H2SMS Practicals KEY

2026-09-07

Table of contents

Preface **1**

Practical sessions . . . 1

I R Basics **3**

R setup **5**

Getting started in R . . . 5

RStudio Projects . . . 5

RStudio Settings . . . 6

R packages and libraries . . . 7

R scripts (.R) vs Quarto documents (.qmd) . . . 7

Writing R code . . . 8

R Basics **9**

Objects & Functions . . . 9

Data Structures . . . 9

Accessing data . . . 11

Logic and relational statements . . . 13

Plotting . . . 14

II Practicals **23**

1 Visualization **25**

1.1 R Review . . . 25

1.2 Basic data exploration . . . 26

1.3 Graphical methods for displaying data . . . 29

1.4 Summary statistics . . . 42

1.5 Conclusions . . . 53

2 Binomial and Poisson distributions **55**

2.1 The Binomial distribution . . . 55

2.2 The Poisson distribution . . . 63

2.3 The Poisson approximation of the binomial model . . . 71

2.4 Conclusions . . . 74

3 Normal distribution **75**

3.1 Using the normal distribution . . . 75
3.2 Normal model adequacy . . . 79
3.3 Quantiles . . . 79
3.4 Testing for normality . . . 83
3.5 Data transformations . . . 85
3.6 The Central Limit Theorem . . . 88
3.7 The standard error of the mean . . . 91
3.8 Normal approximations . . . 93
3.9 Conclusions . . . 96

4 The t-distribution and confidence intervals **97**

4.1 Single sample t-tests . . . 98
4.2 Confidence Intervals . . . 105
4.3 Comparing means (two-sample t tests) . . . 113
4.4 Non-parametric Tests . . . 113
4.5 Conclusions . . . 113

5 Appendix **115**

5.1 Probability . . . 115
5.2 Univariate statistics . . . 115
5.3 The Binomial distribution . . . 116
5.4 The Poisson distribution . . . 116
5.5 Z scores . . . 117
5.6 Samples taken from a population . . . 117
5.7 The t distribution . . . 117
5.8 ANOVA . . . 118
5.9 Regression & correlation . . . 118

III Problem Sets **119**

Problem Set 1 **121**

R Basics . . . 121
Data munging . . . 122
Plotting . . . 124

Problem Set 2 **135**

Binomial distribution . . . 135
Poisson distribution . . . 139

Problem Set 3 **143**

Normal distribution . . . 143
Standard error of the mean . . . 152

Extra: CA1 **159**

Overview . . . 159
Iris . . . 159
Air quality . . . 164
Coral predation . . . 166
Earthquakes . . . 168

Problem Set 4 **171**

Sample mean vs. a reference value . . . 171
Confidence Intervals . . . 175

IV Tutorials 179

***tidyverse* intro** **181**

What makes a dataset tidy vs. messy? . . . 181
Key packages in the *tidyverse* . . . 182
Review: Piping . . . 182
Tibbles . . . 182
Munging data . . . 183
Reshaping data . . . 185
ggplot2 . . . 186
Working with strings . . . 194
Working with dates . . . 197

Preface



Practical sessions

In these practicals, you will apply concepts you learn throughout the H2 Statistics for Marine Science module. 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 Statistics for Marine 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.

Each week, you are asked to complete a short problem set available on Brightspace and submit a rendered .html. Solutions will be posted the following week.

The practicals are 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.

Part I

R Basics

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 **H2SMS** 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 H2SMS_practicalData.xlsx in *Practicals* and move it into your newly created **data** folder.
7. Close RStudio (to learn how to open appropriately)

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.

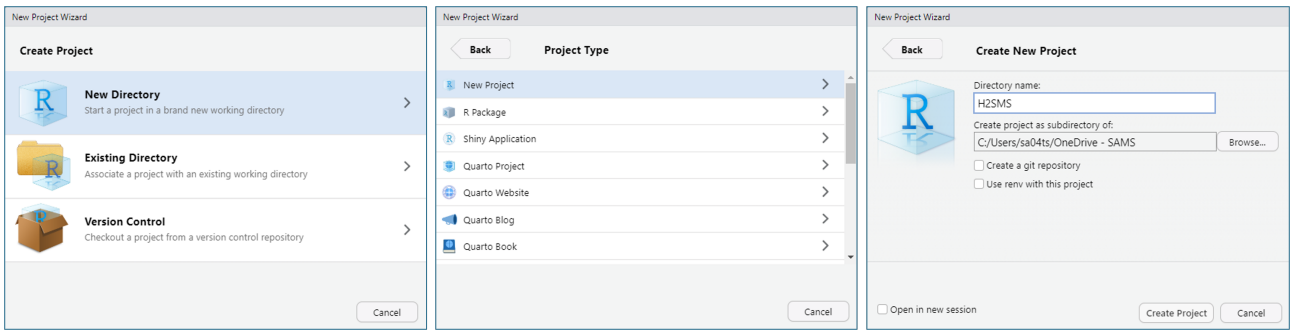


Figure 1: Setting up a new RStudio project.

I repeat: **Open H2SMS.Rproj** instead of the .R or .qmd file you plan to work on!
Opening the .R or .qmd from Windows Explorer will *not* set your R working directory correctly!

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

1. Open Windows Explorer (or Finder on Mac) and find your **H2SMS** folder.
2. Double click on **H2SMS.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. Create a new .R file for that week's practical.

To create a new .R file, click *New Blank File* at the top of the **Files** panel and selecting *R Script*. Name it *Practical_1.R* or similar.

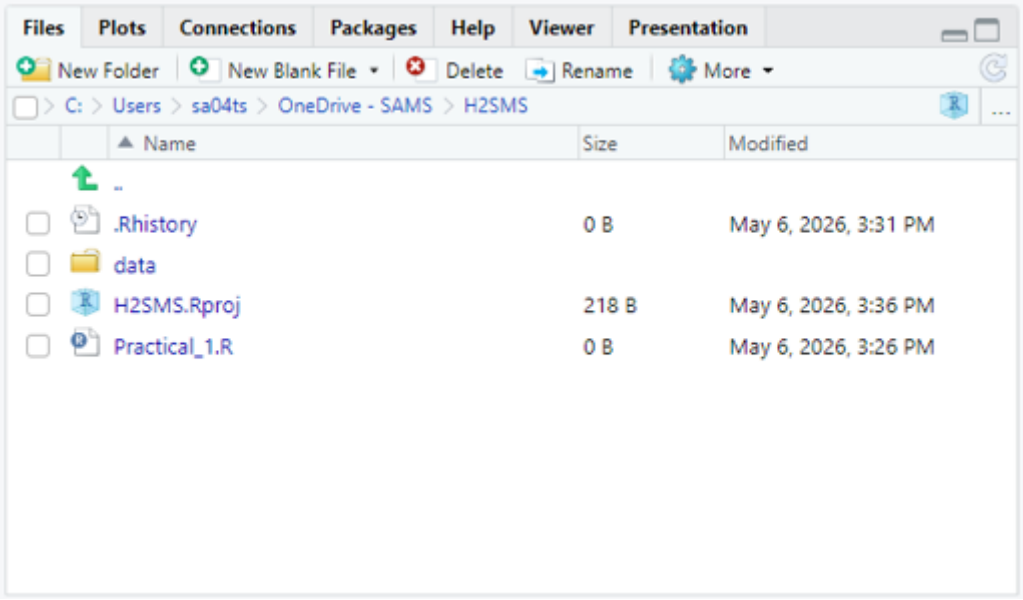


Figure 2: H2SMS directory structure.

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)`.

Install the following packages which we will use this semester:

```
install.packages("tidyverse")
install.packages("emmeans")
install.packages("plotly")
install.packages("cowplot")
```

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. You are welcome to use either .R or .qmd files during the practicals.

Note that the project will be submitted as a .qmd file with a version rendered to .html, and the problem sets should be submitted as .html rendered from the .qmd files linked at the top of each problem set.

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

R Basics

View original

```
library(tidyverse)
theme_set(theme_classic())
```

Objects & Functions

In R, information is stored as *objects*. For our purposes, there are three main structures for storing data. These vary in the number of dimensions and in how many types of variables they can store. We name objects and assign a data structure to that name. This takes the form:

```
# objectName <- dataStructure
```

Functions perform operations on objects. They have names, like objects, but the function name is followed by brackets: `function()`. Functions often have *arguments* that must be specified. For example, the function `mean()` must be provided with data to calculate the mean of.

R will interpret any unquoted text followed by brackets as a function, and any unquoted text as an object name.

Data Structures

Vectors

Vectors are one dimensional and are made up of a sequence of *elements*. Each element stores one value. All values within a vector must be the same data type (e.g., `numeric`, `character`, `logical`, etc). There are several ways to create a vector.

Here are some numeric vectors:

```
vec_a <- c(1, 4, 6)
vec_b <- 12:15
vec_c <- seq(1, 15, by=2)
```

Here are some character and logical vectors:

```
vec_d <- c("Site A", "Site B")
vec_e <- c(TRUE, FALSE, FALSE)
```

There are several useful functions for identifying characteristics of a vector:

```
length(vec_a) # how many elements?
```

```
[1] 3
```

```
class(vec_b) # what type of data?
```

```
[1] "integer"
```

```
str(vec_e) # what is the structure?
```

```
 logi [1:3] TRUE FALSE FALSE
```

All values within a vector must be the same type. If you try to mix data types, R will convert them to a single type.

```
vec_f <- c(1, 3.2, "magnesium")
str(vec_f)
```

```
 chr [1:3] "1" "3.2" "magnesium"
```

Matrices

Matrices are similar to vectors, but are two dimensional. They have rows and columns. They can be created using the function `matrix()`.

```
mx_A <- matrix(1:8, nrow=4, ncol=2)
mx_A
```

```
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

```
str(mx_A)
```

```
 int [1:4, 1:2] 1 2 3 4 5 6 7 8
```

```
mx_B <- matrix(c("A", "B", "C", "D"), nrow=2)
mx_B
```

```
     [,1] [,2]
[1,] "A"  "C"
[2,] "B"  "D"
```

```
str(mx_B)
```

```
 chr [1:2, 1:2] "A" "B" "C" "D"
```

Like vectors, all data must be the same type.

Data frames

Data frames are the most commonly used data structure by most users. They are two dimensional, but each column can be a different data type. They can be created directly using `data.frame()` or by loading data from external sources with functions like `read.csv()` or `read_excel()`.

```
# columnName = vectorOfData
df_A <- data.frame(columnA=1:3,
                   columnB=c("A", "B", "C"),
                   columnC=c(T, F, F))
df_A
```

```
  columnA columnB columnC
1       1       A    TRUE
2       2       B   FALSE
3       3       C   FALSE
```

```
str(df_A)
```

```
'data.frame':   3 obs. of  3 variables:
 $ columnA: int  1 2 3
 $ columnB: chr  "A" "B" "C"
 $ columnC: logi  TRUE FALSE FALSE
```

> Matrices and data frames must be rectangular: all columns have the same number of rows.

Accessing data

We typically perform operations on specific parts of our data. For example, we might calculate the mean of one variable, create a plot of one variable against another, or simply wish to view particular data points. Accessing parts of our data is called *extracting* or *subsetting*, and the method depends on the data structure.

Vectors

R organizes information using indexed elements. Indexes identify which element is which, starting at 1 for the first, and counting upward.

To extract a particular element from a vector, we specify that element's index in square brackets.

```
vec_a
```

```
[1] 1 4 6
```

```
vec_a[1]
```

```
[1] 1
```

```
vec_a[3]
```

```
[1] 6
```

To extract multiple elements, we can use a vector of indexes

```
vec_a[2:3]
```

```
[1] 4 6
```

```
vec_a[c(1, 3)]
```

```
[1] 1 6
```

Matrices

Extracting values from matrices is identical as for vectors, but requires the row index and column index since there are two dimensions.

```
mx_A
```

```
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

```
mx_A[2, 1] # [rows, columns]
```

```
[1] 2
```

```
mx_A[1:2, 2]
```

```
[1] 5 6
```

If we do not specify the rows or the columns, R will return all of them.

```
mx_A[1, ] # row 1, do not specify columns
```

```
[1] 1 5
```

```
mx_A[ , 1] # column 1, do not specify rows
```

```
[1] 1 2 3 4
```

Data frames

Square brackets work for data frames identically to matrices. An additional method for data frames is to use the column names to extract a single column. Columns are accessed using `$` with the syntax `dataframeName$columnName`.

```
df_A
```

```
  columnA columnB columnC
1       1       A    TRUE
2       2       B   FALSE
3       3       C   FALSE
```

```
df_A$columnA
```

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

```
df_A$columnB
```

```
[1] "A" "B" "C"
```

Note that this returns a vector with the column contents. These can be extracted as above.

```
df_A$columnB[2:3]
```

```
[1] "B" "C"
```

Logic and relational statements

We often wish to compare values. R has several operators (a type of function) for making comparisons that return `TRUE` or `FALSE`:

- `==` : equal to
- `!=` : not equal to
- `>` : greater than
- `>=` : greater than or equal to
- `<` : less than
- `<=` : less than or equal to
- `|` : or

These can be used with any data structure.

```
vec_b
```

```
[1] 12 13 14 15
```

```
vec_b == 13
```

```
[1] FALSE  TRUE FALSE FALSE
```

```
vec_c
```

```
[1]  1  3  5  7  9 11 13 15
```

```
vec_c <= 5
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
```

```
vec_d
```

```
[1] "Site A" "Site B"
```

```
vec_d == "Site A"
```

```
[1]  TRUE FALSE
```

```
vec_d != "Site A"
```

```
[1] FALSE  TRUE
```

These statements are often useful for subsetting data, since a logical vector (`TRUE`/`FALSE`) can be used inside square brackets to specify which elements should be extracted.

```
vec_c
```

```
[1]  1  3  5  7  9 11 13 15
```

```
vec_c <= 5
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
```

```
vec_c[vec_c <= 5]
```

```
[1] 1 3 5
```

And for 2D objects like data frames, we can use a logical vector to identify which rows we'd like, and then state which columns to keep.

```
df_A
```

```
  columnA columnB columnC
1       1       A    TRUE
2       2       B   FALSE
3       3       C   FALSE
```

```
df_A$columnB[df_A$columnB == "C"]
```

```
[1] "C"
```

```
df_A[df_A$columnB == "C", ] # specify rows, do not specify columns
```

```
  columnA columnB columnC
3       3       C   FALSE
```

```
df_A[df_A$columnB == "C", 1:2] # specify rows, specify columns
```

```
  columnA columnB
3       3       C
```

> **i** Note that these objects never actually changed because we never assigned the output (using `<-`). To store the output of an operation, you can overwrite the original object or create a new object:
>
> ```
> vec_x <- 1:5
> vec_x2 <- vec_x[3:5] # store as a new object; vec_x is unchanged
> vec_x <- vec_x[4:5] # overwrite; vec_x is now length 2
> ```

Plotting

Scatter plots

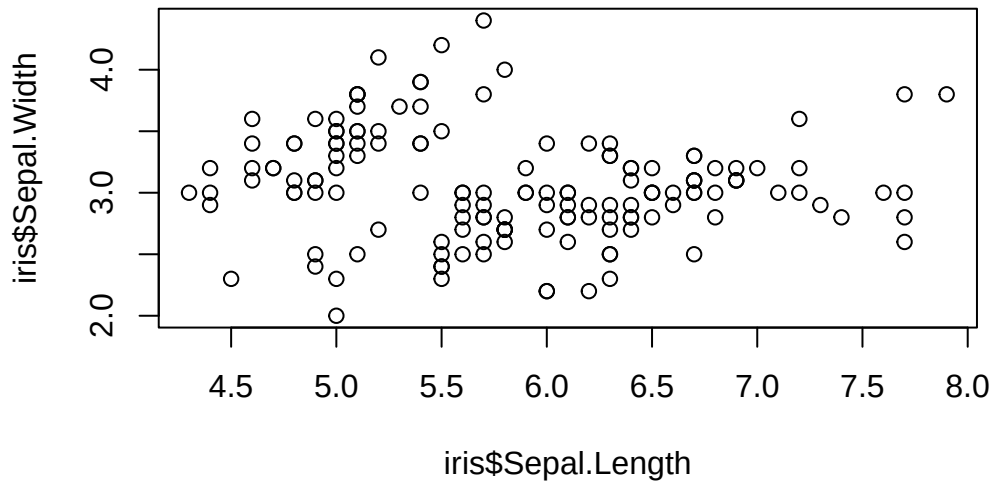
To make a scatter plot of two continuous variables, we can use either `plot()` or `ggplot()`. Further, there are two different approaches for `plot()`.

For illustration, we'll plot `Sepal.Length` on the x-axis and `Sepal.Width` on the y-axis.

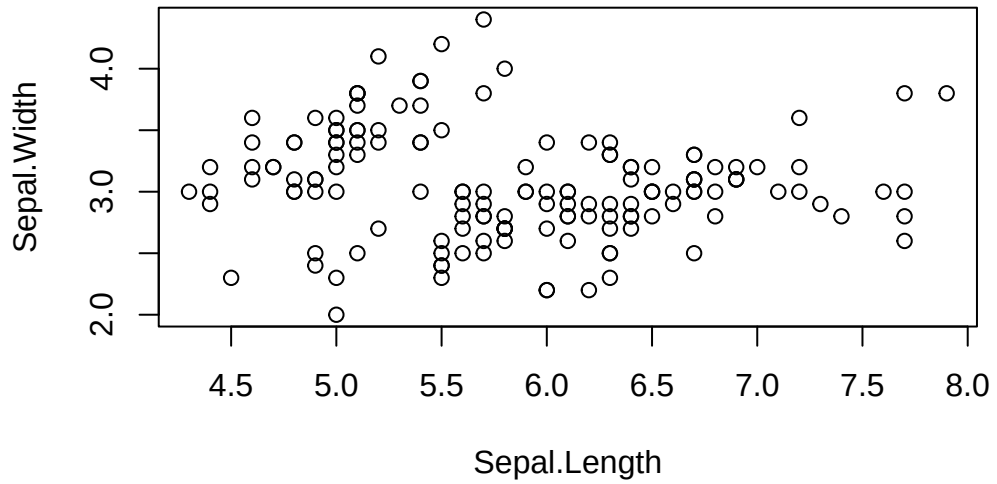
```
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

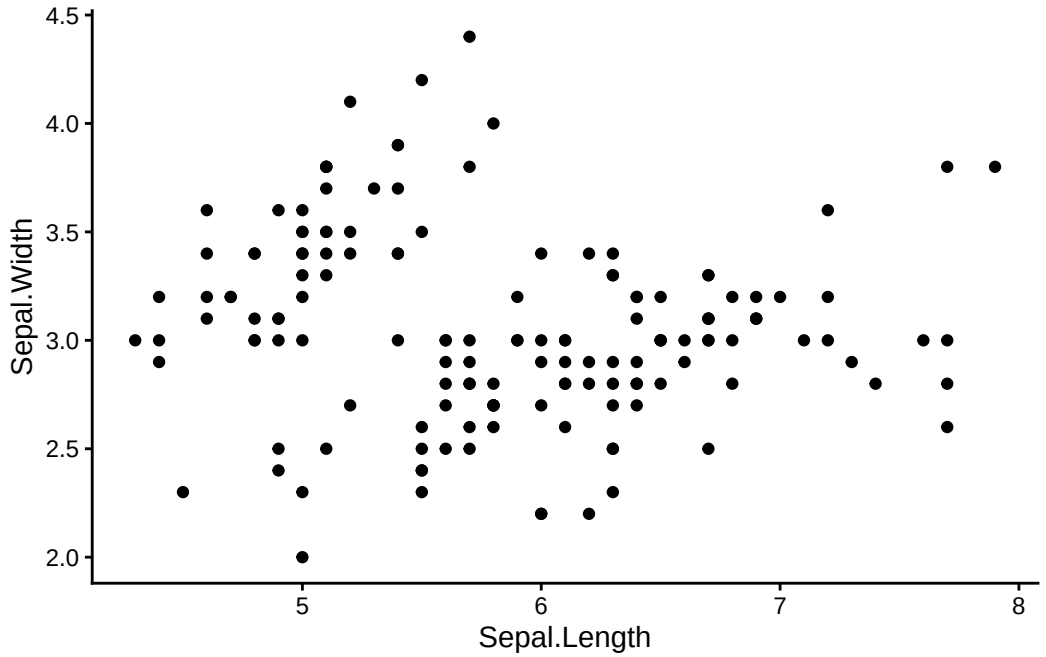
```
# plot(x_vector, y_vector)
plot(iris$Sepal.Length, iris$Sepal.Width)
```




```
# plot(y_column ~ x_column, data=dataframe_name)
plot(Sepal.Width ~ Sepal.Length, data=iris)
```

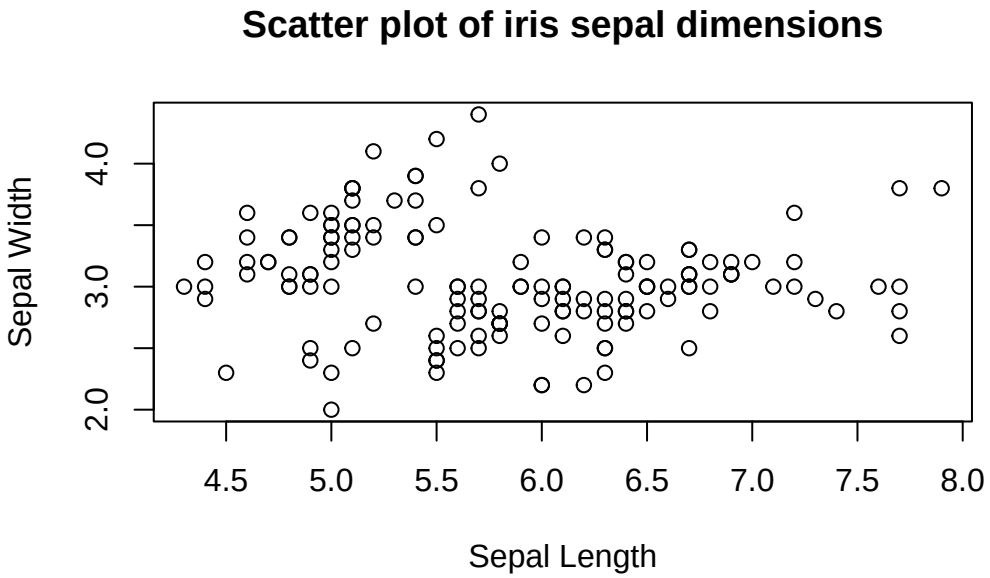


```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width)) +
  geom_point()
```

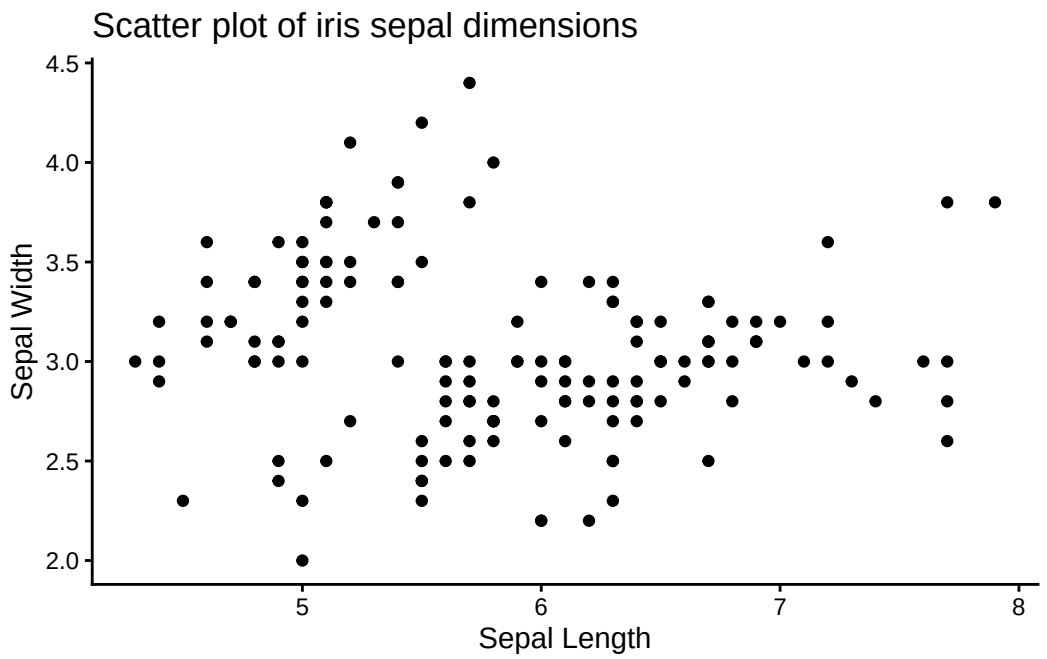


`plot()` is convenient for simple plots, but adding extra detail is often easier with `ggplot()`. Axis labels and titles can be added to each.

```
plot(Sepal.Width ~ Sepal.Length, data=iris,
     xlab="Sepal Length", ylab="Sepal Width",
     main="Scatter plot of iris sepal dimensions")
```



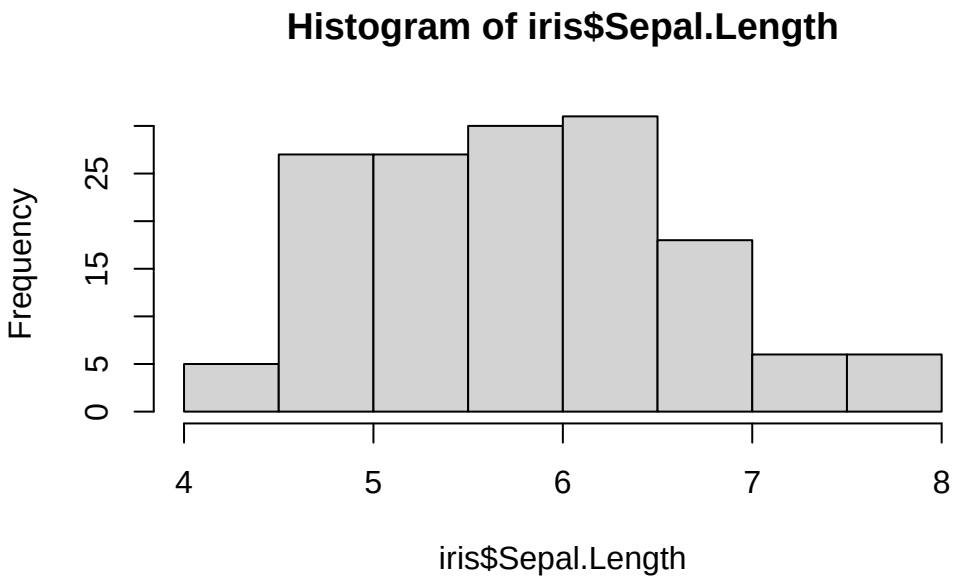

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width)) +
  geom_point() +
  labs(x="Sepal Length",
       y="Sepal Width",
       title="Scatter plot of iris sepal dimensions")
```



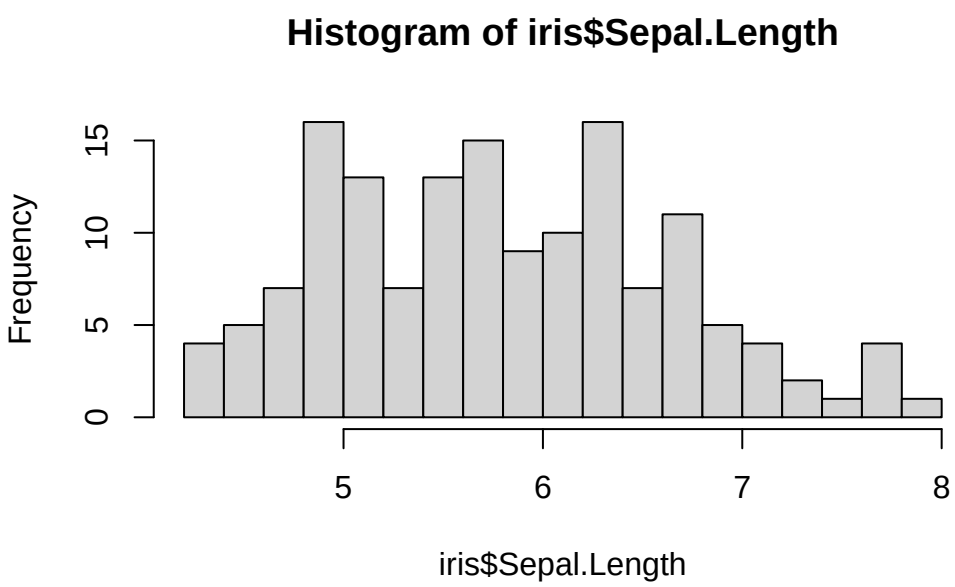
Histograms

Histograms are a common way to visualize the distribution of a continuous variable. We can use `hist()` or `ggplot()`. Note that these have different defaults for bin width, which should always be explored a bit anyway.

```
hist(iris$Sepal.Length)
```

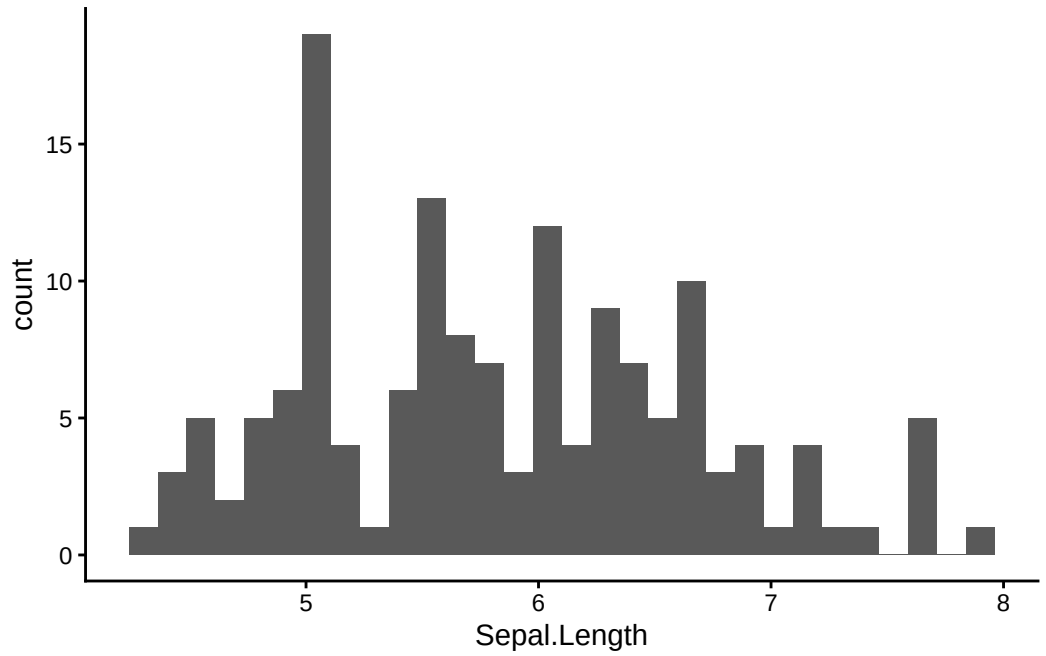


```
hist(iris$Sepal.Length, breaks=15)
```

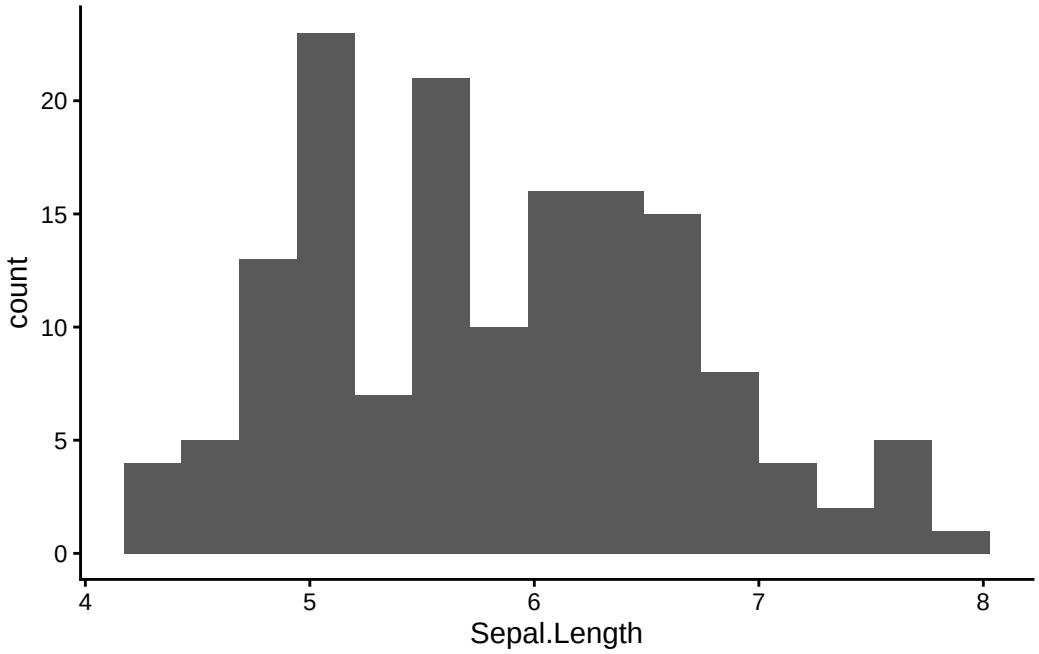



```
ggplot(iris, aes(Sepal.Length)) +
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```




```
ggplot(iris, aes(Sepal.Length)) +
  geom_histogram(bins=15)
```

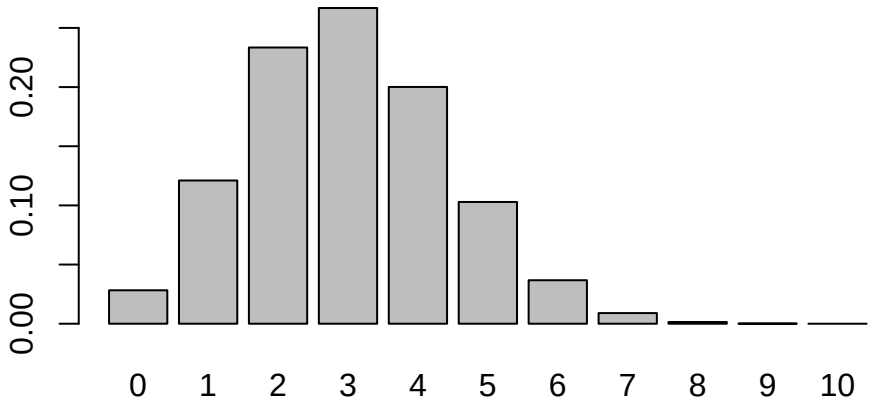


Bar plots

Bar plots visualize discrete distributions. They show the frequency of observations for each possible value. They are suited for discrete variables (e.g., counts), ordinal variables, or categorical variables.

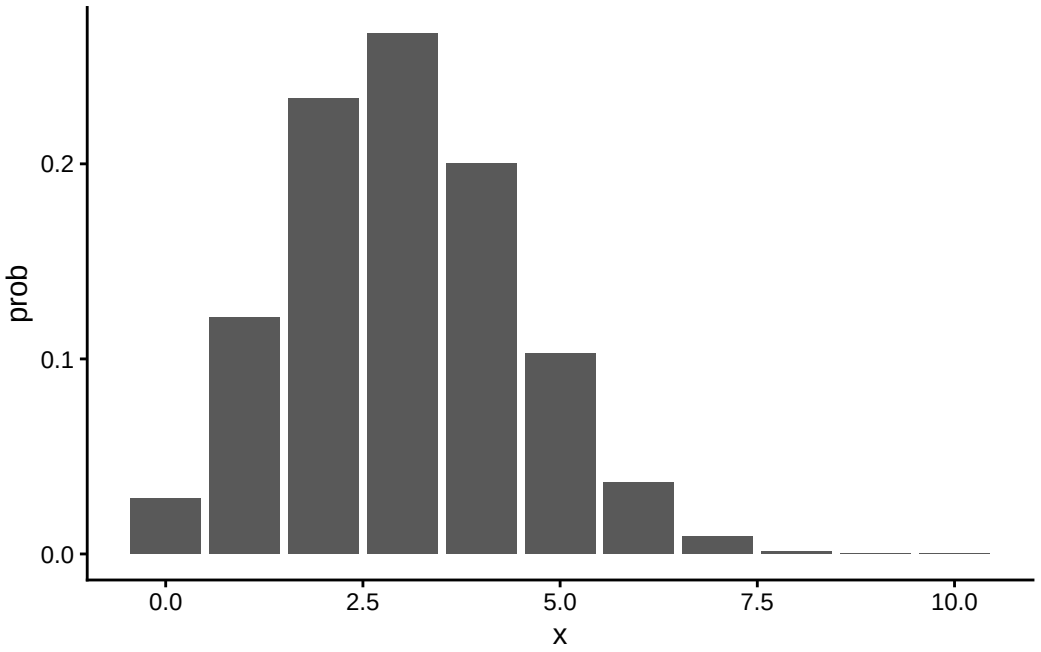
The `barplot()` function and `ggplot()` equivalent are somewhat different in this case. `barplot()` requires a vector of bar heights and an optional vector of names using the `names.arg=` argument.

```
binom_prob_df <- data.frame(x=0:10) |>
  mutate(prob=dbinom(x, size=10, prob=0.3))
barplot(binom_prob_df$prob, names.arg=binom_prob_df$x)
```



With `ggplot()`, we can use bar heights as above, but need to specify `stat="identity"` so that `geom_bar()` does not calculate any summary statistics. By default, `geom_bar()` automatically counts the number of observations.

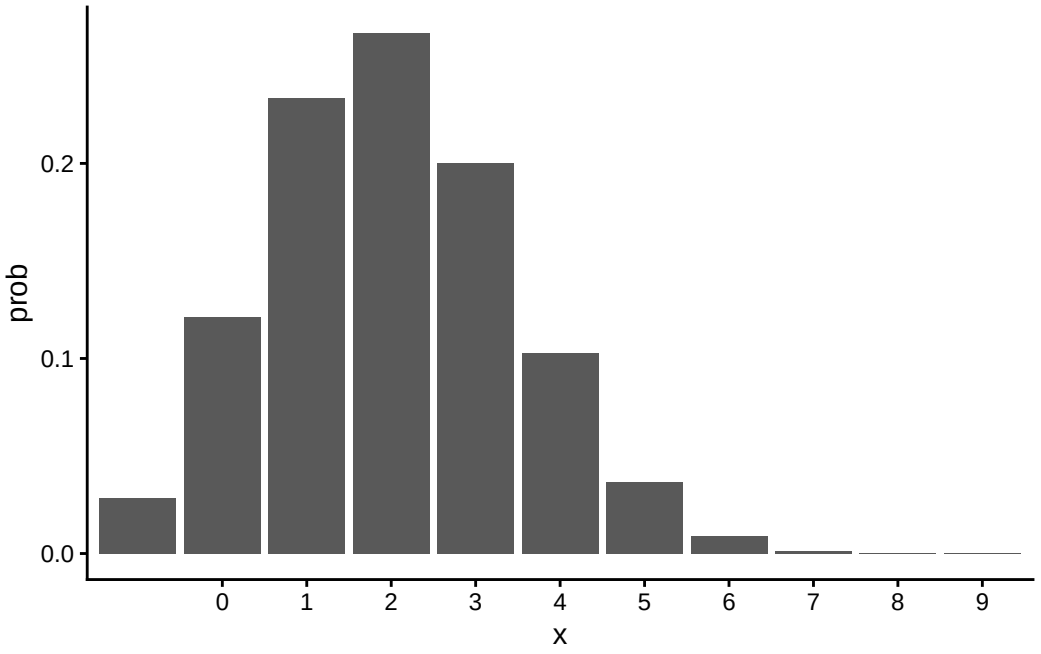
```
ggplot(binom_prob_df, aes(x=x, y=prob)) +
  geom_bar(stat="identity")
```



To make the x-axis a little prettier, we can add on an additional function.

```
ggplot(binom_prob_df, aes(x=x, y=prob)) +
  geom_bar(stat="identity") +
  scale_x_discrete(limits=0:10)
```

```
Warning in scale_x_discrete(limits = 0:10): Continuous limits supplied to discrete scale.
i Did you mean `limits = factor(...)` or `scale_*_continuous()`?
```

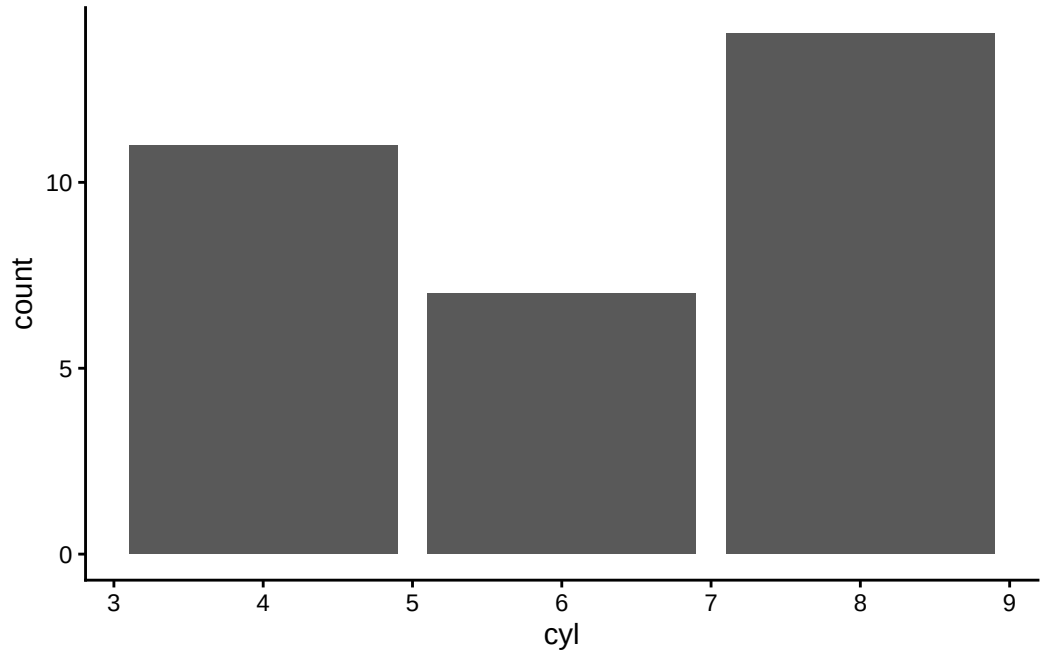


If we have recorded a suitable variable, `geom_bar()` can produce a plot of the frequencies:

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
ggplot(mtcars, aes(cyl)) +
  geom_bar()
```



There are many, many, *many* variations possible. See the tidyverse tutorial for a start, or explore the online documentation for ggplot2.

Part II

Practicals

Chapter 1

Visualization

View original

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 descriptive statistics: how to appropriately display and summarise. You already know 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 critical purposes of data visualization:

1. Getting a 'gut' feel for your data. 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 later).
2. Communicating your results. Focus on clarity, 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.

1.1 R Review

We *very briefly* review key basics of R here. See **R Basics** for a more comprehensive review.

It's best practice to load necessary packages at the top of your document. Remember, packages need to be *installed once per machine*, but *loaded in each R session*.

To install a package, either click the link in the yellow banner at the top of your script (RStudio usually does this when needed), or run:

```
install.packages("tidyverse")
install.packages("cowplot")
```

You will likely not need to do that again this semester! However, you will need to use `library(packageName)` each time you open R.

```
library(tidyverse)
library(readxl) # installed with tidyverse, but not loaded in library(tidyverse)
library(cowplot)
set.seed(2026) # for fully reproducible code
theme_set(theme_classic()) # plotting theme
```

1.2 Basic data exploration

1.2.1 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, bundled into the `datasets` package which is automatically loaded.

```
# functions for basic details of objects
str(penguins) # structure overview
glimpse(penguins) # slightly different overview
names(penguins) # column names
head(penguins) # first few rows
```

Functions have *arguments*, which let you control their settings. They typically have defaults, which you can see in the help page. For example, `head()` shows the first 6 rows by default (`?head`: note `n = 6L`). Specify the argument to adjust the behaviour:

```
head(penguins, n=2)
```

```
  species    island bill_len bill_dep flipper_len body_mass    sex year
1  Adelie Torgersen     39.1     18.7         181      3750   male 2007
2  Adelie Torgersen     39.5     17.4         186      3800 female 2007
```

```
tail(penguins, n=2)
```

```
      species island bill_len bill_dep flipper_len body_mass    sex year
343 Chinstrap  Dream     50.8     19.0         210      4100   male 2009
344 Chinstrap  Dream     50.2     18.7         198      3775 female 2009
```

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

> Function arguments do not need to be named if they are listed in the order given in the help documentation. For example, `head(penguins, 2)` will do exactly the same thing as `head(x=penguins, n=2)`.

1.2.2 Subsetting, renaming, and rearranging

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

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

```
penguins$bill_len # whole column
```

```
penguins[1, 3] # row 1, column 3
```

```
[1] 39.1
```

```
penguins[1:5, 3] # rows 1-5, column 3
```

```
[1] 39.1 39.5 40.3   NA 36.7
```

```
penguins[2:4, ] # leaving the 'columns' blank will return all columns
```

```
  species    island bill_len bill_dep flipper_len body_mass    sex year
2  Adelie Torgersen     39.5     17.4         186      3800 female 2007
3  Adelie Torgersen     40.3     18.0         195      3250 female 2007
4  Adelie Torgersen       NA       NA          NA        NA   <NA> 2007
```

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

```
penguins_alt <- penguins
names(penguins_alt)
```

```
[1] "species"     "island"      "bill_len"    "bill_dep"    "flipper_len"
[6] "body_mass"   "sex"         "year"
```

```
names(penguins_alt)[1] <- "species_name" # change first column name
names(penguins_alt)
```

```
[1] "species_name" "island"       "bill_len"     "bill_dep"     "flipper_len"
[6] "body_mass"    "sex"          "year"
```

Create a new column by specifying the name with `df$colName` and *assigning* a vector.

```
penguins_alt$bill_area <- penguins_alt$bill_dep * penguins_alt$bill_len
head(penguins_alt, 2)
```

```
  species_name    island bill_len bill_dep flipper_len body_mass    sex year
1       Adelie Torgersen     39.1     18.7         181      3750   male 2007
2       Adelie Torgersen     39.5     17.4         186      3800 female 2007
  bill_area
1    731.17
2    687.30
```

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

You can also subset based on criteria. Say we only want rows with observations from 2009:

```
penguins_2009 <- penguins_alt[penguins_alt$year == 2009, ]
```

It is always good to check that you've done what you think you've done:

```
nrow(penguins_2009)
```

```
[1] 120
```

```
head(penguins_2009, 2)
```

```
    species_name island bill_len bill_dep flipper_len body_mass    sex year
101       Adelie Biscoe       35     17.9         192      3725 female 2009
102       Adelie Biscoe       41     20.0         203      4725   male 2009
    bill_area
101     626.5
102     820.0
```

```
count(penguins_alt, year)
```

```
  year   n
1 2007 110
2 2008 114
3 2009 120
```

```
count(penguins_2009, year)
```

```
  year   n
1 2009 120
```

1.2.3 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(penguins_alt$sex)
```

R converts logical values (i.e., `TRUE`/`FALSE`) to numeric (i.e., `1`/`0`) automatically, allowing convenient summaries:

```
sum(is.na(penguins_alt$sex)) # how many are NA?
```

```
[1] 11
```

```
mean(is.na(penguins_alt$sex)) # what proportion are NA?
```

```
[1] 0.03197674
```

Another very useful check is `summary()`:

```
summary(penguins_alt)
```

```
   species_name       island       bill_len        bill_dep
 Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
 Chinstrap: 68   Dream    :124   1st Qu.:39.23   1st Qu.:15.60
 Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
                                 Mean   :43.92   Mean   :17.15
                                 3rd Qu.:48.50   3rd Qu.:18.70
                                 Max.   :59.60   Max.   :21.50
                                 NA's   :2       NA's   :2
  flipper_len      body_mass       sex           year        bill_area
 Min.   :172.0   Min.   :2700   female:165   Min.   :2007   Min.   : 497.6
 1st Qu.:190.0   1st Qu.:3550   male  :168   1st Qu.:2007   1st Qu.: 667.7
 Median :197.0   Median :4050   NA's  : 11   Median :2008   Median : 730.8
 Mean   :200.9   Mean   :4202                Mean   :2008   Mean   : 750.8
 3rd Qu.:213.0   3rd Qu.:4750                3rd Qu.:2009   3rd Qu.: 807.2
 Max.   :231.0   Max.   :6300                Max.   :2009   Max.   :1127.4
 NA's   :2       NA's   :2                                  NA's   :2
```

```
summary(penguins_alt)
```

Once you are confident that your `data.frame` looks sensible, that it contains the data you expect, and that the data types are correct, 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.

1.3 Graphical methods for displaying data

Always keep in mind that **the primary reason for data visualization is to communicate concisely and accurately to your reader**. Graphics should be clear, concise and easy to understand.

Real data is usually complex, and creating effective figures takes careful thought and often many revisions.

> We will primarily use the `ggplot2` package because it is more straightforward in many ways. It loads when you run `library(tidyverse)`. However, we will also note equivalent base R functions from time to time.

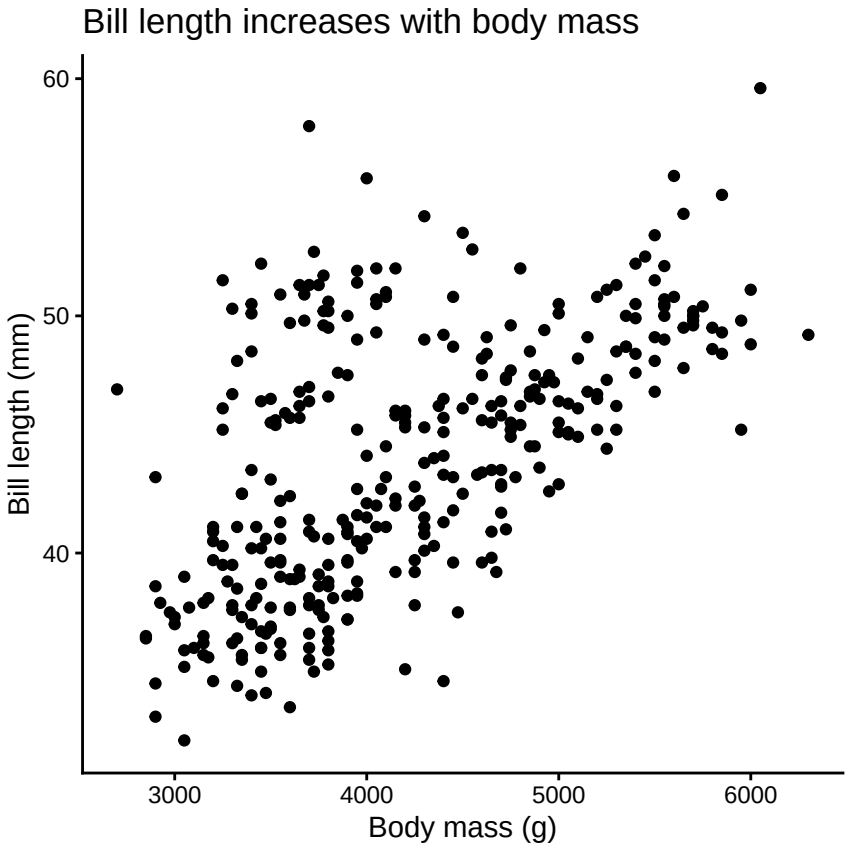
1.3.1 Scatter plots

Scatter plots display two continuous variables against each other. They are commonly used for analyses like correlation or linear regression.

With `ggplot2`:

```
ggplot(penguins_alt, aes(x=body_mass, y=bill_len)) +
  geom_point() +
  labs(x="Body mass (g)",
       y="Bill length (mm)",
       title="Bill length increases with body mass")
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```



The `ggplot()` function takes the dataframe as the first argument. The plot *aesthetics* are specified with `aes()`, such that variables (=columns) are *mapped* to plot aesthetics. Then, layers are added using `geom_*()`. Additional functions can be 'added' to further modify the plot. See a comprehensive list in the ggplot2 manual.

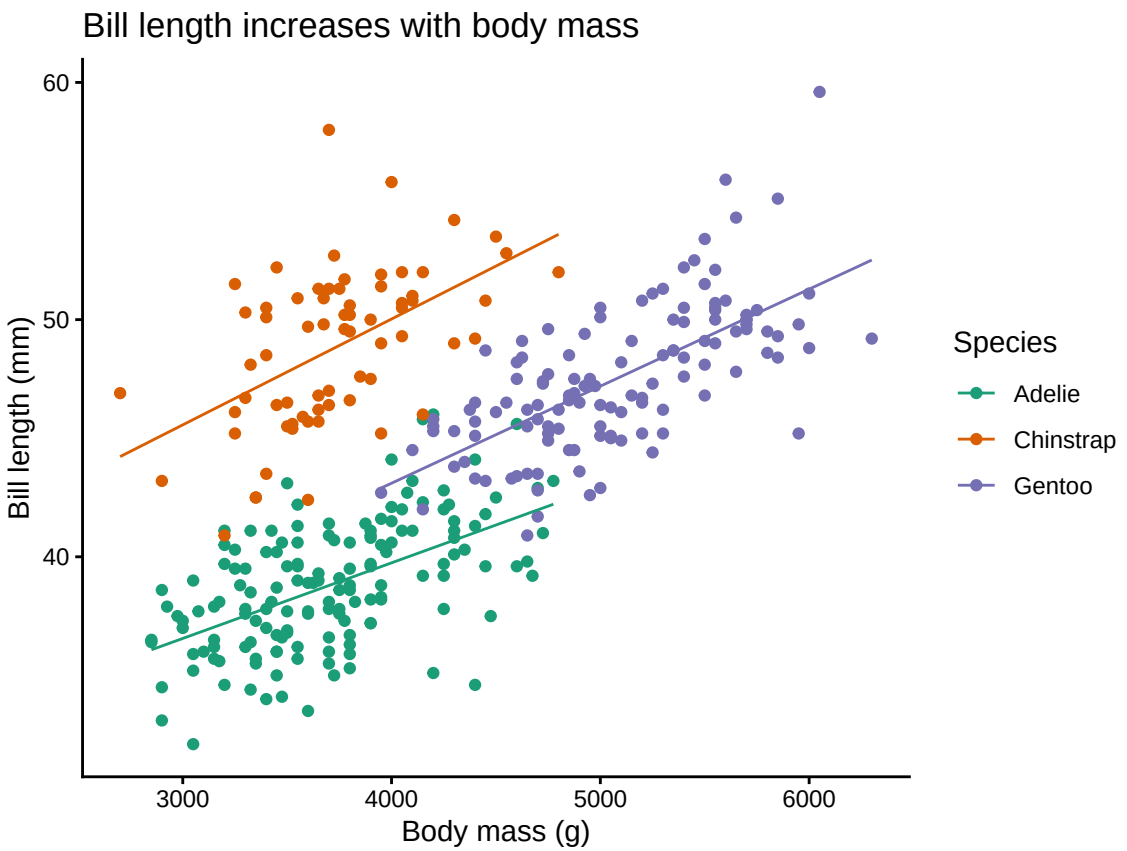
Once you understand the basic structure, it becomes easy to create more interesting figures:

```
ggplot(penguins_alt, aes(x=body_mass, y=bill_len, colour=species_name)) +
  geom_point() +
  geom_line(stat="smooth", method="lm") +
  scale_colour_brewer("Species", palette="Dark2") +
  labs(x="Body mass (g)",
       y="Bill length (mm)",
       title="Bill length increases with body mass")
```

```
`geom_smooth()` using formula = 'y ~ x'

Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_smooth()`).

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```



With base R, the `plot()` function will create a scatter plot if given two numeric variables. There are two options for specifying the variables.

Using `data_df` with columns `x` and `y`, 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()`, but it is clunkier compared to `ggplot2`.

Q. 1.1

With the `penguins` dataset, plot bill length by body mass for only Chinstrap penguins. Make sure the plot is fully labelled.

View solution

```
ggplot(penguins[penguins$species=="Chinstrap",], aes(body_mass, bill_len)) +
  geom_point() +
  labs(x="Body mass (g)",
       y="Bill length (mm)",
       title="Chinstrap penguins")
```

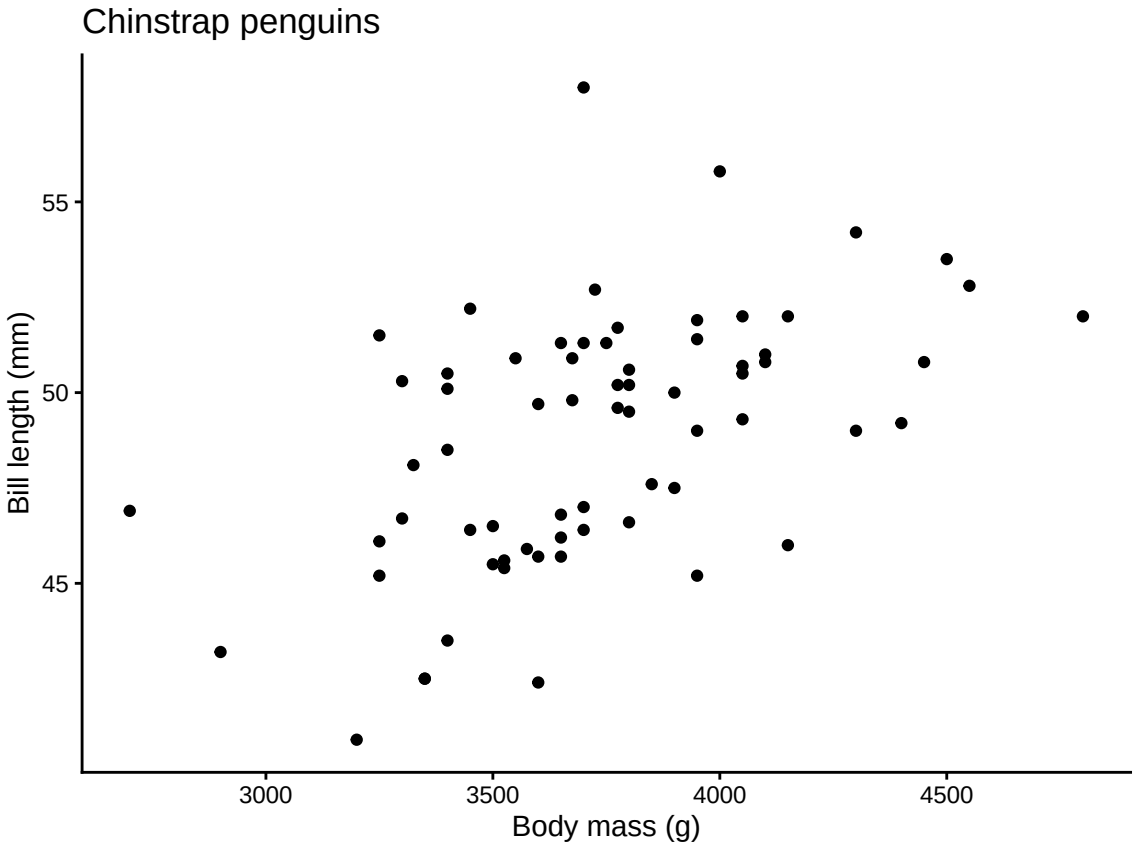


Figure 1.1: Chinstrap penguin allometrics.

Q. 1.2

Using `ggplot()`, create a figure where each species is a different shape. See the **Aesthetics** section in `?geom_point` for help.

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```

1.3.2 Boxplots

Box plots are used to summarise a continuous variable by levels of a factor.

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

```
'data.frame':   344 obs. of  8 variables:
 $ species    : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ island     : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
```

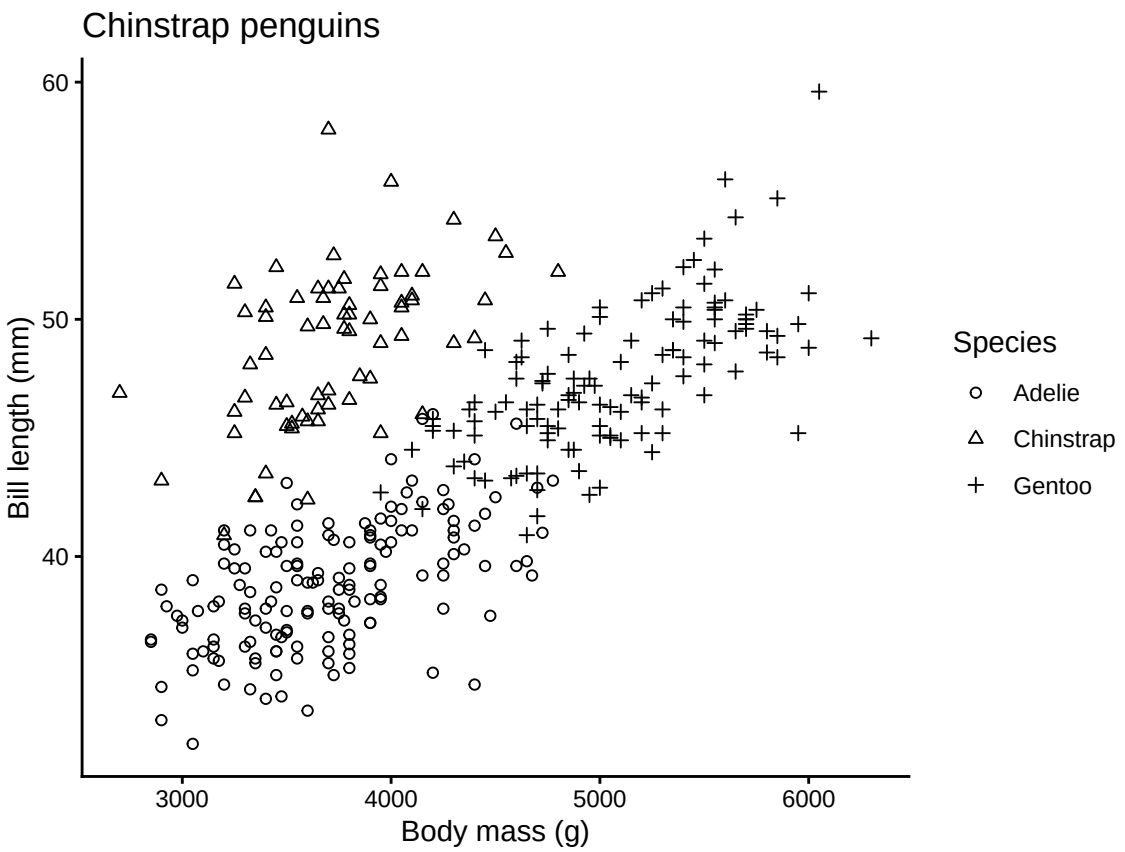


Figure 1.2: Penguin allometrics.

```
 $ bill_len   : num  39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
 $ bill_dep   : num  18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
 $ flipper_len: int  181 186 195 NA 193 190 181 195 193 190 ...
 $ body_mass  : int  3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
 $ sex        : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ...
 $ year       : int  2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

```
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_boxplot()`).
```

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

Q. 1.3

Reproduce the plot shown in Figure 1.3.

View solution

```
ggplot(penguins_alt, aes(species_name, bill_area/body_mass, fill=species_name)) +
  geom_boxplot() +
  labs(x="Species",
       y="Bill area (mm^2) scaled by body mass (g)")
```

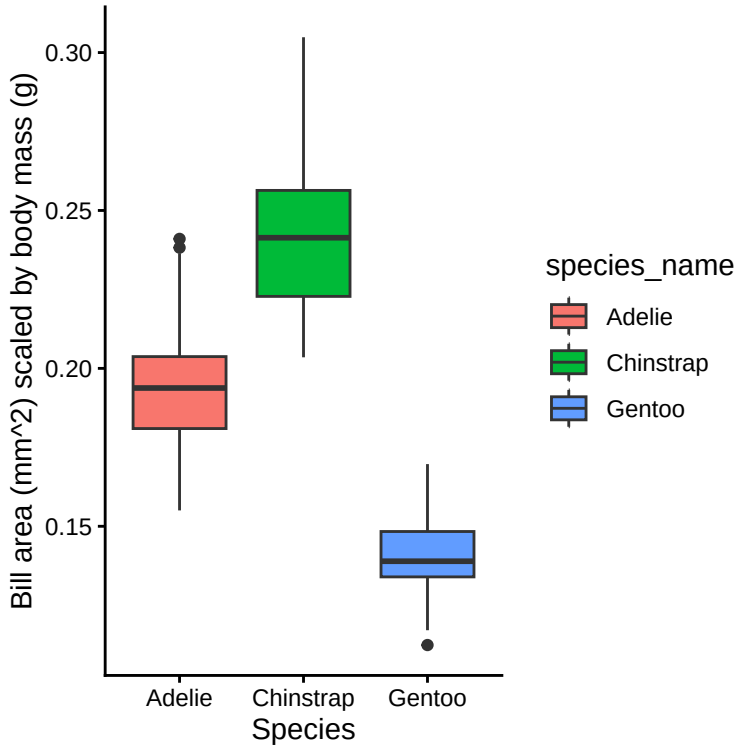


Figure 1.3: Boxplot showing bill area by species

Q. 1.4

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

In base R, use `boxplot(y ~ x, data=dataframe)` to create a boxplot.

Note that box plots are not the most visually intuitive, since *smaller* box areas correspond with *more* observations. Packages like `ggplot2` (and extensions) make alternatives like those in Figure 1.4 simple to produce. We will cover some of these later.

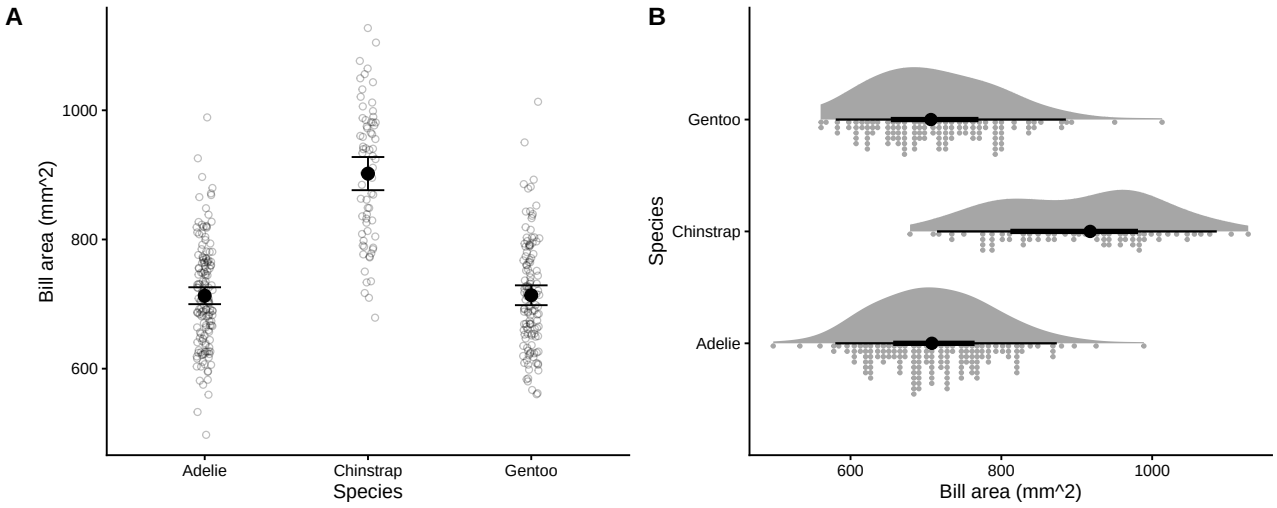


Figure 1.4: Alternatives to boxplots. (A) Strip plot which shows all data along with means and 2 standard errors. (B) Rain cloud plot showing data distributions along with medians with middle 50% and 95%.

1.3.3 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)
lynx_df <- as.data.frame(lynx)
str(lynx_df)
lynx_df$Year <- seq(from=1821, to=1934, by=1)
lynx_df$Trappings <- as.numeric(lynx_df$x) # Time-Series is complicated.
str(lynx_df)
head(lynx_df, 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. 1.5

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)
```

Q. 1.6

Using `ggplot()`, produce a line plot similar to Figure 1.5.

View solution

```
ggplot(lynx_df, aes(Year, Trappings)) +
  geom_point() +
  geom_line() +
  labs(title="Lynx trapping in Canada (1820-1934)")
```

In base R, use `plot(y ~ x, data=dataframe, type='b')` to make plots with **b**oth points and lines, or `type='l'` for just **l**ines.

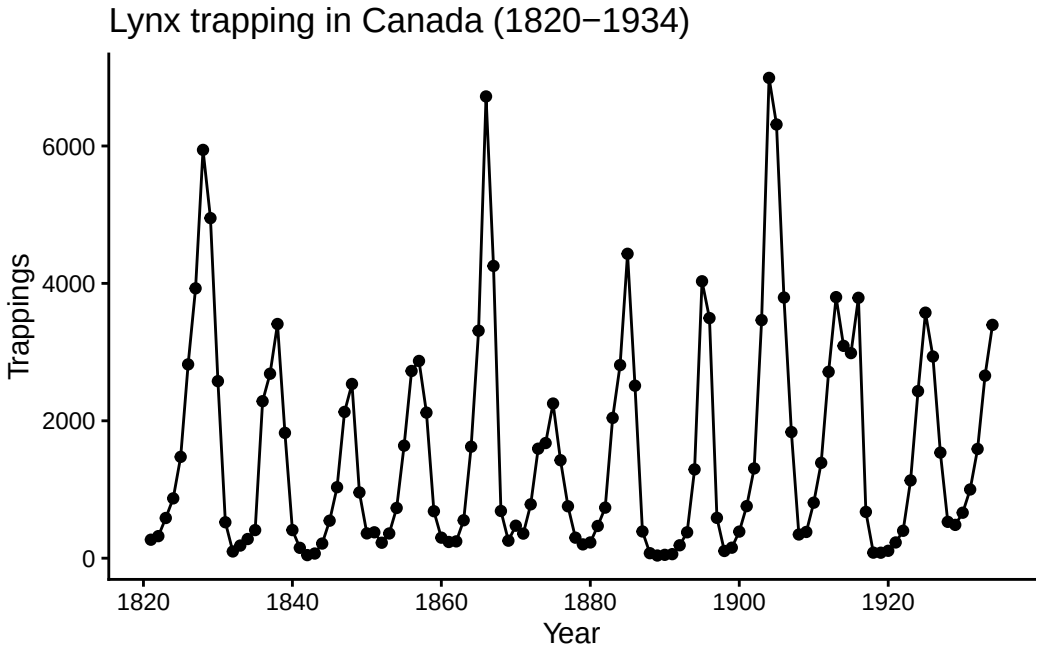


Figure 1.5: The number of lynx trapped in Canada (1820-1934)

Q. 1.7

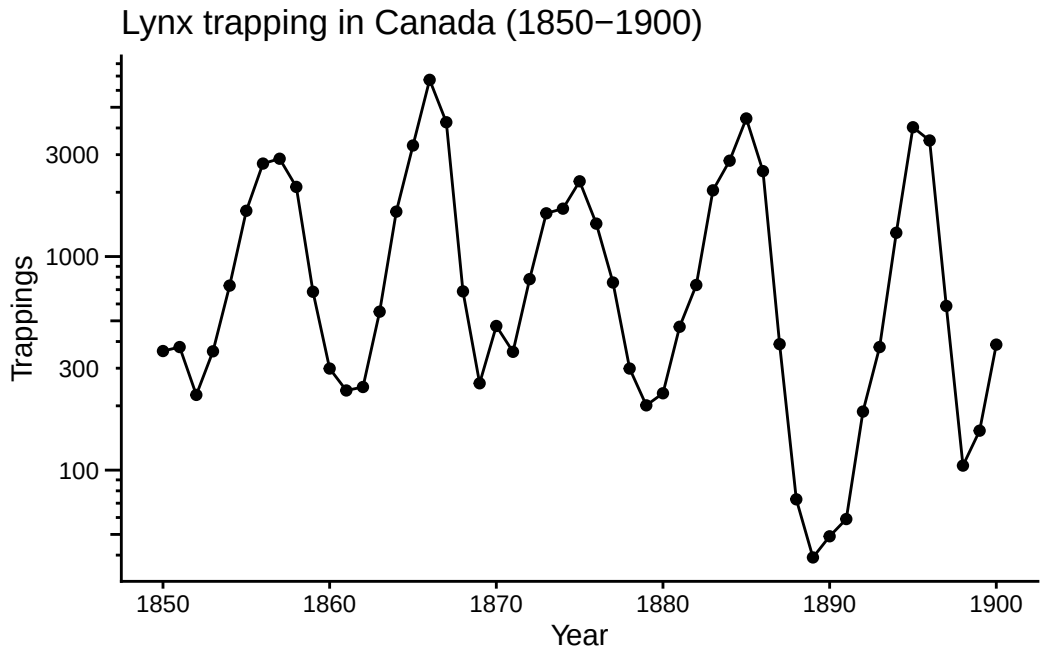
Create a plot that shows the **log-transformed** number of trappings from **only** 1850 to 1900.

View solution

```
ggplot(lynx_df[lynx_df$Year >= 1850 & lynx_df$Year <= 1900, ],
       aes(Year, log(Trappings))) +
  geom_point() +
  geom_line() +
  labs(title="Lynx trapping in Canada (1850-1900)")
```

Alternatively (and better, in this case) is to modify the *axis* rather than the variable.

```
ggplot(lynx_df[lynx_df$Year >= 1850 & lynx_df$Year <= 1900, ],
       aes(Year, Trappings)) +
  geom_point() +
  geom_line() +
  scale_y_log10() +
  guides(y="axis_logticks") +
  labs(title="Lynx trapping in Canada (1850-1900)")
```



1.3.4 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.

```
ggplot(lynx_df, aes(Trappings)) + # specify x-axis, bun not y-axis
  geom_histogram(colour="grey30", fill="grey90") +
  labs(x="Trapped lynx per year",
       title="Lynx trapping")
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```

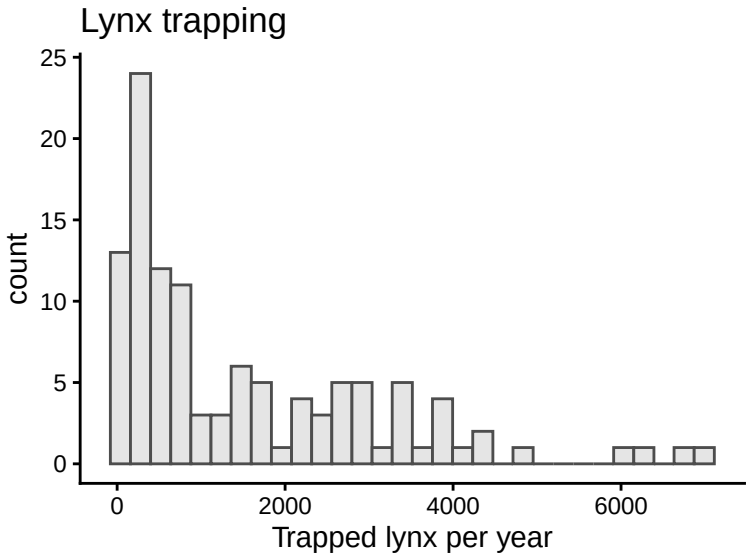


Figure 1.6: Lynx pelts per year with default settings.

Q. 1.8

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

! Low values (< 1000) are most common, with a second mode between 2000-3000.

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

i To make a multi-panel figure with ggplot, use the `plot_grid()` function in the `cowplot` package.

```
library(cowplot)
pA <- ggplot(lynx_df, aes(Trappings)) +
  geom_histogram(colour="grey30", fill="grey90",
                 bins=5) +
  labs(x="Trapped lynx per year",
       title="Lynx trapping")
pB <- ggplot(lynx_df, aes(Trappings)) +
  geom_histogram(colour="grey30", fill="grey90",
                 breaks=c(0, 500, 1000, 2000, 5000, 10000)) +
  labs(x="Trapped lynx per year",
       title="Lynx trapping")
plot_grid(pA, pB, nrow=1, labels="AUTO")
```

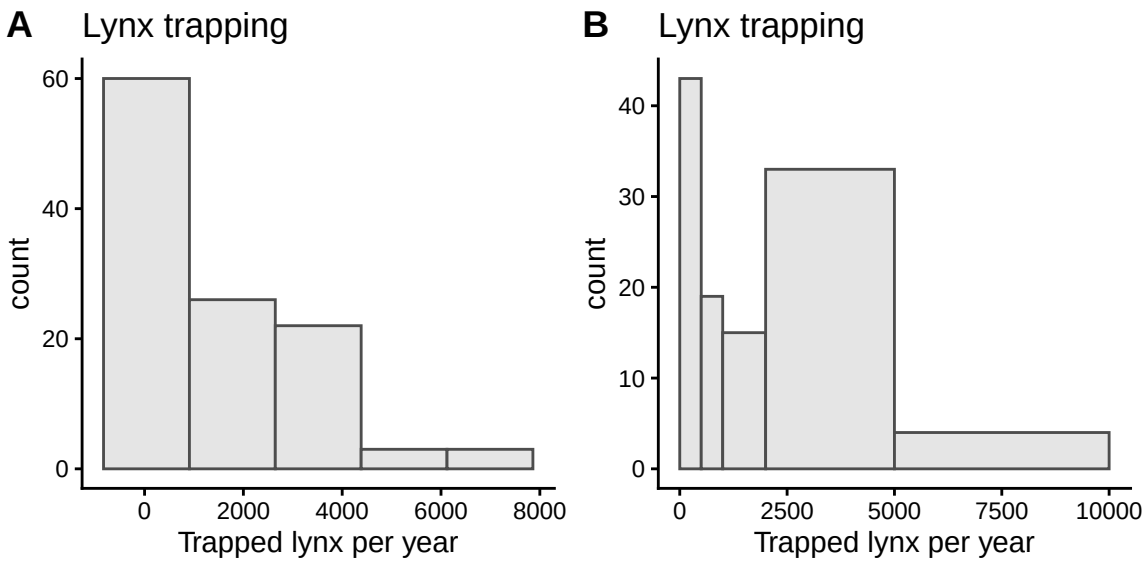


Figure 1.7: Lynx pelts per year with breaks=5 (left) and a vector of breaks (right).

```
pA <- ggplot(lynx_df, aes(Trappings)) +
  geom_histogram(colour="grey30", fill="grey90",
                 breaks=seq(0, 10000, by=100)) +
  labs(x="Trapped lynx per year")
pB <- ggplot(lynx_df, aes(Trappings)) +
  geom_histogram(colour="grey30", fill="grey90",
                 breaks=seq(0, 10000, by=500)) +
  labs(x="Trapped lynx per year")
pC <- ggplot(lynx_df, aes(Trappings)) +
  geom_histogram(colour="grey30", fill="grey90",
                 breaks=seq(0, 10000, by=1000)) +
  labs(x="Trapped lynx per year")
pD <- ggplot(lynx_df, aes(Trappings)) +
  geom_histogram(colour="grey30", fill="grey90",
                 breaks=seq(0, 10000, by=2000)) +
  labs(x="Trapped lynx per year")
plot_grid(pA, pB, pC, pD, nrow=2, ncol=2,
          align="hv", axis="tblr", labels="AUTO")
```

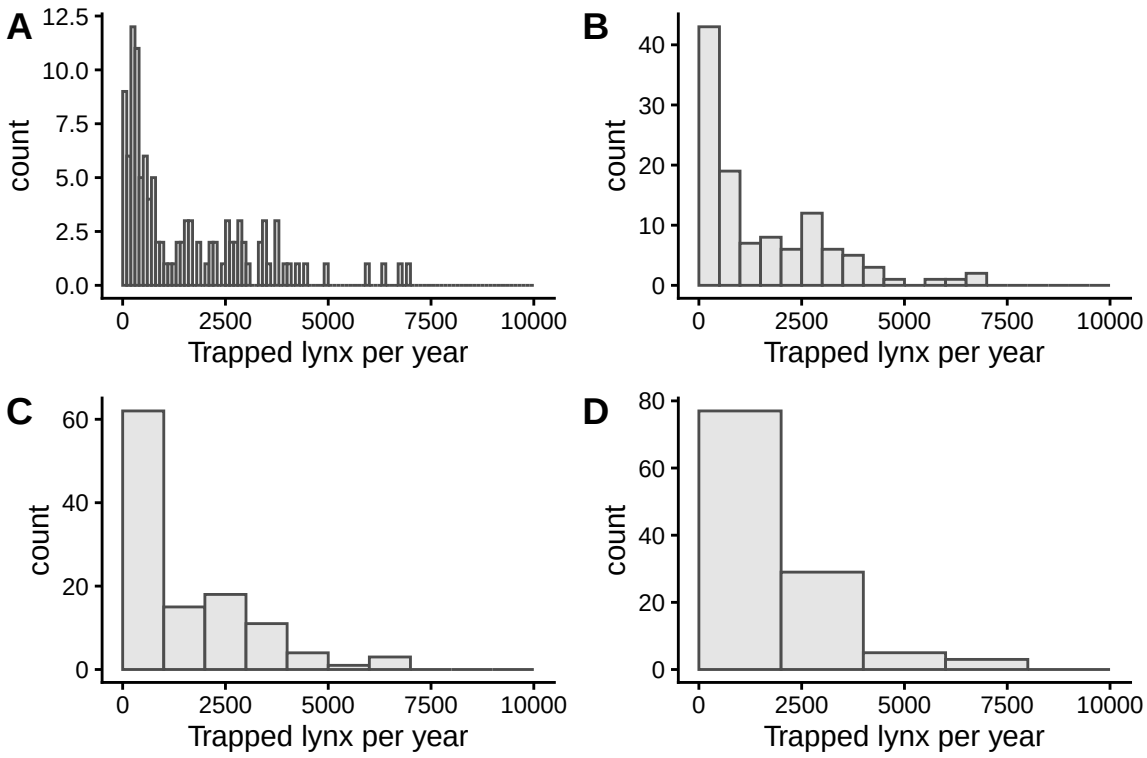


Figure 1.8: Histograms of lynx pelts per year with breaks every (A) 100, (B) 500, (C) 1,000, or (D) 2,000.

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.

> ℹ In base R, use `hist(dataframe$column)`.

1.3.5 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) # 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
```

An initial attempt at a barplot makes a terrible figure.

```
ggplot(islands_df, aes(LandMass, Area)) +
  geom_bar(stat="identity") # 'identity' means it won't calculate any summaries
```

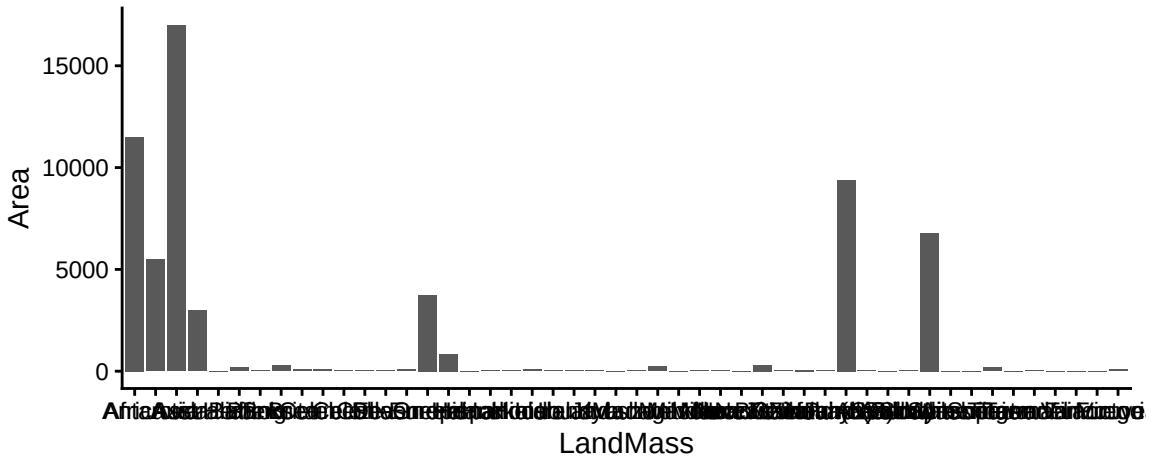


Figure 1.9: Island areas with barplot defaults

The whole dataset includes a lot of very small areas, so let's cut it down to just the 10 largest. We also want to order the land masses by area in our plot.

We will use the `arrange()` function to order our dataframe, and then use `scale_x_discrete()` to specify the breaks in the desired order.

```
islands_df <- arrange(islands_df, desc(Area)) # desc() makes it largest to smallest
```

```
ggplot(islands_df[1:10,], aes(LandMass, Area)) +
  geom_bar(stat="identity") +
  scale_x_discrete(limits=islands_df$LandMass[1:10])
```

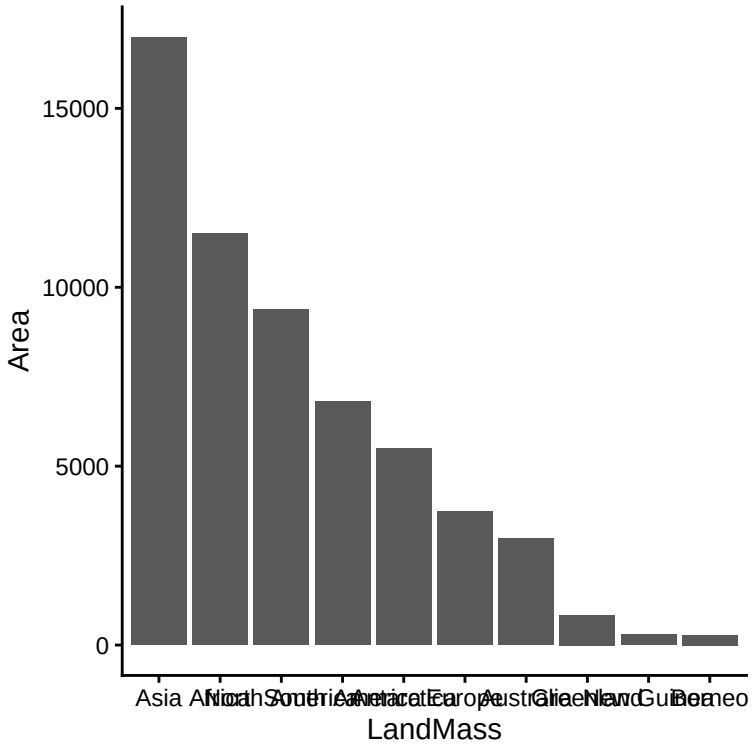


Figure 1.10: Top 10 island areas

It would be more effective sideways:

```
ggplot(islands_df[1:10,], aes(Area, LandMass)) +
  geom_bar(stat="identity") +
  scale_y_discrete(limits=islands_df$LandMass[10:1]) +
  labs(x="Land mass",
       y="Area (km^2)")
```

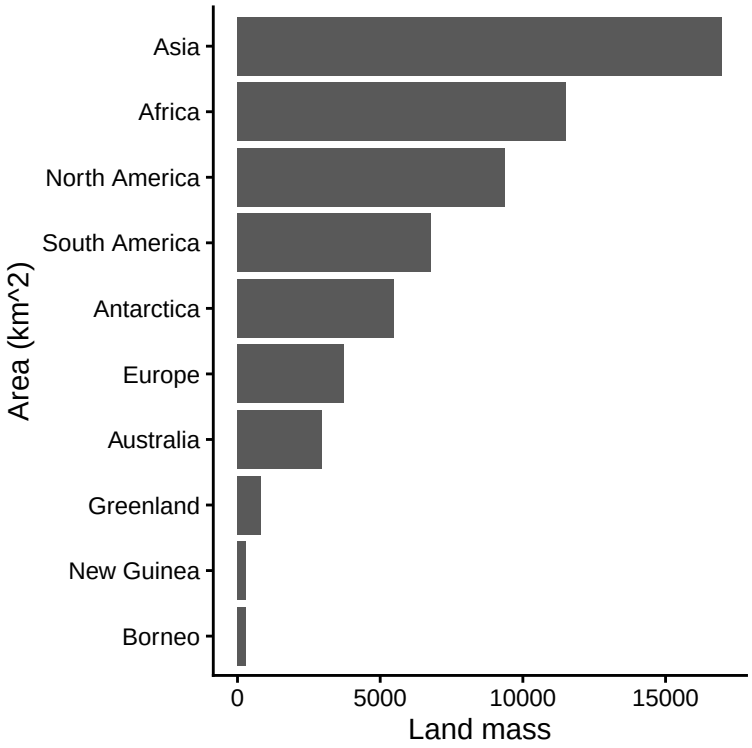


Figure 1.11: Finally! Did you know Antarctica is bigger than Europe?

Data visualization is an iterative process with lots of trial and error to find a plot that communicates the message within the data well.

1.4 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. 1.9

What are the three main measures of central tendency?

! Mean, median, and mode

Q. 1.10

What are three measures of variability?

! Range, interquartile range (IQR), and variance / standard deviation

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.

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)` or by clicking on the icon in the *Environment* pane.

```
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 ...
```

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
```

1.4.1 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/H2SMS_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 ...
```

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

Q. 1.11

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

! There is an outlier with value 99.00

Q. 1.12

Are these data likely to be continuous or discontinuous?

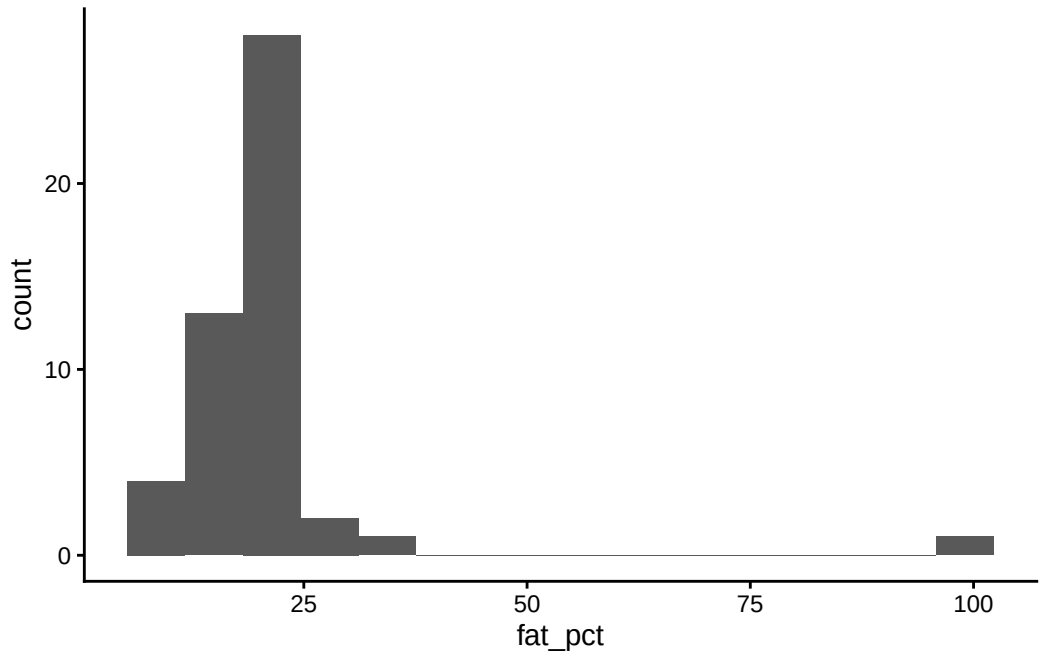
! Continuous

Q. 1.13

Create a plot to visualize the distribution of these data.

!

```r
ggplot(scallop_df, aes(fat_pct)) +
  geom_histogram(bins=15)
```



Q. 1.14

Do you spot any issues?

! As noted above, there is a clear outlier.

View solution

```
ggplot(scallop_df, aes(fat_pct)) +
  geom_histogram(bins=15)
```

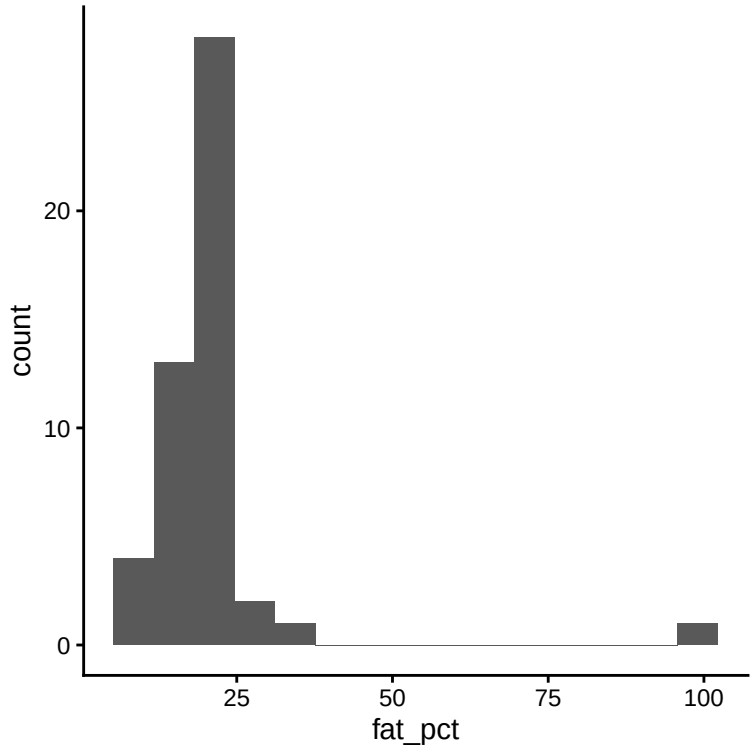


Figure 1.12: Histogram of fat percentage.

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
```

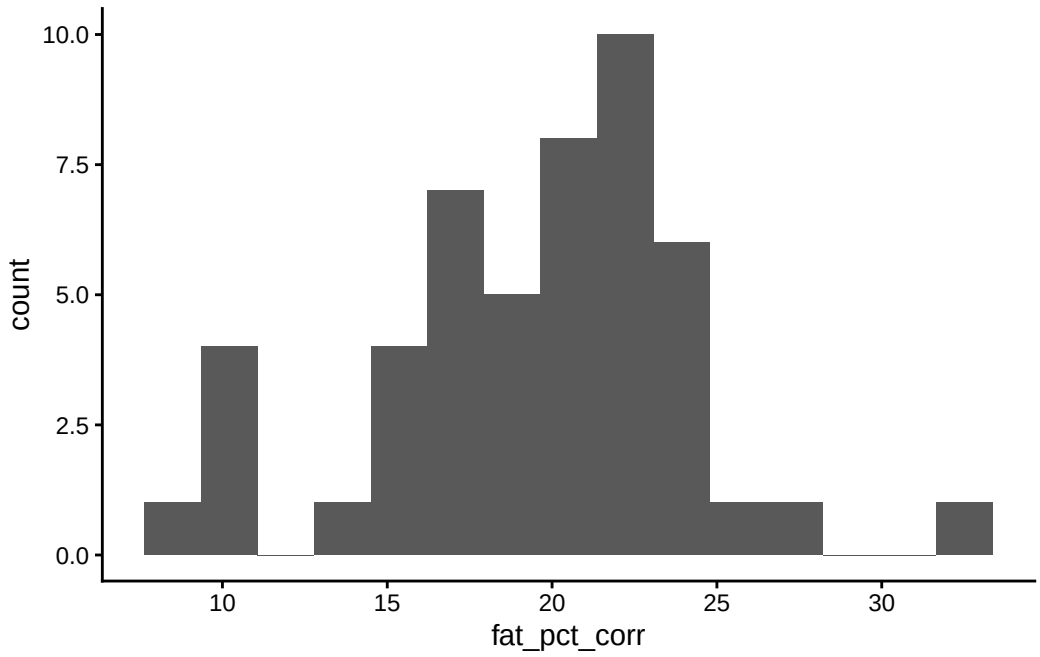
Now summarise `scallop_df` using some of the methods above.

Q. 1.15

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

! The outlier is gone.

```
ggplot(scallop_df, aes(fat_pct_corr)) +
  geom_histogram(bins=15)
```



Q. 1.16

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

! Could do this line by line, or modify the code above:

```
scallop_df |>
  summarise(mean=mean(fat_pct_orig),
            var=var(fat_pct_orig),
            med=median(fat_pct_orig),
            iqr=IQR(fat_pct_orig),
            min=min(fat_pct_orig),
            max=max(fat_pct_orig),
            rng=diff(range(fat_pct_orig)))
```

```
# A tibble: 1 x 7
   mean   var   med   iqr   min   max   rng
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21.4  149.  20.6   5.3   8.5    99  90.5
```

```
scallop_df |>
  summarise(mean=mean(fat_pct_corr),
            var=var(fat_pct_corr),
            med=median(fat_pct_corr),
            iqr=IQR(fat_pct_corr),
            min=min(fat_pct_corr),
            max=max(fat_pct_corr),
            rng=diff(range(fat_pct_corr)))
```

```
# A tibble: 1 x 7
   mean   var   med   iqr   min   max   rng
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  19.6  22.8  20.6   5.6   8.5  32.5    24
```

Q. 1.17

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.

!

```
scallop_df$fat_pct_badTypo <- scallop_df$fat_pct_orig
scallop_df$fat_pct_badTypo[36] <- 999
```

Q. 1.18

Calculate the same summary statistics.

```
scallop_df |>
  summarise(mean=mean(fat_pct_badTypo),
            var=var(fat_pct_badTypo),
            med=median(fat_pct_badTypo),
            iqr=IQR(fat_pct_badTypo),
            min=min(fat_pct_badTypo),
            max=max(fat_pct_badTypo),
            rng=diff(range(fat_pct_badTypo)))
```

```
# A tibble: 1 x 7
   mean    var   med   iqr   min   max   rng
  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  39.7 19591.  20.6   5.3   8.5   999  990.
```

Q. 1.19

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

Median and IQR, since they are based on ranks.

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. 1.20

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.

! Median and IQR are most robust to entry-related errors.

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

```
hake_df <- read_excel("data/H2SMS_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. 1.21

What type of variable is `length`?

! Numeric, continuous

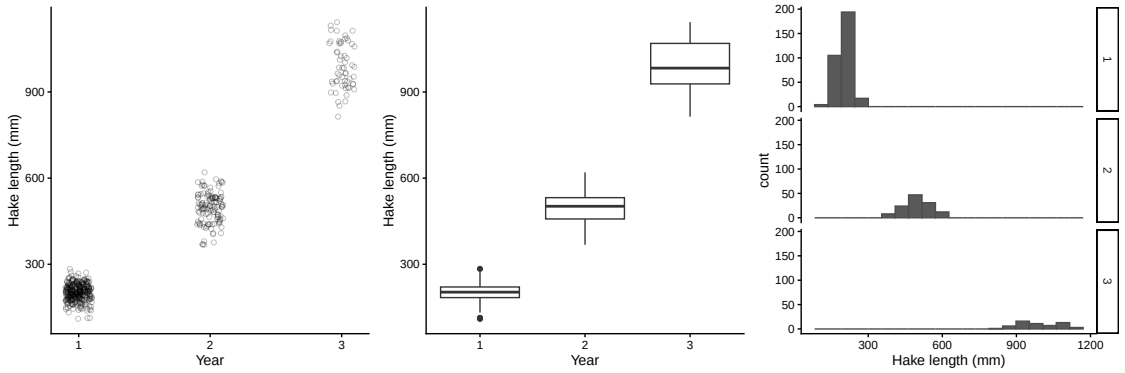
Q. 1.22

Select an appropriate graphical method and display these data.

Hint: `ggplot(..., aes(..., group=Year)) + geom_boxplot()`

!

```
pA <- ggplot(hake_df, aes(Year, `Hake length (mm)`)) +
  geom_jitter(alpha=0.25, shape=1, width=0.1, height=0) + # alpha sets the opacity
  scale_x_continuous(breaks=1:3)
pB <- ggplot(hake_df, aes(Year, `Hake length (mm)`, group=Year)) +
  geom_boxplot()
pC <- ggplot(hake_df, aes(`Hake length (mm)`)) +
  geom_histogram(bins=20, colour="grey30") +
  facet_grid(Year~.)
plot_grid(pA, pB, pC, nrow=1)
```



Q. 1.23

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

!

```
hake_df |>
  group_by(Year) |>
  summarise(mean=mean(`Hake length (mm)`),
            var=var(`Hake length (mm)`),
            med=median(`Hake length (mm)`),
            iqr=IQR(`Hake length (mm)`),
            min=min(`Hake length (mm)`),
            max=max(`Hake length (mm)`),
            rng=diff(range(`Hake length (mm)`)))
```

```
# A tibble: 3 x 8
   Year  mean   var   med   iqr   min   max   rng
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     1  202.  837.   203  37     110   284   174
2     2  497. 2809.   502  74.2   368   620   252
3     3  989. 6450.   983 141     814  1143   329
```

```
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/H2SMS_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 ...
```

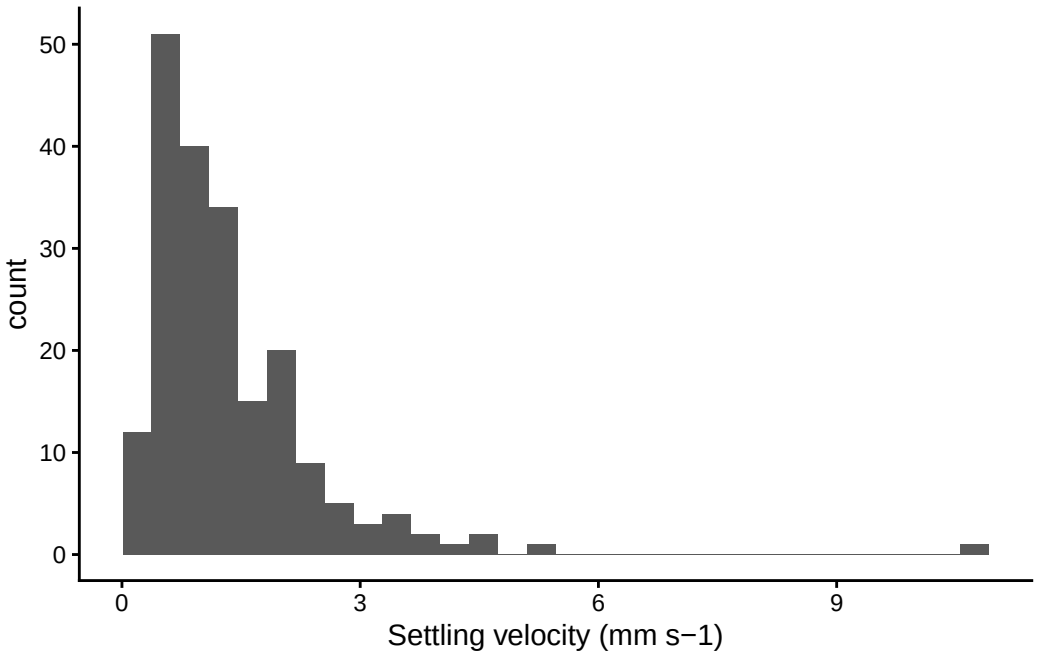
Q. 1.24

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

! It is right skewed.

```
ggplot(fishPoo_df, aes(`Settling velocity (mm s-1)`)) + geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



Q. 1.25

Which measures of central tendency and variability are most appropriate?

! Median and IQR, most likely.

Q. 1.26

Considering the distribution, is the mean or median larger?

! The mean is larger.

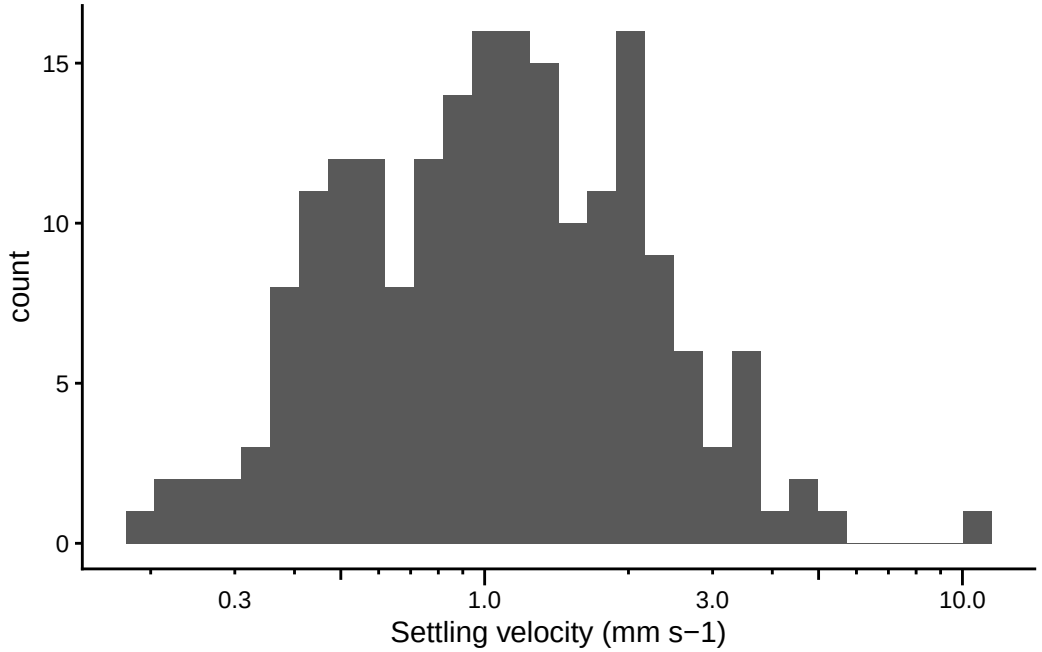
Q. 1.27

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

!

```
ggplot(fishPoo_df, aes(`Settling velocity (mm s-1)`)) +
  geom_histogram() +
  scale_x_log10() +
  guides(x="axis_logticks")
```

```
`stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



Q. 1.28

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.

! Any, but mean and variance / standard deviation would be best.

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?

1.5 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

View original

We often wish to determine the probability of events occurring given our current understanding of the processes involved. This requires a mathematical description of a theoretical relationship: 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); theme_set(theme_classic())
library(readxl)
library(cowplot)
set.seed(2026)
```

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?

! 0.5

This is different to asking whether (100 heads) or (99 heads + 1 tails) is more likely in a throw of 100 coins. Each single flip is an independent event.

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, sometimes called 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 calculations

2.1.3.1 Sand tiger sharks

Sand tiger shark (*Carcharias taurus*) females have two uterine horns, each with as many as 50 embryos. However, they display 'embryophagy', where the largest embryo in each uterine horn eats all of the smaller embryos, so that females typically give birth to two pups.

Assume that the average sex ratio of sand tiger shark pups is 50:50. We are interested in the proportion of pups we expect to be female from one clutch (two pups).

Q. 2.2

What is the event?

! Each pup – whether or not it is female.

Q. 2.3

What is p?

! 0.5

Q. 2.4

What is the number of trials?

! 2

Q. 2.5

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

! Three:

- two female
- one female, one male
- two male

Q. 2.6

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

! $y \sim Binomial(n = k = 2, p = 0.5)$

Q. 2.7

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

!
(a) 1/4
(b) 1/4
(c) 1/2

2.1.3.2 Great white sharks

Great white sharks (*Carcharodon carcharias*) typically give birth to 2-10 pups at a time. You've captured a female for tagging and found that she has 8 developing embryos.

We are interested in the proportion of pups we expect to be female from this shark's clutch (8 pups).

Q. 2.8

What is the Bernoulli event?

! Each offspring: Is it male or female?

Q. 2.9

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

! 0 to 8

Q. 2.10

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

! 0F 8M, 1F 7M, ..., 8F 0M = 9 outcomes

Assume that the average sex ratio is even, such 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. 2.11

Write down the model for the number of female pups for this shark's brood: $y \sim Binom(n, p)$.

! $y \sim Binom(8, 0.5)$

Q. 2.12

Calculate the probability of obtaining eight male offspring.

! Could use the binomial formula, but the coefficient reduces to 1 since there is only one pathway to 8 males:

```
0.5^8
```

```
[1] 0.00390625
```

Q. 2.13

What is the mean number of females you would expect in great white shark broods of 8 pups?

! With p=0.5, expectation is 4

It gets more complicated when you want to know the probability of getting, say, 1 male and 7 females from a brood of eight pups.

Q. 2.14

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

! Symmetric

R can calculate probabilities for specific outcomes from a massive array of theoretical probability distributions, including the binomial distribution.

For ease, we can save the binomial distribution parameters as objects:

```
num_pups <- 8
p_female <- 0.5
```

To calculate the probability of 4 females:

```
# for a single probability: y~Binom(n=8, p=0.5) determine P(y_i=4)
dbinom(4, num_pups, p_female) # dbinom(4, 8, 0.5)
```

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 pups?"* Here, <4 equates to the cumulative probability P(0) + P(1) + P(2) + P(3).

In this case, we can use either `pbinom(...)` or `sum(dbinom(...))` to give the cumulative probability.

```
dbinom(0:3, num_pups, p_female)
sum(dbinom(0:3, num_pups, p_female))
```

```
[1] 0.00390625 0.03125000 0.10937500 0.21875000
[1] 0.3632813
```

```
pbinom(3, num_pups, p_female)
```

```
[1] 0.3632813
```

Q. 2.15

Why do we parameterise `pbinom()` with `3` rather than `4`?

! `pbinom(x, n, p)` gives the cumulative probability up to and including `x`, and in this case we want to know *less than 4* instead of *less than or equal to 4*.

Q. 2.16

Update the code to calculate $P(\leq 4)$?

!
```
pbinom(4, num_pups, p_female)
```

```
[1] 0.6367187
```

Q. 2.17

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

!
```
sum(dbinom(0:3, num_pups, p_female))
```

```
[1] 0.3632813
```

Q. 2.18

What is the probability of getting 3 females?

!
```
dbinom(3, num_pups, p_female)
```

```
[1] 0.21875
```

Q. 2.19

What is the probability of getting 8 females?

```
dbinom(8, num_pups, p_female)
```

```
[1] 0.00390625
```

Q. 2.20

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

From 0 to 2, which could be calculated with `pbinom(2, num_pups, p_female)`

Q. 2.21

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

Q. 2.22

What is the probability of getting < 6 females?

```
pbinom(5, num_pups, p_female)
```

```
[1] 0.8554688
```

Q. 2.23

What is the probability of getting ≤ 6 females?

```
pbinom(6, num_pups, p_female)
```

```
[1] 0.9648438
```

Q. 2.24

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

0

Q. 2.25

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

```
1 - pbinom(1, num_pups, p_female)
```

```
[1] 0.9648438
```

```
sum(dbinom(2:8, num_pups, p_female))
```

```
[1] 0.9648438
```

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
```

```
ggplot(prFemale_df, aes(num_female, prob)) +
  geom_bar(stat="identity") +
  labs(x="Number of females",
       y="Probability")
```

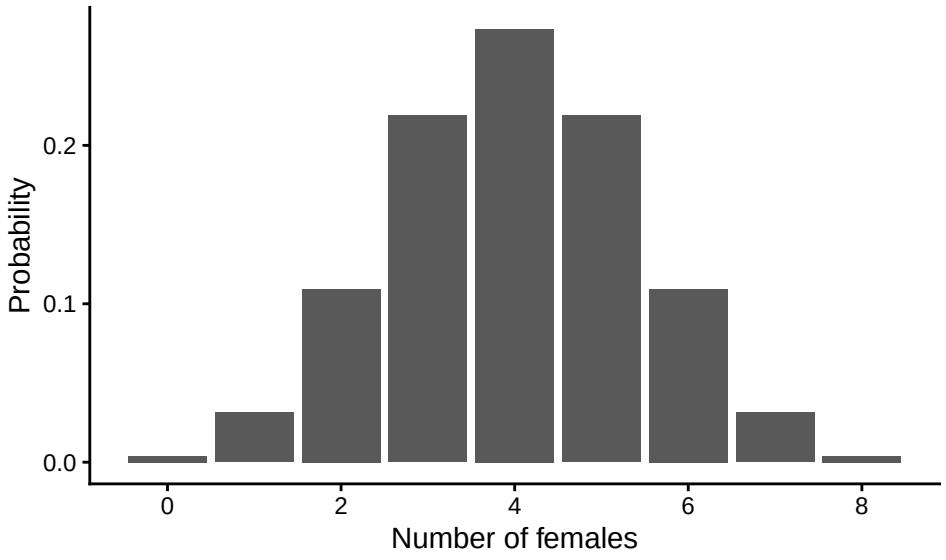


Figure 2.1: Binomial probability distribution

```
prFemale_df$cumul_prob <- cumsum(prFemale_df$prob)

ggplot(prFemale_df, aes(num_female, cumul_prob)) +
  geom_bar(stat="identity") +
  labs(x="Enter the correct label!",
       y="Enter the correct label!")
```

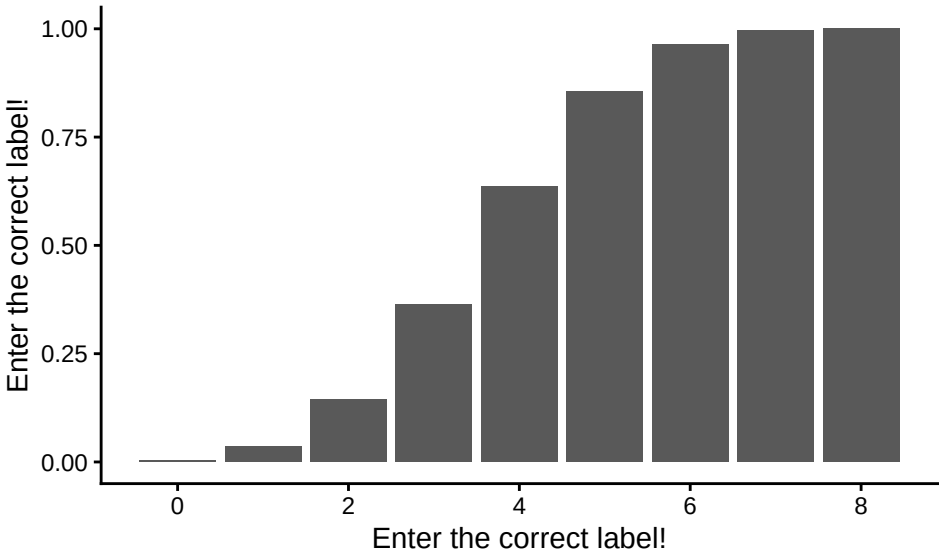


Figure 2.2: Cumulative binomial probability distribution

Try re-plotting so that the two panels appear side by side (hint: `library(cowplot)`, then `?plot_grid`).

Note $P(y_i = x)$ is read as 'the probability that 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, in a brood of 8 pups where each pup has 50% probability of being female, the probability that all are 8 female is 0.00391.

In other words, if we have 500 broods, each with 8 pups, we expect $500 * 0.00391 = 1.95 \approx 2$ broods to be all female. You wonderously manage to identify the sex of your shark's pups, and find that all 8 are, in fact, female. Is there something strange about your shark or is it just one of the 2/500 by chance?

Q. 2.26

Why is a bar graph the appropriate plot here?

! The outcome is a discrete variable: integers.

Q. 2.27

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

! It is symmetric.

Q. 2.28

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

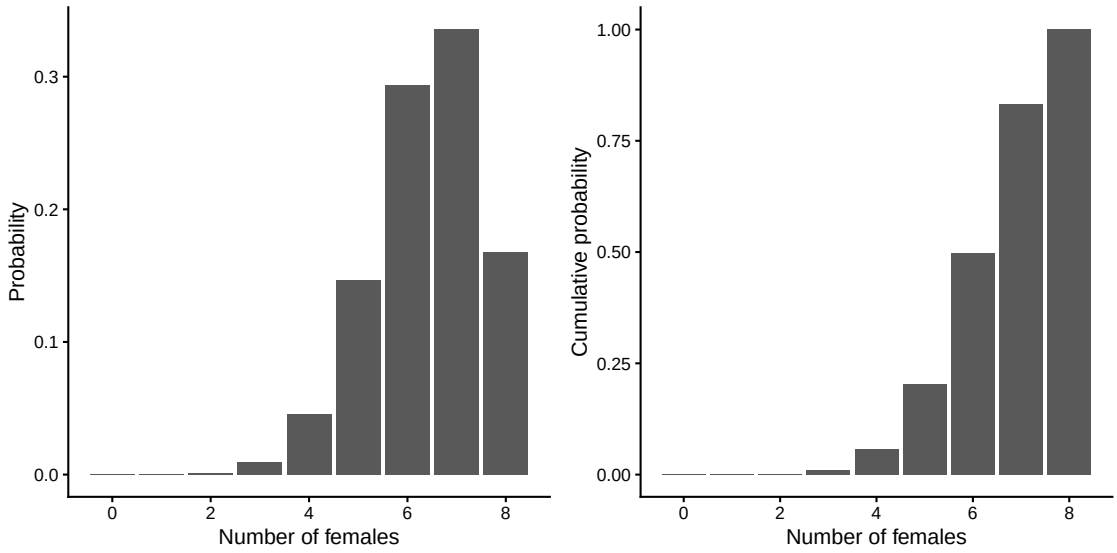
! The distribution is skewed left, with more probability mass toward larger numbers.

```
num_female <- 0:8
p_female <- 0.8
prFemale_df <- data.frame(num_female=num_female,
                          prob=dbinom(num_female, max(num_female), p_female))
prFemale_df$cumul_prob <- cumsum(prFemale_df$prob)
prFemale_df
```

```
  num_female       prob cumul_prob
1          0 0.00000256 0.00000256
2          1 0.00008192 0.00008448
3          2 0.00114688 0.00123136
4          3 0.00917504 0.01040640
5          4 0.04587520 0.05628160
6          5 0.14680064 0.20308224
7          6 0.29360128 0.49668352
8          7 0.33554432 0.83222784
9          8 0.16777216 1.00000000
```

```
pA <- ggplot(prFemale_df, aes(num_female, prob)) +
  geom_bar(stat="identity") +
  labs(x="Number of females",
       y="Probability")

pB <- ggplot(prFemale_df, aes(num_female, cumul_prob)) +
  geom_bar(stat="identity") +
  labs(x="Number of females", y="Cumulative probability")
plot_grid(pA, pB)
```



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 5.

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 $\lambda = \mu = \sigma^2$ which defines the Poisson distribution.

Q. 2.29

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

```
calc_poisson_prob <- function(x, y_bar) {
  # Translate the formula here using x and y_bar
}
```

```
# In R, e^2 = exp(2)
```

```
# In R, 3! = factorial(3)
```

!

```
calc_poisson_prob <- function(x, y_bar) {
  (y_bar^x * exp(-y_bar)) / factorial(x)
}
```

2.2.1 Poisson distributions by hand

When using the Poisson distribution, 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.

My observations can be found in the "WormCasts" sheet in "H2SMS_practicalData.xlsx". Read in the data and calculate the mean number of worm casts per quadrat as well as the range of observations.

```
worm_df <- read_excel("data/H2SMS_practicalData.xlsx", sheet="WormCasts")
head(worm_df)
```

```
# A tibble: 6 x 2
  quadrat casts
    <dbl> <dbl>
1       1     0
2       2     0
3       3     1
4       4     1
5       5     0
6       6     1
```

You should find a mean of 1.17, with values ranging between 0-6.

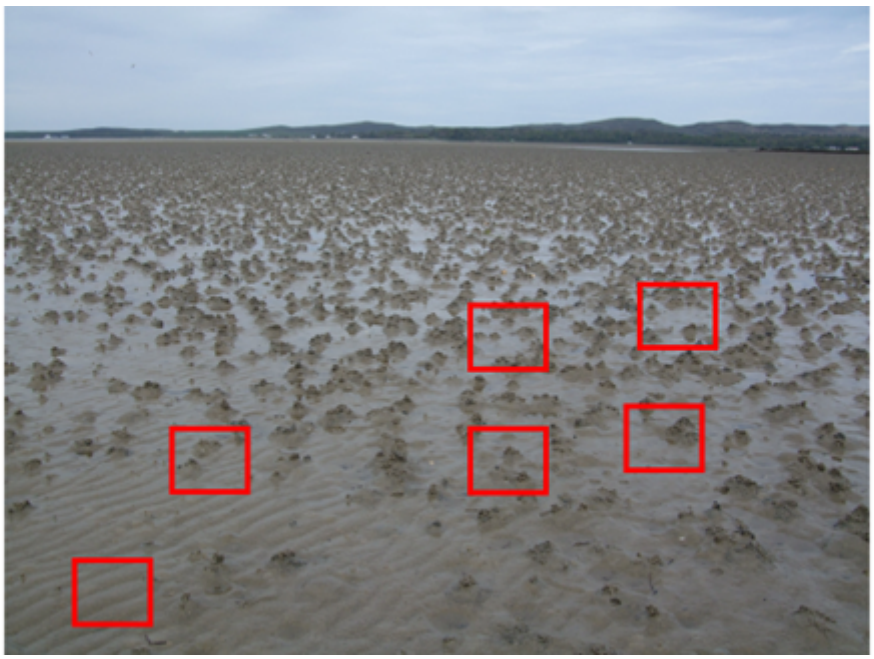


Figure 2.3: Worm casts on a sandy beach with 1x1 m quadrats.

```
mean_casts <- mean(worm_df$casts)
```

We wish to predict the proportion of quadrats that would contain 0, 1, 2, 3, 4, 5, and 6 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.

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. 2.30

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

!

```
calc_poisson_prob(x=0:2, y_bar=mean_casts)
```

```
[1] 0.3103669 0.3631293 0.2124307
```

Q. 2.31

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

! The upper limit is positive infinity, so to fully calculate the probability, use $1 - P(y_i < 1)$

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 using the mean count of worms per quadrat.

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 1.17 worms per quadrat, $P(y_i \leq 5) = 0.999$ (3 sf). This means that if your data are Poisson distributed with $\lambda = 1.17$, it is highly unlikely to find more than five worms in a quadrat.

Q. 2.32

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

! More likely, since the mean is larger. Proof:

```
dpois(5, 1.17)
```

```
[1] 0.005670528
```

```
dpois(5, 3)
```

```
[1] 0.1008188
```

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 <- 1.17
dpois(a, lambda) # probability of observing 'a' worms per quadrat: dpois()
```

```
[1] 0.005670528
```

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

```
[1] 0.006992103
```

Q. 2.33

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

! As with `pbinom()`, the specified value is included in the sum. So we would use `a` instead of `a-1` if we were interested in $P(y_i \leq a)$ instead of $P(y_i < a)$.
For the converse (greater than, as in the example above), the same logic applies.

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

```
[1] 0.3103669 0.6734963 0.8859269 0.9687749 0.9930079 0.9986784
```

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

```
[1] 0.310 0.673 0.886 0.969 0.993 0.999
```

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

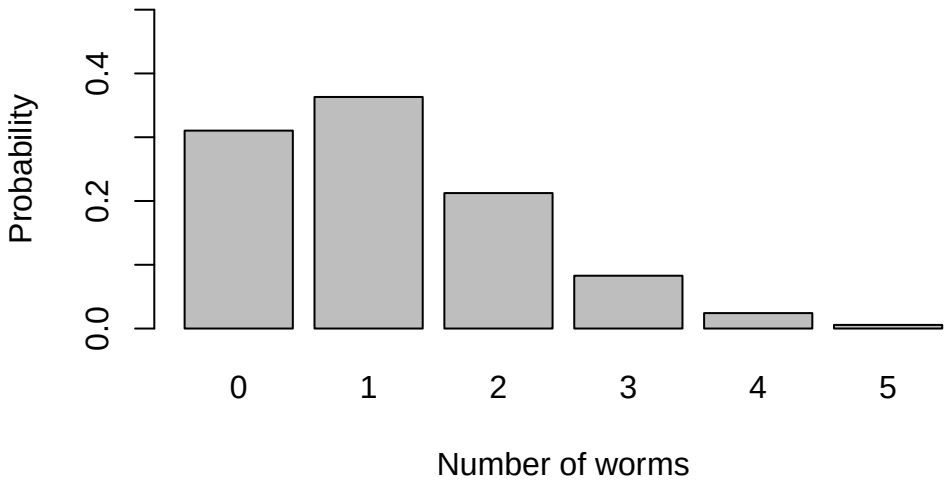


Figure 2.4: Poisson probability distribution.

Q. 2.34

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

!

```
barplot(ppois(0:a, mean_casts),
        ylab="Cumulative probability", xlab="Number of worms",
        space=0.2, ylim=c(0, 1), names.arg=0:a)
```



To compare my observations with these theoretical Poisson expectations, we need to tally the number of 0, 1, etc worms. The easiest way is with `count()`.

```
worm_summary <- worm_df |>
  count(casts) |>
  mutate(prob=dpois(casts, mean_casts),
         cumul_prob=cumsum(prob))
```

```
# Make Table 2.1. Use packageName::function() instead of loading with library()
knitr::kable(worm_summary, digits=5)
```

Table 2.1: Worm cast observations and expected probabilities.

casts	n	prob	cumul_prob
0	33	0.31037	0.31037
1	37	0.36313	0.67350
2	16	0.21243	0.88593
3	9	0.08285	0.96877
4	4	0.02423	0.99301
5	1	0.00567	0.99868

Q. 2.35

Format the probabilities in Table 2.1 to 3 decimal places.

```
knitr::kable(worm_summary, digits=3)
```

casts	n	prob	cumul_prob
0	33	0.310	0.310
1	37	0.363	0.673
2	16	0.212	0.886
3	9	0.083	0.969
4	4	0.024	0.993
5	1	0.006	0.999

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. 2.36

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

```
worm_summary$n_expected <- 100 * worm_summary$prob
```

This is the number of quadrats that you would expect to contain 0, 1, ..., 5 worms, given that the mean density of worms is 1.17 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. 2.37

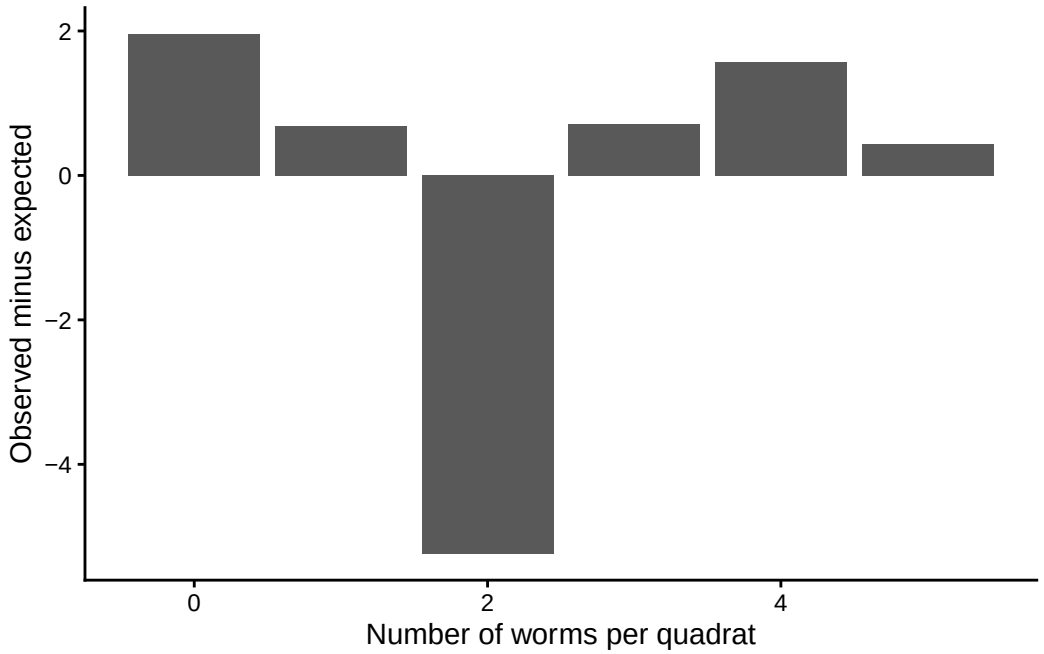
Add another column that is the difference in the observed number of quadrats and `n_expected`.

```
worm_summary$diff <- worm_summary$n - worm_summary$n_expected
```

Q. 2.38

Produce a bar graph of this difference.

```
ggplot(worm_summary, aes(casts, diff)) +
  geom_bar(stat="identity") +
  labs(x="Number of worms per quadrat",
       y="Observed minus expected")
```



Q. 2.39

Calculate the observed variance. Do these observations tend to be overdispersed or underdispersed (= clumped)?

```
var(worm_df$casts)
```

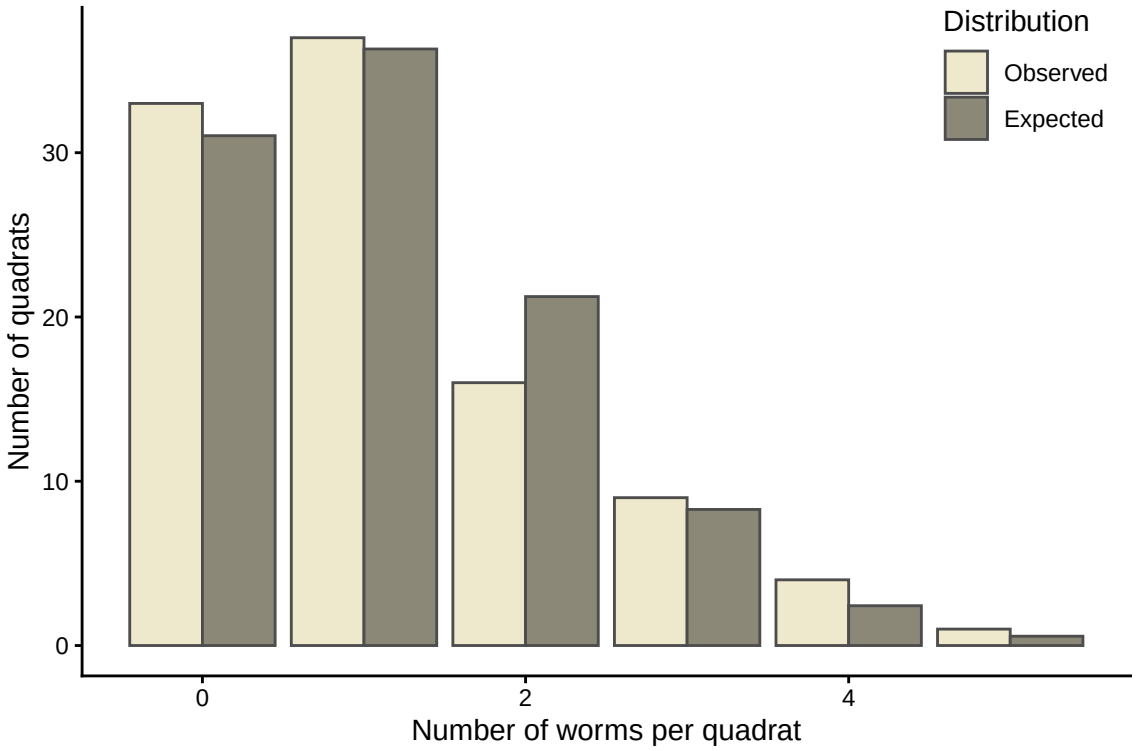
```
[1] 1.354646
```

They tend to be overdispersed, with higher variance than expected under a Poisson distribution. That is, there are more low and high values than predicted and fewer intermediate values than expected.

Here is a figure showing the distributions side-by-side:

```
worm_summary |>
  select(casts, n, n_expected) |>
  pivot_longer(2:3) |>
  ggplot(aes(casts, value, fill=name)) +
  geom_bar(stat="identity", position="dodge", colour="grey30") +
  scale_fill_manual("Distribution",
                    values=c("cornsilk2", "cornsilk4"),
                    labels=c("Observed", "Expected")) +
  labs(x="Number of worms per quadrat",
       y="Number of quadrats") +
  theme(legend.position="inside",
```

Table 2.2: Worm cast observations and expected count from the Poisson distribution.




```
        legend.position.inside=c(0.9, 0.9))
```

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))
```

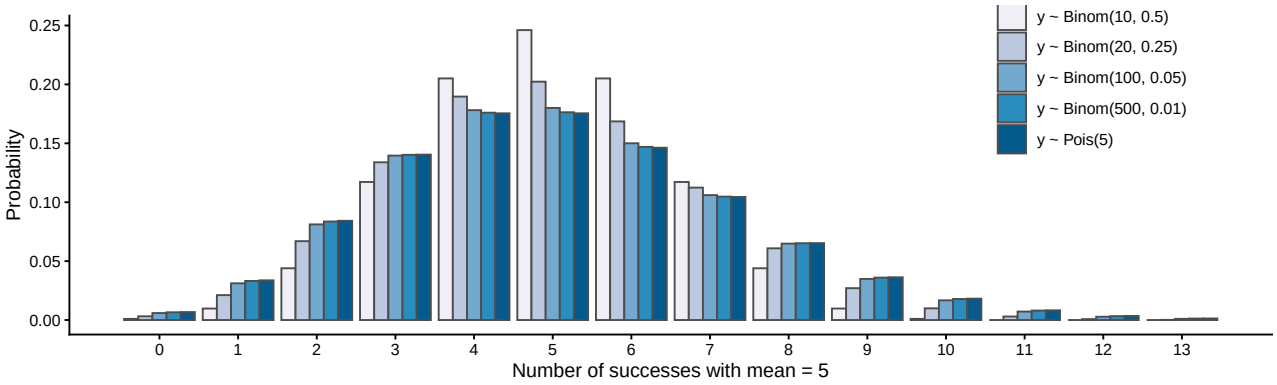


Figure 2.5: Binomial and Poisson distributions converge with larger numbers of events, depending on p.

Q. 2.40

What is the modal value in each of these distributions?

! 5 in all cases.

Q. 2.41

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

```
y1 <- rbinom(1000, 10, 0.5)
y2 <- rbinom(1000, 20, 0.25)
y3 <- rbinom(1000, 100, 0.05)
y4 <- rbinom(1000, 500, 0.01)
y5 <- rpois(1000, 5)

paste0("Mean: ", signif(mean(y1), 3), ", variance: ", signif(var(y1), 3))
```

```
[1] "Mean: 4.98, variance: 2.56"
```

```
paste0("Mean: ", signif(mean(y2), 3), ", variance: ", signif(var(y2), 3))
```

```
[1] "Mean: 5.01, variance: 3.67"
```

```
paste0("Mean: ", signif(mean(y3), 3), ", variance: ", signif(var(y3), 3))
```

```
[1] "Mean: 4.97, variance: 4.73"
```

```
paste0("Mean: ", signif(mean(y4), 3), ", variance: ", signif(var(y4), 3))
```

```
[1] "Mean: 4.96, variance: 4.37"
```

```
paste0("Mean: ", signif(mean(y5), 3), ", variance: ", signif(var(y5), 3))
```

```
[1] "Mean: 4.98, variance: 4.96"
```

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

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

```
[1] "Mean: 5, variance: 5.19"
```

Q. 2.42

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

They are very similar, but not identical since this is a sample.

Q. 2.43

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?

They become increasingly similar. Yes, this is expected because this is what the text above says and we (briefly) discussed it in the lecture.

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

View original

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?

! Binomial and Poisson

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); theme_set(theme_classic())
library(readxl)
set.seed(2026)
```

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?

! `-Inf` to `Inf`

Q. 3.3

Do you think normality a fair assumption for these data?

! Yes, because the impossible values (e.g. < 0) are so low probability that we can ignore them: >99% of the data fall within 3 sd of the mean in a normal distribution (here, $37.6 \pm 3 * 1.20 = 34.0, 41.2$).

Q. 3.4

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

! $y \sim Normal(\mu = 37.6cm, \sigma = 1.20cm)$

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.

```
# NB: This code is not required! It is here for the curious.

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()
```

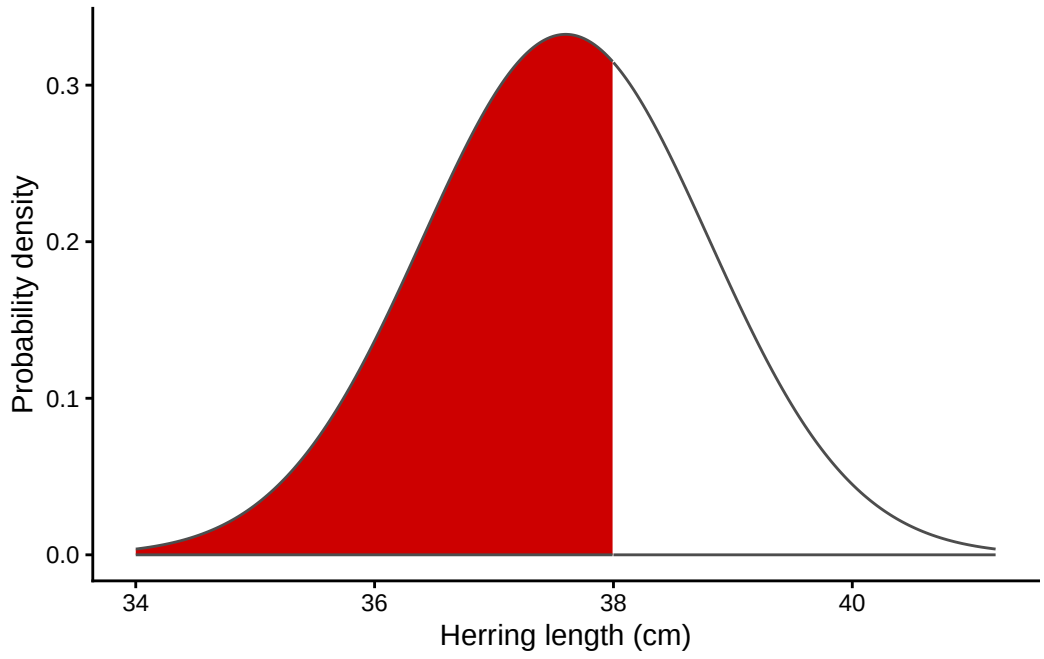


Figure 3.1: Normal distribution of herring lengths.

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?

! Loosely S-shaped, as shown in the lower left panel of Figure 3.2.

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")
```

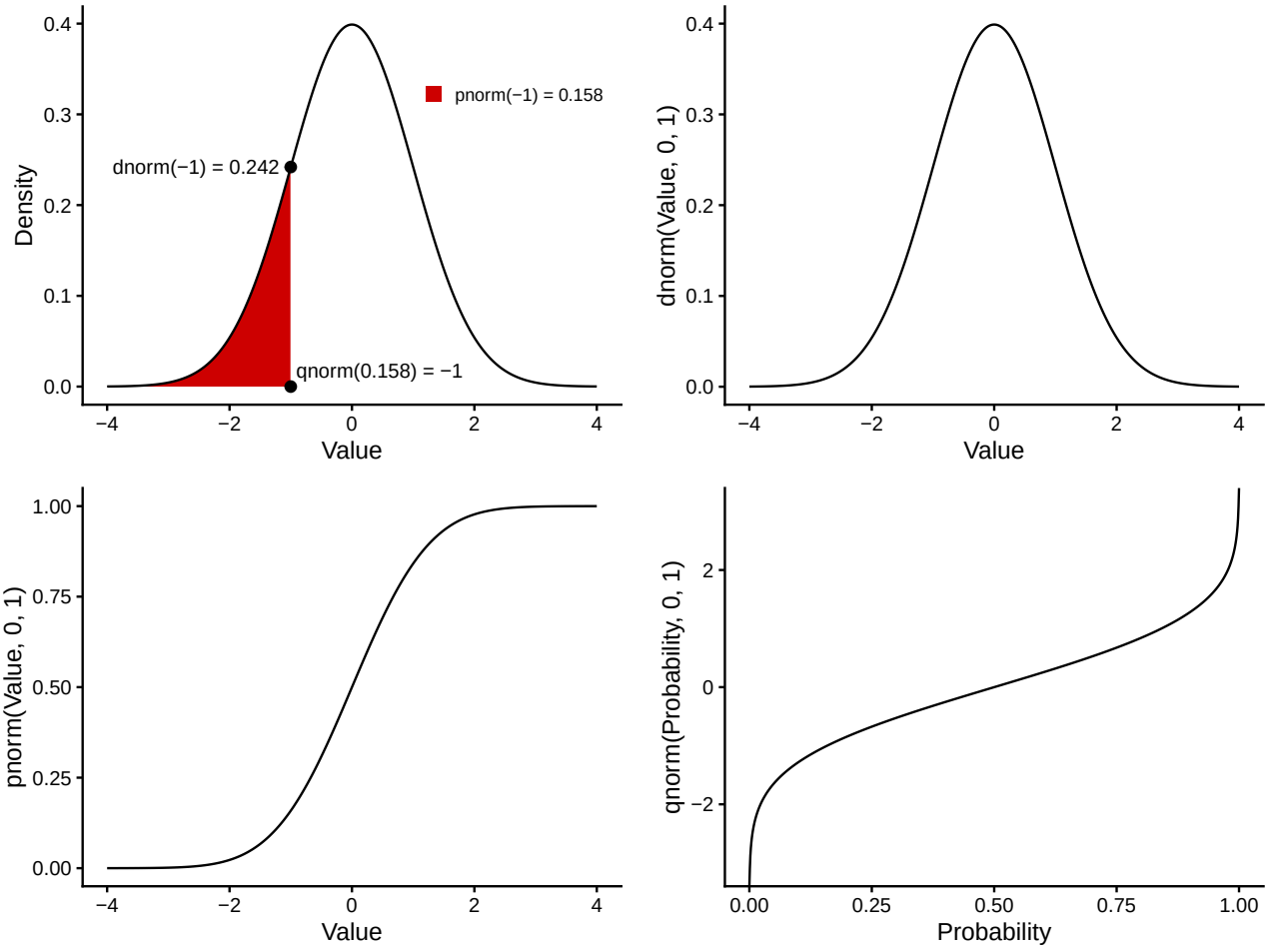


Figure 3.2: Functions for distributions in R illustrated with a standard normal.

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?

```
pnorm(35, 37.6, 1.2)
```

```
[1] 0.01513014
```

Q. 3.7

What proportion of individuals are greater than 39 cm?

```
1 - pnorm(39, 37.6, 1.2)
```

```
[1] 0.1216725
```

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?

! The mean is 15.2 cm and the standard deviation is 25.1 cm.

Q. 3.9

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

!

```
pnorm(0, 15.2, 25.1)
```

```
[1] 0.2723976
```

Q. 3.10

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

! It is not reasonable to assume that this population is normally distributed.

Q. 3.11

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

! Yes.

```
pnorm(0, 15.2, 2.51)
```

```
[1] 6.987079e-10
```

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 $y \sim Norm(37.6, 1.2)$.

! Middle 95%:

```
qnorm(c(0.025, 0.975), 37.6, 1.2)
```

```
[1] 35.24804 39.95196
```

Middle 90%:

```
qnorm(c(0.05, 0.95), 37.6, 1.2)
```

```
[1] 35.62618 39.57382
```

Q. 3.13

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

! 37.6, since for the symmetric normal distribution, the mean is equal to the median.

Q. 3.14

Check your answer above using `qnorm()`.

!

```
qnorm(0.5, 37.6, 1.2)
```

```
[1] 37.6
```

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

Copy and paste the code below:

```
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
```

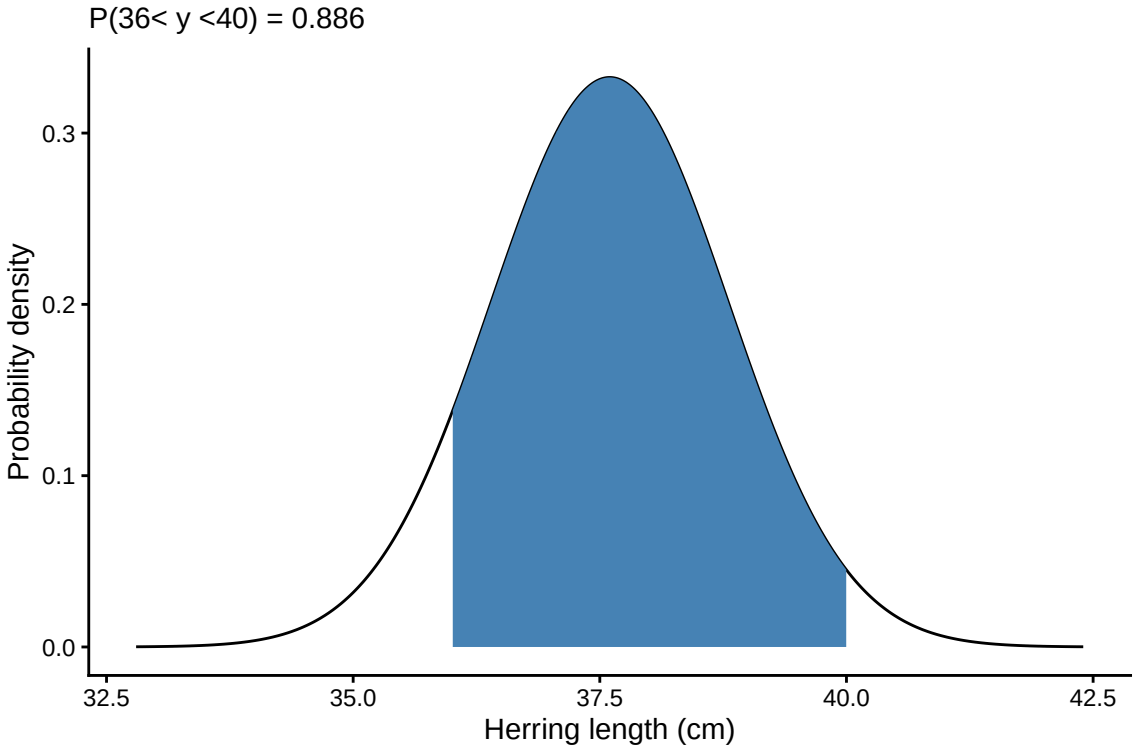


Figure 3.3: The normal distribution illustrating probability of a random herring from your population having a length between 36 and 40 cm.

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.

! Should change `lb <- qnorm(0.025, 37.6, 1.2) # ~35.25` and `ub <- qnorm(0.975, 37.6, 1.2) # ~39.95`.

Q. 3.16

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

! Nothing. Probability is always calculated as an integral (area under the curve) between two points.

Q. 3.17

Does this also apply to discontinuous data?

! No, since the outcomes are discrete.

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.

```
# Double
qnorm(c(0.025, 0.975), 37.6, 1.44*2)
```

```
[1] 31.9553 43.2447
```

```
# Quadruple
qnorm(c(0.025, 0.975), 37.6, 1.44*4)
```

```
[1] 26.31061 48.88939
```

```
# Half
qnorm(c(0.025, 0.975), 37.6, 1.44/2)
```

```
[1] 36.18883 39.01117
```

Q. 3.19

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

Hopefully. A larger standard deviation represents a more dispersed population, so the intervals become wider.

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?

```
(34 - 37.6)/1.2
```

```
[1] -3
```

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/H2SMS_practicalData.xlsx", sheet = "Mood shrimp")
# check the data using head(), str() etc.
```

We'll learn more advanced methods in **?@sec-P5** 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
```

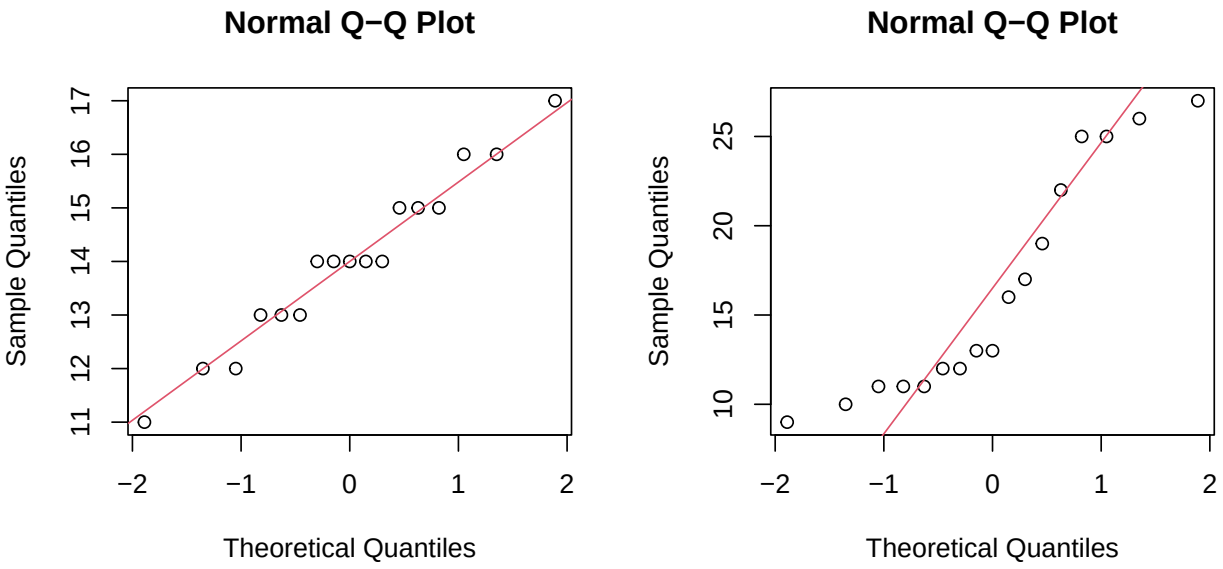


Figure 3.4: Normal (QQ) plots. The left indicates that the normality assumption might be reasonable, not so on the right.

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.

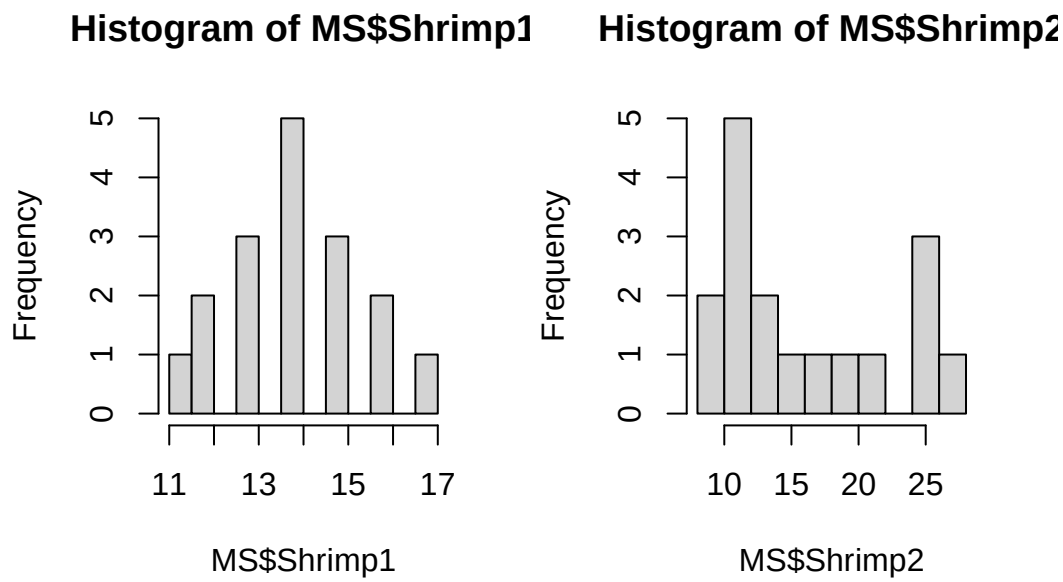
As sample size decreases, it becomes increasingly difficult to see if your data are reasonably approximated by a normal distribution. There are many formal statistical methods for assessing normality, but as we already know, it is impossible for a population to be distributed perfectly normally and this 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.

! Both are small samples (n=NULL), but Shrimp1 is approximately symmetric and looks more normal than Shrimp2, which is heavily skewed right.

```
par(mfrow=c(1,2))
hist(MS$Shrimp1, breaks=10)
hist(MS$Shrimp2, breaks=10)
```



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.

The best solution is to use an appropriate member of the GLM family.

An alternative 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 **?@sec-P5**) 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., **?@sec-P6**), which is unrelated to model assumptions. Values must

be >0.

- 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.

```
library(ggdist)

tibble(x_norm=rnorm(0.5e3, 2, 0.5)) |>
  mutate(x_sq=x_norm^2,
         x_exp=exp(x_norm),
         x_recip=1/x_norm) |>
  select(-x_norm) |>
  pivot_longer(everything()) |>
  bind_rows(
    tibble(x_invlogit=rnorm(0.5e3, 0, 1.5)) |>
      mutate(x_logit=boot::inv.logit(x_invlogit)) |>
      select(-x_invlogit) |>
      pivot_longer(everything())
  ) |>
  mutate(name=factor(name,
                     levels=paste0("x_", c("exp", "sq", "logit", "recip")),
                     labels=paste(c("Log", "Square-root", "Logit", "Reciprocal")))) |>
  ggplot(aes(value, group=name)) +
  geom_dots(side="bottom", scale=1) +
  stat_halfeye(scale=1, .width=c(0.5, 0.95), normalize="panels") +
  facet_wrap(~name, scales="free", nrow=1) +
  theme(axis.text.y=element_blank(),
        axis.title.y=element_blank(),
        axis.ticks.y=element_blank())
```

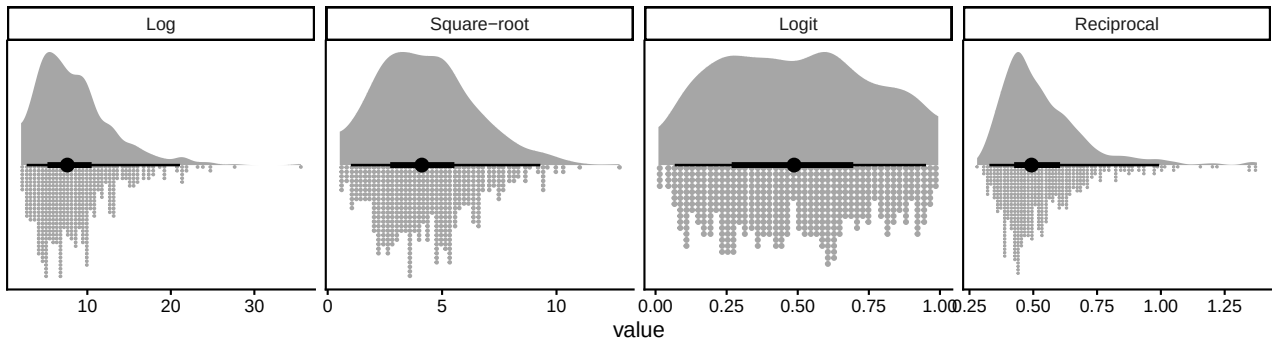


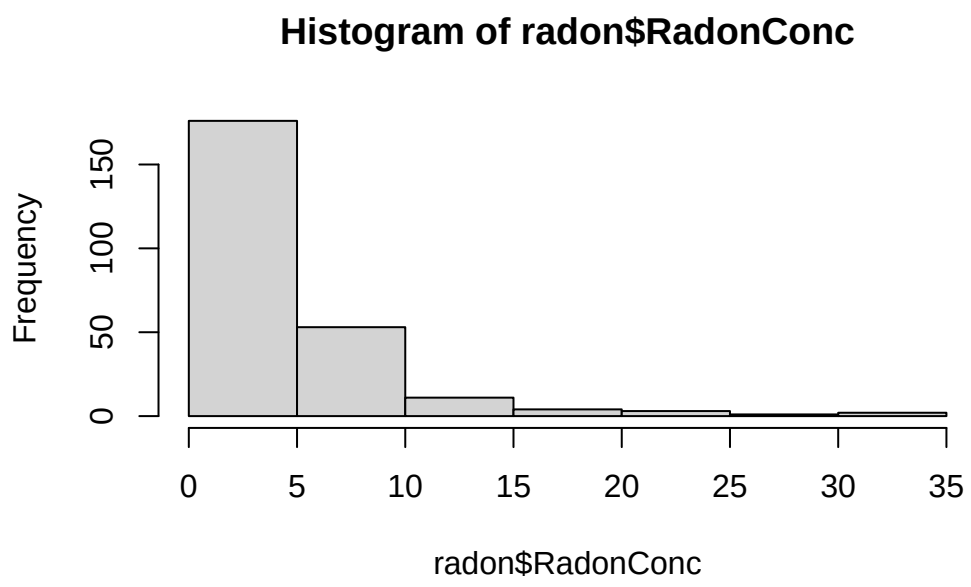
Figure 3.5: Distributions needing transformation for normality. For each panel, applying the stated transformation will result in an approximately normal distribution. Note that often there are multiple transformations that will work.

Q. 3.22

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

! Definitely not.

```
radon <- read_xlsx("data/H2SMS_practicalData.xlsx", sheet="Radon (ppb)")
hist(radon$RadonConc)
```



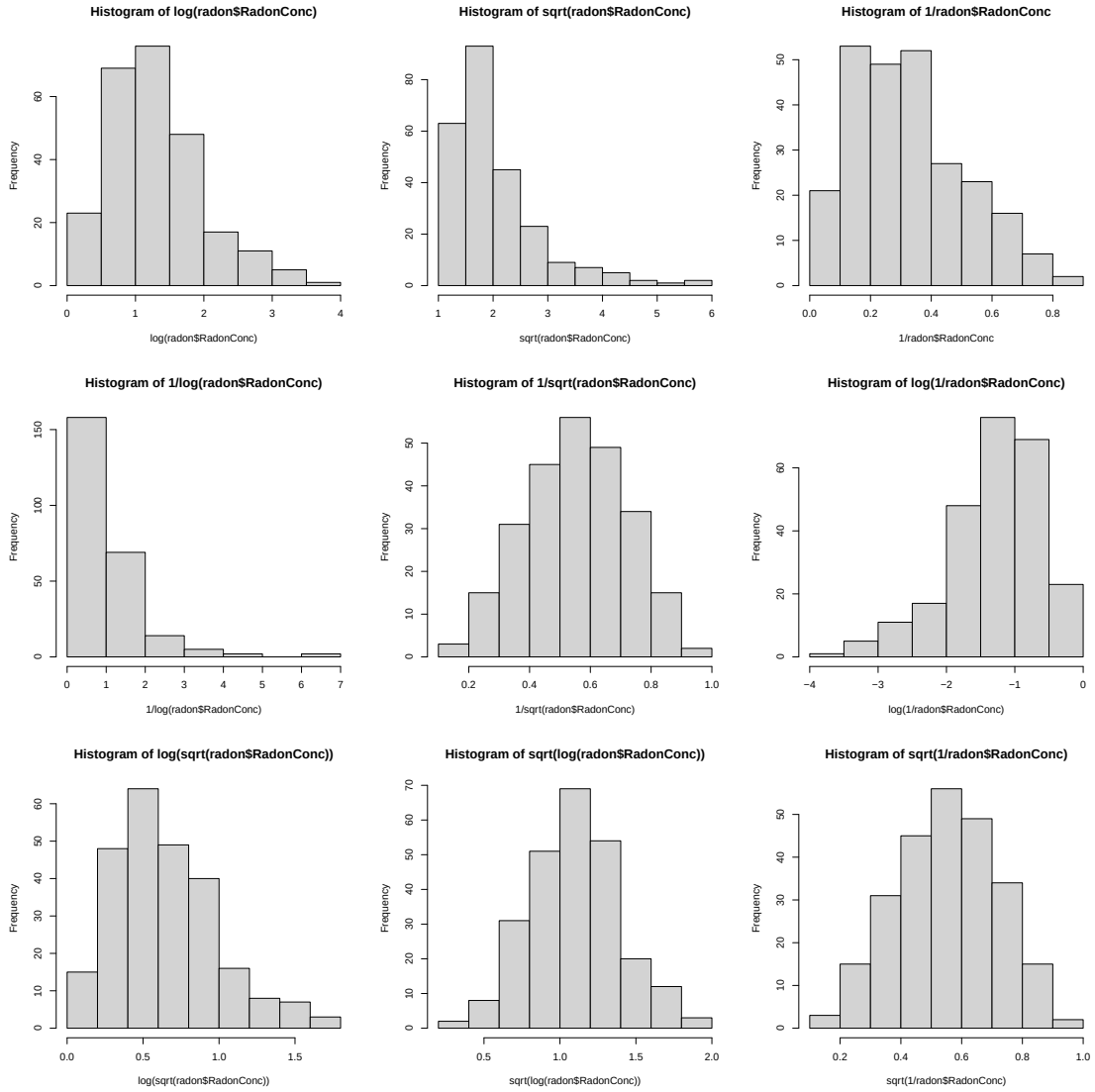

```
# get the Radon data into R; the units are parts per billion
readxl::excel_sheets("data/H2SMS_practicalData.xlsx")
```

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).

!

```
par(mfrow=c(3,3))
hist(log(radon$RadonConc))
hist(sqrt(radon$RadonConc))
hist(1/radon$RadonConc)
hist(1/log(radon$RadonConc))
hist(1/sqrt(radon$RadonConc))
hist(log(1/radon$RadonConc))
hist(log(sqrt(radon$RadonConc)))
hist(sqrt(log(radon$RadonConc)))
hist(sqrt(1/radon$RadonConc))
```



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

Copy 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)
```

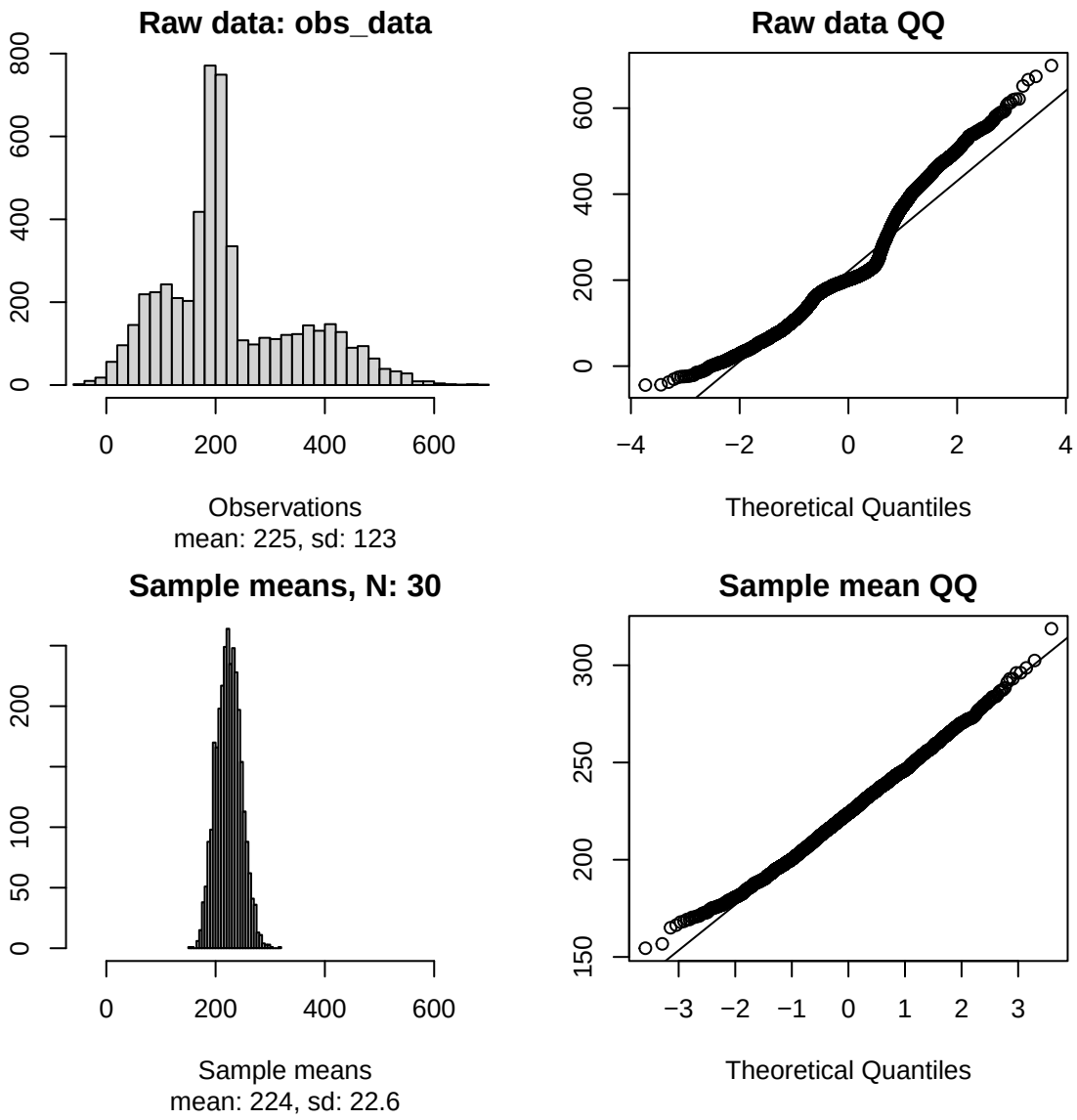


Figure 3.6: The central limit theorem in action.

Q. 3.24

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

! The mean of the sample means is located at the population mean regardless of sample size.

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?

! Smaller sample sizes result in more variability in sample means. The standard deviation of the distribution of sample means is the standard error, and it is a function of sample size.

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 3.6 : `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.

Copy and paste the code below:

```
# 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()
```

Copy and paste the code below:

```
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()
```

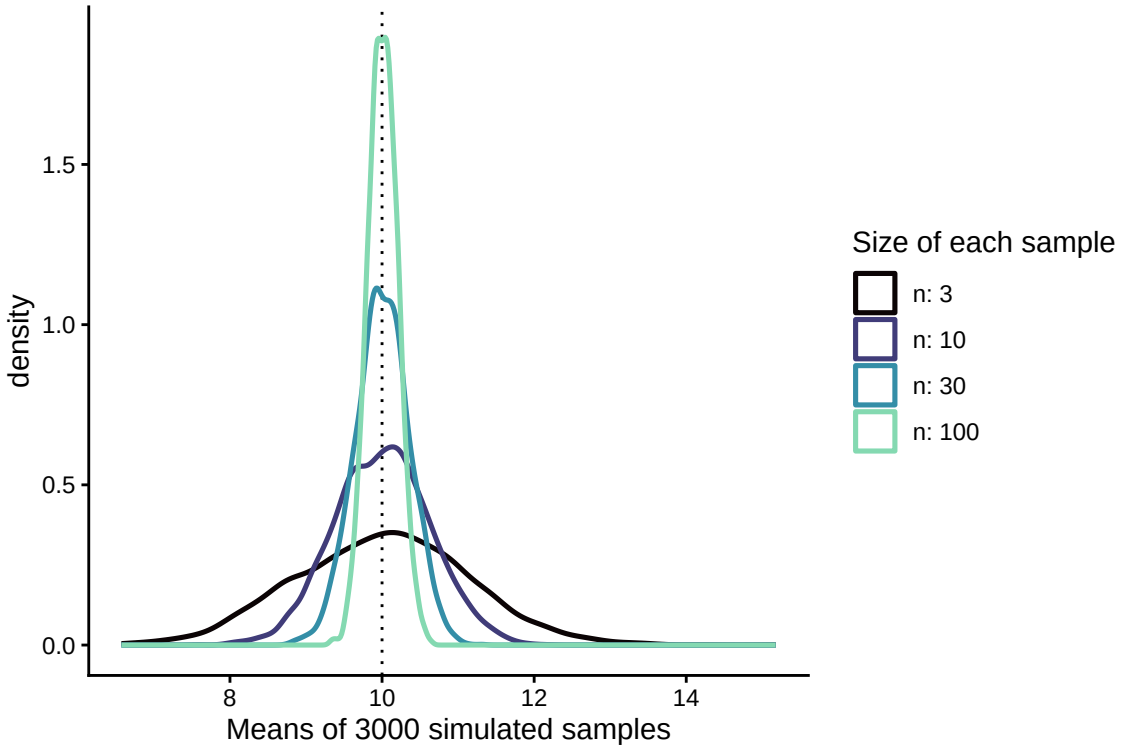


Figure 3.7: Simulated sample means. Each curve shows the sampling distribution for means of samples with size n. The standard deviation of each curve is the standard error of the mean.

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    1.14         1.15
2    10    0.629        0.632
3    30    0.356        0.365
4   100    0.198        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()`.

! The CLT should hold ($\bar{y} \sim Normal(\mu, SEM)$) for N > 30 or so. With N < 30, the means may no longer be approximately normally distributed.

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).

! They should.

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.

i 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: $y \sim Norm(...)$.

! In a Poisson distribution, the mean is equal to the variance, so $y \sim Normal(\mu = 10, \sigma = \sqrt{10})$.

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 $y \sim Pois(\lambda = 40)$ and compare it to the same probability under the normal approximation of this distribution.

! Reasonably close:

```
ppois(35, 40)
```

```
[1] 0.2424142
```

```
pnorm(35, 40, sqrt(40))
```

```
[1] 0.2145977
```

Q. 3.31

Evaluate $P(y_i < 3 \mid \lambda = 2)$ using the Poisson model and its normal approximation.

! Rather further

```
ppois(2, 2)
```

```
[1] 0.6766764
```

```
pnorm(2, 2, sqrt(2))
```

```
[1] 0.5
```

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. Recall that q is the probability of the other outcome $(1-p)$. Roughly speaking, if $np > 5$ and $nq > 5$, 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 0.4 and clutch size is 40.

Q. 3.32

We've observed several clutches of eggs on a particular island with only 10 males. What is the probability of observing 10 or fewer males in this scenario?

!

```
pbinom(10, 40, 0.4)
```

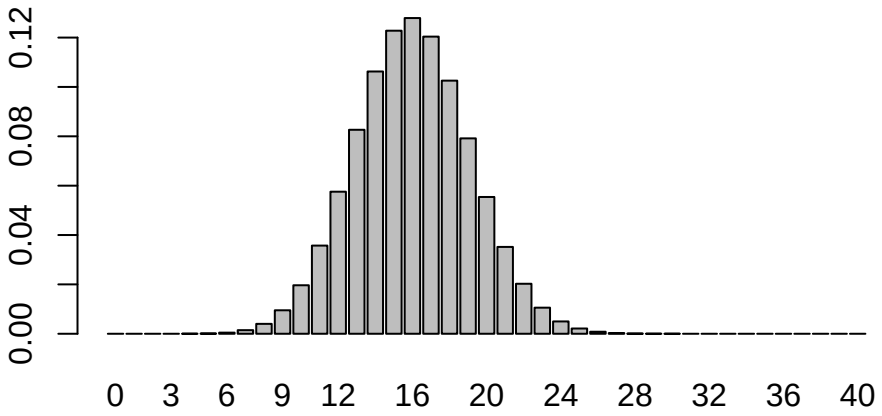
```
[1] 0.03522248
```

Q. 3.33

Plot the probability from 0 to 40 males as a bar graph and comment on its shape.

! It is symmetric and looks similar in shape to a normal distribution

```
barplot(dbinom(0:40, size=40, p=0.4), names.arg=0:40)
```



Q. 3.34

Calculate the mean and variance of this population.

! Mean $= np$ and variance $= npq$, so:

```
40 * 0.4 # mean
```

```
[1] 16
```

```
40 * 0.4 * 0.6 # variance
```

```
[1] 9.6
```

Q. 3.35

Are np and nq both > 5?

! Yes.

```
40 * 0.4
```

```
[1] 16
```

```
40 * 0.6
```

```
[1] 24
```

Q. 3.36

Specify the normal distribution that approximates this binomial distribution.

! $y \sim Normal(\mu = 16, \sigma = \sqrt{9.6})$

Q. 3.37

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

!

```
pnorm(10, 16, sqrt(9.6))
```

```
[1] 0.02640376
```

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

View original

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); theme_set(theme_classic())
library(readxl)
set.seed(2026)
```

```
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))
```

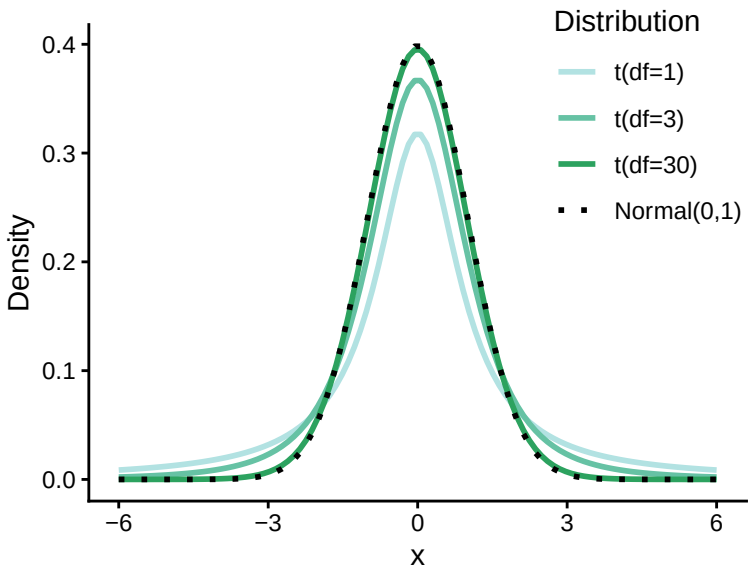


Figure 4.1: Comparison of different t-distributions (note that at df=30, the distribution is nearly identical to the normal distribution).

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?

! Clearly not.

Q. 4.2

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

! A larger sample leads to a more precise estimate (narrower confidence intervals) on average.

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?

! One tailed.

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/H2SMS_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)
```

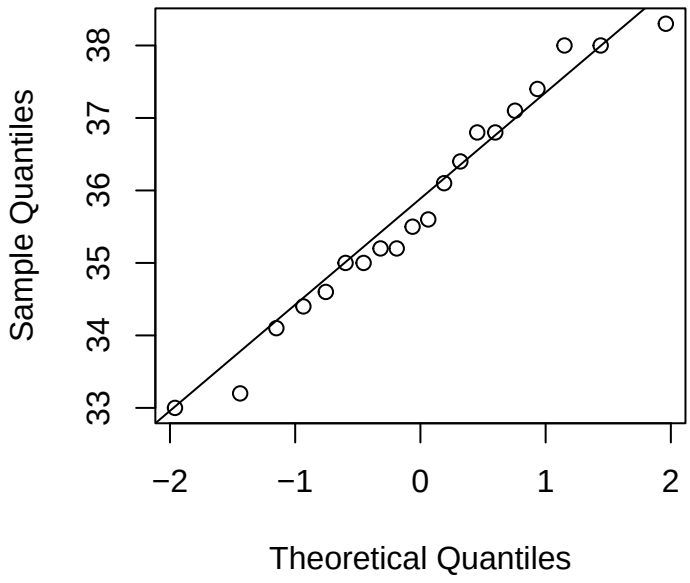


Figure 4.2: QQ-plot for the sample of cod.

Q. 4.4

Do you think the normality assumption is reasonable?

! Yes, the qqplot shows that the data are approximately normally distributed.

Q. 4.5

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

! We are interested in the population mean μ, which we approximate using the sample mean $\bar{y}$ which is our sample statistic. The distribution of possible sample means is centred on the population mean (assuming random samples), becomes narrower as sample size increases, and tends to be approximately normal when a) the data are approximately normal, or b) sample size is large enough (about 30 or more).

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 5).

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?

! There is about a 1.5% chance of collecting a sample (n=20) with a mean this small if the population mean is 36.6 or more. You may thus conclude illegal practices, or a follow-up sample with a larger sample size.

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?

! Yes, if $\alpha = 0.05$.

Q. 4.12

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

! No.

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?

! Decreasing α lowers the Type I error rate but raises the Type II error rate. You are less likely to falsely reject H_0 but more likely to falsely 'accept' it.

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?

! $\sigma = 10$

Q. 4.15

What does the symbol '~' mean?

! "Is modelled as", "is distributed as", or "is predicted by"

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.

Copy the following code:

```
# 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))
}
```

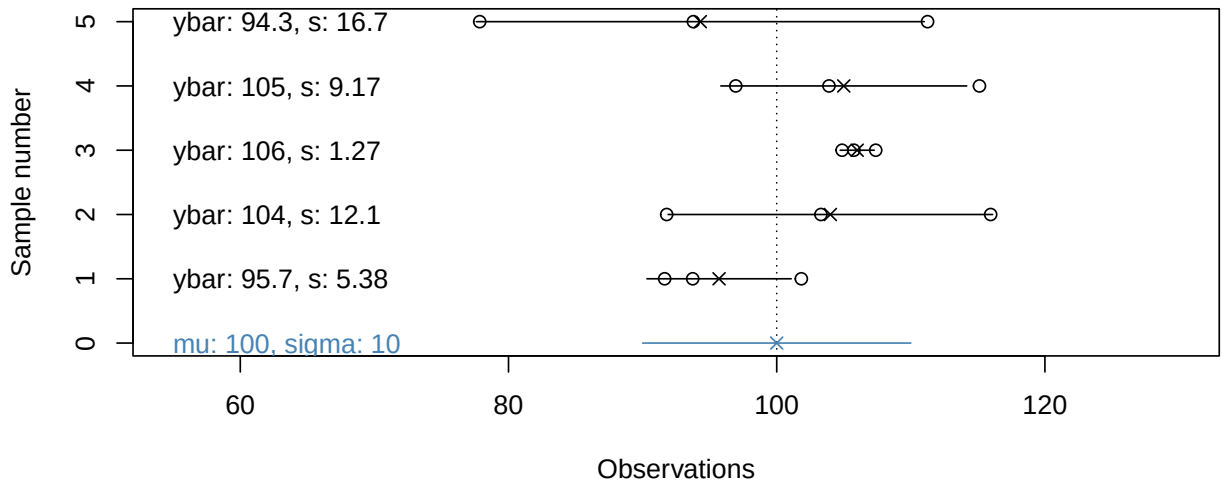


Figure 4.3: Repeated samples from a normal distribution. Open points are observations in the sample, with 'x' giving the mean and lines showing 1 sd.

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?

! Yes, there is a discrepancy. The relative variability in the sample standard deviation seems larger (often more than an order of magnitude between the smallest and largest estimates).

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?

! The T-statistic is the value from this distribution *for a particular sample*. The t-distribution gives the distribution of T-statistics we would expect for a given sample size if the null hypothesis were true. The P-value gives the area of this distribution that is more extreme than the specific T-statistic of interest.

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.

! The distribution looks increasingly normal as `sample_size` increases. At small sample sizes, the peak is shorter and the tails are fatter because larger T-statistics are more likely.

Copy the following code:

```
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)"))
```

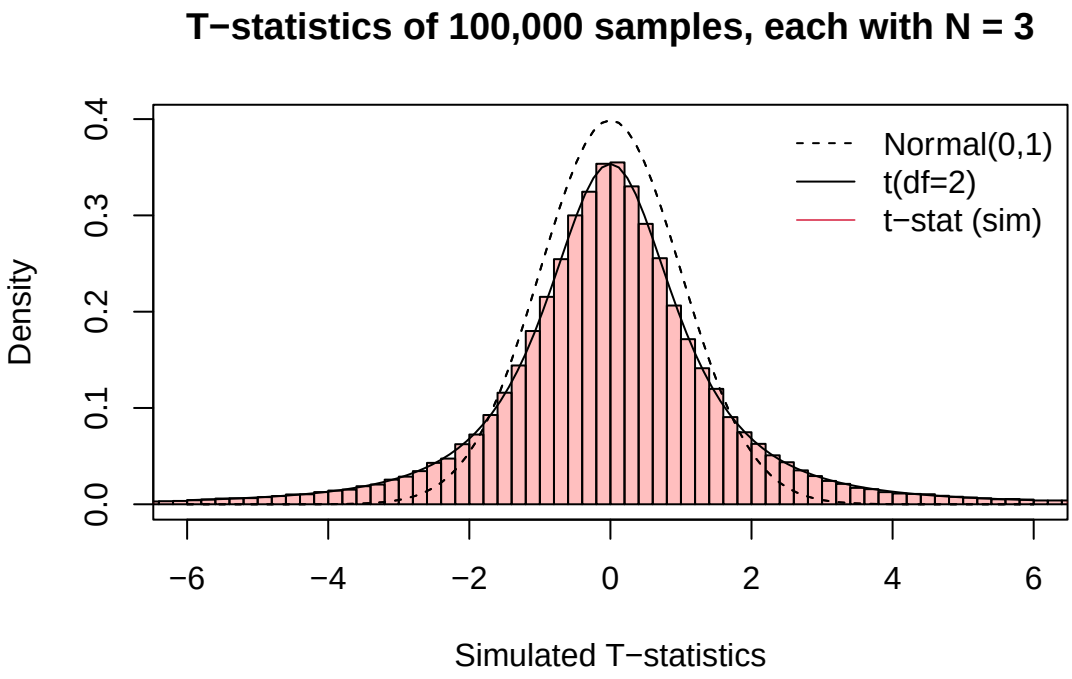


Figure 4.4: Histogram of 100,000 T-statistics calculated from 100,000 samples, along with the corresponding theoretical t-distribution (solid line) and a standard normal (dotted line).

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)))
```

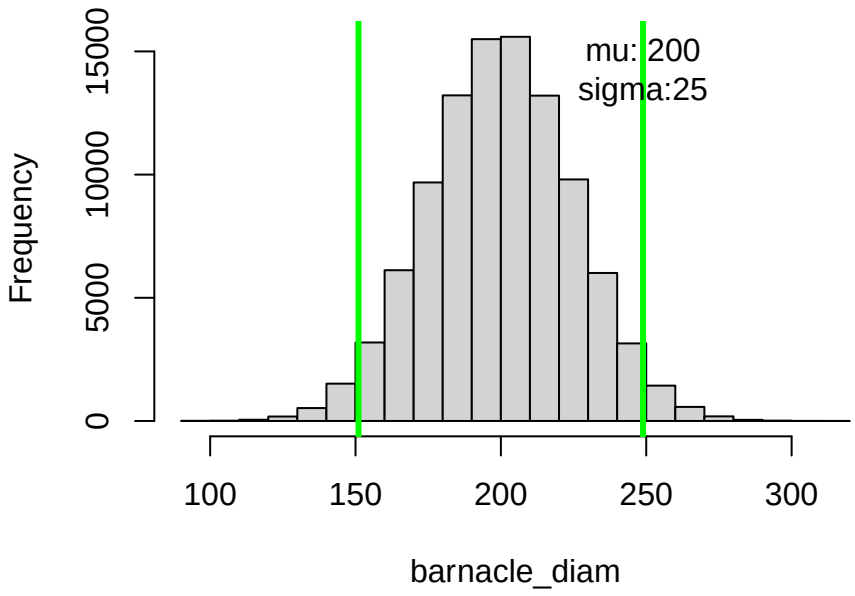


Figure 4.5: Histogram of a simulated barnacle population with 2.5% and 97.5% quantiles.

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
```

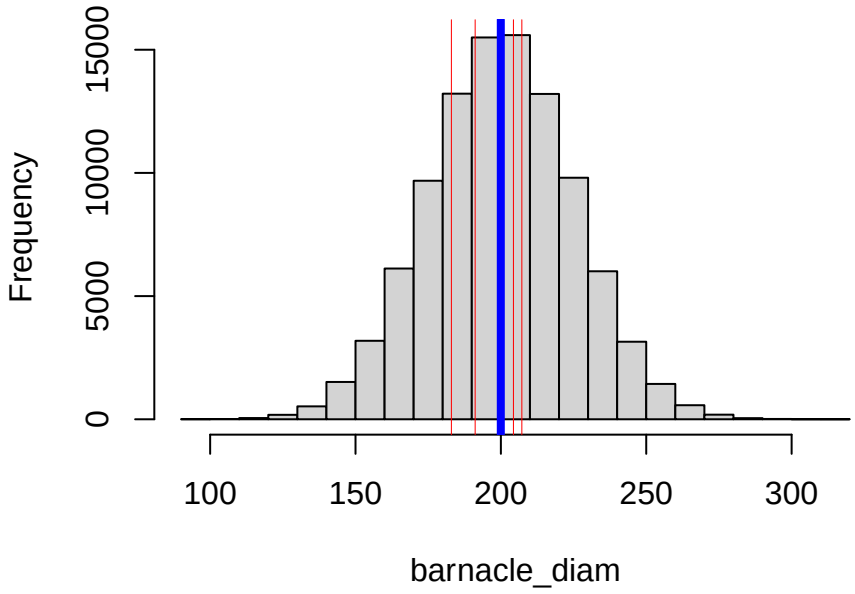


Figure 4.6: Histogram of the barnacle population showing location of 4 sample means (red lines), each with N = 5. The blue line shows the true population mean mu.

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)))
}
```

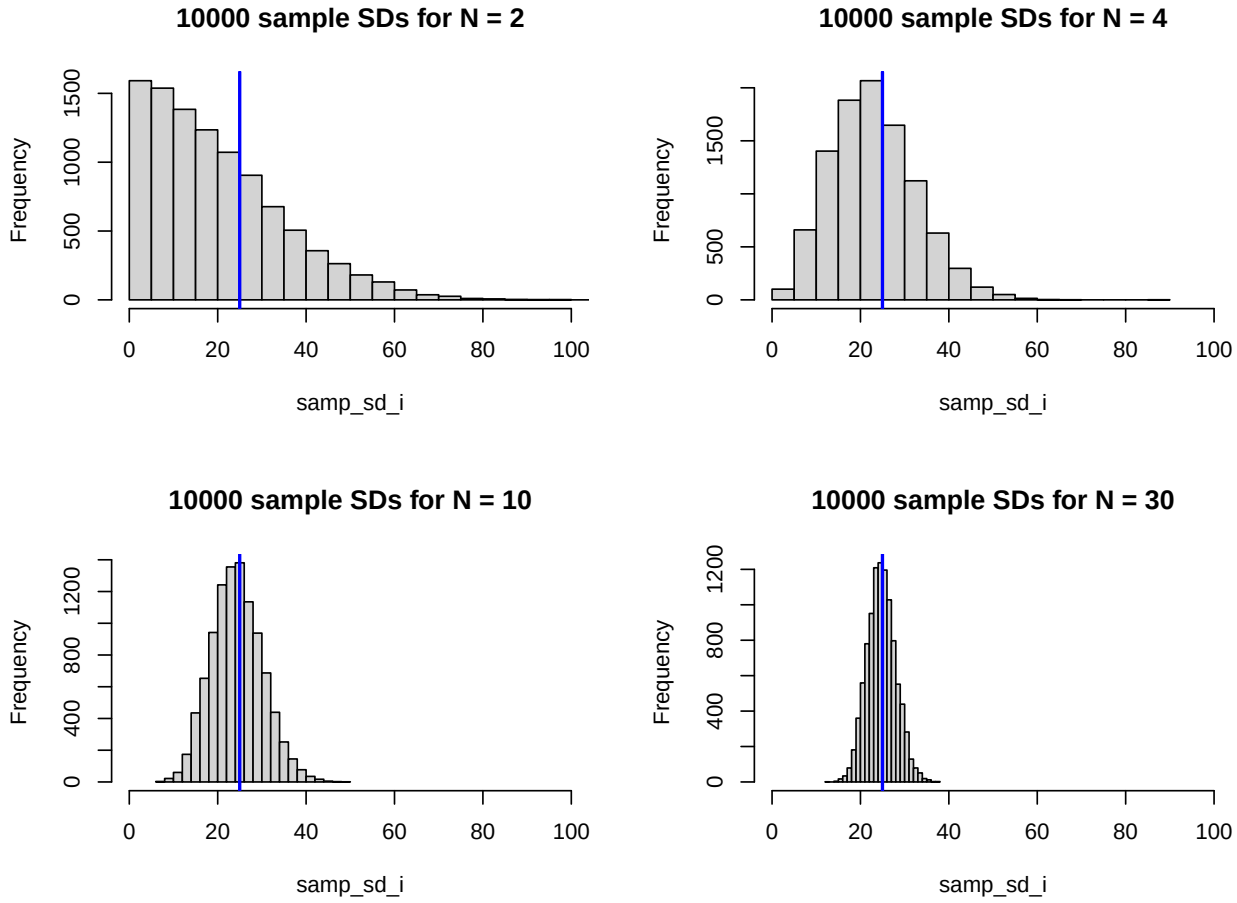


Figure 4.7: Histograms of sample standard deviations from repeated samples of the same barnacle population. The true population standard deviation is shown in blue.

```
par(mfrow=c(1,2))
plot(sim_df$N, sim_df$sd_median, col = "dodgerblue",
  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")
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, col = "dodgerblue",
  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")
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"
)
```

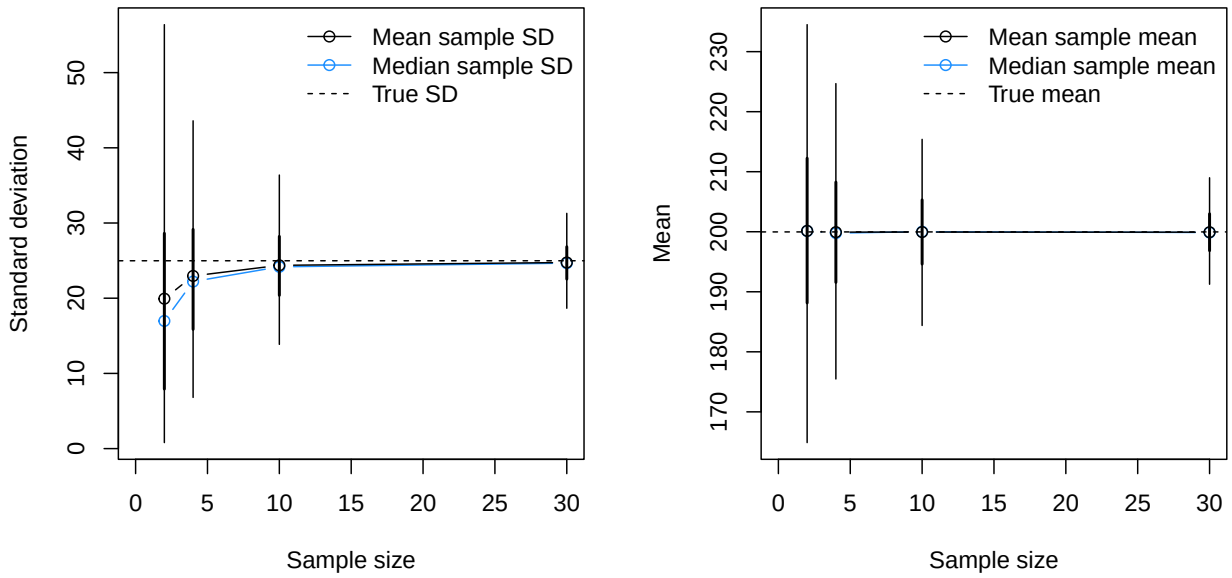


Figure 4.8: Mean (black), median (blue), and 50% and 95% quantiles (vertical lines) for (left) sample standard deviations at each sample size compared to the true population standard deviation (dotted line) or for the (right) sample means.

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?

! Yes. The distribution is skewed right, so the mean is larger than the median.

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.

i 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.

Copy the following code:

```
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)
}
```

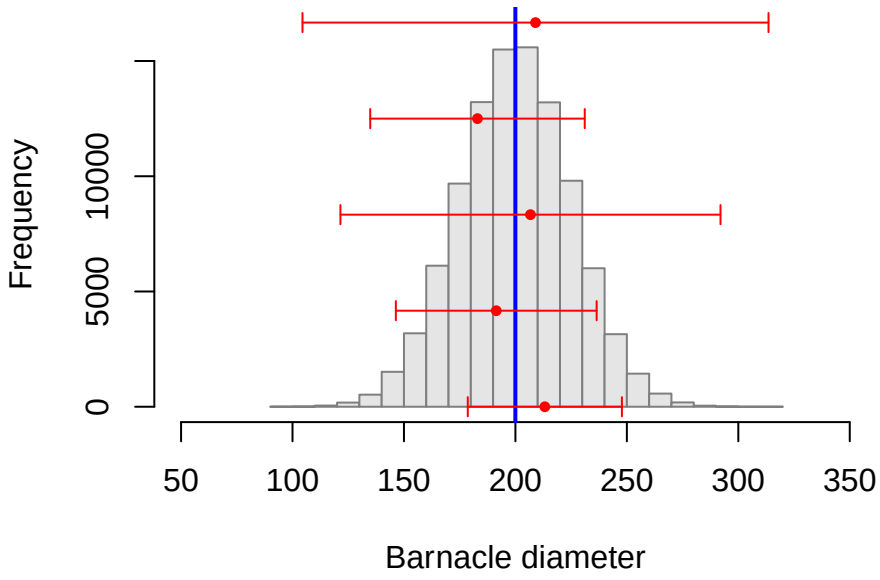


Figure 4.9: Histogram illustrating the barnacle population with population mean (blue) and sample means with 95% CIs (red) repeated across 5 samples.

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?

! Larger sample sizes produce narrower CIs on average.

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`?

! 95% of 95% CIs will capture the true value (on average) regardless of the sample size. This is how they are defined. Larger sample sizes decrease the width of the CIs but still result in 5% of samples having 95% CIs that *miss* the true value.

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()`).

! The first argument of `qt()` needs to be changed to `0.005` and `0.995` for 99% CIs and to `0.105` and `0.895` for 79% CIs.
Determine the percent that needs to be excluded, then divide that into upper and lower areas.

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())
```

```
Warning: `geom_errorbarh()` was deprecated in ggplot2 4.0.0.
i Please use the `orientation` argument of `geom_errorbar()` instead.

`height` was translated to `width`.
```

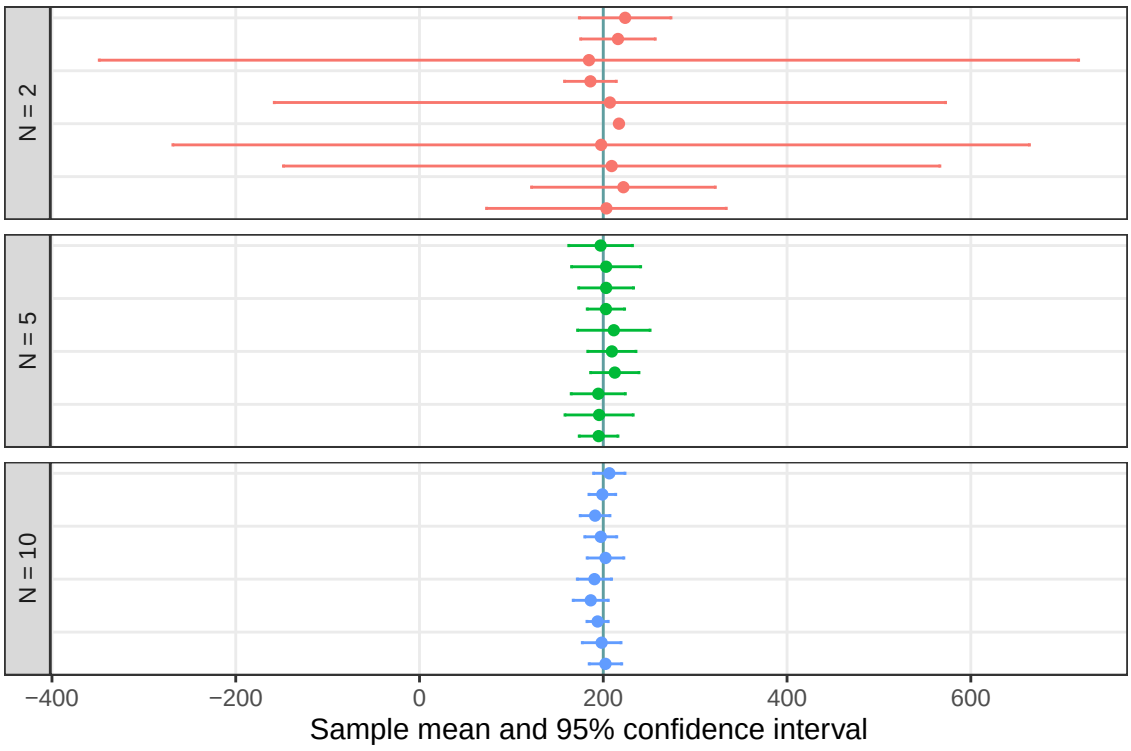


Figure 4.10: Confidence interval size vs. sample size.

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?

! The average width decreases as sample size increases. The variability in width decreases as well.

Q. 4.25

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

! Always 95% no matter what the sample size is.

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 **?@sec-P5**.

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 **?@sec-P5**).

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

Appendix

5.1 Probability

5.1.1 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.

5.1.2 Multiplication and addition

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

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

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

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

and

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

and when A and B are mutually exclusive, then:

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

5.1.3 Bayes' theorem

$P(A|B) = \frac{P(B|A)*P(A)}{P(B)}$, where $P(B) = P(A) * P(B|A) + P(A') * P(B|A')$.

5.2 Univariate statistics

5.2.1 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

	Population	Sample
Standard deviation	σ	s

5.2.2 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 $Q3 + 1.5 * IQR$.

5.2.3 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.

5.2.4 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: $s = \sqrt{\frac{SS}{n-1}}$

5.3 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** $= n * p * q$.

5.4 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

5.5 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.

5.6 Samples taken from a population

`sd()`, `sqrt()`, `length()`, `mean()`, `sqrt()`

5.6.1 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.

5.7 The t distribution

`dt()`, `pt()`, `qt()`, `rt()`

The t-distribution describes the distribution of T statistics expected under the null hypothesis.

5.7.1 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)`.

5.7.2 Confidence intervals for sample means

`t.test()`, `qt()`, `mean()`, `sd()`, `length()`, `sqrt()`

The 95% CI for a mean is calculated as:

$\bar{y} \pm t_{\alpha/2, df=n-1} * s_{\bar{y}}$

5.8 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)`.

5.9 Regression & correlation

`lm()`, `cor()`, `confint()`, `predict()`

We do not bother calculating regression coefficients by hand.

Part III

Problem Sets

Problem Set 1

View original

Download file
Please render this file to .html and submit via Brightspace.

```
library(tidyverse); theme_set(theme_classic())
library(readxl)
set.seed(2026)
```

R Basics

Data types

Create a vector called `site_num` with the values 1 through 10.

```
site_num <- 1:10
site_num <- seq(1, 10, by=1)
site_num <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Create a matrix called `pop_abund` with the values 101 through 120, with 2 columns and 10 rows.

```
pop_abund <- matrix(101:120, ncol=2, nrow=10)
```

Create a copy of the dataframe `iris` (already available as `iris`) and name it `iris_df`. View the column names and create a new column called `Petal.Area`, calculated as `Petal.Width` times `Petal.Length`.

!

```
iris_df <- iris
```

```
names(iris_df)
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
str(iris_df)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
glimpse(iris_df)
```

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

```
iris_df$Petal.Area <- iris_df$Petal.Width * iris_df$Petal.Length
```

```
iris_df <- iris_df |>
  mutate(Petal.Area=Petal.Width*Petal.Length)
```

Data munging

The *SmoltWeights* sheet from H2SMS_practicalData.xlsx contains the weight in grams (column `weight_g`) from a sample of 34 wild salmon smolts (column `fish_id`) during their out-migration from a river to the ocean. Load the dataset and perform basic summaries (range, mean, median, number of missing values for each column). Identify (using code) which fish are missing values for `weight_g`.

```
smolt_df <- read_xlsx("data/H2SMS_practicalData.xlsx", sheet="SmoltWeights")
str(smolt_df)
```

```
tibble [34 x 2] (S3: tbl_df/tbl/data.frame)
 $ fish_id : num [1:34] 1 2 3 4 5 6 7 8 9 10 ...
 $ weight_g: num [1:34] 44.2 40.2 39.7 36 40.9 33.8 51.2 41.1 36.7 37.2 ...
```

```
summary(smolt_df)
```

```
    fish_id         weight_g
 Min.   : 1.00   Min.   :26.60
 1st Qu.: 9.25   1st Qu.:36.00
 Median :17.50   Median :39.70
 Mean   :17.50   Mean   :40.17
 3rd Qu.:25.75   3rd Qu.:46.10
 Max.   :34.00   Max.   :51.20
                 NA's   :1
```

There are many ways to identify the row with missing data.

```
which(is.na(smolt_df$weight_g))
```

```
[1] 12
```

```
smolt_df[is.na(smolt_df$weight_g),] # gives the row from the data.frame
```

```
# A tibble: 1 x 2
  fish_id weight_g
    <dbl>    <dbl>
1      12       NA
```

```
smolt_df |>
  filter(is.na(weight_g))
```

```
# A tibble: 1 x 2
  fish_id weight_g
    <dbl>    <dbl>
1      12       NA
```

Create a new column called `weight_lb` that gives the weights in pounds, where 1 kg = 2.205 lb.

```
smolt_df$weight_lb <- smolt_df$weight_g / 1000 * 2.205
```

What proportion of fish weigh less than 0.2 lb? How many fish is this?

```
mean(smolt_df$weight_lb < 0.2, na.rm=T)
```

```
[1] 1
```

```
sum(smolt_df$weight_lb < 0.2, na.rm=T)
```

```
[1] 33
```

Create a new data.frame called `large_df` that contains only the fish as large or larger than the median weight

and excludes rows with missing weights.

```
# A tibble: 17 x 3
   fish_id weight_g weight_lb
     <dbl>    <dbl>     <dbl>
 1       1     44.2    0.0975
 2       2     40.2    0.0886
 3       3     39.7    0.0875
 4       5     40.9    0.0902
 5       7     51.2    0.113
 6       8     41.1    0.0906
 7      17     46.7    0.103
 8      20     46.1    0.102
 9      21     50.9    0.112
10      22     45.7    0.101
11      23     46.8    0.103
12      24     50.5    0.111
13      25     46.7    0.103
14      29     41.8    0.0922
15      32     41.5    0.0915
16      33     47.3    0.104
17      34     48.9    0.108
```

```
smolt_median <- median(smolt_df$weight_g, na.rm=T)
large_df <- smolt_df[smolt_df$weight_g >= smolt_median & !is.na(smolt_df$weight_g), ]
```

```
large_df <- smolt_df |>
  drop_na(weight_g) |>
  filter(weight_g >= median(weight_g))
```

Create a new data.frame called `extreme_df` that contains only fish more than 1.5 standard deviations greater than the mean *and* excludes rows with missing weights.

```
# A tibble: 3 x 3
  fish_id weight_g weight_lb
    <dbl>    <dbl>     <dbl>
1       7     51.2     0.113
2      21     50.9     0.112
3      24     50.5     0.111
```

```
smolt_mean <- mean(smolt_df$weight_g, na.rm=T)
smolt_sd <- sd(smolt_df$weight_g, na.rm=T)
rows_to_keep <- !is.na(smolt_df$weight_g) &
                        (smolt_df$weight_g > smolt_mean + 1.5*smolt_sd)
extreme_df <- smolt_df[rows_to_keep, ]
```

```
extreme_df <- smolt_df |>
  drop_na(weight_g) |>
  filter(weight_g > mean(weight_g) + 1.5*sd(weight_g))
```

Plotting

Histograms

Create a histogram of the smolt weights, labelling the axes and omitting a plot title. Add a figure legend using `#| fig-cap: "..."` in the quarto document code block. Add `#| label: fig-smoltHist`, and then in *text*

below, reference the figure by typing `@fig-smoltHist`.

When rendered, `@fig-smoltHist` in the text is automatically replaced by "Figure 1", with figure numbers updated automatically. Adding a legend with `#| fig-cap: "..."` in your code block will automatically give that figure a number, making your life easier.
The label must start with `fig-` for this to work.
Note that you *cannot* re-use the same `label`.

Here is some text referencing Figure 5.1 by using the label I've used in the code chunk (`@fig-smoltHist`).

````
```{r}
#| code-fold: true
#| code-summary: "Code chunk"
#| label: fig-smoltHist
#| fig-cap: "Histogram of smolt weights."
#| fig-width: 4
#| fig-height: 4

hist(smolt_df$weight_g,
     main=NULL,
     xlab="Smolt weight (g)",
     ylab="Number of fish")
```
````

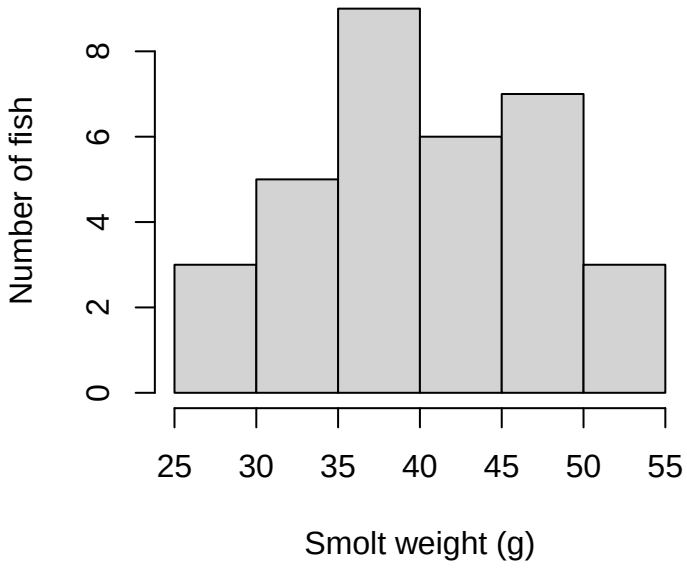


Figure 5.1: Histogram of smolt weights.

Line plots

The dataset `beaver1` is available in R, giving the body temperature through time of a female beaver (*Castor canadensis*), with observations every 10 minutes.

```
str(beaver1)
```

```
'data.frame':   114 obs. of  4 variables:
 $ day  : num  346 346 346 346 346 346 346 346 346 346 ...
 $ time : num  840 850 900 910 920 930 940 950 1000 1010 ...
 $ temp : num  36.3 36.3 36.4 36.4 36.5 ...
 $ activ: num  0 0 0 0 0 0 0 0 0 0 ...
```

Debug the following code so that it produces a line plot of body temperature by observation id. Note that `time` is given in the form `0330` for 3:30am (`?beaver1`). The `obs_id` column thus gives the ordered observations from first to last.

As an added challenge, add vertical dashed lines indicating when the beaver was active.

> Set `#| eval: true` to run the code when you click Render. If you've successfully debugged, you should then see your plot.

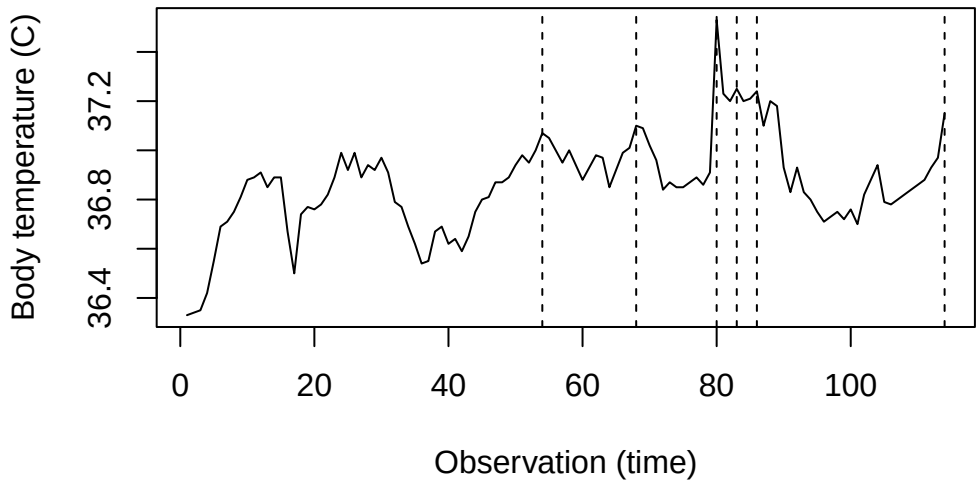
```
beaver_df <- beaver1
beaver_df$obs_id <- 1:nrow(beaver_df)
plot(obs_id ~ Temp,
     data=beaver_df,
     type="p",
     xLab="Observation (time)",
     ylab="Body temperature (C)")
```

! Original:

```
beaver_df <- beaver1
beaver_df$obs_id <- 1:nrow(beaver_df)
plot(obs_id ~ Temp, # Order is flipped (y ~ x) and `temp` should be lower case
     data=beaver_df,
     type="p", # Should be "l" (line) or "b" (line with points)
     xLab="Observation (time)", # typo: lower case l
     ylab="Body temperature (C)")
```

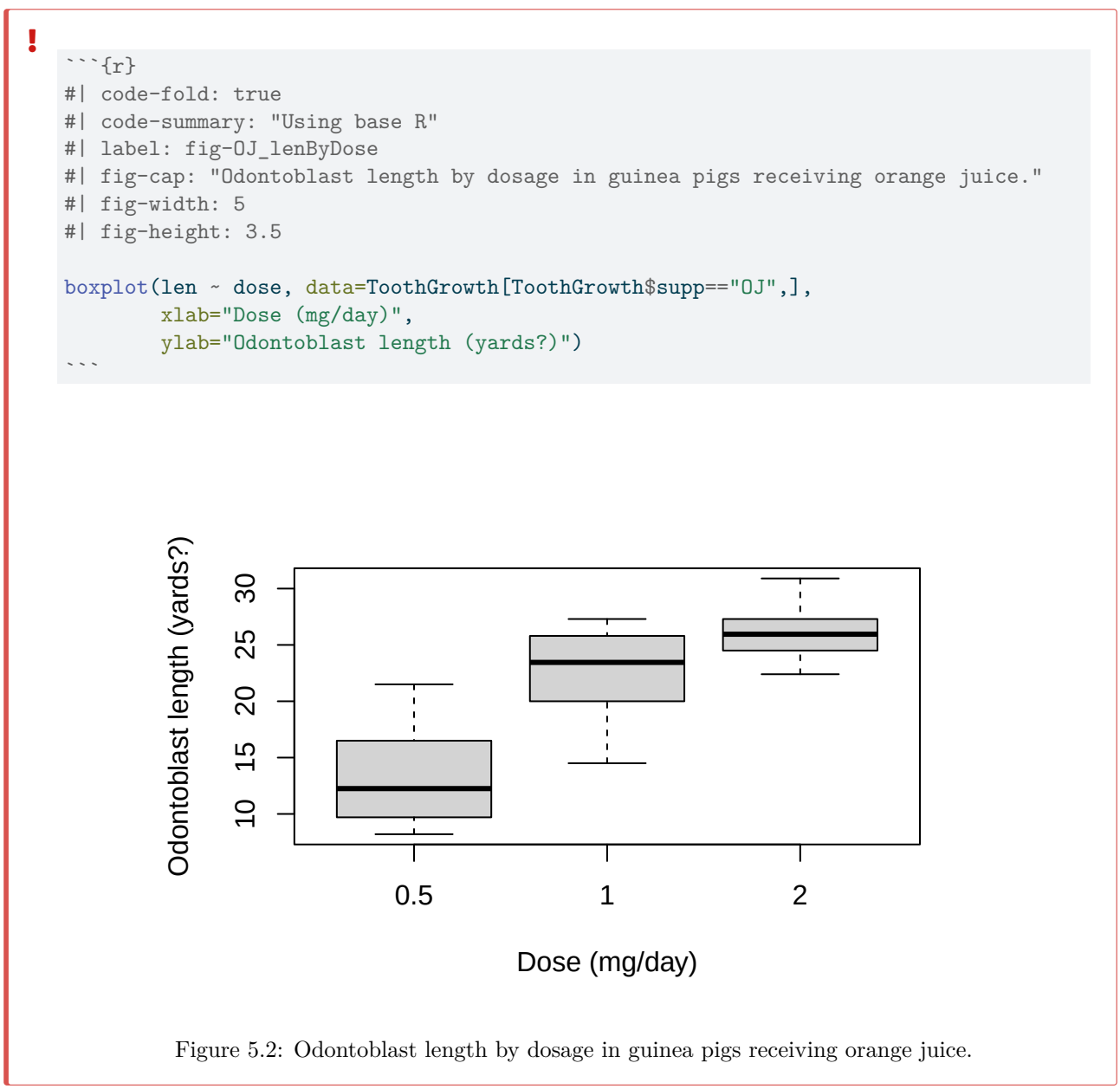
Corrected:

```
beaver_df <- beaver1
beaver_df$obs_id <- 1:nrow(beaver_df)
plot(temp ~ obs_id, data=beaver_df, type="l",
     xlab="Observation (time)",
     ylab="Body temperature (C)")
# ?abline
abline(v=beaver_df$obs_id[beaver_df$activ==1], lty=2)
```



Boxplots

The dataset `ToothGrowth` is available in R, giving the length (`len`) of tooth-growth cells (odontoblasts) in 60 guinea pigs under different vitamin C regimes (`supp`: OJ=orange juice, VC=ascorbic acid; `dose`: 0.5, 1, 2 mg/day). Create a boxplot of the length by dose for only the guinea pigs receiving orange juice. Write a figure caption and add a label as above.



```r
#| code-fold: true
#| code-summary: "Using base R"
#| label: fig-OJ_lenByDose
#| fig-cap: "Odontoblast length by dosage in guinea pigs receiving orange juice."
#| fig-width: 5
#| fig-height: 3.5

boxplot(len ~ dose, data=ToothGrowth[ToothGrowth$supp=="OJ",],
        xlab="Dose (mg/day)",
        ylab="Odontoblast length (yards?)")
```


Figure 5.2: Odontoblast length by dosage in guinea pigs receiving orange juice.

Use `ggplot()` to make the same boxplot, but with `supp` as the fill colour.

Remember to check the column types first!

!

```
str(ToothGrowth)
```

```
'data.frame':   60 obs. of  3 variables:
 $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
 $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

Since `dose` is numeric, there are two options to produce different boxes for each combination of `dose` and `supp`.

```
ggplot(ToothGrowth, aes(factor(dose), len, fill=supp)) +
  geom_boxplot() +
  scale_fill_manual("Supplement", values=c("orange", "steelblue")) +
  labs(x="Dose (mg/day)",
       y="Odontoblast length (nm)")
```

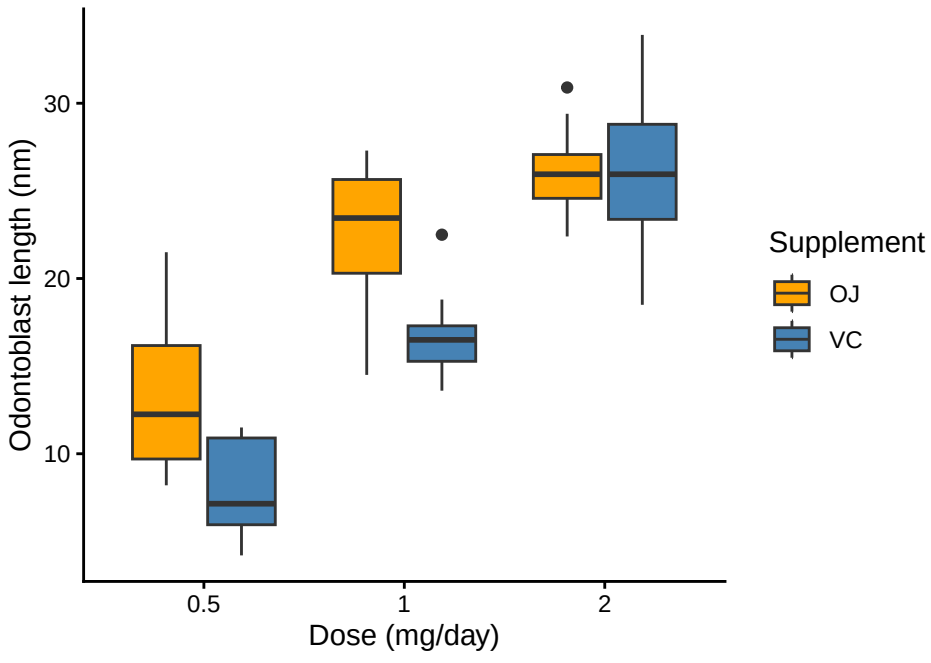


Figure 5.3: Odontoblast length by dosage and supplement type in guinea pigs.

```
```{r}
#| code-fold: true
```
```

Scatter plots

Using the same `ToothGrowth` dataset, produce a scatter plot of odontoblast length as a function of dose, where `supp` is the point colour.

! Since `supp` is a factor, the *levels* can be coded into colours automatically in `plot()`.

```
plot(len ~ dose, data=ToothGrowth, col=supp,
     xlab="Dose (mg/day)",
     ylab="Odontoblast growth (AU)")
legend("topleft",
       legend=levels(ToothGrowth$supp),
       col=1:2,
       pch=1,
       bty="n")
```

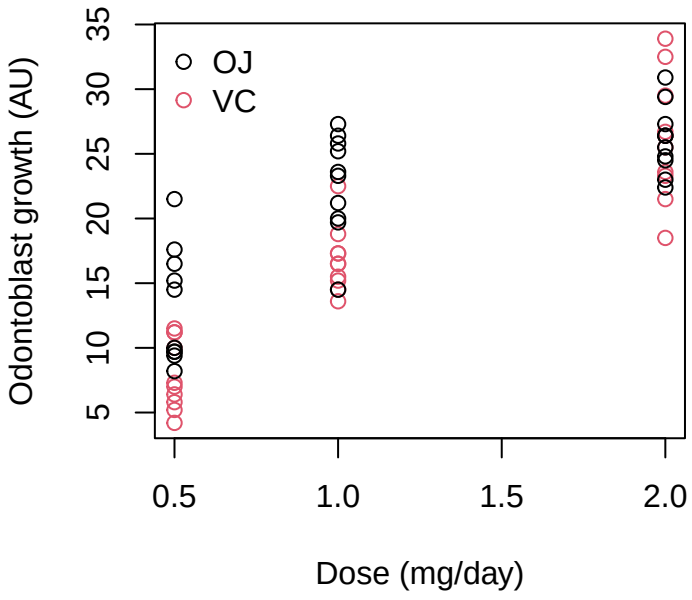


Figure 5.5: Odontoblast length by dosage and supplement type in guinea pigs.

In my opinion, it is easier to use ggplot as your figures get more complex.

```
ggplot(ToothGrowth, aes(dose, len, colour=supp)) +
  geom_point(shape=1, size=2, stroke=0.8) +
  scale_colour_manual("Supplement", values=c("orange", "steelblue")) +
  labs(x="Dose (mg/day)",
       y="Odontoblast length (nm)") +
  theme_classic()
```



Problem Set 2

View original

Download file
Please render this file to .html and submit via Brightspace.

```
library(tidyverse); theme_set(theme_classic())
library(readxl)
set.seed(2026)
```

Binomial distribution

Scenario 1

Alex is interested in how likely their tulip bulbs are to flower in their garden. They buy three identical packages of five bulbs from Tesco and plant them in November. In April, they see that 12 have successfully produced flowers.

What is $k = n$?

```
k <- 15
k
```

```
[1] 15
```

What is their best estimate for the probability that a given tulip bulb will flower ($\hat{p}$)?

```
p_hat <- 12/k
p_hat
```

```
[1] 0.8
```

What is the standard error on this sample proportion?

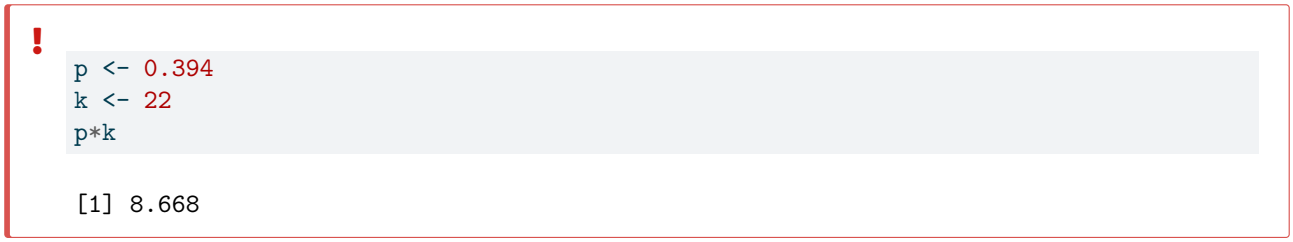
```
sqrt( (p_hat*(1-p_hat))/(k-1) )
```

```
[1] 0.1069045
```

Scenario 2

Mary is interested in the effect of winter storms on the detachment rates of the kelp *Laminaria hyperborea*. She is studying a small population near the beach at Wee Ganavan with $n = 22$ individuals forming the canopy. She reads that in similar locations, storms cause an average of 39.4% of canopy plants to dislodge over the winter.

Using these values, what is the mean expected number of dislodged plants in her population?



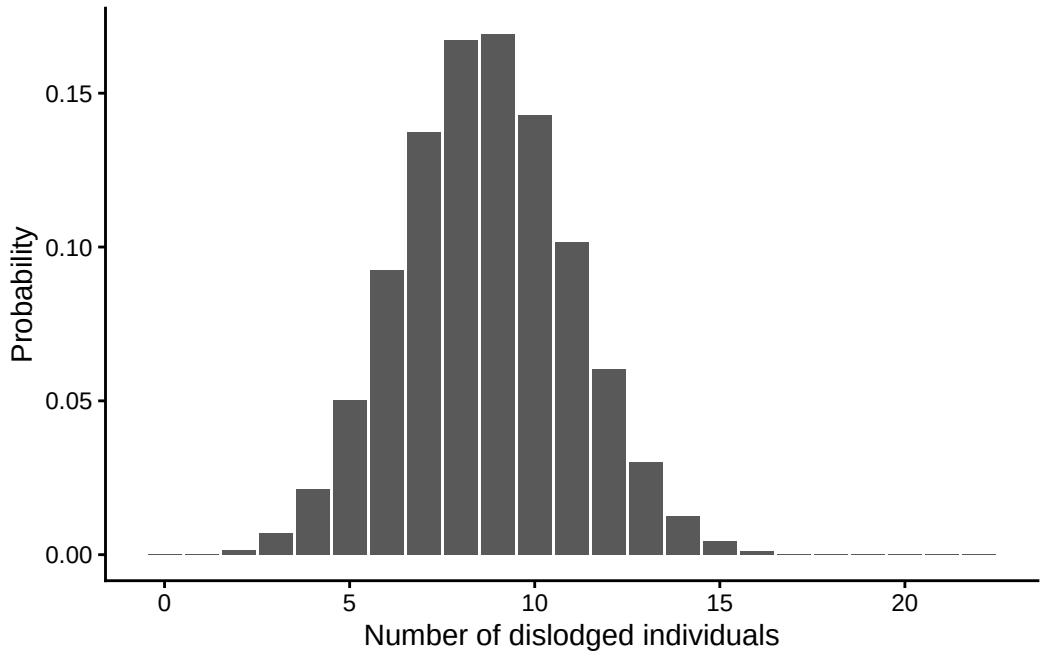
```
p <- 0.394
k <- 22
p*k
```

```
[1] 8.668
```

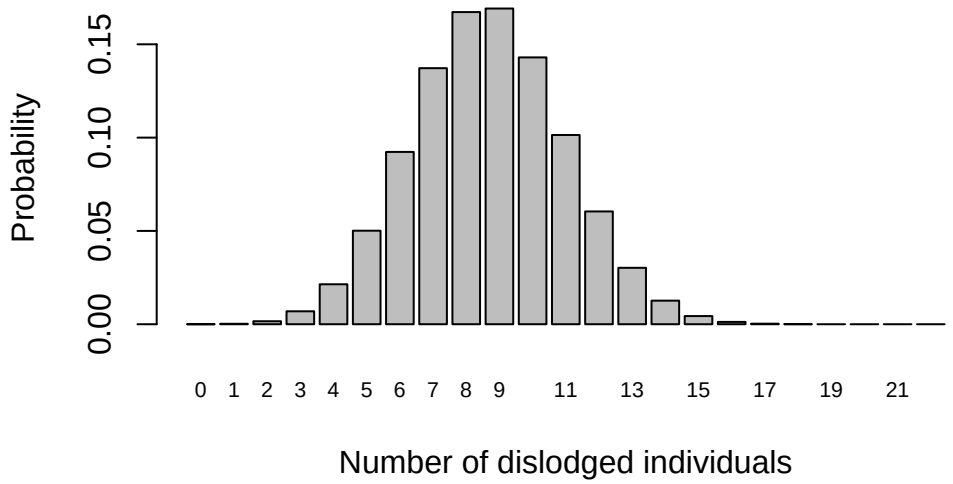
Make a barplot showing the probability of each possible number dislodged individuals.

!

```
# Here I created a dataframe (tibble) 'on the fly' using a pipe |> instead
# of storing it as an object. I'm only using it for the plot, so don't really
# care to keep it around in my R environment
tibble(nDislodged=0:k,
       prob=dbinom(nDislodged, k, p)) |>
  ggplot(aes(nDislodged, prob)) +
  geom_bar(stat="identity") +
  labs(x="Number of dislodged individuals",
       y="Probability")
```




```
barplot(dbinom(0:k, k, p),
        names.arg=0:k,
        xlab="Number of dislodged individuals",
        ylab="Probability",
        cex.names=0.7) # Changes the size of the 'names' (default = 1)
```



What is the probability that 9 individuals are dislodged?

```
dbinom(9, k, p)
```

```
[1] 0.1691739
```

What is the probability that 5 or fewer dislodge (≤ 5)?

```
sum(dbinom(0:5, k, p)) # pbinom(5, k, p)
```

```
[1] 0.08032148
```

How many out of every 10 years would you expect more than half of this population to dislodge (i.e., > 11), assuming dislodgement occurs independently among years?

```
10 * sum(dbinom(12:k, k, p))  # 1 - pbinom(11, k, p)
```

```
[1] 1.0928
```

Poisson distribution

Scenario 1

The *Dogwhelks* sheet from H2SMS_practicalData.xlsx contains the number of dog whelks (*Nucella lapillus*) in 40 quadrats on a beach on Skye. Load the dataset and calculate the mean and variance of the number of dog whelks per quadrat.

```
dw_df <- read_xlsx("data/H2SMS_practicalData.xlsx", sheet="Dogwhelks")
str(dw_df)
```

```
tibble [40 x 2] (S3: tbl_df/tbl/data.frame)
 $ quadrat   : num [1:40] 1 2 3 4 5 6 7 8 9 10 ...
 $ num_whelks: num [1:40] 0 1 0 0 0 2 1 2 0 0 ...
```

```
mean(dw_df$num_whelks)
var(dw_df$num_whelks)
```

```
[1] 0.6
[1] 0.5025641
```

What is the proportion of quadrats with 0, 1, and 2 dogwhelks?

```r
mean(dw_df$num_whelks == 0)
mean(dw_df$num_whelks == 1)
mean(dw_df$num_whelks == 2)
```

```
[1] 0.525
[1] 0.35
[1] 0.125
```

```r
dw_df |>
  count(num_whelks) |> # count occurrences of each value, put in new column `n`
  mutate(proportion=n/sum(n))
```

```
# A tibble: 3 x 3
  num_whelks     n proportion
       <dbl> <int>      <dbl>
1          0    21      0.525
2          1    14      0.35
3          2     5      0.125
```

```r
table(dw_df$num_whelks)/nrow(dw_df) # table() counts occurrences of each value
```

```
    0     1     2
0.525 0.350 0.125
```

Using the sample mean as λ, what is the probability of observing more than 2 dogwhelks in the next quadrat?

```r
lambda <- mean(dw_df$num_whelks)
1 - ppois(2, lambda)
```

```
[1] 0.02311529
```

If you were to repeat the sampling ($n = 40$ quadrats), in how many would you expect 1 dogwhelk?

```r
dpois(1, lambda) * 40
```

```
[1] 13.17148
```

Scenario 2

In July from 2018-2022, there were an average of 14.2 days where an orca sighting was reported in Shetland (shetland.org).

For the following questions, use the Poisson distribution to approximate the distribution of the number of days with reported sightings. Assume that the days with reported sightings occur independently and that there is no change in methodology or ecology.

What is the probability that there are 7 or fewer days with reported sightings next July?

```r
lambda <- 14.2
ppois(7, lambda)
```

```
[1] 0.02831033
```

What is the probability of having 14 days with sightings?

```
dpois(14, lambda)
```

```
[1] 0.1058393
```

What is the probability of having 14 or more days with sightings?

```
1 - ppois(13, lambda)
```

```
[1] 0.5565904
```

What is the maximum possible number of days with sightings?

```
# 31 days
```

According to the Poisson distribution, what is more likely: a July with 0 days with reported sightings or a July with more than 31 days of reported sightings?

```
dpois(0, lambda)
```

```
[1] 6.807981e-07
```

```
1 - ppois(31, lambda)
```

```
[1] 3.341259e-05
```

```
# There is a 0.00007% chance of 0 days with sightings and a
# 0.003% chance of >31 days with sightings.

# Remember to always consider the limits of your model!
```

Problem Set 3

View original

Download file
Please render this file to .html and submit via Brightspace.

```
library(tidyverse); theme_set(theme_classic())
library(readxl)
set.seed(2026)
```

Normal distribution

Scenario 1: Using `*norm()`

For a particular population of brown seaweed, the winter carbon:nitrogen ratio is distributed approximately normally among individuals as $y \sim Norm(\mu = 27.6, \sigma = 2.5)$.

What proportion of the population would you expect to have a C:N ratio greater than 30?

```
mu <- 27.6
sigma <- 2.5

1 - pnorm(30, mu, sigma)
```

```
[1] 0.1685276
```

What is the probability of sampling an individual with a ratio between 23 and 24?

```
pnorm(24, mu, sigma) - pnorm(23, mu, sigma)
```

```
[1] 0.04204958
```

What ratio would delineate the lowest 30% of the population?

```
qnorm(0.3, mu, sigma)
```

```
[1] 26.289
```

Generate a random sample of 12 individuals from this population. Calculate the sample mean and sample standard deviation.

```r
obs <- rnorm(12, mu, sigma)
```

```r
# Mean:
mean(obs)

# sd:
sd(obs)
```

```
[1] 26.15386
[1] 1.957169
```

What is the difference between your sample mean and the true population mean called? Why does it occur and what are two aspects of your sampling design that affect it?

```r
# This is sampling error: the difference between the true population parameter
# and the statistic calculated from a sample. It is inevitable that a statistic
# from a sample (e.g., a sample mean) will be different from the parameter for
# the entire population (e.g., the population mean) because a sample is only
# a subset of the whole population.
```

```r
# Sampling error is strongly affected by:
# 1) Sample size: A larger sample will, on average, have smaller sampling error
#     since differences among individuals 'average out'
```

```r
# Sampling error is strongly affected by:
# 2) Randomization: A non-random sample introduces bias, making some individuals
#     more likely to be included. This bias will tend to make sampling error
#     biased as well.
```

```r
# Sampling error is strongly affected by:
# 3) Population variability: A more variable population (larger sigma) means the
#     observations in a sample are more variable, and so sampling error will
#     tend to be larger. Standardizing conditions as much as possible can reduce
#     this source of noise.
```

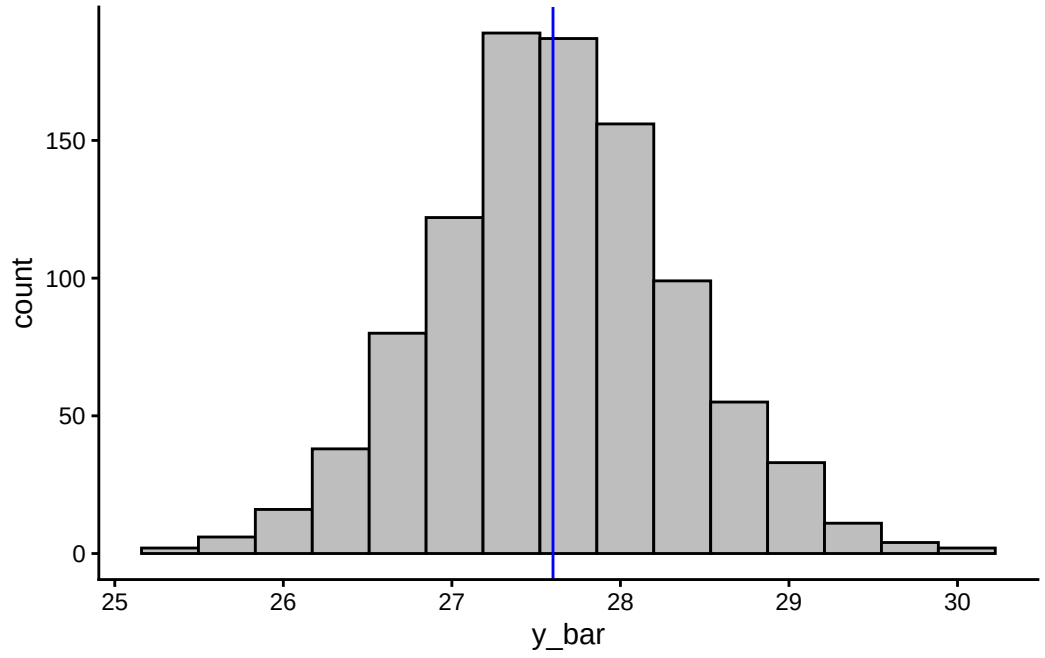
Repeat your random sampling 1000 times, calculate a sample mean for each of the 1000 samples, and make a histogram of them. Calculate the mean of these sample means.

```r
# y_bar <- rep(0, 1000)
# for(i in 1:1000) {
#   y_bar[i] <- ...
# }
```

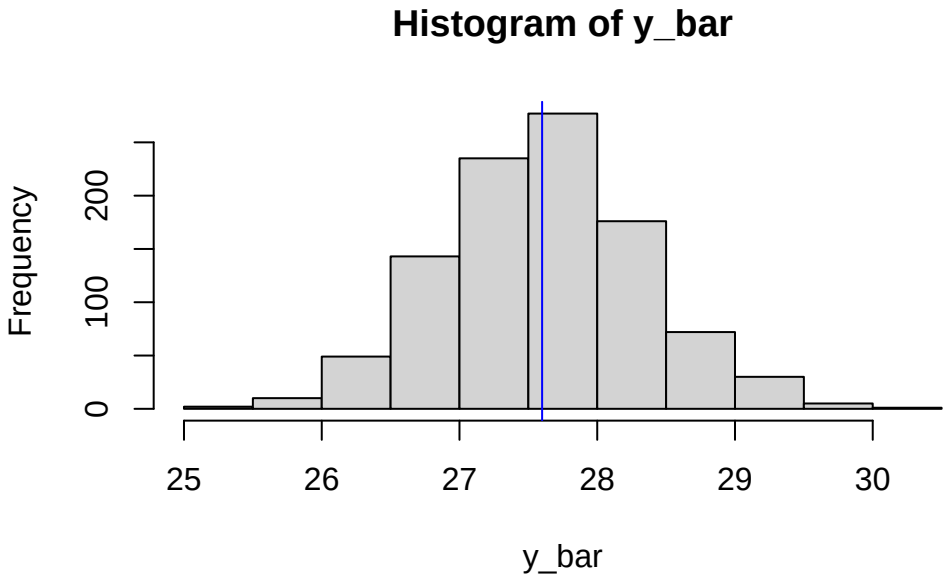
!

```
y_bar <- rep(0, 1000)
for(i in 1:1000) {
  y_bar[i] <- mean(rnorm(12, mu, sigma))
}
```

```
tibble(y_bar=y_bar) |>
  ggplot(aes(y_bar)) +
  geom_histogram(bins=15, colour="black", fill="grey") +
  geom_vline(xintercept=mu, colour="blue")
```




```
hist(y_bar, breaks=15); abline(v=mu, col="blue")
```




```
signif(mean(y_bar), 3)
```

```
[1] 27.6
```

How does this mean (and the histogram) compare to the population mean?

!
```
# The mean of sample means is very close to mu, and the histogram of sample means
# is centred on mu.
#
# Note also that the distribution is approximately normal, illustrating the
# Central Limit Theorem (since the population is ~Norm).
#
# This relies on a *random* sample. If the sampling is biased (non-random), this
# pattern can not be relied upon!
```

Scenario 2: Transformations

The `swiss` dataset in R gives standardised fertility and socioeconomic indicators for each of the 47 French-speaking provinces of Switzerland in ca. 1888:

```
str(swiss)
```

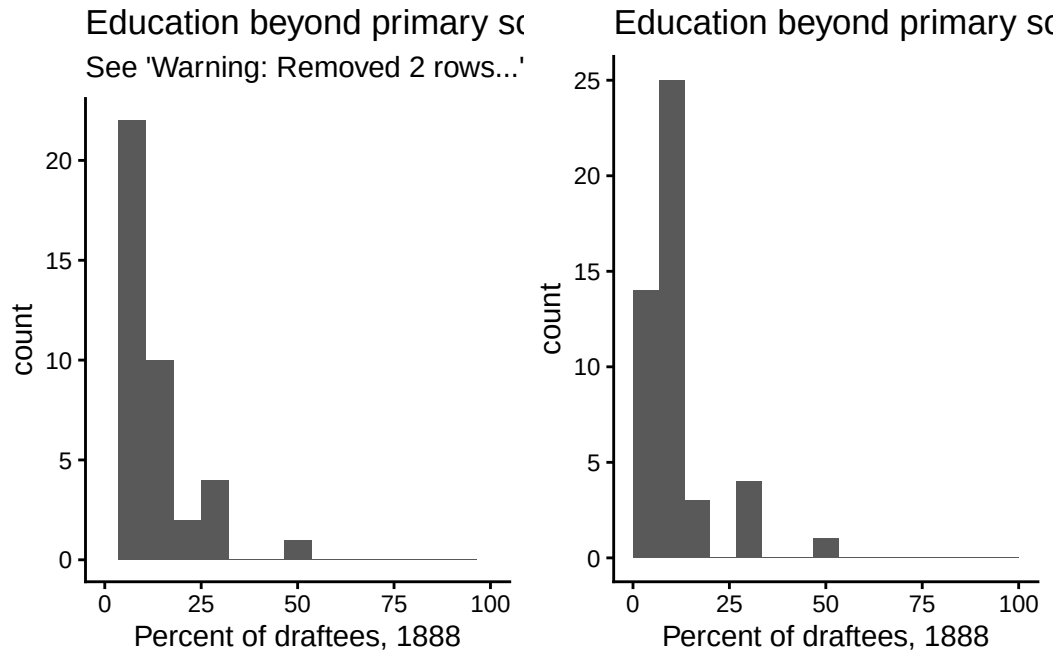
```
'data.frame':   47 obs. of  6 variables:
 $ Fertility       : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.4 82.9 ...
 $ Agriculture     : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3 45.2 ...
 $ Examination     : int  15 6 5 12 17 9 16 14 12 16 ...
 $ Education       : int  12 9 5 7 15 7 7 8 7 13 ...
 $ Catholic        : num  9.96 84.84 93.4 33.77 5.16 ...
 $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21 24.4 ...
```

The values are given as percentages ranging from 0-100. Make a histogram for the `Education` column, setting the x-axis limits to the theoretical range of the variable.

!

```
cowplot::plot_grid(
  # default geom_histogram() settings: Notice that the 0's get dropped when
  # setting xlim(0, 100)! This is because by default the bars were centred
  # on 0, and so setting xlim(0,...) means it can't draw the left half of the
  # bar, so it drops it. At least there is a warning...
  ggplot(swiss, aes(Education)) +
    geom_histogram(bins=15) +
    xlim(0, 100) +
    labs(title="Education beyond primary school",
         subtitle="See 'Warning: Removed 2 rows...'",
         x="Percent of draftees, 1888"),
  # to avoid this, we can set 'boundary=0' to make 0 the boundary for the bar
  # rather than the centre. In this case, 0 is a theoretical limit, but ggplot
  # doesn't know that.
  ggplot(swiss, aes(Education)) +
    xlim(0, 100) +
    geom_histogram(bins=15, boundary=0) +
    labs(title="Education beyond primary school",
         x="Percent of draftees, 1888")
)
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```




```
hist(swiss$Education,
     xlim=c(0, 100),
     breaks=14,
     main="Education beyond primary school",
```

What transformation would be most appropriate for these kinds of data?

!

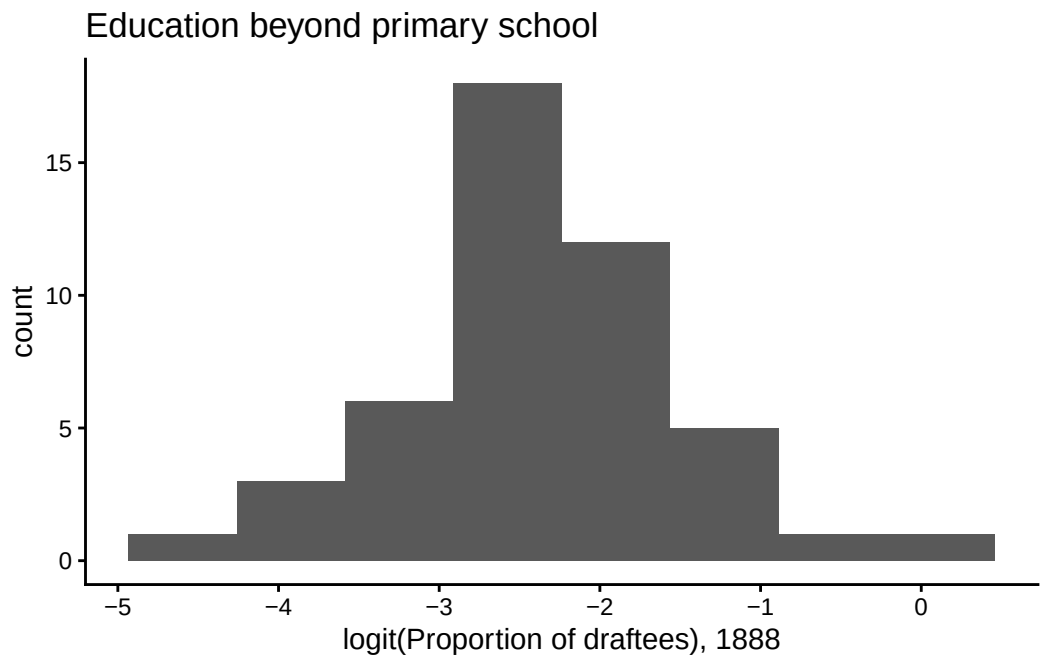
```
# Logistic: Data are percentages (convert to proportions by dividing by 100)
# and tend to be smaller values near the lower boundary (0%) with a long tail
# of larger values (right skew).

# A log transformation may also be suitable.
```

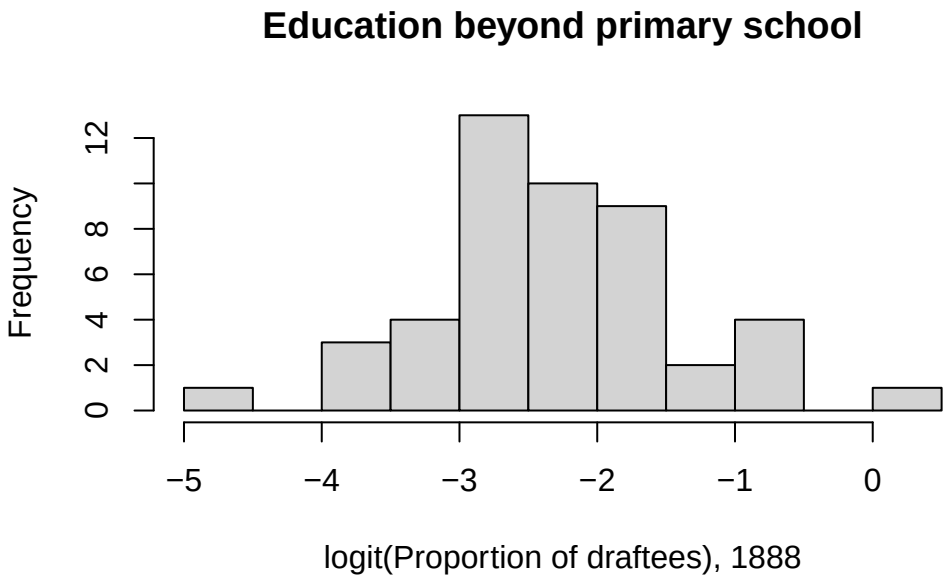
Apply this transformation and make a histogram of the transformed values.

!

```
ggplot(swiss, aes(boot::logit(Education/100))) +
  geom_histogram(bins=8) +
  labs(title="Education beyond primary school",
       x="logit(Proportion of draftees), 1888")
```




```
hist(boot::logit(swiss$Education/100),
     breaks=8,
     main="Education beyond primary school",
     xlab="logit(Proportion of draftees), 1888")
```



What do you notice about the shape of the original and transformed distributions?

```
# The transformed distribution looks much more like a normal distribution.
```

Standard error of the mean

For the seaweed population in Scenario 1, we simulated a sample of 12 individuals from the population. In my particular sample, I found a sample mean of 26.154. Recall that the true population was distributed as $y \sim Norm(\mu = 27.6, \sigma = 2.5)$.

Using the true population standard deviation $\sigma = 2.5$, calculate the standard error of the mean.

```
sigma <- 2.5
n <- 12

sigma / sqrt(n)
```

```
[1] 0.7216878
```

In your own words, describe what the standard error of the mean is.

```
# The standard error of the mean is the standard deviation of the distribution
# of possible sample means from a given population. It describes the breadth
# of the distribution of sample means we would expect if we were to collect
# many, many different samples, each with size n, and calculate a mean from
# each sample.
```

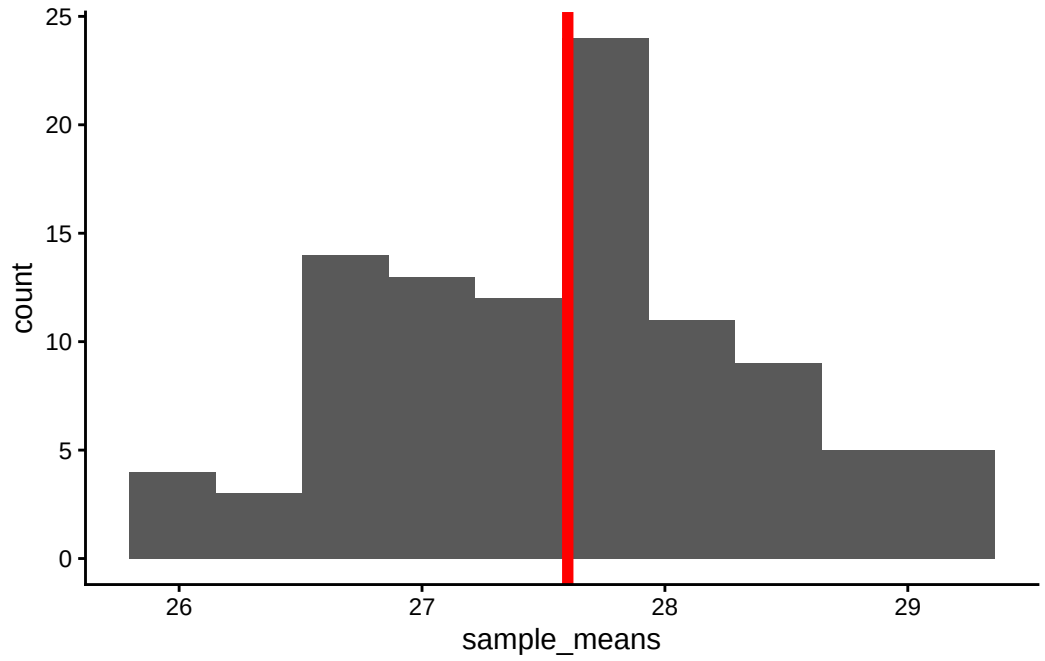
The distribution of sample means can be simulated in R. Write code to 1) simulate 100 different samples from the seaweed population with $n = 12$, and 2) calculate a sample mean from each of these 100 samples.

```
n_samples <- 100
n <- 12
sample_means <- rep(0, n_samples)
for(i in 1:n_samples) {
  sample_means[i] <- rnorm(n, mu, sigma) |> mean()
}
```

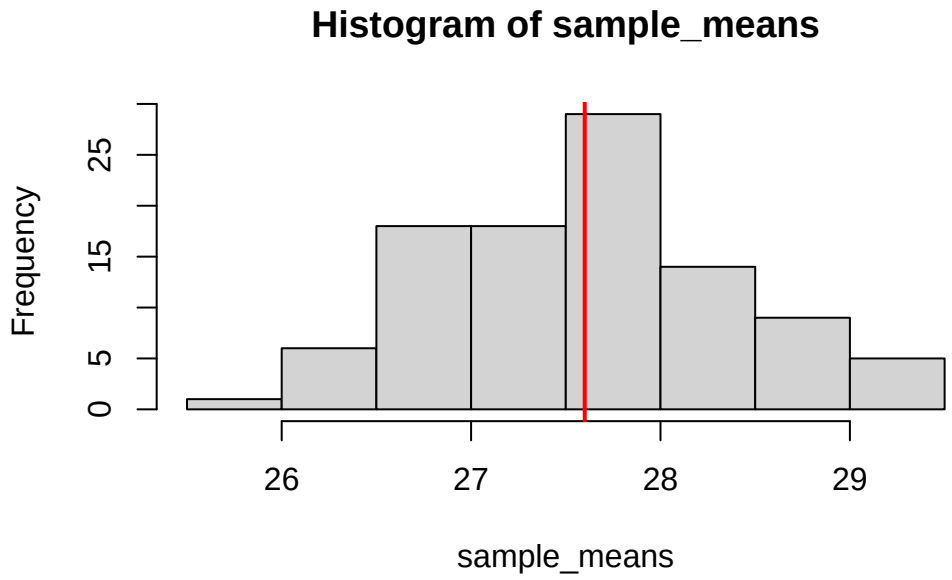
Make a plot to show the distribution of these sample means, and indicate the true population mean μ.

!

```
tibble(sample_means=sample_means) |>
  ggplot(aes(sample_means)) +
  geom_histogram(bins=10) +
  geom_vline(xintercept=mu, colour="red", linewidth=2)
```




```
hist(sample_means, breaks=10)
abline(v=mu, col="red", lwd=2)
```



Calculate the standard deviation of the 100 sample means.

```
sd(sample_means)
```

```
[1] 0.7922183
```

How does it compare to the standard error of the mean you calculated above? Try increasing the number of samples to 3,000 and recalculate.

```
# The standard deviation of the sample means should be close. Increasing the
# number of samples should bring this value even closer to the standard error
# of the mean calculated analytically. Note that this is because we are
# averaging over sampling error!
```

Lastly, repeat the process, this time with $n = 120$ for each sample. Calculate the standard error of the mean, simulate 100 sample means, and compare the standard deviation of these with the calculated standard error.

!

```
n <- 120

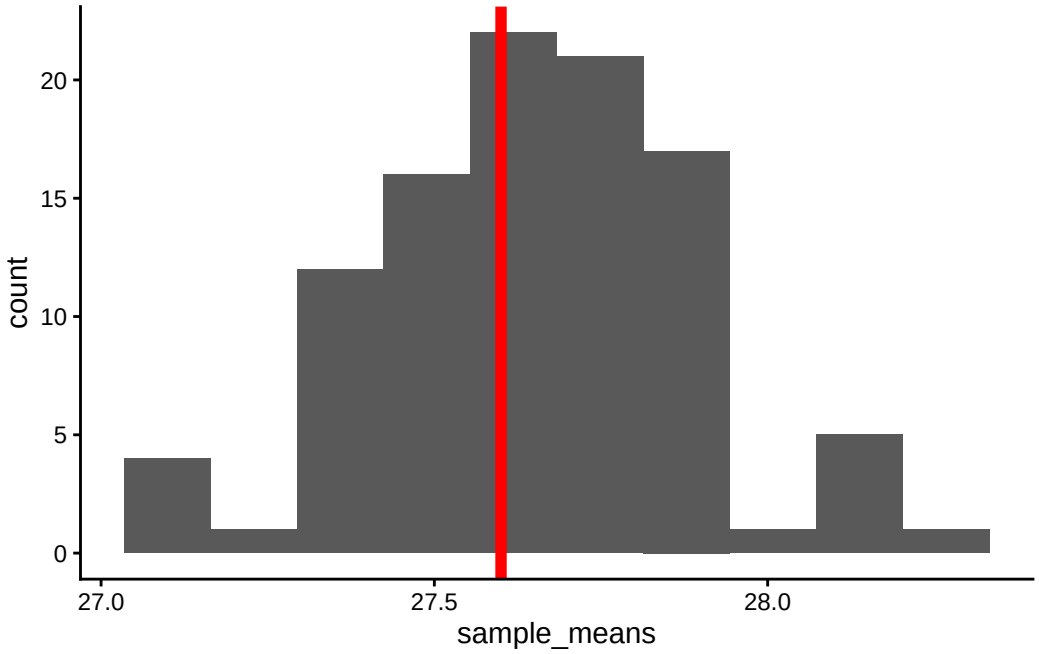
sigma / sqrt(n)
```

```
[1] 0.2282177
```

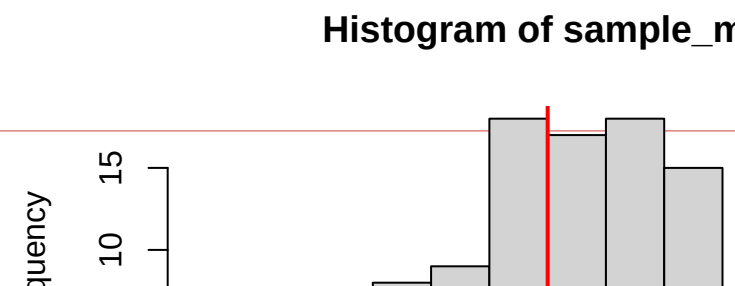
```
n_samples <- 100
sample_means <- rep(0, n_samples)
for(i in 1:n_samples) {
  sample_means[i] <- rnorm(n, mu, sigma) |> mean()
}
sd(sample_means)
```

```
[1] 0.2279312
```

```
tibble(sample_means=sample_means) |>
  ggplot(aes(sample_means)) +
  geom_histogram(bins=10) +
  geom_vline(xintercept=mu, colour="red", linewidth=2)
```




```
hist(sample_means, breaks=10)
abline(v=mu, col="red", lwd=2)
```



Why does the standard error decrease? What does this tell us about our level of confidence in the sample statistics we calculate ($\bar{y}$, s) and our inferences about the population parameters we are interested in (μ, σ)?

!

```
# Sample means will vary less (among samples) as the size of each sample
# increases. The variability among individuals averages out, and our samples
# become more representative of the population. According to the equation, the
# standard error decreases with the square root of the sample size.

# So, with a larger sample size, we can be more confident that our sample mean
# is near to the population mean.

# Note that this assumes proper sampling methodology!
# (See Experimental Design lectures)
```

Extra: CA1

View original

Download file

Overview

Coverage: L01-L09, P1-P3

This is a document to give you additional practice working in R. It will include working with data (manipulating, summarising, plotting) and analyzing the distributions we've covered.

Please do not treat this as a comprehensive study guide for CA1!

Note that there are often many possible paths to the correct solution.

```
library(tidyverse); theme_set(theme_classic())
library(readxl)
set.seed(2026)
```

Iris

The `iris` dataset is available in R upon start-up.

What are the dimensions of the object (rows, columns) and what data type is it?

```
str(iris)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Create a scatter plot of sepal area on the x-axis and petal area on the y-axis.

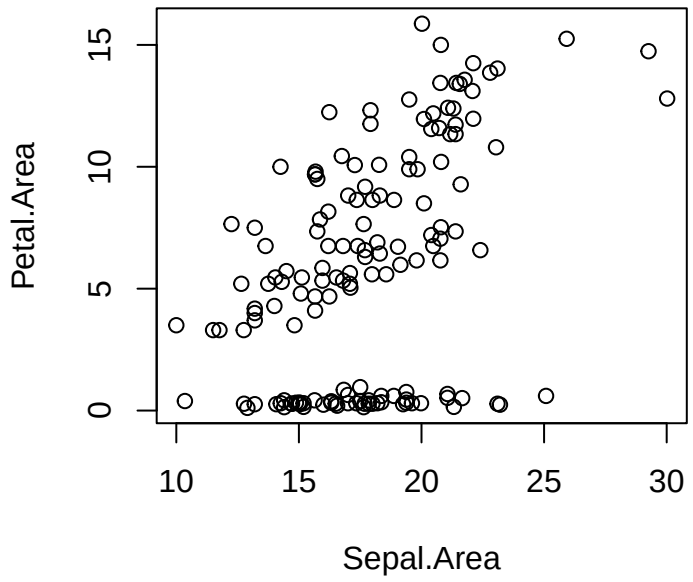
! First create new columns `Sepal.Area` and `Petal.Area`.

```
# method 1
iris_df <- iris |>
  mutate(Sepal.Area=Sepal.Length*Sepal.Width,
         Petal.Area=Petal.Length*Petal.Width)

# method 2
iris_df <- iris
iris$Sepal.Area <- iris$Sepal.Length * iris$Sepal.Width
iris$Petal.Area <- iris$Petal.Length * iris$Petal.Width
```

Then make the scatter plot:

```
plot(Petal.Area ~ Sepal.Area, data=iris)
```



For each species, what is the mean and standard deviation of sepal area?

```
# method 1
iris |>
  group_by(Species) |>
  summarise(mean_SepalArea=mean(Sepal.Area),
            sd_SepalArea=sd(Sepal.Area))
```

```
# A tibble: 3 x 3
  Species    mean_SepalArea sd_SepalArea
  <fct>               <dbl>        <dbl>
1 setosa               17.3         2.93
2 versicolor           16.5         2.87
3 virginica            19.7         3.46
```

```
# method2
summary(iris$Species) # unique values of Species
mean(iris$Sepal.Area[iris$Species=="setosa"])
mean(iris$Sepal.Area[iris$Species=="versicolor"])
mean(iris$Sepal.Area[iris$Species=="virginica"])
sd(iris$Sepal.Area[iris$Species=="setosa"])
sd(iris$Sepal.Area[iris$Species=="versicolor"])
sd(iris$Sepal.Area[iris$Species=="virginica"])
```

Create a plot with two panels, each with a boxplot with species on the x-axis. In the first panel, plot sepal area on the y-axis. In the second panel, plot petal area on the y-axis.

```
par(mfrow=c(1,2))
boxplot(Sepal.Area ~ Species, data=iris, cex.axis=0.7)
boxplot(Petal.Area ~ Species, data=iris, cex.axis=0.7)
```



Extract the even numbered rows from `iris`, and the columns `Species`, `Sepal.Area`, and `Petal.Area` in that order.

!
```
iris[seq(2, nrow(iris), by=2), c("Species", "Sepal.Area", "Petal.Area")]
```

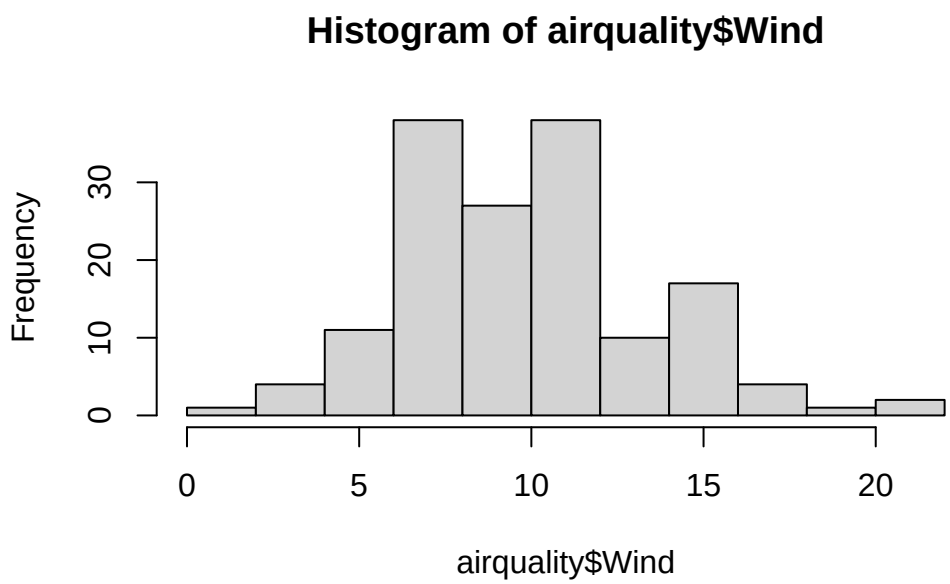
```
       Species Sepal.Area Petal.Area
2       setosa      14.70       0.28
4       setosa      14.26       0.30
6       setosa      21.06       0.68
8       setosa      17.00       0.30
10      setosa      15.19       0.15
12      setosa      16.32       0.32
14      setosa      12.90       0.11
16      setosa      25.08       0.60
18      setosa      17.85       0.42
20      setosa      19.38       0.45
22      setosa      18.87       0.60
24      setosa      16.83       0.85
26      setosa      15.00       0.32
28      setosa      18.20       0.30
30      setosa      15.04       0.32
32      setosa      18.36       0.60
34      setosa      23.10       0.28
36      setosa      16.00       0.24
38      setosa      17.64       0.14
40      setosa      17.34       0.30
42      setosa      10.35       0.39
44      setosa      17.50       0.96
46      setosa      14.40       0.42
48      setosa      14.72       0.28
50      setosa      16.50       0.28
52  versicolor      20.48       6.75
54  versicolor      12.65       5.20
56  versicolor      15.96       5.85
58  versicolor      11.76       3.30
60  versicolor      14.04       5.46
62  versicolor      17.70       6.30
64  versicolor      17.69       6.58
66  versicolor      20.77       6.16
68  versicolor      15.66       4.10
70  versicolor      14.00       4.29
72  versicolor      17.08       5.20
74  versicolor      17.08       5.64
76  versicolor      19.80       6.16
78  versicolor      20.10       8.50
80  versicolor      14.82       3.50
82  versicolor      13.20       3.70
84  versicolor      16.20       8.16
```

Air quality

The `airquality` dataset is available in R upon start-up, and gives daily values of mean ozone, solar radiation, mean wind speed, and max temperature at LaGuardia Airport in New York City (`?airquality`).

Make a histogram of wind speed.

```
hist(airquality$Wind)
```



Assume wind speed is normally distributed in this dataset with $\mu = 9.99mph$ and $\sigma = 3.51mph$.

What proportion of days would you expect to have a wind speed less than 5 mph?

```
pnorm(5, 9.99, 3.51)
```

```
[1] 0.07756359
```

You have a kite that can only be flown safely in winds between 14 mph and 20 mph. What proportion of days would be suitable?

```
pnorm(20, 9.99, 3.51) - pnorm(14, 9.99, 3.51)
```

```
[1] 0.1244602
```

How many days would be suitable in a 31-day month?

```
31 * (pnorm(20, 9.99, 3.51) - pnorm(14, 9.99, 3.51))
```

```
[1] 3.858267
```

What proportion of days in the dataset were actually suitable?

```
mean(airquality$Wind >= 14 & airquality$Wind <= 20)
```

```
[1] 0.1437908
```

What are the z-scores for 14 and 20 mph?

```
(14 - 9.99)/3.51
```

```
[1] 1.14245
```

```
(20 - 9.99)/3.51
```

```
[1] 2.851852
```

What are the quantiles for the middle 95%?

```
qnorm(c(0.025, 0.975), 9.99, 3.51)
```

```
[1]  3.110526 16.869474
```

What is the quantile for the largest 5%?

```
qnorm(0.95, 9.99, 3.51)
```

```
[1] 15.76344
```

Given the population parameters above, what is the z score of the sample mean in the dataset?

```
(mean(airquality$Wind) - 9.99)/(3.51/sqrt(nrow(airquality)))
```

```
[1] -0.1144731
```

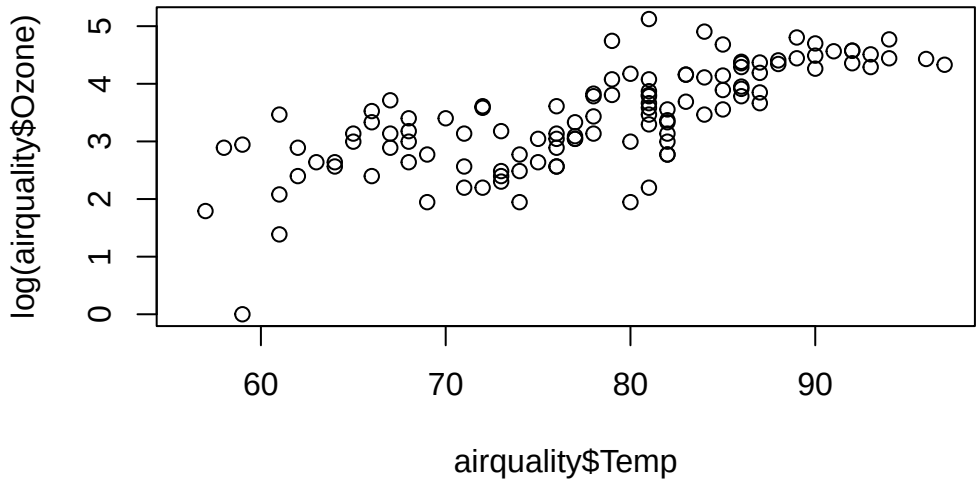
Given the population parameters above, what is the probability of observing a sample mean smaller than the one in this dataset?

```
pnorm(mean(airquality$Wind), 9.99, 3.51/sqrt(nrow(airquality)))
```

```
[1] 0.4544314
```

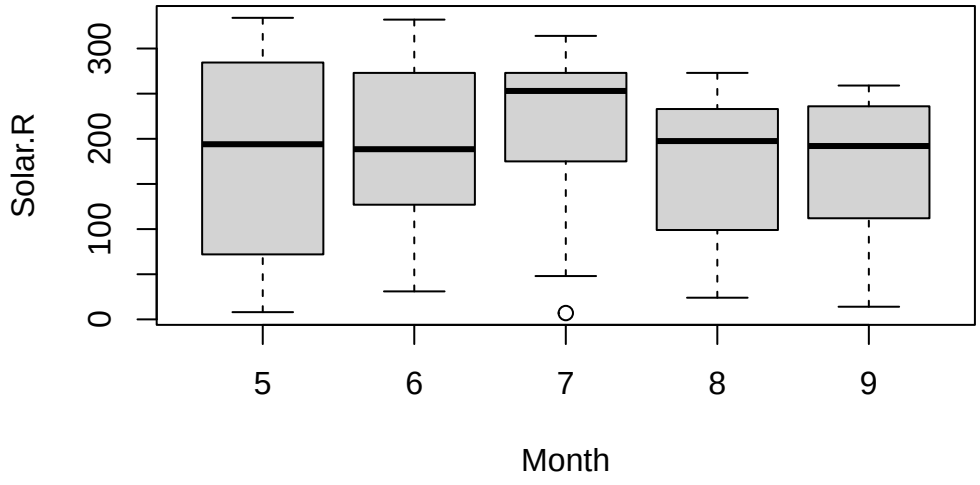
Create a scatter plot with temperature on the x-axis and log-transformed ozone on the y-axis.

```
plot(airquality$Temp, log(airquality$Ozone))
```



Create a boxplot with the solar radiation on the y-axis and the month on the x-axis.

```
boxplot(Solar.R ~ Month, data=airquality)
```



Coral predation

You are studying predation of the Flamingo Tongue Snail on soft corals on South Water Caye in Belize. In a distinct patch of reef, you randomly select a sample of 14 corals and identify that 9 of these have feeding scars.

What distribution would you expect this variable (i.e., the number of corals with feeding scars of those sampled)

to follow?

! Binomial

What is your sample proportion and its standard error?

!
```
p_hat <- 9/14
p_hat
```

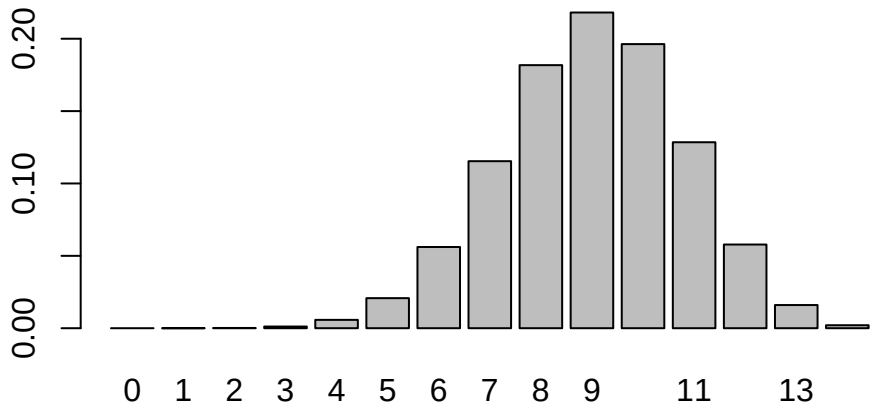
```
[1] 0.6428571
```

```
sqrt((p_hat)*(1-p_hat)/(14-1))
```

```
[1] 0.1328944
```

Create a bar graph of the binomial distribution corresponding to these parameters, with the x-axis spanning the full range of possible outcomes.

!
```
barplot(dbinom(0:14, 14, prob=9/14), names.arg=0:14)
```



Based on this distribution, if you were to collect another random sample of the same size, what is the probability that all 14 would have feeding scars?

!
```
dbinom(14, 14, 9/14)
```

```
[1] 0.002058745
```

What is the probability you would find *fewer* with feeding scars than in the first sample?

!
```
pbinom(8, 14, 9/14)
```

```
[1] 0.3811746
```

What is the probability of finding *more* with feeding scars than in the first sample?

```
1-pbinom(9, 14, 9/14)
```

```
[1] 0.400701
```

What is the probability of a sample with 4, 5, or 6 corals with feeding scars?

```
sum(dbinom(4:6, 14, 9/14))
```

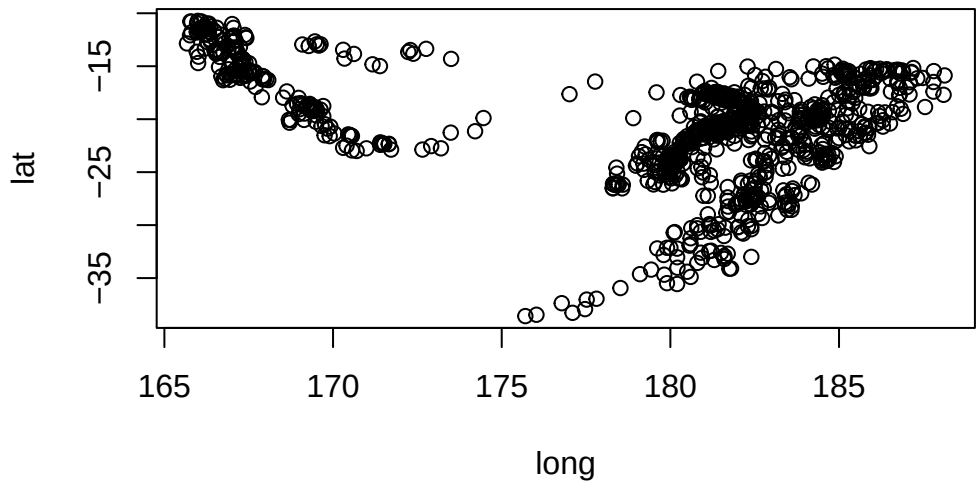
```
[1] 0.08265225
```

Earthquakes

The dataset `quakes` is available in R upon start-up, and gives magnitudes and locations for 1000 seismic events near Fiji since 1964.

Plot the locations of the earthquakes (x-axis: longitude, y-axis: latitude).

```
plot(lat ~ long, data=quakes)
```



Create a column called `mag_cat` with the magnitude truncated to integer (i.e., 4.0-4.99 become 4). Hint: `?trunc`

```
quakes$mag_cat <- trunc(quakes$mag)
```

For each magnitude category (`mag_cat`), count the number of occurrences in the dataset.

```
summary(quakes) # ranges 3-6
```

```
      lat              long           depth            mag       
 Min.   :-38.59   Min.   :165.7   Min.   : 40.0   Min.   :4.00  
 1st Qu.:-23.47   1st Qu.:179.6   1st Qu.: 99.0   1st Qu.:4.30  
 Median :-20.30   Median :181.4   Median :247.0   Median :4.60  
 Mean   :-20.64   Mean   :179.5   Mean   :311.4   Mean   :4.62  
 3rd Qu.:-17.64   3rd Qu.:183.2   3rd Qu.:543.0   3rd Qu.:4.90  
 Max.   :-10.72   Max.   :188.1   Max.   :680.0   Max.   :6.40  
    stations         mag_cat     
 Min.   : 10.00   Min.   :4.000  
 1st Qu.: 18.00   1st Qu.:4.000  
 Median : 27.00   Median :4.000  
 Mean   : 33.42   Mean   :4.203  
 3rd Qu.: 42.00   3rd Qu.:4.000  
 Max.   :132.00   Max.   :6.000  
```

```
# Handy trick:
summary(as.factor(quakes$mag_cat))
```

```
  4   5   6 
802 193   5 
```

Let's assume the dataset spans 60 years (1964-2023). What is the mean number of magnitude category 5 earthquakes per year?

```
lambda_cat5 <- sum(quakes$mag_cat==5)/60
lambda_cat5
```

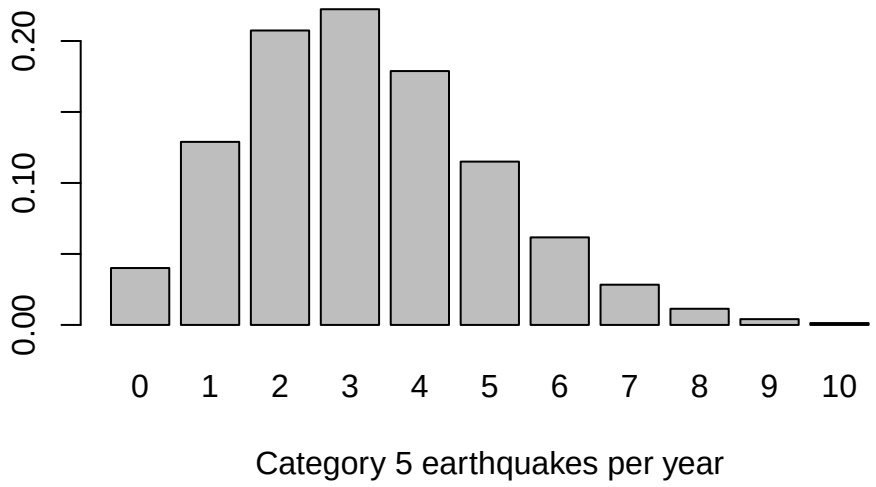
```
[1] 3.216667
```

What distribution is most suitable for the number of category 5 earthquakes per year?

Poisson

Using the above value as λ, create a bar plot of the probability distribution of the number of category 5 earthquakes per year, ranging from 0 to 10 earthquakes.

```
barplot(dpois(0:10, lambda_cat5), names.arg=0:10, xlab="Category 5 earthquakes per year")
```



What is the probability that next year has no category 5 earthquakes?

```
dpois(0, lambda_cat5)
```

```
[1] 0.04008846
```

Of the next 10 years, how many would you expect to have:

No category 5 earthquakes?

```
dpois(0, lambda_cat5)*10
```

```
[1] 0.4008846
```

Fewer than 2 earthquakes?

```
ppois(1, lambda_cat5)*10
```

```
[1] 1.690397
```

At least 5?

```
(1-ppois(4, lambda_cat5))*10
```

```
[1] 2.223618
```

Problem Set 4

View original

Download file
Please render this file to .html and submit via Brightspace.

```
library(tidyverse); theme_set(theme_classic())
library(readxl)
set.seed(2026)
```

Sample mean vs. a reference value

Scenario 1

A marine protected area (MPA) was established five years ago. Previous surveys from nearby fished reefs found a mean fish biomass of 12.2 kg 100m^{-2}.

Researchers survey 15 transects inside the MPA and want to test whether fish biomass is higher than the regional average of fished reefs.

The data are available from the "MPA" sheet in "H2SMS_practicalData.xlsx". Load and check the data.

```
MPA_df <- read_xlsx("data/H2SMS_practicalData.xlsx", sheet="MPA")
str(MPA_df)
```

```
tibble [15 x 2] (S3: tbl_df/tbl/data.frame)
 $ transect        : num [1:15] 1 2 3 4 5 6 7 8 9 10 ...
 $ biomass_kg_100m2: num [1:15] 12.8 13.2 12.7 12.6 10.4 ...
```

```
summary(MPA_df)
```

```
    transect    biomass_kg_100m2
 Min.   : 1.0   Min.   :10.30
 1st Qu.: 4.5   1st Qu.:12.14
 Median : 8.0   Median :12.79
 Mean   : 8.0   Mean   :12.75
 3rd Qu.:11.5   3rd Qu.:13.46
 Max.   :15.0   Max.   :15.02
```

What is the response variable?

! Fish biomass (kg $100m^{-2}$)

What is the null hypothesis?

! $\mu_{MPA} \leq 12.2kg \cdot 100m^{-2}$
The mean fish biomass within the MPA is **less than or equal to** the mean fish biomass outside the MPA.
Note that the hypothesis is about the *population mean*, not the *sample mean*.

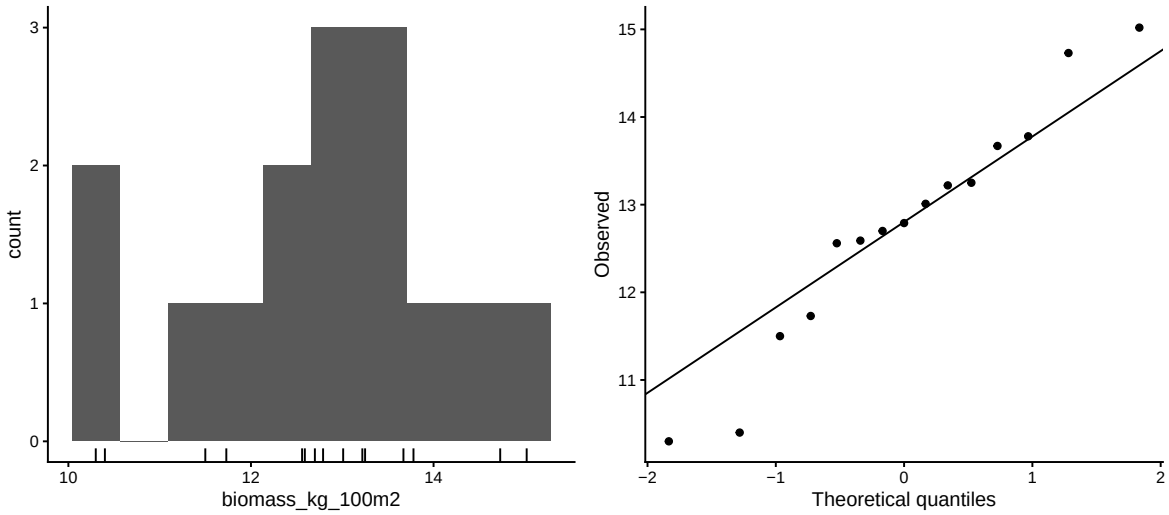
What is the alternative hypothesis?

! $\mu_{MPA} > 12.2kg \cdot 100m^{-2}$
The mean fish biomass within the MPA is **greater than** the mean fish biomass outside the MPA.
Note that the hypothesis is about the *population mean*, not the *sample mean*.

Evaluate the null hypothesis using $\alpha = 0.05$. Perform an appropriate analysis (including assumption checks), and report the relevant results in written text.

!

```
pA <- ggplot(MPA_df, aes(biomass_kg_100m2)) +
  geom_histogram(bins=10) +
  geom_rug()
pB <- ggplot(MPA_df, aes(sample=biomass_kg_100m2)) +
  geom_qq() + geom_qq_line() +
  labs(x="Theoretical quantiles",
       y="Observed")
cowplot::plot_grid(pA, pB, nrow=1, align="h", axis="tb")
```



It's a small sample, but the observations seem approximately normal.

```
t.test(MPA_df$biomass_kg_100m2, mu=12.2, alternative="greater")
```

```
    One Sample t-test

data:  MPA_df$biomass_kg_100m2
t = 1.5718, df = 14, p-value = 0.06916
alternative hypothesis: true mean is greater than 12.2
95 percent confidence interval:
 12.13369      Inf
sample estimates:
mean of x
    12.75
```

In a one-tailed one sample t-test, the mean fish biomass within the MPA (12.8 kg 100m^2; 95% confidence limit: 12.1 kg 100m^{-2}) was not significantly greater than the regional mean outside the MPA of 12.2 kg 100m^{-2} (T_{14}=1.57, p=0.069).

In your own words, explain what the p-value represents.

!

The p-value gives the probability of these observations if the null hypothesis were true. That is, it is the likelihood of these observed values if the MPA mean were equal to 12.2 kg 100^{-2}.

Scenario 2

In New Zealand, Brewin et al. (2000) studied fecundity and spawning of the sea urchin *Evechinus chloroticus*, finding a mean annual gamete output of 0.068 g g^{-1} yr^{-1} between 1990-1994 at Dieffenbach Point. On my holiday, I decided to repeat that aspect of their experiment (securing all necessary permits and approvals).

The data are in the "UrchinGametes" sheet in "H2SMS_practicalData.xlsx", already summarised to grams of gamete per grams of urchin per year. Load the data and perform appropriate checks.

!

```
urchin_df <- read_xlsx("data/H2SMS_practicalData.xlsx", sheet="UrchinGametes")
str(urchin_df)
```

```
tibble [20 x 2] (S3: tbl_df/tbl/data.frame)
 $ urchin_id: num [1:20] 1 2 3 4 5 6 7 8 9 10 ...
 $ output   : num [1:20] 0.0956 0.0476 0.0842 0.0775 0.06 ...
```

```
summary(urchin_df)
```

```
   urchin_id         output
 Min.   : 1.00   Min.   :0.00359
 1st Qu.: 5.75   1st Qu.:0.05743
 Median :10.50   Median :0.07049
 Mean   :10.50   Mean   :0.06820
 3rd Qu.:15.25   3rd Qu.:0.08477
 Max.   :20.00   Max.   :0.12041
```

What is the response variable?

! Annual gamete output (g gamete per gram of urchin per year)

What is the null hypothesis?

! $\mu_{new} = 0.068g \cdot g^{-1} \cdot yr^{-1}$
The mean gamete output of the current population is **equal to** the historic mean.
Note that the hypothesis is about the *population mean*, not the *sample mean*.

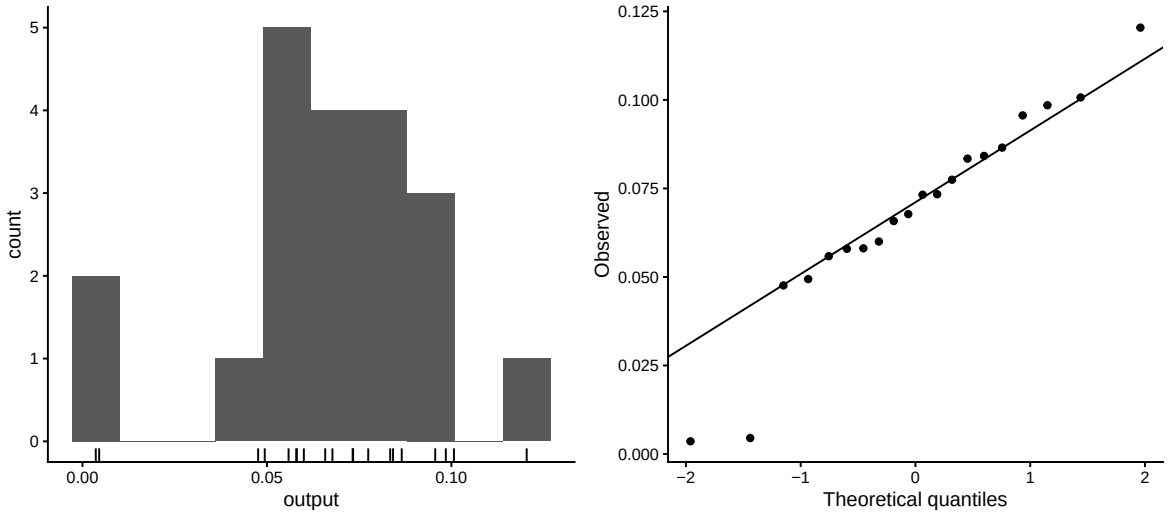
What is the alternative hypothesis?

! $\mu_{new} \neq 0.068g \cdot g^{-1} \cdot yr^{-1}$
The mean gamete output of the current population is **not equal to** the historic mean.
Note that the hypothesis is about the *population mean*, not the *sample mean*.

Evaluate the null hypothesis using $\alpha = 0.05$. Perform an appropriate analysis (including assumption checks), and report the relevant results in written text.

!

```
pA <- ggplot(urchin_df, aes(output)) +
  geom_histogram(bins=10) +
  geom_rug()
pB <- ggplot(urchin_df, aes(sample=output)) +
  geom_qq() + geom_qq_line() +
  labs(x="Theoretical quantiles",
       y="Observed")
cowplot::plot_grid(pA, pB, nrow=1, align="h", axis="tb")
```



The observations seem approximately normal, given the sample size.

```
t.test(urchin_df$output, mu=0.068)
```

```
    One Sample t-test

data:  urchin_df$output
t = 0.030502, df = 19, p-value = 0.976
alternative hypothesis: true mean is not equal to 0.068
95 percent confidence interval:
 0.05467906 0.08171494
sample estimates:
mean of x
 0.068197
```

In a two-tailed one sample t-test, the mean gamete output (0.0682 g g^{-1} yr^{-1}; 95% CI: 0.0547-0.0817 g g^{-1} yr^{-1}) was not significantly different than the historic average of 0.068 g g^{-1} yr^{-1} (T_{19}=0.03, p=0.976).

In your own words, interpret your results with regard to the null hypothesis and the p-value.

!

The observed gamete output was entirely plausible under the null hypothesis that the population mean was the same as in 1990-1994.

Confidence Intervals

The sheet “Plastics” in “H2SMS_practicalData.xlsx” contains observations of the number of plastic pellets per kilogram. Load the data.

```
plastic_df <- read_xlsx("data/H2SMS_practicalData.xlsx", sheet="Plastics")
str(plastic_df)
```

```
tibble [100,000 x 2] (S3: tbl_df/tbl/data.frame)
 $ id     : num [1:100000] 1 2 3 4 5 6 7 8 9 10 ...
 $ pellets: num [1:100000] 431 673 548 560 429 498 406 591 438 406 ...
```

```
summary(plastic_df)
```

```
       id            pellets
 Min.   :     1   Min.   : 66.0
 1st Qu.: 25001   1st Qu.:432.0
 Median : 50001   Median :500.0
 Mean   : 50001   Mean   :499.7
 3rd Qu.: 75000   3rd Qu.:567.0
 Max.   :100000   Max.   :924.0
```

Consider this the entire *population* of interest.

Write code to take a sample of 25 observations from this population and store it in a vector called `obs`.

Hint

`sample()`

```
obs <- sample(plastic_df$pellets, size=25)
```

How different is your sample mean from the true population mean?

```
mean(obs) - mean(plastic_df$pellets)
```

```
[1] -4.3144
```

The true mean is 499.7 and my sample mean was 495.4.

What is the term that describes this difference?

Sampling error: The difference between a sample statistic and the population parameter.

Calculate 95% confidence intervals for your sample mean using only your sample. Do they include the true value?

```
t.test(obs)
```

```
    One Sample t-test

data:  obs
t = 20.866, df = 24, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 446.3995 544.4005
sample estimates:
mean of x
    495.4
```

```
# y_bar +/- T[df,95%] * SE
mean(obs) + qt(c(0.025, 0.975), length(obs)-1) * sd(obs)/sqrt(length(obs))
```

```
[1] 446.3995 544.4005
```

In this case, yes.

Assuming each student starts with a different `set.seed`, what proportion of students would answer 'yes' to the previous question?

0.95 = 95%

If you instead calculated 80% CIs, a) would you be more likely to capture the true mean or less? And, b) would the width of your confidence interval be larger or smaller?

a) Less likely and b) smaller. An 80% CI will only include the true value in 80% of samples (under ideal conditions!), while a 95% CI will include the true value in 95% of samples.

Calculate the 80% CI for your sample and compare.

!

```r
ci_95 <- t.test(obs)$conf.int
ci_80 <- t.test(obs, conf.level=0.8)$conf.int

ci_95
```

```
[1] 446.3995 544.4005
attr(,"conf.level")
[1] 0.95
```

```r
ci_80
```

```
[1] 464.1123 526.6877
attr(,"conf.level")
[1] 0.8
```

```r
expand_grid(confidence=c(0.8, 0.95),
            limit=c("lo", "high")) |>
  mutate(limit=c(ci_80, ci_95)) |>
  ggplot(aes(limit, confidence, group=confidence)) +
  geom_line() +
  geom_point(shape="|") +
  geom_vline(xintercept=mean(plastic_df$pellets), linetype=3) +
  labs(x="Value", y="Confidence level")
```

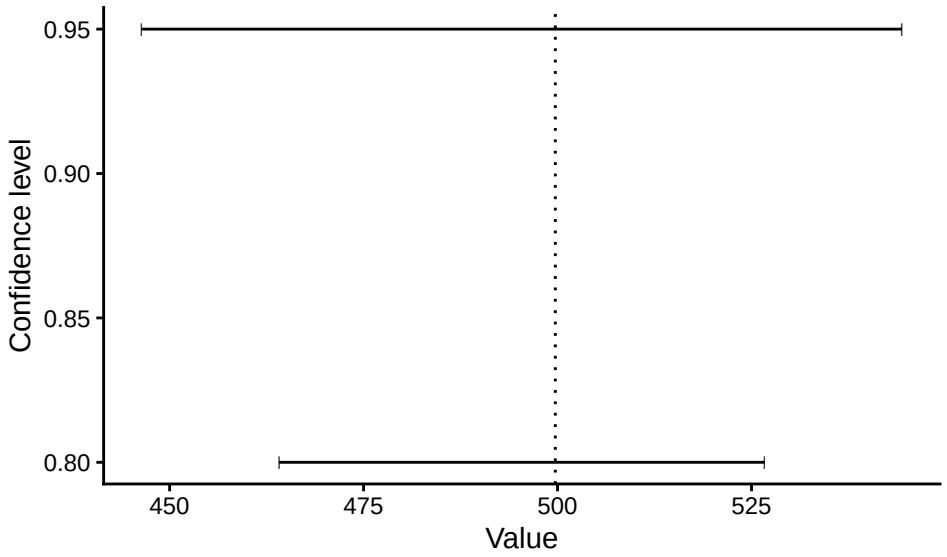


Figure 5.7: True value (dotted line) compared to the 80% and 95% confidence interval from one sample.

If you were to run your code again (without resetting `set.seed`), what is the probability that you would collect a sample with a 90% CI that includes the true value?

! 90% - each (future) sample will have a 90% chance of having 90% CIs that include the true value.

Part IV

Tutorials

tidyverse intro

Unlike in your coursework, real world datasets are often messy. Data often comes from several sources, was collected by different people, entered into a spreadsheet by even different people, and arrives on your computer in multiple arrangements. There may be missing or incorrect values, multiple labels for the same treatment, and many unnecessary columns.

Even with clean datasets, you should spend a good amount of time looking at your data in different ways. This is called *Exploratory Data Analysis* (EDA). Quick plots and summaries are a great way to identify potential errors and can help you develop an intuitive sense of complex datasets.

The *tidyverse* is "an opinionated collection of R packages designed for data science" that is commonly used in the R-based data science world. The packages share a common philosophy with a distinct coding grammar and data structures. They work collaboratively to make data science easier, cleaner, and more reproducible.

This is a brief overview – there is much, much more that we will not cover.

What makes a dataset tidy vs. messy?

Within the *tidyverse* framework, a tidy dataset is one where:

1. Each variable is a single column
2. Each observation is a single row
3. Each type of observational unit is a single table

If you have multiple related spreadsheets, each should have a column with the same name that allows them to be joined or merged appropriately.

Guidelines for data entry

When entering your data, try to keep to the following guidelines:

1. **Be consistent.** Use a single way of entering a variable (Y/N *or* Yes/No *or* 0/1).
2. **Choose good names.** Use underscores, camelCase, or SnakeCase instead of spaces. Use short, descriptive names. In R, object names can't start with numbers or special characters, so try to avoid that in your factor levels as well.
3. **Write dates as YYYY-MM-DD.** This is the global ISO 8601 standard. It sorts well and might keep Excel from mangling your data.
4. **No empty cells.** Enter NA instead.
5. **Put just one thing in a cell.** Have a column `weight_kg` filled with numbers rather than a column called `weight` filled with values like "5.5 kg".
6. **Don't use font color or highlighting as data.** Sometimes used to flag values. Instead, create a new column to store the information.
7. **Save data as plain text files.** Proprietary formats like .xlsx can be handy, but plain text (.csv, .txt, .tsv, etc) does not require special software and can be read by anyone on any machine.

Key packages in the *tidyverse*

There are many packages associated with the tidyverse. We will cover **aspects** of only a select few:

- *tibble*: a re-imagining of the dataframe
- *dplyr*: data manipulation
- *tidyr*: tidying data
- *ggplot2*: visualization
- *stringr*: working with strings
- *lubridate*: working with dates and date-times

When you install the *tidyverse* package, all of these are installed (and more).

```
# Not needed if you did this once already
install.packages("tidyverse")
```

```
# Needed each time you restart R
library(tidyverse)
```

Review: Piping

As seen in Practical 1, *piping* helps keep your code legible. The 'pipe operator' allows you to string together functions by taking the output of one and sending it as input to the next. Many functions in the *tidyverse* are designed to work nicely in pipes.

The native R pipe is `|>` and was introduced in R 4.1.0. Prior to this, piping required the `%>%` function in the *magrittr* package, which you may see in help pages or online documentation.

```
iris$Sepal.Length |>
  sort() |>
  head(n=4)
```

The convenience will become clear as we go.

Tibbles

Tibbles are dataframes, but with a few extra tweaks. See the Tibbles chapter in R for Data Science for a more comprehensive overview.

```
str(msleep)
```

```
tibble [83 x 11] (S3: tbl_df/tbl/data.frame)
 $ name        : chr [1:83] "Cheetah" "Owl monkey" "Mountain beaver" "Greater short-tailed shrew" ...
 $ genus       : chr [1:83] "Acinonyx" "Aotus" "Aplodontia" "Blarina" ...
 $ vore        : chr [1:83] "carni" "omni" "herbi" "omni" ...
 $ order       : chr [1:83] "Carnivora" "Primates" "Rodentia" "Soricomorpha" ...
 $ conservation: chr [1:83] "lc" NA "nt" "lc" ...
 $ sleep_total : num [1:83] 12.1 17 14.4 14.9 4 14.4 8.7 7 10.1 3 ...
 $ sleep_rem   : num [1:83] NA 1.8 2.4 2.3 0.7 2.2 1.4 NA 2.9 NA ...
 $ sleep_cycle : num [1:83] NA NA NA 0.133 0.667 ...
 $ awake       : num [1:83] 11.9 7 9.6 9.1 20 9.6 15.3 17 13.9 21 ...
 $ brainwt     : num [1:83] NA 0.0155 NA 0.00029 0.423 NA NA NA 0.07 0.0982 ...
 $ bodywt      : num [1:83] 50 0.48 1.35 0.019 600 ...
```

Munging data

It is best to keep your data file in its original form and do all cleaning, filtering, transforming, etc. in R for full transparency and reproducibility. Editing by hand leaves no trail of what you did; mistakes are easy to make but difficult to find.

The dplyr package provides functions for munging data. See the Data Transformation chapter in R for Data Science for a more comprehensive overview.

Key functions:

- `filter()` filters **rows** based on a `TRUE/FALSE` logical statement
- `select()` selects **columns** by name, number, or a *tidyselect* helper function
- `arrange()` reorders the tibble according to the column(s) provided
- `mutate()` creates new columns
- `group_by()` groups the tibble by the column(s) provided
- `summarise()` creates summaries, either for the whole tibble or by group

`filter()`

Use `filter()` to subset data based on logical statements.

```
msleep |>
  filter(vore=="herbi")
msleep |>
  filter(vore=="herbi" & !is.na(sleep_rem))
```

`select()`

Use `select()` to select (or exclude) particular columns. There are many *selection helpers* available, such as `starts_with()` or `ends_with()`. Use a negative sign to exclude specified columns.

```
msleep |>
  select(name, order, starts_with("sleep"))
msleep |>
  select(-conservation)
```

`arrange()`

Use `arrange()` to reorder your data by one or more columns.

```
msleep |>
  filter(vore=="herbi") |>
  arrange(bodywt)
```

`mutate()`

Use `mutate()` to create new columns. If you use `group_by()` first, calculations will be for each group.

```
msleep |>
  filter(!is.na(brainwt) & !is.na(bodywt)) |>
  mutate(brain_pct=brainwt/bodywt*100) |>
  arrange(desc(brain_pct))
msleep |>
  select(name, vore, awake) |>
  mutate(vs_grandMean=awake - mean(awake)) |>
  group_by(vore) |>
  mutate(vs_voreMean=awake - mean(awake))
```

`summarise()`

Use `summarise()` to create new columns that summarise your data. If you use `group_by()` first, the summaries will be calculated for each group.

```
msleep |>
  filter(!is.na(awake)) |>
  summarise(nSpp=n_distinct(name),
            nGenera=n_distinct(genus),
            awake_mean=mean(awake),
            awake_sd=sd(awake))
msleep |>
  filter(!is.na(awake)) |>
  group_by(vore) |>
  summarise(nSpp=n_distinct(name),
            nGenera=n_distinct(genus),
            awake_mean=mean(awake),
            awake_sd=sd(awake))
```

! With the `msleep` dataset, remove all rows that do not have a conservation status and sort the rows so that they are in alphabetical order by 1) order and 2) genus.

View solution

```
msleep |>
  filter(!is.na(conservation)) |>
  arrange(order, genus)
```

```
# A tibble: 54 x 11
   name   genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
   <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
 1 Cow    Bos   herbi Arti~ domesticated         4         0.7       0.667  20
 2 Roe d~ Capr~ herbi Arti~ lc                   3        NA        NA      21
 3 Goat   Capri herbi Arti~ lc                   5.3       0.6      NA      18.7
 4 Giraf~ Gira~ herbi Arti~ cd                   1.9       0.4      NA      22.1
 5 Sheep  Ovis  herbi Arti~ domesticated         3.8       0.6      NA      20.2
 6 Pig    Sus   omni  Arti~ domesticated         9.1       2.4       0.5    14.9
 7 Cheet~ Acin~ carni Carn~ lc                  12.1      NA        NA      11.9
 8 North~ Call~ carni Carn~ vu                   8.7       1.4       0.383  15.3
 9 Dog    Canis carni Carn~ domesticated        10.1       2.9       0.333  13.9
10 Domes~ Felis carni Carn~ domesticated        12.5       3.2       0.417  11.5
# i 44 more rows
# i 2 more variables: brainwt <dbl>, bodywt <dbl>
```

! In the `msleep` dataset, how many carnivores sleep more than 8 hours? What proportion of carnivore species is this?

View solution

```
msleep |>
  filter(vore=="carni") |>
  summarise(n_gt8=sum(sleep_total > 8),
            prop_gt8=mean(sleep_total > 8)) # or sum(...)/n()
```

```
# A tibble: 1 x 2
  n_gt8 prop_gt8
  <int>    <dbl>
1    13    0.684
```

Reshaping data

The tidyr package has many functions for tidying your datasets. We will cover just two that tend to be particularly useful.

Key functions:

- `pivot_longer()` takes a series of columns and stacks them into one
- `pivot_wider()` takes a single column and spreads it into several

`pivot_longer()`

Use `pivot_longer()` to stack columns (`cols=`), specifying new columns to store the original column values (`values_to=`) and original column names (`names_to=`).

```
msleep |>
  select(name, brainwt, bodywt) |>
  pivot_longer(cols=c("brainwt", "bodywt"),
               names_to="structure",
               values_to="weight")
```

`pivot_wider()`

Use `pivot_wider()` to spread the values from single column (`values_from=`) into several based on a categorical column (`names_from=`).

```
iris |>
  group_by(Species) |>
  mutate(Plant_id=row_number()) |>
  ungroup() |>
  select(Plant_id, Species, Petal.Width) |>
  pivot_wider(names_from="Species", values_from="Petal.Width")
```

! With the `iris` dataset, select the columns for species, petal width, and sepal width, then transform it to be longer with columns `Species`, `FlowerPart`, and `Width`.

View solution

```
iris |>
  select(Species, Petal.Width, Sepal.Width) |>
  pivot_longer(c("Petal.Width", "Sepal.Width"),
               names_to="FlowerPart",
               values_to="Width")
```

```
# A tibble: 300 x 3
   Species FlowerPart  Width
   <fct>   <chr>       <dbl>
 1 setosa  Petal.Width   0.2
 2 setosa  Sepal.Width   3.5
 3 setosa  Petal.Width   0.2
 4 setosa  Sepal.Width   3
 5 setosa  Petal.Width   0.2
 6 setosa  Sepal.Width   3.2
 7 setosa  Petal.Width   0.2
 8 setosa  Sepal.Width   3.1
 9 setosa  Petal.Width   0.2
10 setosa  Sepal.Width   3.6
# i 290 more rows
```

With the `msleep` dataset, calculate the mean awake time for each feeding type (vore) and then transform it so that there is a single row and a column for each feeding type.

View solution

```
msleep |>
  group_by(vore) |>
  summarise(mean=mean(awake)) |>
  pivot_wider(names_from="vore", values_from="mean")
```

```
# A tibble: 1 x 5
  carni herbi insecti  omni  `NA`
  <dbl> <dbl>   <dbl> <dbl> <dbl>
1  13.6  14.5    9.06  13.1  13.8
```

---

ggplot2

The ggplot2 package is fantastic for exploratory data analysis, as well as creating polished plots for final presentation. See the 'reference' tab on the package website for all plotting functions, and the Exploratory Data Analysis chapter and the Graphics for Communication chapter in R for Data Science for many examples and much more information.

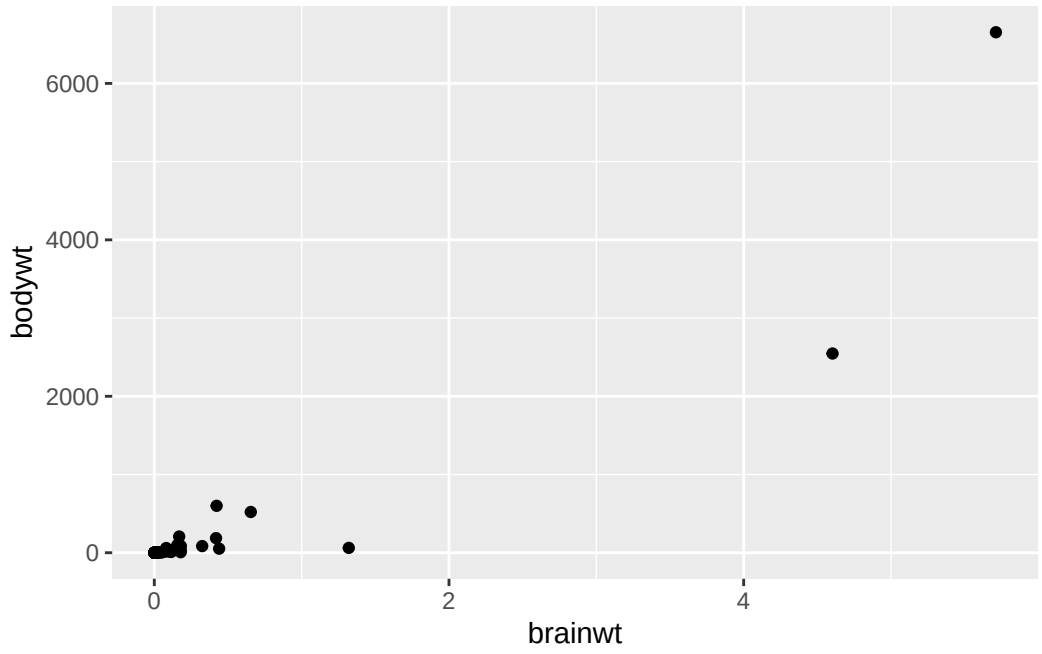
Key functions:

- `ggplot()` initializes a plot object; always used first, followed by `+`
- `geom_*()` plot elements (e.g., `geom_point()`, `geom_line()`, etc)
- `geom_smooth()` calculates and plots a trend line
- `scale_colour_*()` specifies a custom colour palette
- `scale_fill_*()` specifies a custom fill palette
- `scale_x/y_*()` specifies a custom x or y axis

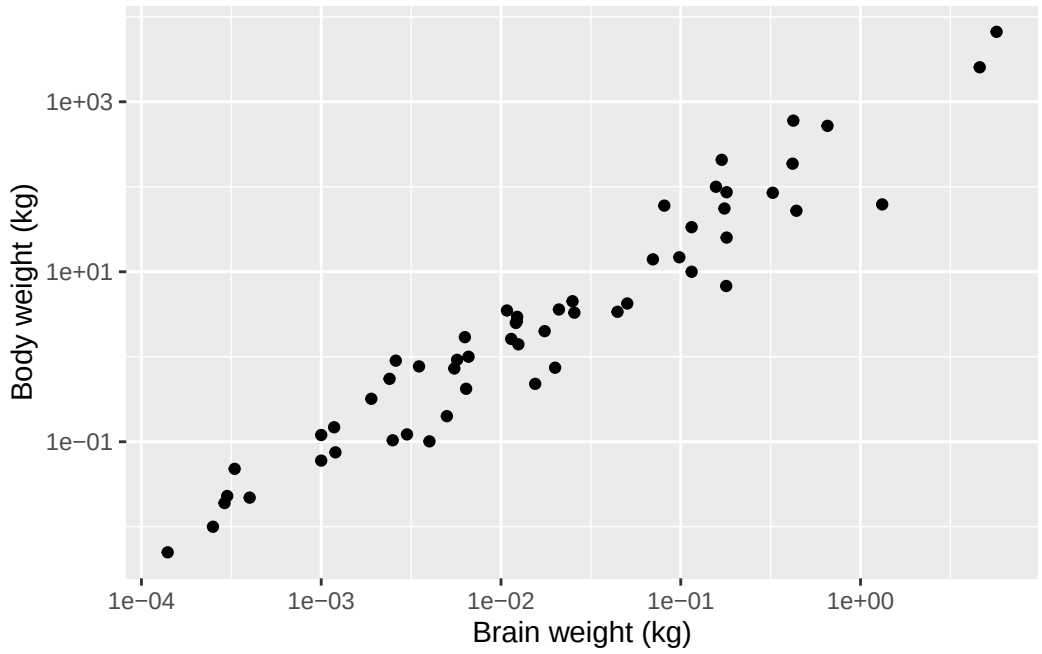
- `labs()` adds labels (x, y, title, subtitle)
- `facet_wrap()`, `facet_grid()` partition data into panels by the specified variable(s)

Scatter plots

```
# scatter plot
msleep |>
  ggplot(aes(brainwt, bodywt)) +
  geom_point()
```

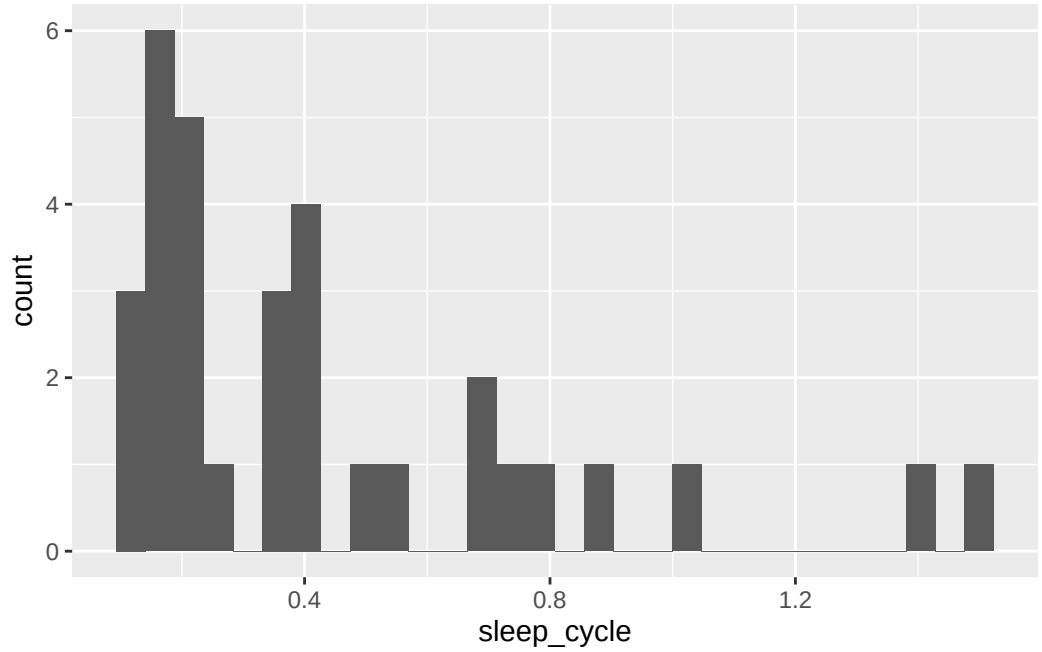



```
msleep |>
  ggplot(aes(brainwt, bodywt)) +
  geom_point() +
  scale_x_log10("Brain weight (kg)") +
  scale_y_log10("Body weight (kg)")
```

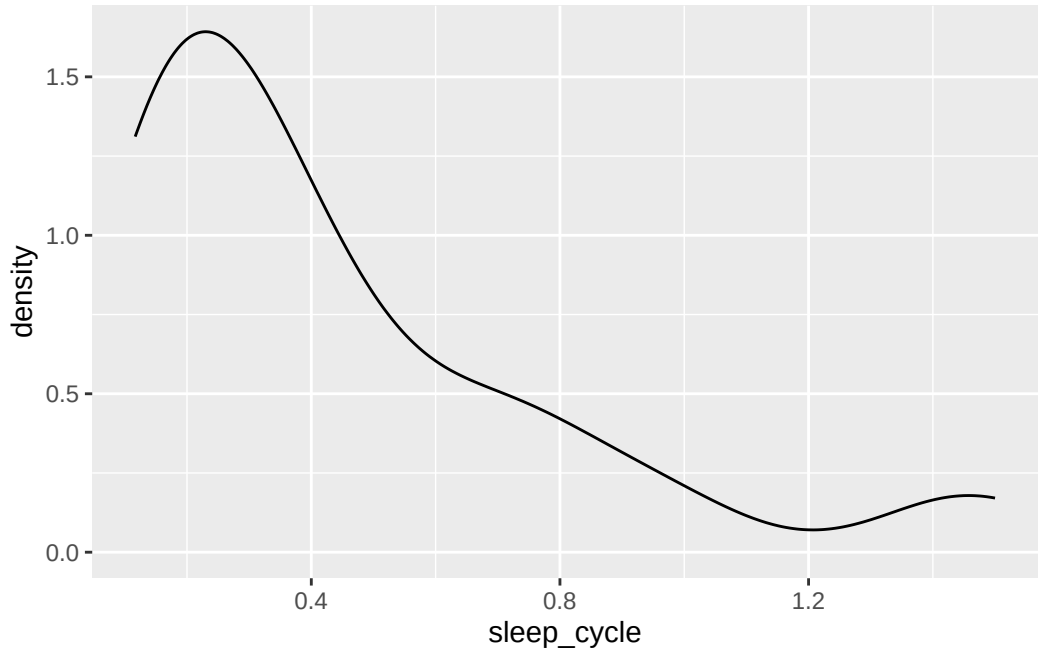


Histograms, etc

```
# histogram
msleep |>
  ggplot(aes(sleep_cycle)) +
  geom_histogram()
```

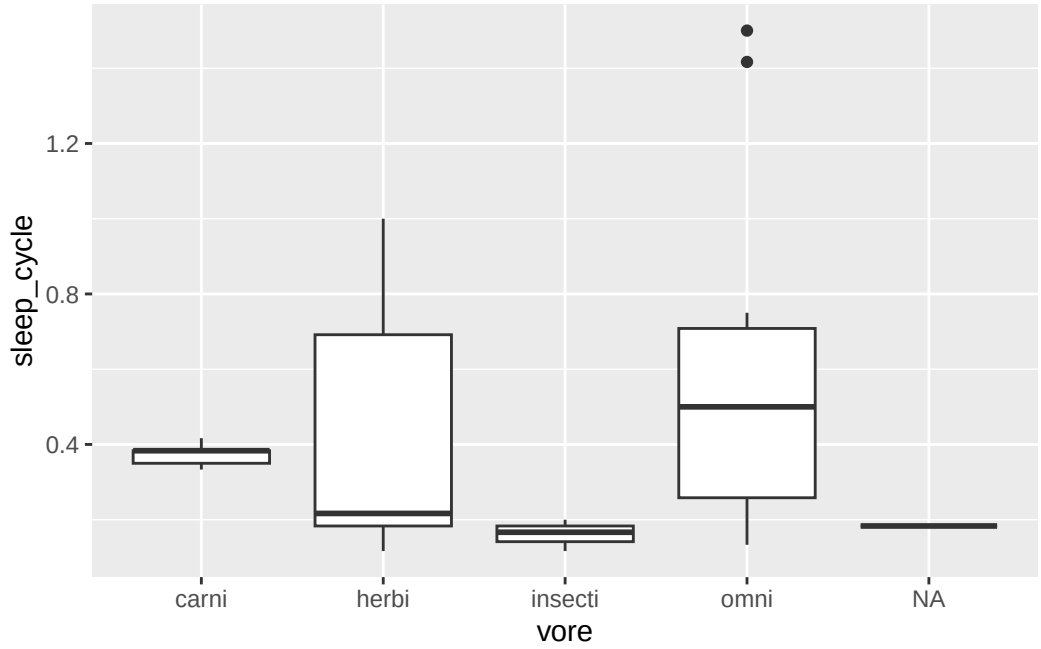



```
# density plot
msleep |>
  ggplot(aes(sleep_cycle)) +
  geom_density()
```

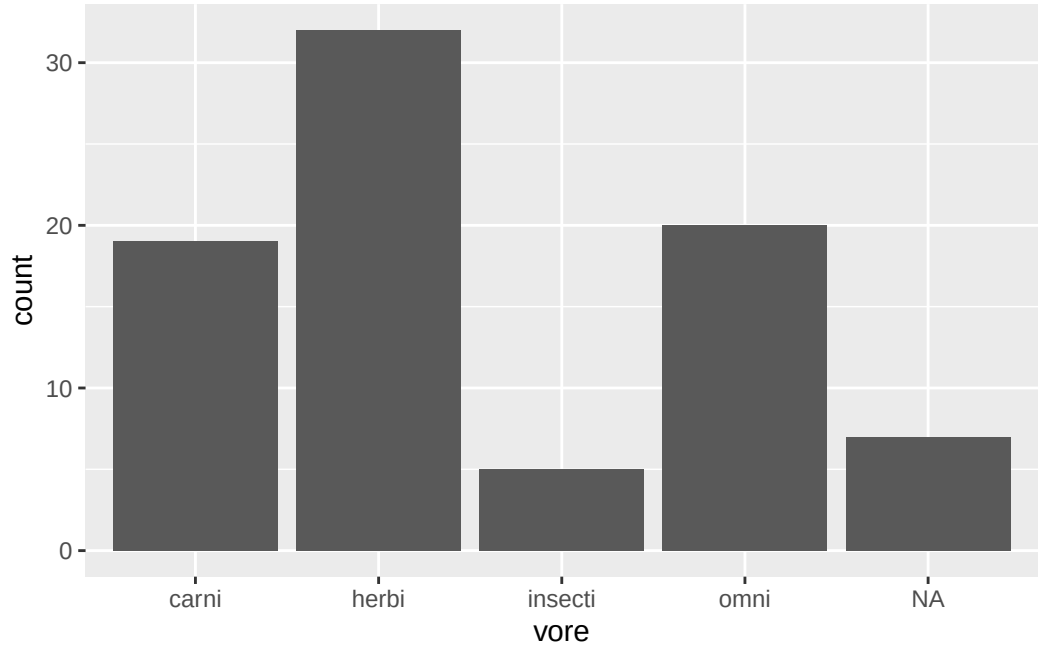



```
# box plot
msleep |>
```

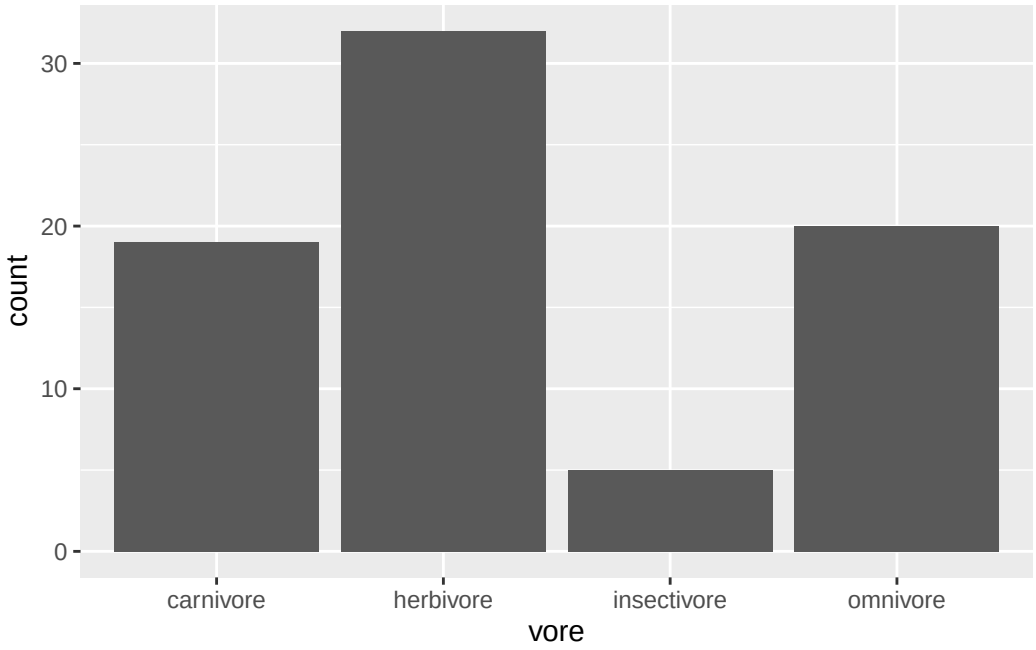
```
  ggplot(aes(vore, sleep_cycle)) +
  geom_boxplot()
```




```
# bar plot
msleep |>
  ggplot(aes(vore)) +
  geom_bar()
```

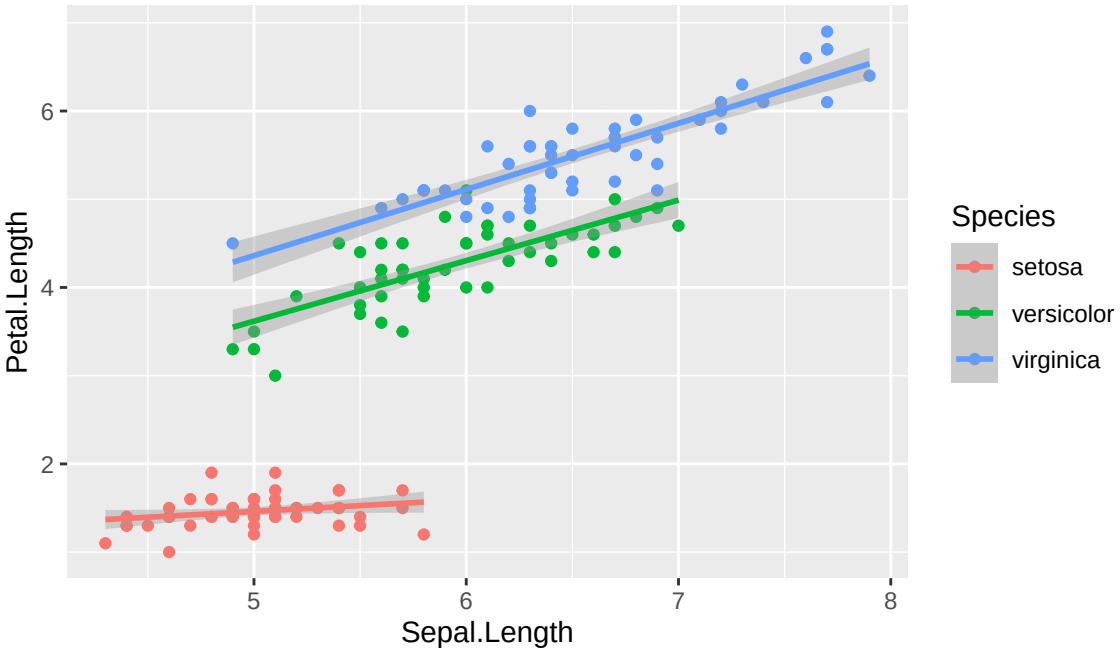



```
msleep |>
  filter(!is.na(vore)) |>
  mutate(vore=paste0(vore, "vore")) |>
  ggplot(aes(vore)) +
  geom_bar()
```

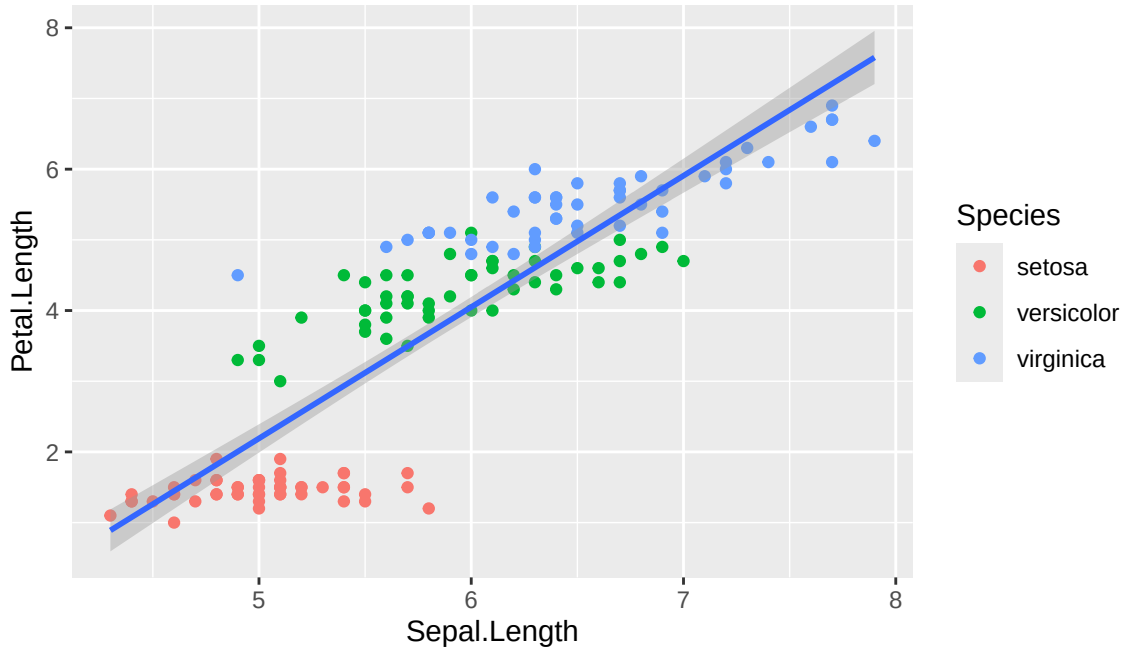


Trend lines

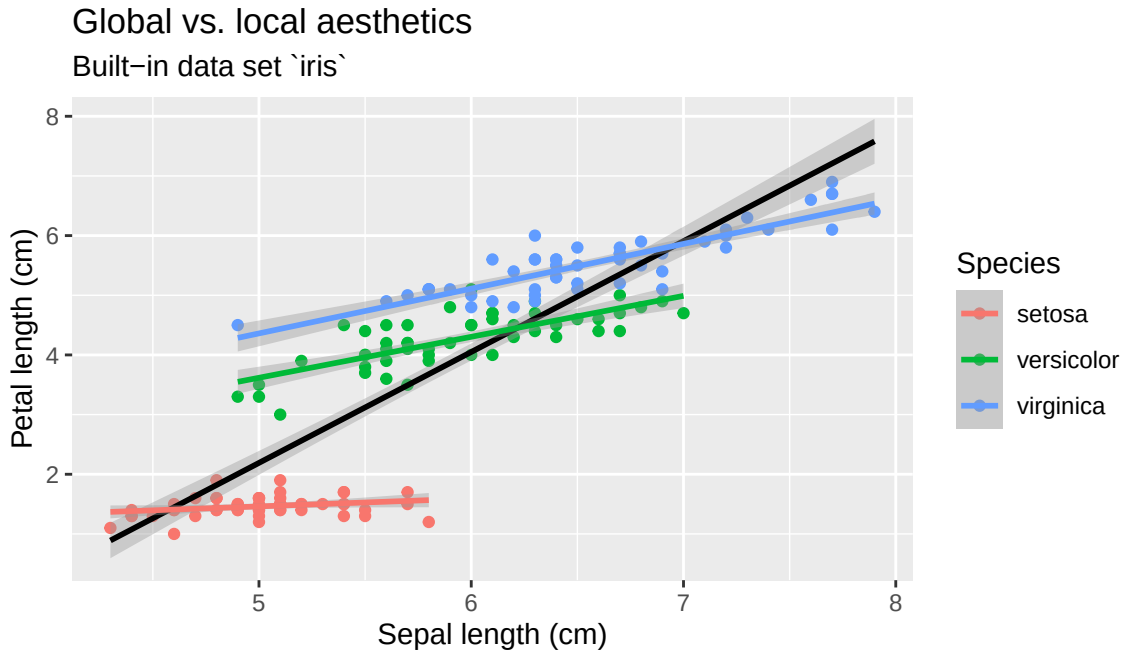
```
# scatter plot with group-level trend lines
iris |>
  ggplot(aes(Sepal.Length, Petal.Length, colour=Species)) +
  geom_point() +
  geom_smooth(method="lm")
```




```
# scatter plot with global trend line
iris |>
  ggplot(aes(Sepal.Length, Petal.Length)) +
  geom_point(aes(colour=Species)) +
  geom_smooth(method="lm")
```




```
# scatter plot with global and group-level trend lines
iris |>
  ggplot(aes(Sepal.Length, Petal.Length)) +
  geom_point(aes(colour=Species)) +
  geom_smooth(method="lm", colour="black") +
  geom_smooth(aes(colour=Species), method="lm") +
  labs(x="Sepal length (cm)",
       y="Petal length (cm)",
       title="Global vs. local aesthetics",
       subtitle="Built-in data set `iris`")
```

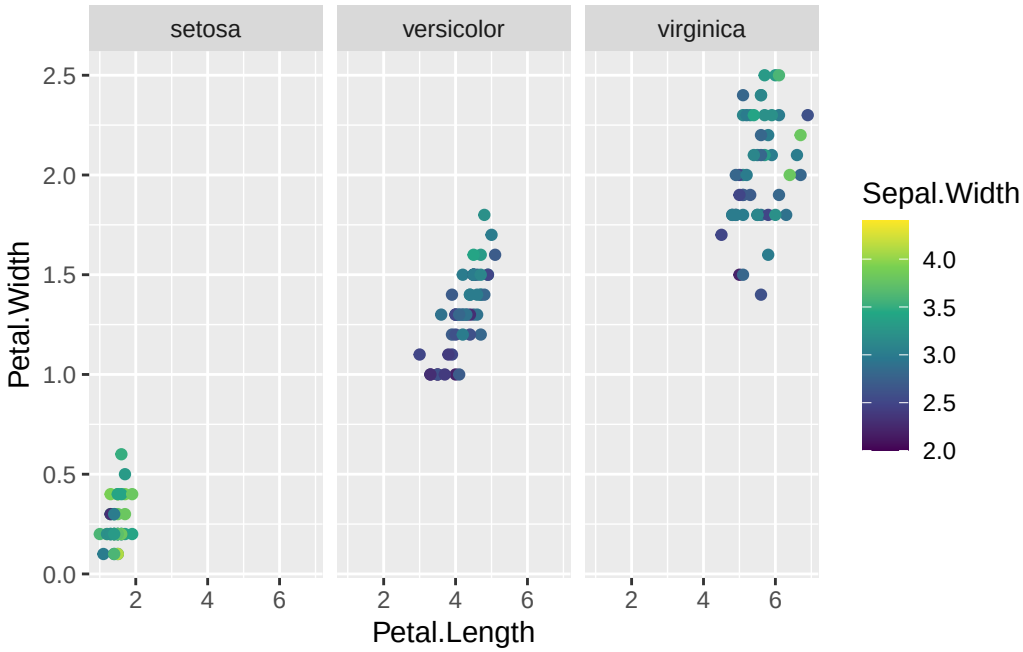


Bells and whistles

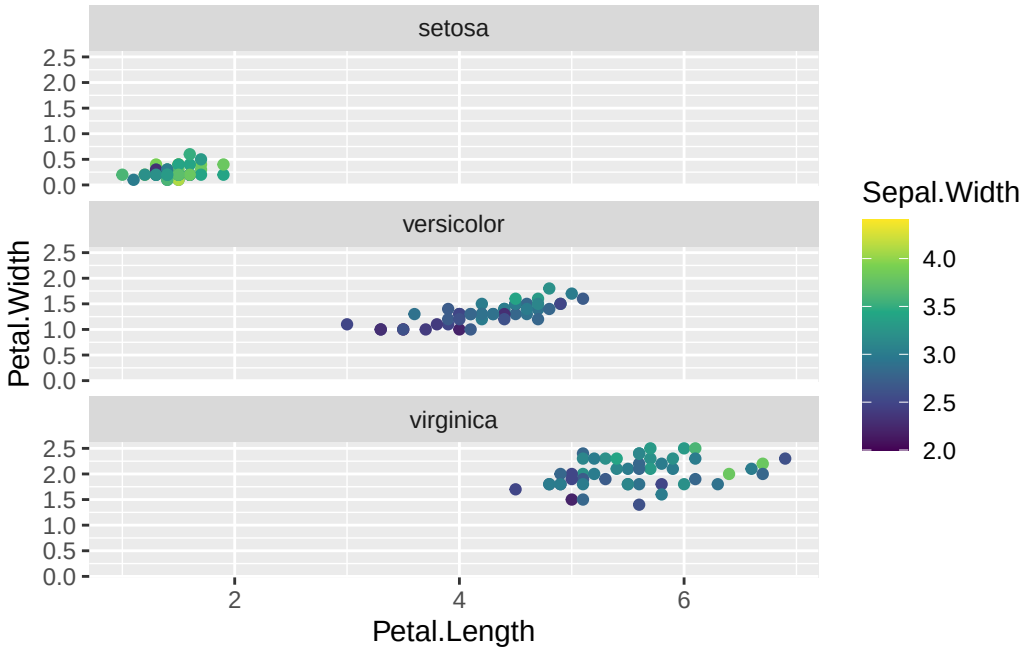
There are endless options for making your plot just as you like. Colour can be specified with functions taking the form `scale_colour_*()` or `scale_fill_*()` (e.g., see `?scale_fill_manual`). The details of the

plot elements can be specified within the `geom_*()` functions (e.g., see `?geom_point`). Panels can be added using `facet_wrap(~col_A)` or `facet_grid(col_A ~ col_B)`. Axes also have their own functions (e.g., see `?scale_x_continuous`). Plots also have themes, which can be specified manually using `theme()` or by adding pre-specified themes like `theme_classic()` or `theme_bw()`.

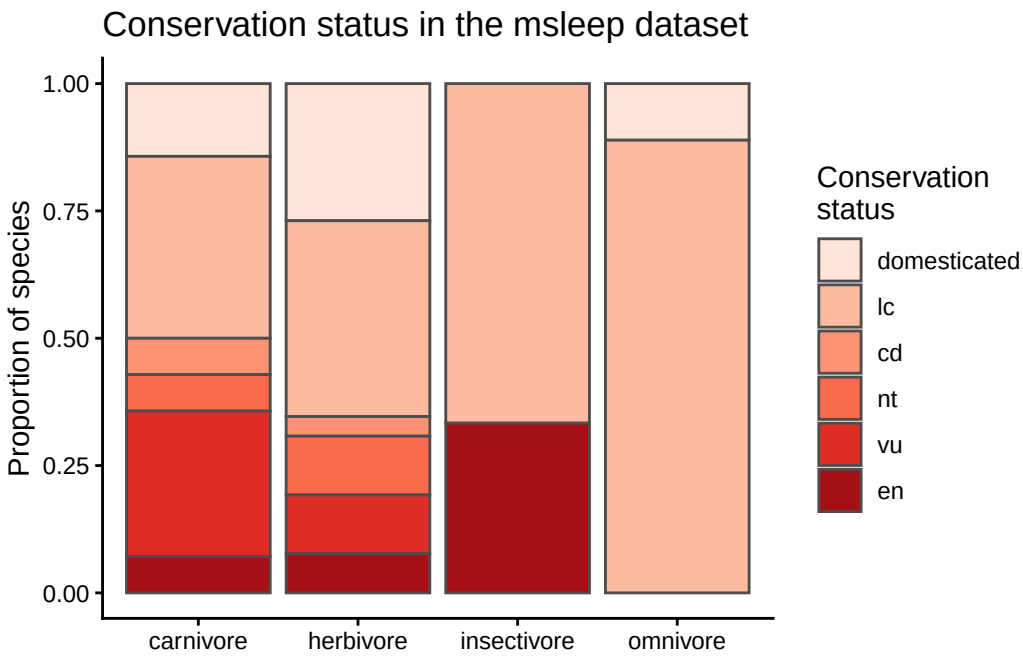
```
# specify a colour scale with species as panels
iris |>
  mutate(Sepal.Area=Sepal.Length*Sepal.Width) |>
  ggplot(aes(Petal.Length, Petal.Width, colour=Sepal.Width)) +
  geom_point() +
  scale_colour_viridis_c() +
  facet_wrap(~Species)
```




```
# use arguments within facet_wrap() to have 1 column of panels
iris |>
  mutate(Sepal.Area=Sepal.Length*Sepal.Width) |>
  ggplot(aes(Petal.Length, Petal.Width, colour=Sepal.Width)) +
  geom_point() +
  scale_colour_viridis_c() +
  facet_wrap(~Species, ncol=1)
```




```
# specify geom_bar() arguments, add a fill, labels, theme, etc
msleep |>
  filter(!is.na(vore) & !is.na(conservation)) |>
  mutate(vore=paste0(vore, "vore"),
         conservation=factor(conservation,
                             levels=c("domesticated", "lc", "cd", "nt", "vu", "en"))) |>
  ggplot(aes(vore, fill=conservation)) +
  geom_bar(position="fill", colour="grey30") +
  scale_fill_brewer("Conservation\nstatus", type="seq", palette="Reds") +
  labs(y="Proportion of species",
       title="Conservation status in the msleep dataset") +
  theme_classic() +
  theme(axis.title.x=element_blank())
```



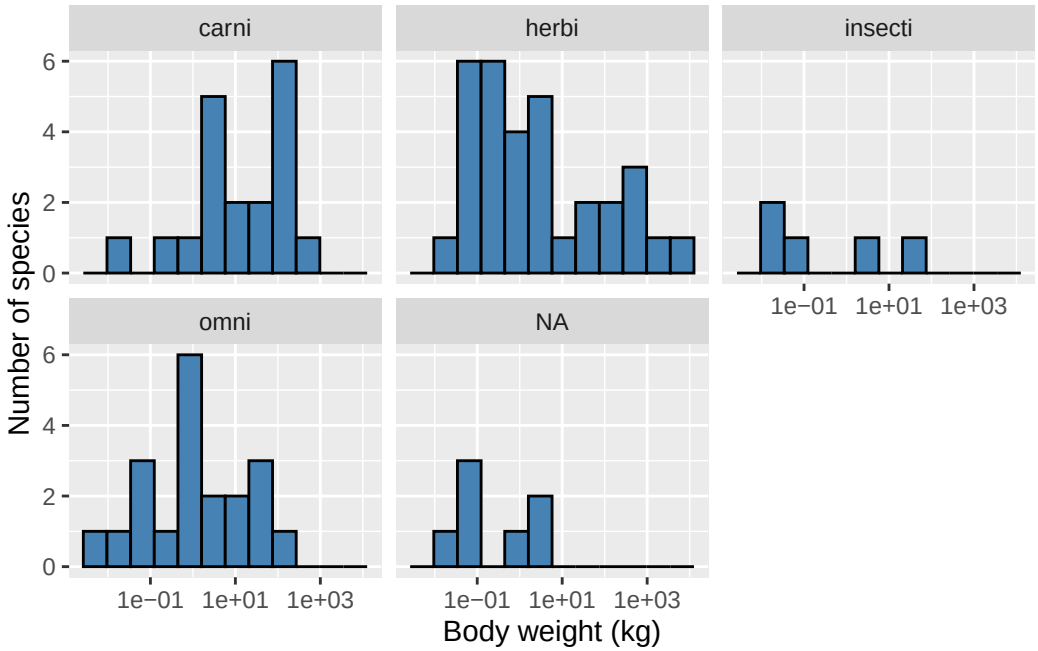
These can all be combined in any way you can imagine.

! Explore the `msleep` dataset further using ggplot. For example, is there an association between sleep and brain weight as a proportion of body weight? Does this depend on feeding strategy? Do larger species spend more time sleeping?

! Create a histogram of body weight with each feeding strategy in a different panel. Try transforming the x-axis to log-10 scale. Adjust the number of bins in the histograms and change the fill and colour of the bars. Add axis labels.

View solution

```
msleep |>
  ggplot(aes(bodywt)) +
  geom_histogram(bins=12, colour="black", fill="steelblue") +
  scale_x_log10() +
  facet_wrap(~vore) +
  labs(x="Body weight (kg)", y="Number of species")
```



Working with strings

The stringr package has many convenient functions for working with strings (text). They generally follow the naming form of `str_verb()`. See the Strings chapter in R for Data Science for a more comprehensive overview.

Key functions:

- `str_remove()` removes a *pattern* from each element in the vector
- `str_replace()` replaces a *pattern* from each element in the vector
- `str_sub()` extracts a subset of characters from each element in the vector
- `str_split()` splits each element in the vector based on a *pattern*

Working with dates

Date-time data are notoriously prickly, and R is no exception. The lubridate package makes them much less so. See the Dates and Times chapter in R for Data Science for a more comprehensive overview.

Key functions:

- `ymd()`, `mdy()`, `dmy()` convert a string with **y**ear, **m**onth, and **d**ay into a date
- `ymd_hms()` converts a string to a date-time, including **h**our, **m**inutes, and **s**econds
- `year()`, `month()`, `mday()`, `yday()`, `week()` extract the corresponding component
- adding and subtracting work intuitively
- plotting with *ggplot2* is straightforward