() == "html") {

arrow_df <- tibble(element=row.names(PCA_leach$rotation),
                   PC1=PCA_leach$rotation[,1]*6,
                   PC2=PCA_leach$rotation[,2]*6,
                   PC3=PCA_leach$rotation[,3]*6)

PCA_leach$x |>
  as_tibble() |>
  mutate(Treat_day = leach_df_wide$Treat_day) |>
  separate_wider_delim(Treat_day, delim="_", names=c("Treatment", "Days")) |>
  mutate(Days=as.numeric(Days)) |>
  plot_ly(type="scatter3d", x=~PC1, y=~PC2, z=~PC3, symbol=~Treatment, color=~Days, mode="markers",
          symbols=c("circle-open", "diamond-open", "cross")) |>
  add_text(data=arrow_df, x=~PC1, y=~PC2, z=~PC3, text=~element, color=I("black"), symbol=NULL)
}
```

The ordination plots the relative positions (in terms of similarity) of the samples. If there are numerous label overlaps, this makes the interpretation of the ordination difficult. If you were producing this for publication you would need to sort this out to present the reader with a figure that communicates the message effectively.

Q. 6.14

Which elements are positively associated with granite and concrete, particularly 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 `|`?

```
#?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)
```

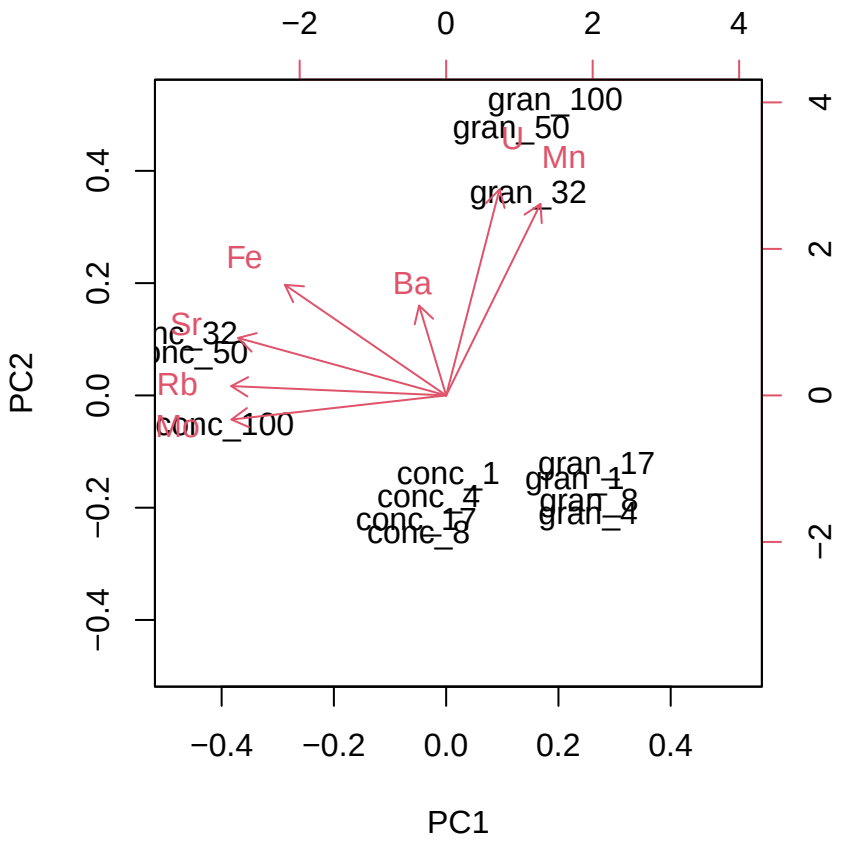

Figure 6.9: PCA of the concrete and granite, illustrating temporal and treatment differences.

Q. 6.15

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/or the cumulative proportion). If the cumulative proportion for the 1st two components is high, then the

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 the variables, you can assess the relative importance of each in driving the patterns in the 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 absolute values on PC1) while Mn and U have high values for PC2 (you could say that PC2 accounts for Mn and U).

6.6 Conclusions

Multivariate analyses are extremely useful in a variety of contexts. The form of analysis is often more qualitative and less inferential compared to univariate analyses we've covered, but this is an active field of research and there are well-developed methods and R packages for more complex analyses (e.g., `ade4`.

Chapter 7

Appendix

7.1 Probability

7.1.1 Probability for equally likely outcomes

The probability of event A occurring is $P(A) = \frac{x}{n}$, where n is the number of trials and x 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:

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

and

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

and when A and B are mutually exclusive, then:

$P(A\ or\ B) = P(A) + P(B)$

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	μ	$\bar{y}$
Variance	σ^2	s^2
Standard deviation	σ	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 formulas are used:

Variance: $\sigma^2 = \frac{SS}{n}$

Standard deviation: $\sigma = \sqrt{\frac{SS}{n}} = \sqrt{\sigma^2}$

When dealing with **samples** the following formulas 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:

$P(x) = \frac{\lambda^x e^{-\lambda}}{x!}$

where λ is the mean and x is the value (count) of interest. Note that in a sample, we use the sample mean $\bar{y}$ as our estimate for λ

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, μ is the mean of the population and σ 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 σ 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, μ 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, μ 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.