Refining your plots, reports and code

MSc in Statistics 2025/26, Imperial College London

Zak Varty

Welcome Back!

Today's objectives

1. use R code within Quarto for single-document literate programming tasks
2. critique and refine default R outputs (such as plots and tables) for reports
3. understand how and when to show code within reports
4. use style guides and directory templates to provide a consistent structure across projects

Literate Programming

What is literate programming?

- A paradigm that combines:
  - **Code** (that executes)
  - **Narrative** (that explains)
- Result: documents that are both **human-readable** and **machine-executable**

Adding code blocks

~~~
```{r}
# use :: to access data or functions from within a package
penguins <- palmerpenguins::penguins
```
~~~

```
# use :: to access data or functions from within a package
penguins <- palmerpenguins::penguins
```

Unwanted Warnings

Sometimes you get warnings that you don't need to draw attention to

```
library(palmerpenguins)
```

```


Attaching package: 'palmerpenguins'


The following objects are masked from 'package:datasets':

    penguins, penguins_raw
```

Suppressing Warnings

~~~
```{r}
#| warning: false
library(palmerpenguins)
```
~~~

```
library(palmerpenguins)
```

Code blocks that show but don't run

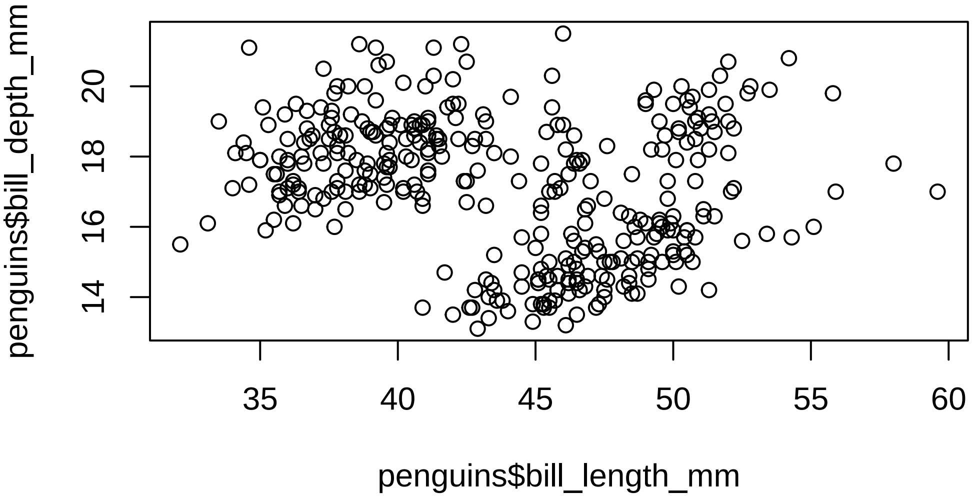
Sometimes you want people to do as you say, not as you do. For example, I might want to ask you to install a package but without downloading it every time these notes are rendered.

~~~
```{r}
#| eval: false
install.packages("palmerpengins")
```
~~~

```
install.packages("palmerpengins")
```

Code blocks that run but don’t show

~~~
```{r}
#| echo: false
plot(x = penguins$bill_length_mm,
     y = penguins$bill_depth_mm)
```
~~~



Multiple outputs

When you have multiple objects being printed in a single block, these will often get split up when the document is rendered.

~~~
```{r}
mean(penguins$body_mass_g, na.rm = TRUE)
median(penguins$body_mass_g, na.rm = TRUE)
```
~~~

```
mean(penguins$body_mass_g, na.rm = TRUE)
```

```
[1] 4201.754
```

```
median(penguins$body_mass_g, na.rm = TRUE)
```

```
[1] 4050
```

Multiple outputs (`hold`)

Top keep this displaying as a single block, you can set `results` to `hold`.

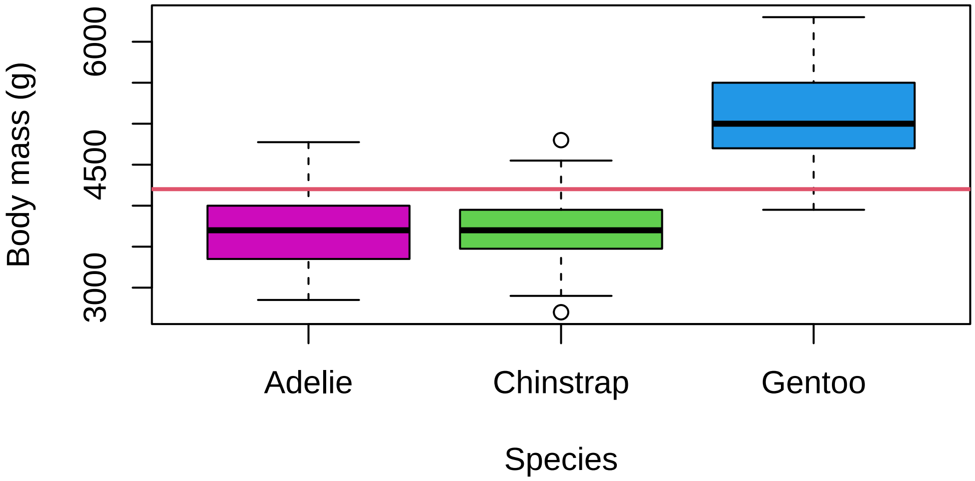
~~~
```{r}
#| results: hold
mean(penguins$body_mass_g, na.rm = TRUE)
median(penguins$body_mass_g, na.rm = TRUE)
```
~~~

```r
mean(penguins$body_mass_g, na.rm = TRUE)
median(penguins$body_mass_g, na.rm = TRUE)
```

```
[1] 4201.754
[1] 4050
```

Motivating Example

To give us an example to play with, we could consider the following question: are the average weights of all species of penguin the same?



Tables

How many penguins of each species do we have?

```r
table(penguins$species)
```

```

   Adelie Chinstrap    Gentoo 
      152        68       124 
```

```r
# fit linear model
mass_lm <- lm(formula = body_mass_g ~ 1 + species, penguins)

# extract coefficients and standard errors
mass_lm_coefficients <- summary(mass_lm)$coefficients[,1:2]
mass_lm_coefficients
```

```
                   Estimate Std. Error
(Intercept)      3700.66225   37.61935
speciesChinstrap   32.42598   67.51168
speciesGentoo    1375.35401   56.14797
```

Nicer Tables

Example 1

```r
species_counts <- table(penguins$species)
knitr::kable(species_counts, col.names = c("Species", "Count"))
```

Species	Count
Adelie	152
Chinstrap	68
Gentoo	124

Example 2

```r
knitr::kable(x = mass_lm_coefficients)
```

	Estimate	Std. Error
(Intercept)	3700.66225	37.61935
speciesChinstrap	32.42598	67.51168
speciesGentoo	1375.35401	56.14797

We could set the row names to improve formatting.

```r
row.names(mass_lm_coefficients) <- c("Intercept", "Chinstrap", "Gentoo")
knitr::kable(x = mass_lm_coefficients)
```

	Estimate	Std. Error
Intercept	3700.66225	37.61935
Chinstrap	32.42598	67.51168
Gentoo	1375.35401	56.14797

And then refine the column names.

```r
knitr::kable(
  x = mass_lm_coefficients,
  col.names = c("MLE", "Std. Err"))
```

	MLE	Std. Err
Intercept	3700.66225	37.61935
Chinstrap	32.42598	67.51168
Gentoo	1375.35401	56.14797

Finally, we really don't need to be showing all of those decimal places.

```r
knitr::kable(
  x = round(mass_lm_coefficients, 2),
  col.names = c("MLE", "Std. Err"))
```

	MLE	Std. Err
Intercept	3700.66	37.62
Chinstrap	32.43	67.51
Gentoo	1375.35	56.15

	MLE	Std. Err

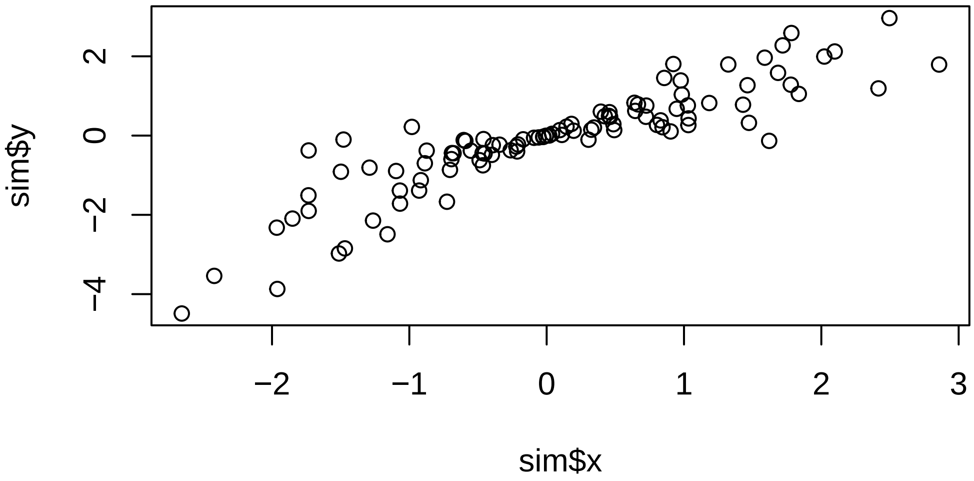
A Semi-scripted approach

In report-style assessments you will not usually display your code in the main body of the report. In that case you can still use literate programming but you should:

- hide all of your code blocks
- clearly section your code with structural comments
- include an extra code block at the end of the report (this is in the template by default).

```r
#######################
# Question 3
#######################

# Part A --------------------------

# simulate data
n <- 100
x <- rnorm(n, mean = 0, sd = 1)
y <- rnorm(n, mean = x, sd = abs(x) / 2)
type <- ifelse(x > 0, "A", "B")

# collect and clean workspace
sim <- data.frame(x, y, type)
rm(n, x, y, type)
```

```r
# Part B --------------------------
plot(x = sim$x, y = sim$y)
```



Labeling and Referencing

Referencing TLDR

- Label items of markdown using the `{#type-label}`
- Label code output with the `#| label: type-label`
- Reference either in the main text using `@type-label`

Example reference types: `sec`, `fig`, `tbl`, `eq`.

For full documentation see Quarto Cross-References.

Referencing Sections and Subsections

```
# Labeling and Referencing {#sec-labeling}
...
@sec-labeling focuses on labelling and referencing parts of your document.
```

Section focuses on labelling and referencing parts of your document.

Referencing Tables

~~~
```{r}
#| label: tbl-palmer
#| tbl-cap: "Example rows of the Palmer penguin dataset."

library(knitr)
kable(head(penguins))
```
~~~

```r
library(knitr)
kable(head(penguins)[ ,1:6])
```

Table 6: Example rows of the Palmer penguin dataset.

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
Adelie	Torgersen	39.1	18.7	181	3750
Adelie	Torgersen	39.5	17.4	186	3800
Adelie	Torgersen	40.3	18.0	195	3250
Adelie	Torgersen	NA	NA	NA	NA
Adelie	Torgersen	36.7	19.3	193	3450
Adelie	Torgersen	39.3	20.6	190	3650

Table 6 gives example data entries from the Palmer penguins dataset. In your reports,all tables should have a caption and be discussed in the main text.

Referencing Figures

```r
plot(x = sim$x, y = sim$y)
```

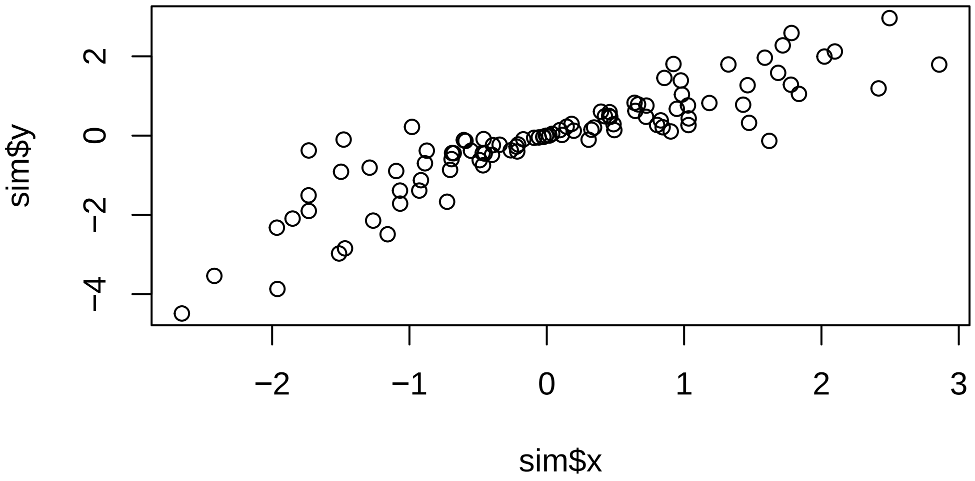


Figure 1

Simulating data can be an important tool to understand how models behave in ideal conditions. Figure 1 shows an example of some bivariate data, simulated in Section .

Referencing Equations

```
$$ e^{\pi i} + 1 = 0.$${#eq-euler}
```

$$e^{\pi i} + 1 = 0. \tag{1}$$

Equation 1 states an elegant and well known result. By convention, equations should only be numbered if they are mentioned in the main text.

Citing Sources

Add a bibliography to your document using the bibliography YAML metadata field. As for LaTeX this should point to a `.bib` file, detailing the sources you will cite. I have provided a small example bibliography file `ref.bib`.

```
---
title: "Refining your plots, reports and code"
bibliography: ref.bib
---
```

By default, Pandoc will automatically generate a list of works cited and place it in the document if the style calls for it. It will be placed in a div with the id refs if one exists:

```
### References

::: {#refs}
:::
```

To cite books or articles you can use a direct citation to talk about a source such as Box and Cox (1964) within a sentence, or a parenthetical reference to evidence a more general claim (Casella and Berger 2024) .

Markdown Syntax:

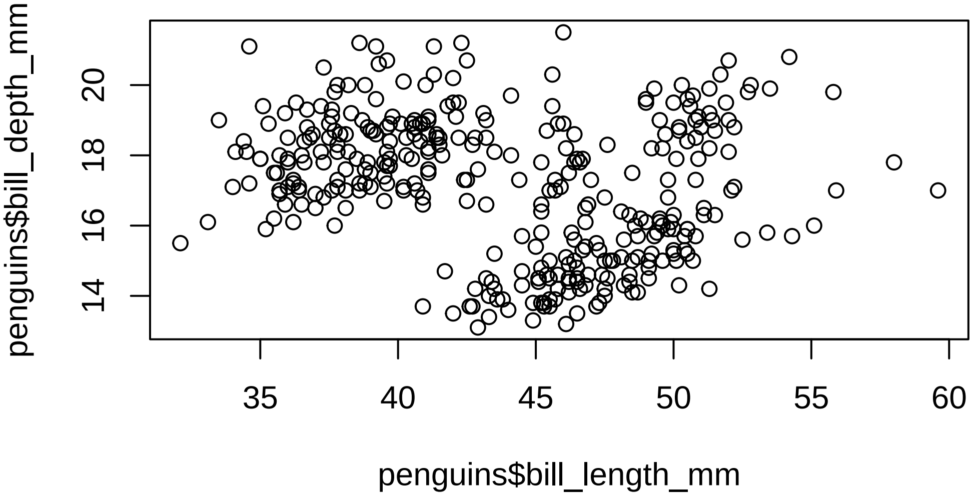
- `@box1964analysis` gives Box and Cox (1964);
- `[@casella2024statistical]` gives (Casella and Berger 2024).

Further information in the citation documentation.

Refining Your Figures

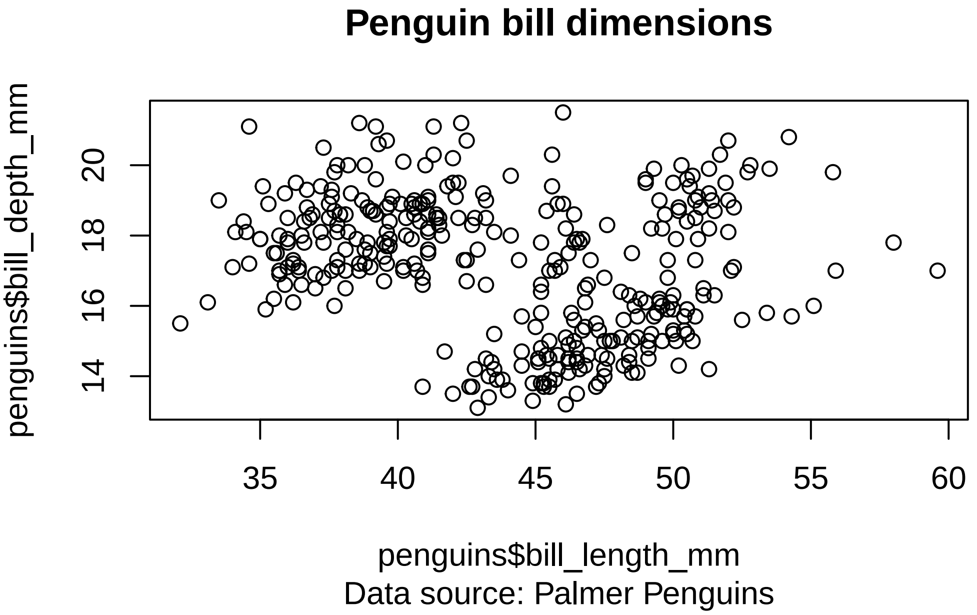
A basic plot

```
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm)
```



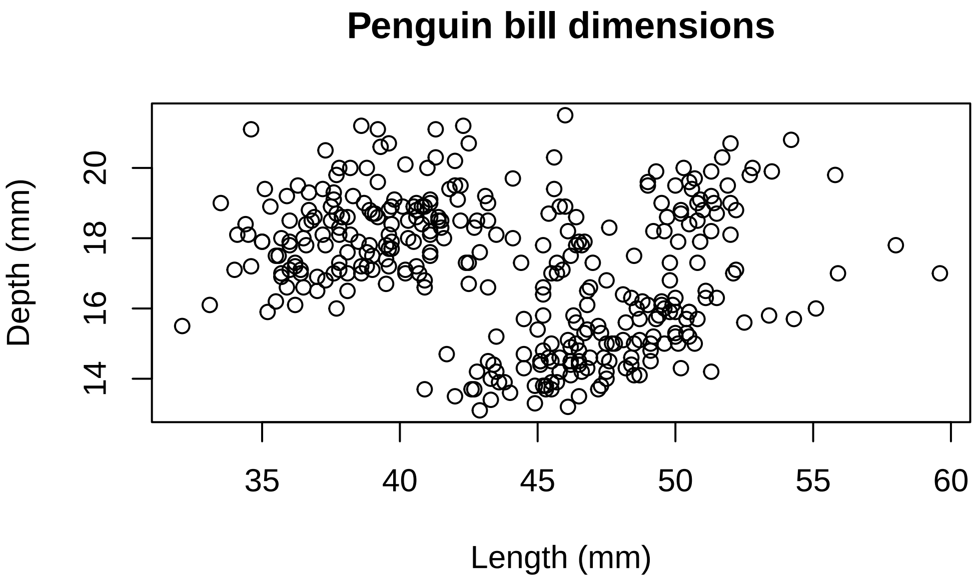
Titles and Subtitles

```
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  main = "Penguin bill dimensions",
  sub = "Data source: Palmer Penguins")
```



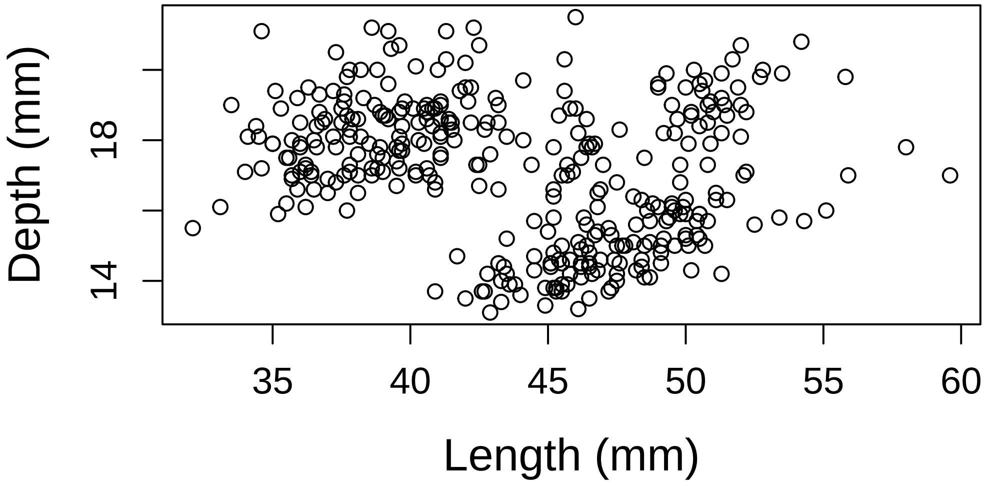
Axis Labels

```
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  main = "Penguin bill dimensions",
  xlab = "Length (mm)",
  ylab = "Depth (mm)")
```



Resizing Axis Labels

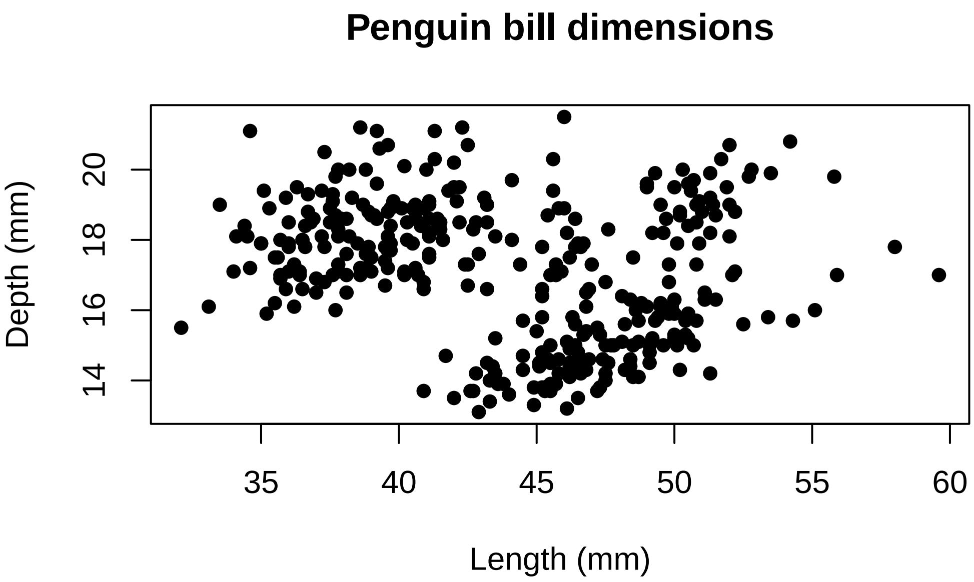
```r
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  xlab = "Length (mm)",
  ylab = "Depth (mm)",
  cex.axis = 1.2, # relative scaling of axis values
  cex.lab = 1.4)  # relative scaling of axis labels
```



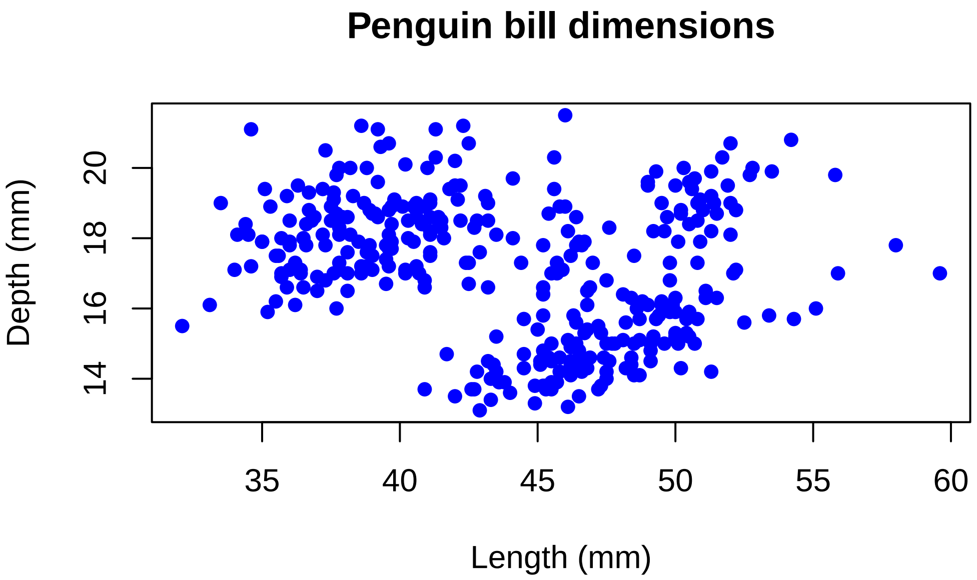
Visual Grammar

Plotting Chatacters

```
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  main = "Penguin bill dimensions",
  xlab = "Length (mm)",
  ylab = "Depth (mm)",
  pch = 16)
```

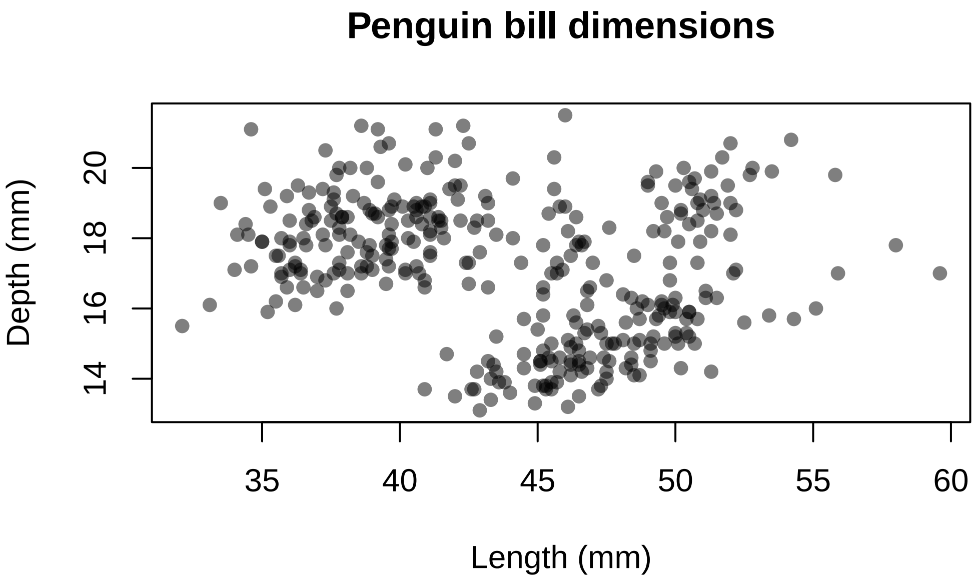


```
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  main = "Penguin bill dimensions",
  xlab = "Length (mm)",
  ylab = "Depth (mm)",
  pch = 16,
  col = "blue")
```



Over-plotting happens when one point or line in a figure obscures another. We can resolve this issue by picking a transparent colour using `rgb()`.

```
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  main = "Penguin bill dimensions",
  xlab = "Length (mm)",
  ylab = "Depth (mm)",
  pch = 16,
  col = rgb(0,0,0,0.5))
```



We could also use colour to encode another aspect of our data visually.

```
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  main = "Penguin bill dimensions",
  xlab = "Length (mm)",
  ylab = "Depth (mm)",
  pch = 16,
  col = penguins$species)
```

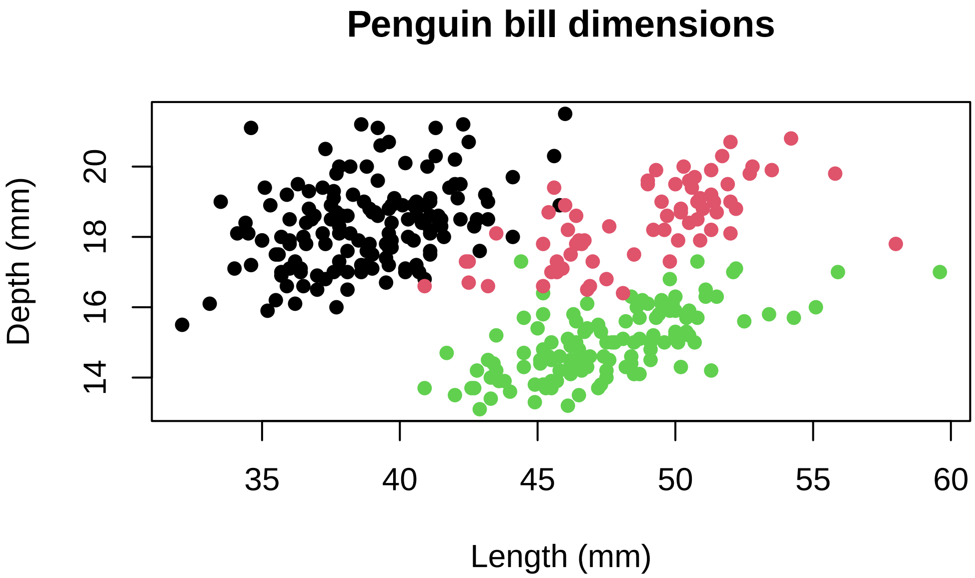


Figure 2: Penguin bill dimensions by species: Adelie (black), Chinstrap (red) and Gentoo (green).

To make the plot more accessible, we could also alter the plotting character for each species.

```r
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  main = "Penguin bill dimensions",
  xlab = "Length (mm)",
  ylab = "Depth (mm)",
  pch = 15 + as.numeric(penguins$species),
  col = penguins$species)
```

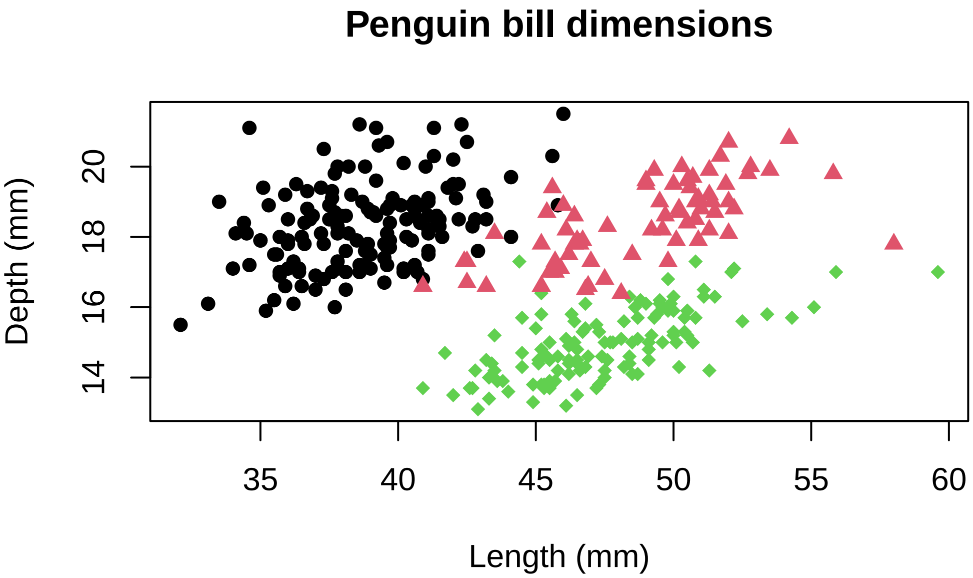


Figure 3: Penguin bill dimensions by species: Adelie (black circles), Chinstrap (red triangles) and Gentoo (green diamonds).

If we are plotting lines instead, we can also change the line width and type.

```r
time <- seq(from = 0, to = 100, by = 0.01)
displacement <- 10 + 5 * sin(time)

plot(
  x = time,
  y = displacement,
  type = "l",
  bty = "n",
  col = "blue",
  lwd = 2, # line width
  lty = 2) # dashed line
```

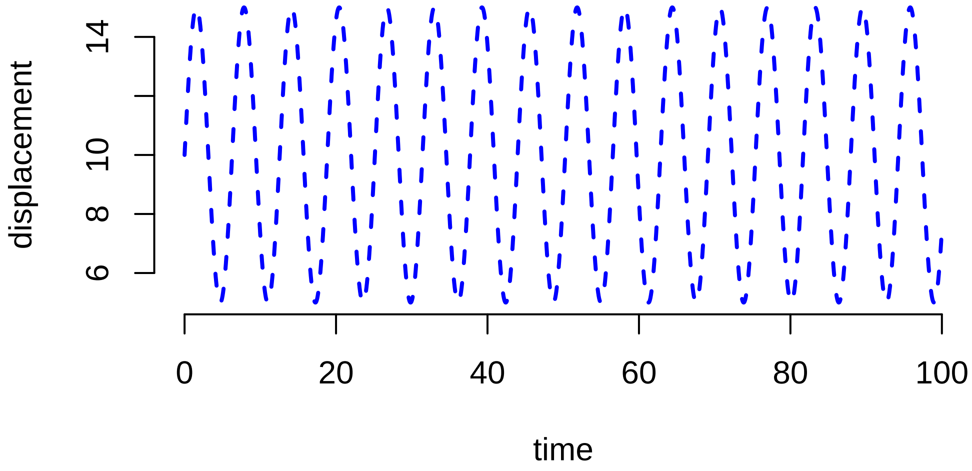
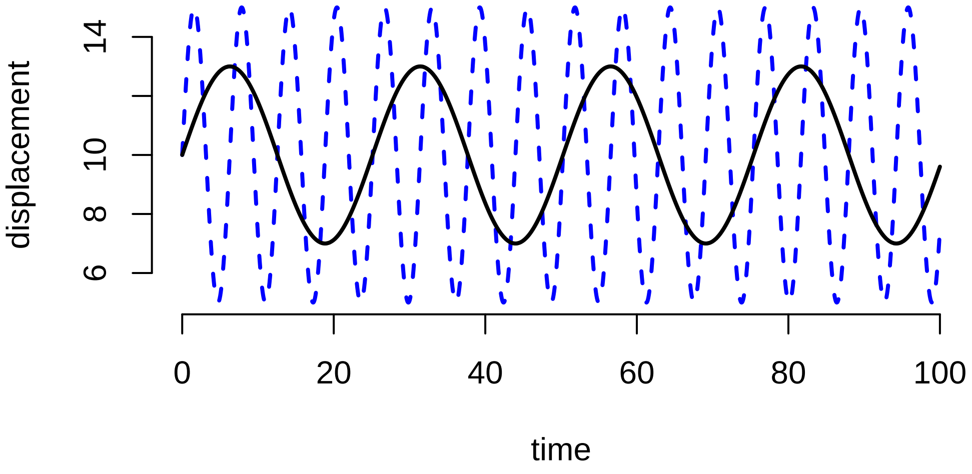


Figure 4: Displacement of an undamped simple harmonic oscillator.

Layering your plots

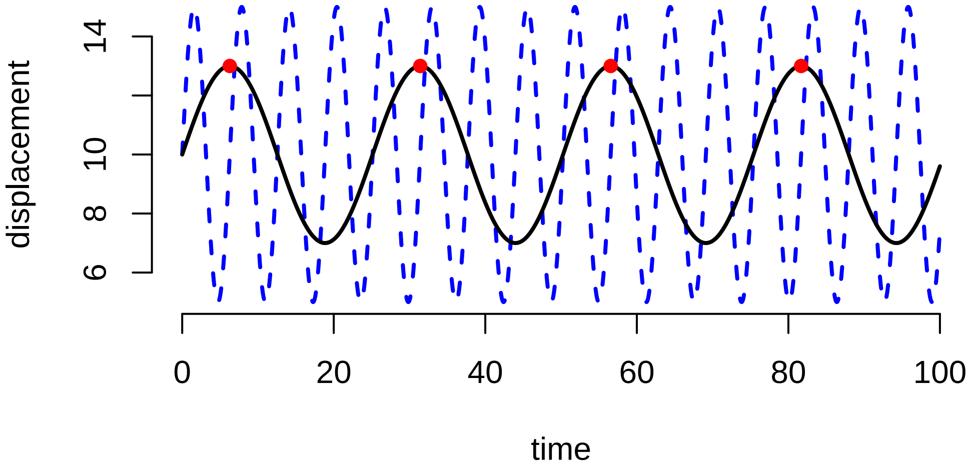
We can overlay a line on our previous plot using `lines()`.

```
plot(
  x = time,
  y = displacement,
  type = "l",
  bty = "n",
  col = "blue",
  lwd = 2, # line width
  lty = 2) # dashed line
spring_2 = 10 + 3 * sin(0.25 * time)
lines(x = time, y = spring_2, lwd = 2)
```



Similarly, we can add one or more points to the plot by using `points()`.

```
plot(
  x = time,
  y = displacement,
  type = "l",
  bty = "n",
  col = "blue",
  lwd = 2,
  lty = 2)
lines(x = time, y = spring_2, lwd = 2)
points(x = c(2, 10, 18, 26) * pi, y = rep(13, 4), pch = 16, col = "red")
```



Finally, we might want to add a legend layer to our earlier scatterplot in Figure 3.

```
plot(
  x = penguins$bill_length_mm,
  y = penguins$bill_depth_mm,
  col = penguins$species,
  pch = 15 + as.numeric(penguins$species),
  xlim = c(29,70), # extend plotting area to make room for legend
  ylim = c(12,23),
  main = "Penguin bill dimensions",
  xlab = "Length (mm)",
  ylab = "Depth (mm)",
  bty = "n")
legend(
  "bottomright",
  title = "Species",
  legend = c("Adelie", "Chinstrap", "Gentoo"),
  pch = 16:18,
  col = 1:3,
  bty = "n")
```

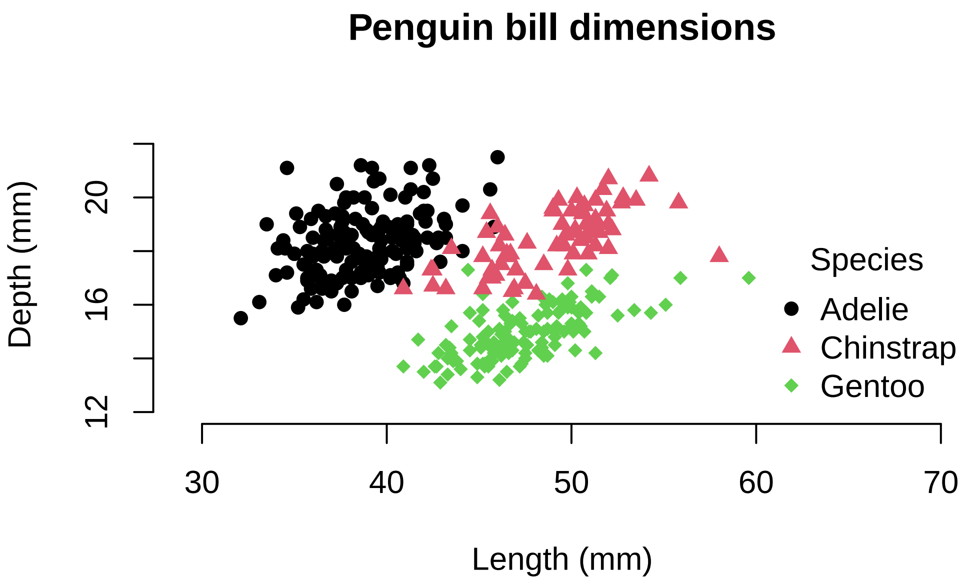


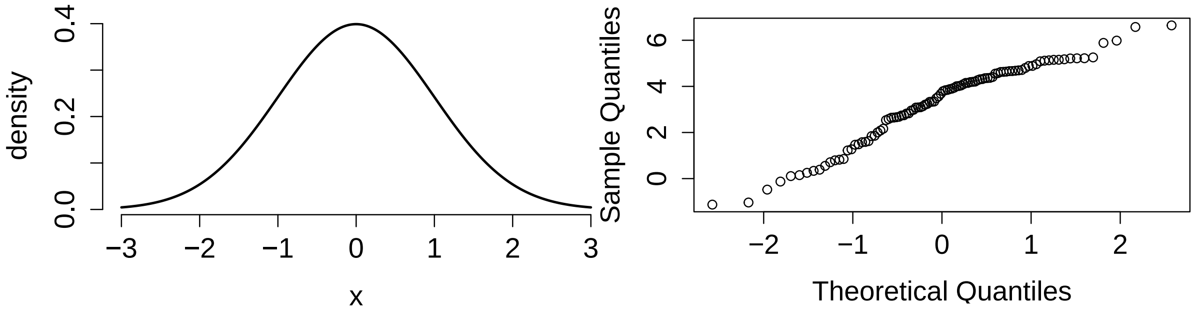
Figure 5: Penguin bill dimensions by species.

Combining your plots

~~~
```{r}
#| layout-ncol: 2
#| label: fig-example-plots
#| fig-cap: "Two example plots relating to the Gaussian distribution."
#| fig-subcap:
#|   - "Gaussian distribution"
#|   - "Gaussian QQ-plot"

x <- seq(from = -3, to = 3,by = 0.01)
density <- dnorm(x, mean = 0, sd = 1)

plot(
  x = x,
  y = density,
  bty = "n",
  cex.axis = 1.4,
  cex.lab = 1.4,
  type = "l",
  lwd = 2)

y <- rnorm(n = 100, mean = 3, sd = 2)
qqnorm(y, cex.axis = 1.4, cex.lab = 1.4, main = "")
```
~~~

```r
x <- seq(from = -3, to = 3,by = 0.01)
density <- dnorm(x, mean = 0, sd = 1)

plot(
  x = x,
  y = density,
  bty = "n",
  cex.axis = 1.4,
  cex.lab = 1.4,
  type = "l",
  lwd = 2)

y <- rnorm(n = 100, mean = 3, sd = 2)
qqnorm(y, cex.axis = 1.4, cex.lab = 1.4, main = "")
```

Figure 6 shows how we can create multiple plots within a code block and then have these appear



(a) Gaussian distribution

(b) Gaussian QQ-plot

Figure 6: Two example plots relating to the Gaussian distribution.

within the same figure as sub-figures. A probability density function is shown in Figure 6a, while a quantile-quantile plot is shown in Figure 6b.

What goes where? Tiles, Captions and Descriptions

Aspect	Plot Title	Figure Caption
Location	Inside or above the plot	Outside the plot
Purpose	Quick context for the plot	Formal description with interpretive context
Referencing	Rarely referenced in the text	Commonly referenced (e.g., “see Figure 1”)

Where a figure or table is mentioned in the main text you should explain *how* it supports any claims that you are making.

> BAD: Figure 3 demonstrates Simpson’s paradox.
>
> GOOD: Figure 3 shows that while there appears to be a positive association between bill length and depth *within* each species, this association disappears or perhaps reverses when considering all species together. This is an example of Simpson’s paradox [simpson1951contingency].

Useful Resources

- Imperial Coursework Report Template
- Quarto documentation
- Make a reprex, please
- Telling Stories With Data
  - 3 - Reproducilbe workflows
- R style guides Google, Tidyverse.

- Effective Data Science
  - 1 - Organising your work
  - 8 - Exploratory data analysis
  - 9 - Data visualisation
- The TidyTuesday Cookbook

Review

1. use R code within Quarto for single-document literate programming tasks
2. critique and refine default R outputs (such as plots and tables) for reports
3. understand how and when to show code within reports
4. use style guides and directory templates to provide a consistent structure across projects

By default, Pandoc will automatically generate a list of works cited and place it in the document if the style calls for it. It will be placed in a div with the id refs if one exists:

References

Box, George EP, and David R Cox. 1964. "An Analysis of Transformations." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 26 (2): 211–43.

Casella, George, and Roger Berger. 2024. *Statistical Inference*. Chapman; Hall/CRC.