Tutorial 01

ENVX2001 – Applied Statistical Methods

Semester 1

Welcome 👋

Welcome to Tutorial 1. In this tutorial we will show you how to:

1. Create an RStudio Project.
2. Create a Quarto document and customise it.
3. Format text and explore some advanced Quarto options.

Getting started

In some instances you will see notes like this one in the margin, which often contain useful tips or reminders.

- These tutorials are designed to be comprehensive yet accessible. I recommend following along and completing the exercises as you go.
- You will need to **create your own Quarto document** to complete the exercises. In RStudio, go to
  - `File → New File → Quarto Document...` or
  - `File → New File → R Markdown...` and follow the intuitive prompts.
- If you have any suggestions for improvement, please let me know! Send me an email (you can find my details on Canvas).

RStudio Projects

RStudio Projects help you keep your work organised and reproducible. While not strictly necessary, they make your life easier, especially when working on multiple project.

Why Use Projects?

RStudio Projects help with organisation by keeping all project files together in one directory. They also ensure reproducibility by making your code work consistently across different computers. Think of them as a way to "tell" RStudio that all your work for a particular project is in one place.

At its heart, **an RStudio Project is simply a text file with the `.Rproj` extension.** This file lives in your folder that you're working in, and tells RStudio to treat that folder as a project. Once created, RStudio manages the project for you and you never have to edit the `.Rproj` file directly.

Setting up a Project

Let's set up a new project for ENVX2001 in RStudio.

1. Open RStudio
2. Click **File -> New Project...**
3. Select either **New Directory** or **Existing Directory**
   - **New Directory** creates a new folder for your project
   - **Existing Directory** lets you choose a folder that already exists
4. Select **New Project** unless you have a specific project you want to work on (e.g. a Quarto Blog)
5. Name your project "ENVX2001" or similar, choose where to save it, and click **Create Project**. You will see other options, but you can ignore them for now.

Give your project a meaningful name, ideally with "ENVX2001" in it so you can identify it later. For example "ENVX2001 Labs".

Recommended folder structure

Modern technology means that some of you may never have to touch the file system. However, it's still good to know how to organise your files. Search the internet for using the Finder (macOS) or the File Explorer (Windows) if you need a refresher.

Here's a recommended folder structure for your project:

```
CODE
ENVX2001/
├── labs/
│   ├── data/
│   ├── Lab01.qmd
│   ├── Lab02.qmd
│   ├── ...
│   └── ENVX2001-labs.Rproj
├── project_1/
│   ├── data/
│   ├── project1.qmd
│   └── ENVX2001-labs.Rproj
├── project_2/
│   ├── data/
│   └── ...
└── ...
```

Opening Projects

From File Explorer (PC) or Finder (Mac)

1. Navigate to your project folder
2. Double-click the `.Rproj` file
   - On Windows: Look for the RStudio cube icon
   - On Mac: Look for the `.Rproj` extension

From Within RStudio

Manually:

1. Click `File` → `Open Project...`
2. Navigate to your project folder
3. Select the `.Rproj` file
4. Click "Open"

Recent projects:

If you've opened the project before, your project may already be open! Check the top-right corner of RStudio – if you see your project name, you're good to go. Click on the dropdown to switch to recently opened projects.

Benefits

When in a project, RStudio will remember your working directory, history, and environment. This means you can close RStudio and come back to your project later, and everything will be as you left it. This is especially useful when working on a project over multiple days. Additionally, adding data files to your project folder will make them easier to access in your code as you can use **relative paths**. For example, instead of using:

```
CODE
read.csv("C:/Users/username/Documents/ENVX2001/data.csv")
```

you can use:

```
CODE
read.csv("data/data.csv")
```

The latter is more portable and **will work on any computer**, as long as the data file is in the `data` folder, while the former will only work on your computer as it would have your user name on it!

What's next?

Set up an RStudio project separately for your labs and projects. This will make it easier to keep your work organised and reproducible. Otherwise, there isn't much else to think about as RStudio will manage the project for you. Just remember to open your project when you start working on it!

Quarto

Getting Started

Let's create and modify a Quarto document about The Beatles. Follow these steps to learn about Quarto's features:

1. In RStudio, go to **File > New File > Quarto Document...**
2. In the dialogue box that appears:
   - Enter "The Beatles: A Musical Journey" as the title
   - Add your name as the author
   - Select "HTML" as the output format
3. Click "Create"

Exercise 1: Setting Up Your Document

1. Delete all the example content in your new document (but keep the YAML header)
2. Replace your YAML header with this enhanced version:

```
CODE
---
title: "The Beatles: A Musical Journey"
subtitle: "From Liverpool to Global Stardom"
author: "Your Name"
date: "today"
format:
  html:
    toc: true
    number-sections: true
    theme: cosmo
---
```

3. Click "Render" to see what an empty document looks like

Exercise 2: Adding Content

Copy and paste this text after your YAML header:

```
CODE
## The Early Years

The Beatles' story begins in Liverpool, England, where John Lennon formed a band called the
Quarrymen in 1957. Paul McCartney joined soon after, followed by George Harrison in 1958. The band
went through several name changes before settling on "The Beatles" in 1960.

## Hamburg and The Cavern Club

### The Hamburg Experience

In 1960, the Beatles began playing in Hamburg, Germany. These extended performances helped them
develop their stage presence and musical style. During this period, they recruited Ringo Starr as
their drummer, completing the lineup that would become legendary.

### The Cavern Club Era

Back in Liverpool, the Beatles became regular performers at The Cavern Club. Between 1961 and
1963, they performed there 292 times. The club became synonymous with their early success and the
birth of the "Merseybeat" sound.

## Global Phenomenon

By 1964, "Beatlemania" had taken hold worldwide. Their appearance on The Ed Sullivan Show in
February 1964 was watched by an estimated 73 million viewers. This marked the beginning of the
"British Invasion" and revolutionised popular music.

## Musical Evolution

The Beatles continuously evolved their musical style throughout their career:

1. Early rock and roll (1962-1964)
2. Folk rock period (1965-1966)
3. Psychedelic era (1966-1968)
4. Return to roots (1969-1970)

Their willingness to experiment with different musical styles and recording techniques set new
standards for popular music.
```

3. Click “Render” again to see the formatted document

Exercise 3: Experimenting with YAML Options

Try these modifications to your YAML header one at a time, rendering after each change:

1. Change the theme:

```
CODE
theme: darkly  # Try: flatly, journal, darkly
```

2. Add a table of contents:

```
CODE
```

```
toc: true
```

3. Add section numbers:

```
CODE
number-sections: true
```

4. Try them all together:

```
CODE
format:
  html:
    toc: true
    theme: cosmo
    number-sections: true
```

Exercise 4: Try Different Output Formats

Microsoft Word

Try creating a Word document by changing your YAML to:

```
CODE
format:
  docx:
    toc: true
    number-sections: true
```

You can learn more about Word output options in the Quarto Word documentation.

Typst (Advanced Option)

Typst is a modern alternative to LaTeX/PDF, but it requires some learning to customise effectively. If you're interested in exploring it:

```
CODE
format:
  typst:
    toc: true
```

Learn more about Typst in: - Quarto's Typst documentation - Official Typst documentation

Multiple Output Formats

You can create multiple output formats simultaneously by specifying them in your YAML header:

```
CODE
format:
  html:
    toc: true
    number-sections: true
```

```
    theme: cosmo
  docx:
    toc: true
    number-sections: true
```

Each format can have its own specific options. For example, you could have different themes for HTML while maintaining consistent structure across all formats.

Quick Tips

1. Use keyboard shortcut to render:
   - Windows/Linux: Ctrl+Shift+K
   - Mac: Cmd+Shift+K
2. Preview different themes from the complete list here
3. Keep experimenting with different YAML options to see what works best for your documents

Writing in Quarto

Let's jump right into it. Create a new Quarto document for this tutorial, and remove the template content except for the YAML header.

Exercise 1: Markdown

First, let's practice some basic markdown formatting.

> **i** Replicate this
>
> Here's a brief overview of **Christopher Nolan's** filmmaking:
>
> 1. His film *Inception* (2010) took **ten years** to write
> 2. He wrote *Memento* based on **two key concepts**:
>   - *Anterograde amnesia*
>   - *Non-linear storytelling*
> 3. His highest-grossing film is *The Dark Knigh Rises* (2012), which earned over **$1 billion** worldwide.
>
> His signature style: complex narratives with time manipulation as seen in *Interstellar* and *Tenet*.
>
> Visit IMDB's Nolan Page for his complete filmography.

Try to replicate the text and formatting above, including the link to IMDB, in your own document, using markdown syntax. For reference, check out the Quarto Markdown Basics.

Exercise 2: Working with code chunks

Now let's focus on code. We will work with data from the `palmerpenguins` package to visualise and summarise some data. Try running the following code chunks in your document:

CODE

````
```{r}
#| label: setup
#| message: false
#| warning: false

library(tidyverse)
library(palmerpenguins)
```
````

Never run package installation code (`install.packages()`) in your Quarto document! Instead, install required packages in the R console:

CODE

```
install.packages(c("tidyverse", "palmerpenguins"))
```

This ensures your document remains reproducible and prevents installation errors during rendering.

Let's break down the chunk options:

- `label:` gives your chunk a unique name
- `message: false` suppresses package loading messages
- `warning: false` suppresses warnings
- The `#|` syntax makes options more readable

Exercise 3: Creating Visualizations

Now let's create some plots with carefully controlled output:

CODE

````
```{r}
#| label: penguin-plot
#| fig-cap: "Penguin body mass by species"
#| fig-width: 8
#| fig-height: 6
#| fig-align: center
#| echo: true

ggplot(penguins, aes(species, body_mass_g, fill = species)) +
  geom_boxplot() +
  theme_minimal() +
  labs(y = "Body Mass (g)",
```
````

```
       x = "Species") +
  theme(legend.position = "none")
```

OUTPUT

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

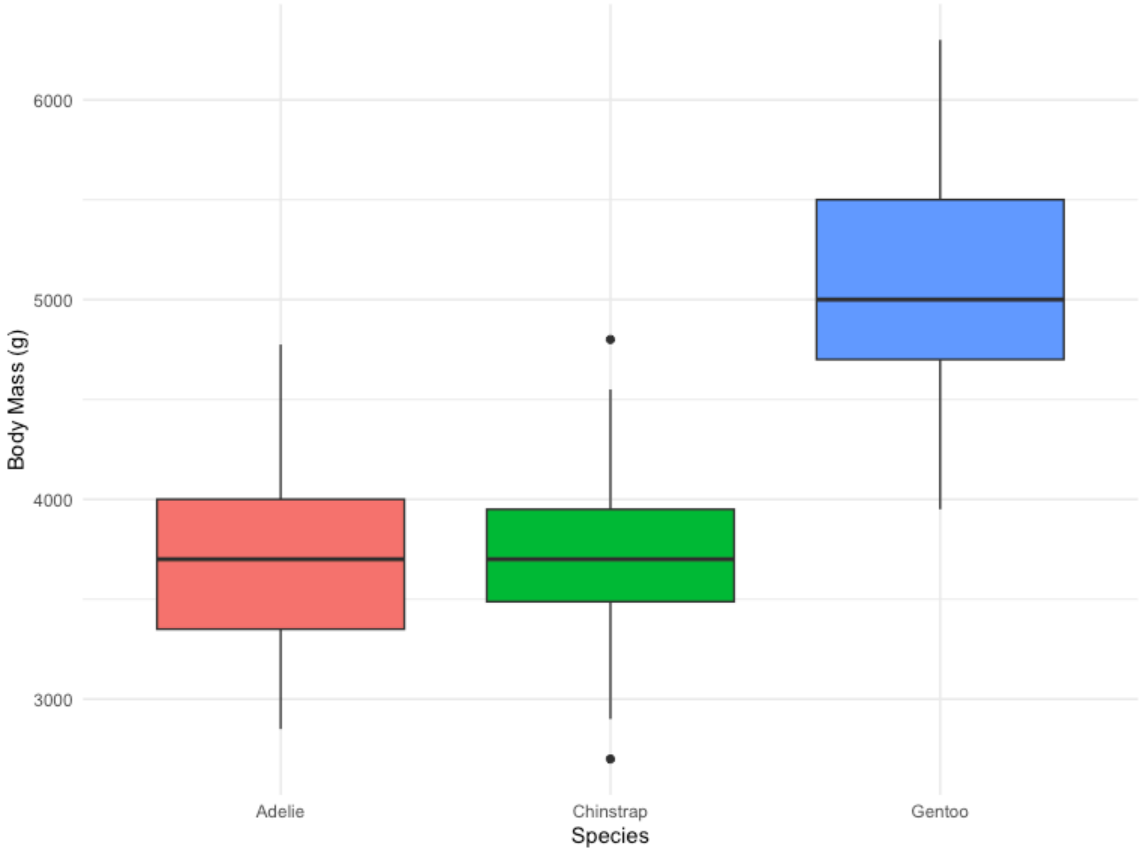


Figure 1: Penguin body mass by species

New options to notice:

- `fig-width` and `fig-height` control plot dimensions
- `fig-align` centers the plot
- `echo: true` shows the code (try changing to `false`)

Exercise 4: Tables and data

Let's explore different ways to display data:

CODE

```{r}
#| label: penguin-summary
#| tbl-cap: "Summary Statistics by Species"

penguins %>%
  group_by(species) %>%
  summarise(
    mean_mass = mean(body_mass_g, na.rm = TRUE),
    sd_mass = sd(body_mass_g, na.rm = TRUE),
    n = n()
  ) %>%
  knitr::kable()
```

species	mean_mass	sd_mass	n
Adelie	3700.662	458.5661	152
Chinstrap	3733.088	384.3351	68
Gentoo	5076.016	504.1162	124

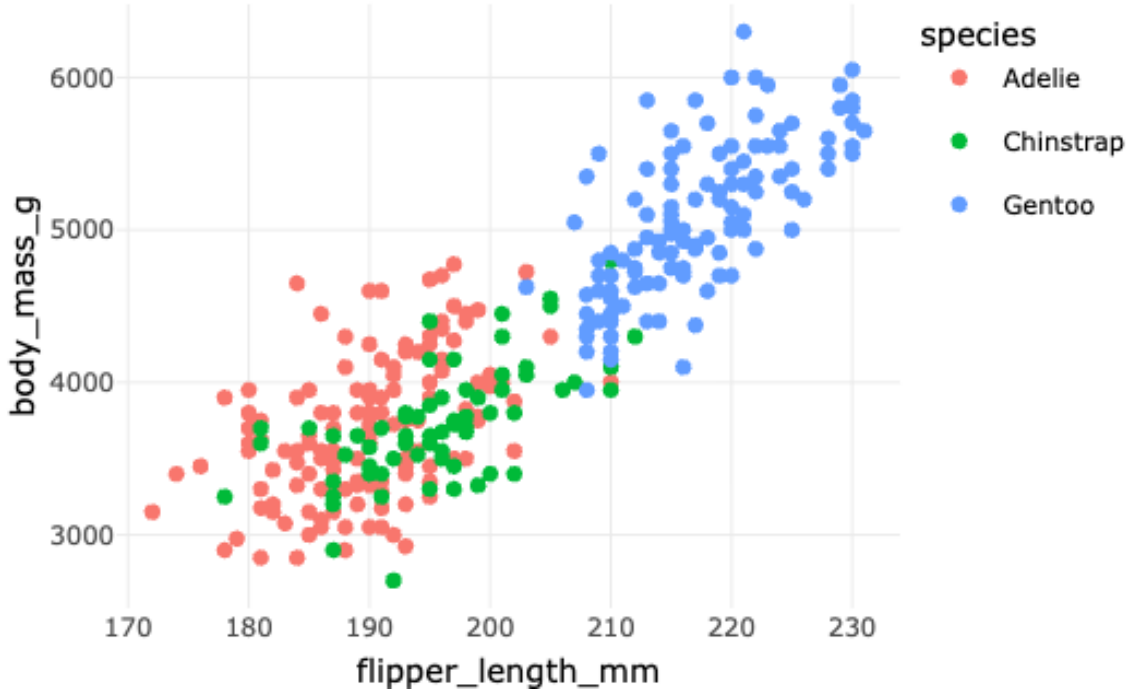
Exercise 5: Interactive features

Quarto excels at creating interactive content. Try this:

CODE
```{r}
#| label: interactive-plot
#| warning: false

library(plotly)
p ← ggplot(penguins,
            aes(flipper_length_mm,
                body_mass_g,
                color = species)) +
  geom_point() +
  theme_minimal()

ggplotly(p)
```



And for interactive tables:

CODE

```
```{r}
#| label: interactive-table

library(DT)
datatable(penguins)
```
```

Show 10 entries

Search:

	species	island	bill_length_mm	bill_depth_mm
1	Adelie	Torgersen	39.1	18.7
2	Adelie	Torgersen	39.5	17.4
3	Adelie	Torgersen	40.3	18
4	Adelie	Torgersen		
5	Adelie	Torgersen	36.7	19.3
6	Adelie	Torgersen	39.3	20.6
7	Adelie	Torgersen	38.9	17.8
8	Adelie	Torgersen	39.2	19.6
9	Adelie	Torgersen	34.1	18.1
10	Adelie	Torgersen	42	20.2

Showing 1 to 10 of 344 entries

Previous 1 2 3 4 5 … 35

Next

Exercise 6: Useful chunk options

Here are some useful chunk options, try them:

1. Cache computations (great for slow calculations):

```
CODE
```{r}
#| label: slow-calculation
#| cache: true

# Your time-consuming code here
Sys.sleep(2)  # Simulating slow computation
```
```

2. Run code but hide everything:

```
CODE
```{r}
#| label: hidden-setup
#| include: false

# This code runs but shows nothing
theme_set(theme_minimal())
```
```

3. Show code but don't run it:

```
CODE
```{r}
#| label: example-code
#| eval: false

# This code is displayed but not executed
ggplot(data) +
  geom_point()
```
```

Try changing the values for options like `echo`, `include`, and others from `true` to `false` (or vice versa). See what happens.

Quick tips

1. Use meaningful chunk labels - they help with debugging
2. Keep chunks focused on single tasks
3. Cache slow computations with `cache: true`
4. Use `include: false` for setup code you don't want to show

Next Steps

That's it! Well done. The Quarto documentation is a great resource for more advanced features, but beware – it's a rabbit hole…

Done!

Thanks!

This is the end of the tutorial.

If you have any questions or suggestions, let me know during the lectures or via email. You know where to find me!