Lab 02 – Statistical programming

ENVX1002 Handbook

Semester 1, 2025

Learning Outcomes

At the end of this practical students should be able to:

- Acknowledge the importance of academic integrity
- Import, subset and inspect data in R
- Calculate simple summary statistics using both R and Excel
- Generate dynamic reports in Quarto using inline R code
- Understand how to debug R code

Before we begin

Please create a new Quarto document in your project folder to practice R code and complete lab exercises. We will go through the process in Exercise 1.

The following packages are required for this practical:

- `tidyverse` – a collection of R packages designed for data science
- `readxl` – a package for reading Excel files
- `modeest` – a package for estimating the mode of a distribution
- `lubridate` – a package for working with dates and times

If you have not already installed these packages, you can do so by adding the following code into a code chunk in your Quarto document:

```
library(tidyverse)
library(readxl)
library(modeest)
library(lubridate)
```

Finally, please download the data file used in this lab here: soil.xlsx.

Academic integrity

This exercise encourages students to discuss academic integrity, and in particular the grey areas often present. Your demonstrator will provide you with a number of scenarios to discuss with each other in smaller groups, and then with the class.

If you are interested in more information on Academic Integrity at the University of Sydney, see the following link: Academic Integrity. Also ensure you have completed the Academic Honesty Education Module (AHEM). This must be complete before your first assessment is due (next week for ENVX1002).

Exercise 1: Setting up (walk-through)

Getting Started - Important!

Before starting each lab session, you will need to create a new Quarto document to work on the exercises. Follow these steps to set up your lab environment:

1. **Open RStudio Project**
   - Any of the following should work:
     a. In RStudio, click on the project name in the top-right corner and select your project from a recent list. (It might already be open, otherwise it would show "Project: (None)")
     b. In RStudio, use File > Open Project to navigate to your `.Rproj` file
     c. Double-click the `.Rproj` file directly from your file explorer

Your project name should appear in the top-right corner of RStudio.

2. **Create a New Quarto Document**
   - Several ways to do it, the easiest is by clicking the "New File" button in the toolbar and selecting "Quarto Document..."
   - Give your document a meaningful name (e.g., `Lab 2 Exercises`), leave other options as default and click "Create"
3. **Save and Render Your Document**
   - **You must save your document now.** This acts as a simple check to see that you are working in an environment that is not restricted.
   - Click File > Save or press Ctrl+S (Windows) / Cmd+S (Mac)
   - Name it something meaningful like `lab02_practice.qmd`
   - Click the "Render" button in the editor toolbar to generate the HTML output

> **i** Note
>
> If you encounter any issues, please ask your demonstrator for help.

Follow our lead

> **!** Important
>
> We've created this section to make sure you don't get lost during the lab. Remember to read "A brief R guide for surviving ENVX1002" available in the Tool Kit section on Canvas for a better understanding of how to use R in this course.

Two golden rules

1. Text are formatted in Markdown (like this text)
2. Code must be written inside code chunks

A code chunk is what makes your document "dynamic" as it can execute code and perform all sorts of tasks.

To create a code chunk:

- **Quick way**: Use keyboard shortcuts
  - Windows/Linux: `Ctrl + Alt + I`
  - Mac: `Cmd + Option + I`
- **Manual way**: Type three backticks followed by {r}, then end with three backticks

Here's what a code chunk looks like:

````
```{r}
mean(c(1, 2, 3))
```
````

Note: the "fence" that we use (the three backticks plus the `{r}`) will only be visible in Source view. Your demonstrator will show you how to switch between Source and Preview views.

Understanding Functions
This is a function:

```
# Input → Process → Output
mean(c(2, 4, 6)) # Takes numbers, calculates average, returns 4
```

```
[1] 4
```

You can recognise it because it has:

- A name (`mean`)
- Parentheses `()` to hold input data
- Input data inside the parentheses (e.g., `c(2, 4, 6)`)

A function will almost always return an output, which you can use in your code. In this case, the output is `4`. Functions are the reason why we can do so much with R – they are like actions that someobe else has written for us to use so that we can complete our tasks with minimal programming.

Let's look at three common functions for central tendency, but at the same time look at what it means to run functions on data we often call "objects":

1. `mean()`:

```r
# Create a vector of numbers and store it in an object called "scores"
scores <- c(85, 92, 78, 95, 8)
# Calculate the mean of the scores
mean(scores)
```

```
[1] 71.6
```

2. `median()`:

```r
heights <- c(160, 165, 168, 170, 185)
median(heights)
```

```
[1] 168
```

3. `mode()`:

```r
votes <- c("yes", "no", "yes", "maybe", "yes")
mfv(votes)
```

```
[1] "yes"
```

Remember: Every function needs:

- Parentheses `()` to work
- Input data (inside the parentheses)
- Sometimes extra options (like `na.rm = TRUE` to handle missing values)

Exercise 2: Water quality

Sulphate (SO_4) is a key indicator in water quality monitoring and can be used to assess environmental impacts from industry and agriculture. In this exercise, we will explore a dataset of SO_4 concentrations (mg/kg) in water samples.

The data is stored in a sheet called "SO4" in the MS Excel file, `soil.xlsx`.

```r
so4 <- read_excel("data/soil.xlsx", sheet = "SO4")
```

Try writing and running the code chunks in your own Quarto document to see their outputs. Results will appear below each chunk. See Lab 01 for more information on inserting and running code chunks in Quarto.

When we load data into R for the first time, it is important to check what it looks like (and whether it loaded correctly). The `str()` function is a good way to quickly inspect the data:

```r
str(so4)
```

```
tibble [39 × 1] (S3: tbl_df/tbl/data.frame)
 $ SO4: num [1:39] 50.6 55.4 56.5 57.5 58.3 63 66.5 64.5 63.4 58.4 ...
```

> Question 1
>
> **What does the output of `str()` tell us about the data?** You may want to look at the documentation `?str` or search online for more information, but ask your demonstrator if you're still unsure about why we use this function. Relate this to the data you are working with.

Since the data is a data frame object, we have a good idea of what functions we can use to explore it. Let's examine the first few rows of our data:

```r
head(so4)
```

Let's calculate some basic descriptive statistics. Read the code and try to understand every part of it, including the text in #.

```r
# Calculate mean, median, and mode
mean_so4 <- mean(so4$SO4)
median_so4 <- median(so4$SO4)
mode_so4 <- mfv(so4$SO4)[1] # Most frequent value using modeest package

# Calculate measures of spread
range_so4 <- range(so4$SO4)
iqr_so4 <- IQR(so4$SO4)
var_so4 <- var(so4$SO4)
sd_so4 <- sd(so4$SO4)
```

Reporting your data

When reporting statistics in a Quarto document, there are two approaches we could take:

1. Basic R output:

```r
mean_so4
```

```
[1] 61.92308
```

```r
sd_so4
```

```
[1] 5.241558
```

```
median_so4
```

```
[1] 62.1
```

2. Inline reporting (recommended):
Consider the following reporting:

> The mean SO_4 concentration in our samples is 61.92 mg/kg, with a standard deviation of 5.24 mg/kg. The median value is 62.1 mg/kg.

Using inline R code (approach 2) has several advantages:

- Statistics are seamlessly integrated into your text
- Numbers are automatically updated if your data changes
- Results are presented in a reader-friendly format

To create inline R code, use backticks with r, like this:

```
The mean SO~4~ concentration in our samples is `{r} round(mean_so4, 2)` mg/kg...
```

Try to recreate the report above in your Quarto document, or use other objects like `mode_so4` and `var_so4` in your report.

Exercise 3: Using MS Excel

Why use Excel when you have R? Well, Excel is an ubiquitous tool in many industries and can be a useful complement to R for quick data exploration and analysis. Sometimes it may even be easier to use Excel for simple tasks.

Let's calculate the same statistics using Excel to compare approaches. First:

1. Open the `soil.xlsx` file in Excel
2. Navigate to the "SO4_excel" sheet (we'll use this sheet to avoid modifying the original data)
3. Ensure the data is properly displayed in columns

MS Excel remains a popular tool for quick and dirty data analysis and can be a valuable resource in just about any field.

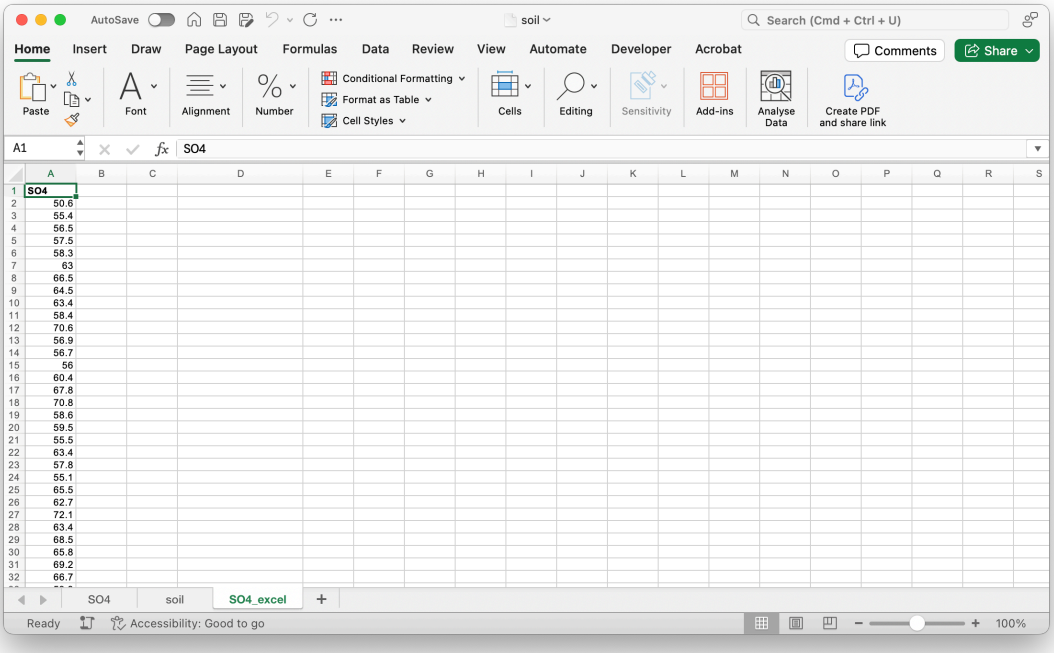

Figure 1: Imported data in Excel. Make sure to use the “SO4_excel” sheet.

For any calculation in Excel:

1. Select an empty cell
2. Type “=” followed by the function name
3. Select your data range
4. Press Enter

For example, to calculate the mean:

1. Click an empty cell
2. Type `=AVERAGE(`
3. Select all SO4 values
4. Type `)` and press Enter

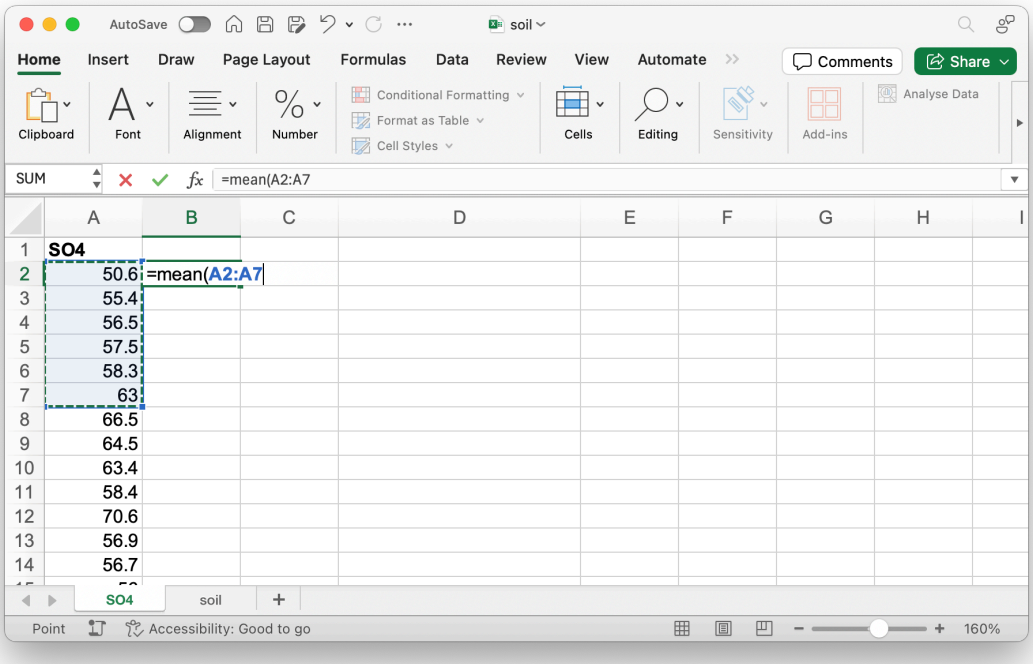

Figure 2: Using the `=AVERAGE()` formula in Excel. Note that this screenshot demonstrates the selection of a specific range of data.

Question 2

Calculate these statistics for the SO_4 data in Excel:

- For central tendency:
  - Mean: Use `=AVERAGE()`
  - Median: Use `=MEDIAN()`
  - Mode: Use `=MODE()`
- For spread:
  - Range: Use `=MAX()` and `=MIN()`
  - IQR: Use `=QUARTILE.INC()` for Q3 and Q1
  - Variance: Use `=VAR.S()`
  - Standard Deviation: Use `=STDEV.S()`

Write notes on how you used these formulas if necessary. You may also want to compare the results with those obtained in R and write down your thoughts on the efficiency and ease of use for each method.

Exercise 4: Soil data

In agricultural science, soil characteristics are essential for understanding soil health and fertility. The `soil` sheet in the `soil.xlsx` file contains data on soil properties at different depths, as well as lithology and land use information.

In this exercise, we'll explore different ways to subset data in R using the soil characteristics data. First, let's load the data:

```
soil <- read_excel("data/soil.xlsx", sheet = "soil")
```

Understanding data structure

Let's examine our data structure:

```
str(soil)
```

```
tibble [55 × 7] (S3: tbl_df/tbl/data.frame)
 $ id       : chr [1:55] "AT1" "AT2" "AT3" "BM1" ...
 $ clay0    : num [1:55] 21.26 21.43 4.52 19.37 40.64 ...
 $ clay60   : num [1:55] 30.4 38.2 42.6 24.6 75.6 ...
 $ ec0      : num [1:55] 52.4 34.9 52.8 35.1 46.4 ...
 $ ec60     : num [1:55] 32.6 25.1 38.5 26.4 35.2 ...
 $ lithology: chr [1:55] "Siliceous Mid" "Siliceous Mid" "Siliceous Lower"
"Siliceous Lower" ...
 $ land_use : chr [1:55] "Grazing" "Grazing" "Grazing" "Grazing" ...
```

```
head(soil)
```

```
# A tibble: 6 × 7
  id    clay0 clay60   ec0  ec60 lithology       land_use
  <chr> <dbl>  <dbl> <dbl> <dbl> <chr>           <chr>
1 AT1   21.3    30.4  52.4  32.6 Siliceous Mid   Grazing
2 AT2   21.4    38.2  34.8  25.1 Siliceous Mid   Grazing
3 AT3    4.52   42.6  52.8  38.5 Siliceous Lower Grazing
4 BM1   19.4    24.6  35.2  26.4 Siliceous Lower Grazing
5 BM2   40.6    75.6  46.4  35.2 Mafic           Grazing
6 BM3   42.7    75.1  50.0  59.7 Mafic           Grazing
```

Question 3

What do you notice about the data structure that is different from the SO_4 data? How does this affect the way we subset the data?

The `soil` data frame contains the following columns:

- `clay0`: Clay content at 0 cm depth
- `clay60`: Clay content at 60 cm depth
- `ec0`: Electrical conductivity at 0 cm depth
- `ec60`: Electrical conductivity at 60 cm depth
- `lithology`: Type of soil composition
- `land_use`: Land usage type

Basic subsetting in R

There are two main ways to subset data in R:

1. Using the `$` operator to select columns
2. Using square brackets `[]` to select rows and columns

Using the $ operator

The `$` operator allows us to select a specific column from our data frame. For example:

```
# Get the land use column
soil$land_use
```

```
 [1] "Grazing"  "Grazing"  "Grazing"  "Grazing"  "Grazing"  "Grazing"
 [7] "Grazing"  "Grazing"  "Grazing"  "Grazing"  "Grazing"  "Grazing"
[13] "Grazing"  "Grazing"  "Grazing"  "Grazing"  "Grazing"  "Grazing"
[19] "Grazing"  "Grazing"  "Grazing"  "Grazing"  "Cropping" "Cropping"
[25] "Cropping" "Cropping" "Grazing"  "Grazing"  "Grazing"  "Grazing"
[31] "Grazing"  "Grazing"  "Cropping" "Cropping" "Cropping" "Grazing"
[37] "Cropping" "Grazing"  "Grazing"  "Grazing"  "Cropping" "Grazing"
[43] "Grazing"  "Grazing"  "Cropping" "Grazing"  "Grazing"  "Grazing"
[49] "Grazing"  "Grazing"  "Natural"  "Natural"  "Grazing"  "Cropping"
[55] "Cropping"
```

Combined with functions, the `$` operator can be used to calculate statistics on specific columns.

```
# Calculate the mean clay content at 0 cm depth
mean_clay0 <- mean(soil$clay0)
mean_clay0
```

```
[1] 23.198
```

Notice how we saved the result of the `mean()` function in a new object `mean_clay0`. This is useful for storing results and using them later in your code (e.g. for inline reporting). However, it means that the result will not be displayed in the console unless you explicitly print it by typing `mean_clay0` again in a new line.

> **Note**
>
> The `$` operator is particularly useful when you want to:
>
> - Access a single column quickly
> - Use column values in calculations
> - Create plots with specific variables

Using square brackets []

Square brackets allow more flexible subsetting using the format: `dataframe[rows, columns]`. It also works with vectors, lists, and matrices. Try the following examples:

```
soil[1:5, ] # First 5 rows, all columns
```

```
# A tibble: 5 × 7
  id    clay0 clay60   ec0  ec60 lithology       land_use
  <chr> <dbl>  <dbl> <dbl> <dbl> <chr>           <chr>
1 AT1   21.3    30.4  52.4  32.6 Siliceous Mid   Grazing
2 AT2   21.4    38.2  34.8  25.1 Siliceous Mid   Grazing
3 AT3    4.52   42.6  52.8  38.5 Siliceous Lower Grazing
4 BM1   19.4    24.6  35.2  26.4 Siliceous Lower Grazing
5 BM2   40.6    75.6  46.4  35.2 Mafic           Grazing
```

```
soil[, c("clay0", "clay60")] # Clay columns
```

```
# A tibble: 55 × 2
   clay0 clay60
   <dbl>  <dbl>
 1 21.3    30.4
 2 21.4    38.2
 3  4.52   42.6
 4 19.4    24.6
 5 40.6    75.6
 6 42.7    75.1
 7 28.1    50.5
 8 41.3    72.6
 9 48.7    78.7
10 26.3    50.5
# ℹ 45 more rows
```

```
soil[1:3, c("ec0", "ec60")] # First 3 rows, electrical conductivity columns
```

```
# A tibble: 3 × 2
    ec0  ec60
  <dbl> <dbl>
1  52.4  32.6
2  34.8  25.1
3  52.8  38.5
```

When using `[]`, leaving the row or column section empty (with just a comma) means "select all"

Basic subsetting: Use square brackets to extract soil samples at 0cm depth that have clay content greater than 40%.

```
soil[soil$clay0 > 40, ]
```

Multiple conditions: Find samples where clay content at 60cm is greater than 30% AND electrical conductivity at 0cm is less than 0.5.

```
soil[soil$clay60 > 30 & soil$ec0 < 0.5, ]
```

Question 4

Practice: Subset the data to find samples where:

- clay content at 0cm depth is less than 50%
- electrical conductivity at 60cm depth is between 25 and 30, inclusive
- clay content at 0cm is less than 10% OR electrical conductivity at 60cm is greater than 25

Exercise 5: Soil statistics

We will continue to work on the soil dataset and practice calculating some basic summary statistics.

Question 5

1. Calculate the mean, median, and mode for clay content at 0cm depth
2. Calculate the range, IQR, variance, and standard deviation for electrical conductivity at 60cm depth
3. Report these statistics in your Quarto document using inline R code

Done!

This is the end of Lab 02. Remember to save your Quarto document ~~and submit it to Canvas~~ when you're done. If you have any questions, feel free to ask your demonstrator for help.

Bonus: Take-home exercises

Take-home exercises are optional but recommended for further practice. You can complete these exercises in your own time.

Exercise 5: Mario Kart statistics

We're not going to go too much about the game – but here is a dataset of character attributes from the popular Mario Kart.

Download it here: mario_kart.csv

The data is stored in a CSV file, which MS Excel can open directly. To read it into R, use the `read_csv()` function from the `readr` package.

The dataset contains the following variables:

- `character`: Character name
- `weight`: Character's weight class (1-5 scale)
- `speed`: Maximum speed rating (1-5 scale)
- `acceleration`: How quickly they reach top speed (1-5 scale)
- `handling`: Steering control rating (1-5 scale)
- `traction`: Grip on different surfaces (1-5 scale)
- `drift_rating`: Skill at power-sliding around corners (1-5 scale)

Complete the following exercises using both R and Excel:

1. **Data Import and Inspection**
   - Load the data into R using `read_csv()`
   - Examine the structure using `str()`
   - View the first few rows using `head()`
   - What do you notice about the data?
2. **Central Tendency** Calculate for the speed attribute:
   - Mean
   - Median
   - Mode

   Which measure best represents the "typical" speed rating? Why?
3. **Spread Analysis** For the weight attribute, calculate:
   - Range
   - IQR
   - Variance
   - Standard deviation

   What do these tell you about the variation in character weights?
4. **Character Comparison**

- Which characters have the highest and lowest acceleration?
- Find all characters with above-average handling
- Identify characters with both speed and weight above 4.0

5. **Advanced Challenge**

- Calculate the mean of all attributes for each character
- Who is the most "well-rounded" character (closest to average in all stats)?
- Create a report comparing Mario and Luigi's stats using inline R code

Attribution