Lecture 03: Exploring and visualising data

ENVX1002 Statistics in Life and Environmental Sciences

Januar Harianto

The University of Sydney

Mar 2026

Learning outcomes

After this week, you will be able to:

1. Understand the importance of data exploration before analysis
2. Apply exploratory functions to explore and summarise datasets
3. Identify different data types and structures in datasets
4. Select appropriate visualization types based on data characteristics
5. Understand the Grammar of Graphics approach to data visualisation
6. Create and customise visualisations in R using the ggplot2 package
7. Build plots layer-by-layer using the ggplot2 framework
8. Interpret distributions, including skewness, kurtosis, and outliers

Quick checklist

By now you should have...

- ⍰ Installed **R** and **RStudio**
- ⍰ Completed Lecture 2 content and read the ENVX1002 R guide
- ⍰ A basic understanding of measures of central tendency and spread
- ⍰ A basic understanding of what a `function(argument = value)` is in R
- ⍰ Rendered a few Quarto documents in RStudio

Core concepts

Data exploration

Why explore data before analysis?

- Identify patterns, outliers, and relationships
- Detect data quality issues
- Guide selection of appropriate statistical methods
- Avoid incorrect conclusions from flawed data

The data exploration workflow

1. Understand data structure and types
2. Examine distributions and summary statistics
3. Visualise relationships between variables
4. Identify patterns and anomalies

Types of data: recap from Week 1

Data in R can be broadly categorized as either **categorical** or **continuous**.

- Different data types require different analysis approaches
- Understanding data types helps select appropriate visualisations
- **R stores different data types in specific formats** (which is why we need to know what they are when we import data!)

Categorical data

- **Nominal**: no natural order, e.g.
  - Species (dog, cat, fish)
  - Hair colour (black, brown, blonde)
  - Blood type (A, B, AB, O)
- **Ordinal**: natural order exists, e.g.
  - Education (primary, secondary, tertiary)
  - Pain scale (mild, moderate, severe)
  - T-shirt sizes (S, M, L, XL)

Continuous data

- **Interval**: equal intervals, no true zero, e.g.
  - Temperature in °C (0°C isn't "no temperature")
  - Calendar dates
  - pH scale
- **Ratio**: equal intervals with true zero, e.g.
  - Height (0 cm = no height)
  - Weight (0 kg = no weight)
  - Age (0 years = birth)

Different types of data are distributed differently

Understanding how data is distributed is crucial for selecting appropriate ways to explain it to others.

Normal distribution: Introduction

- Bell-shaped, symmetric curve
- Defined by mean (μ) and standard deviation (σ)
- Many natural phenomena follow this distribution
  - Heights of individuals in a population
  - Measurement errors
  - Many physiological traits

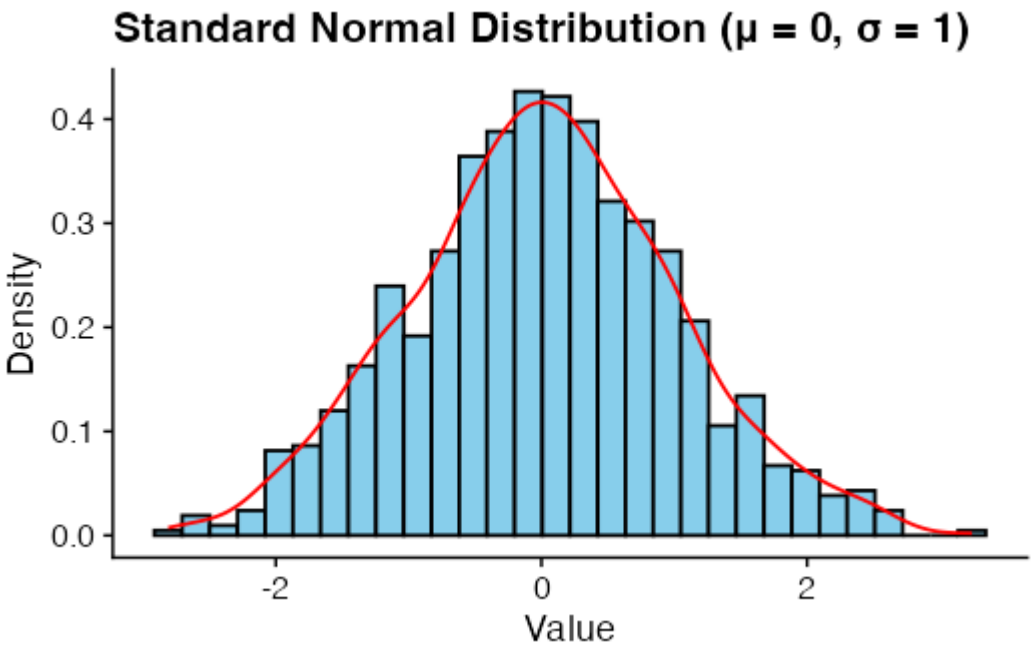
$$X \sim N(\mu, \sigma^2)$$

> The random variable X follows a normal distribution with mean μ and variance σ^2

What does a normal distribution look like?

```r
# Generate normal distribution data
set.seed(123)
normal_data ← rnorm(1000, mean = 0, sd = 1)

# Plot normal distribution
ggplot(data.frame(x = normal_data), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 30,
                 fill = "skyblue",
                 colour = "black") +
  geom_density(colour = "red") +
  labs(title = "Standard Normal Distribution (μ = 0, σ = 1)",
       x = "Value",
       y = "Density")
```


Standard Normal Distribution (μ = 0, σ = 1)
Density
0.0
0.1
0.2
0.3
0.4
-2
0
2
Value

Properties of normal distribution: The empirical rule

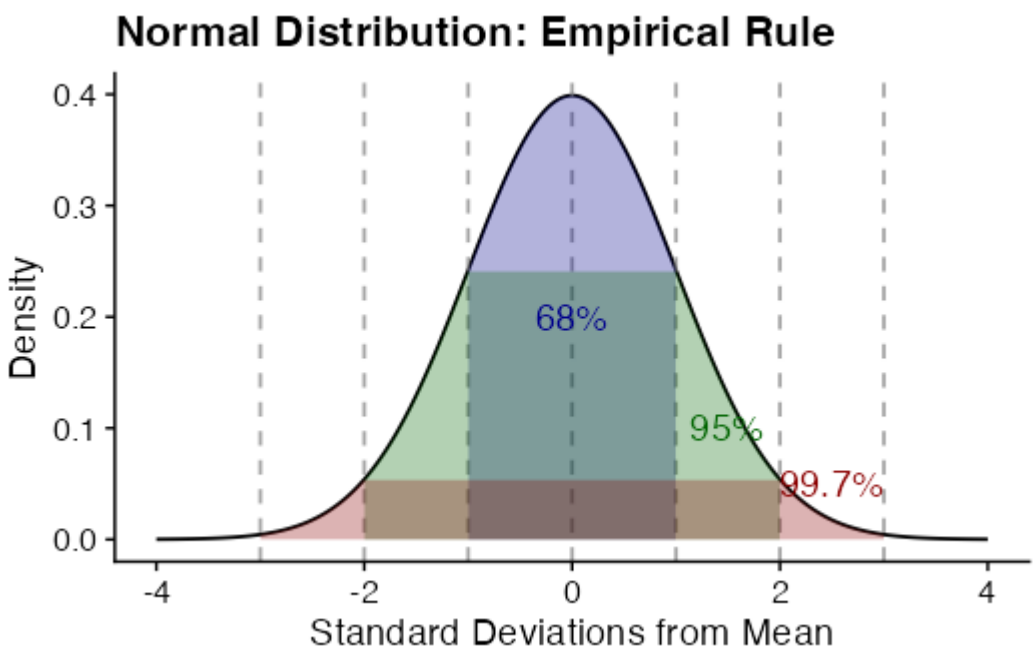
- Mean = median = mode
- ~68% of data within 1σ of mean
- ~95% of data within 2σ of mean
- ~99.7% of data within 3σ of mean

The empirical rule visualised

```
# Create a standard normal distribution
x ← seq(-4, 4, length.out = 1000)
y ← dnorm(x)
df ← data.frame(x = x, y = y)

# Plot with empirical rule highlighted
ggplot(df, aes(x = x, y = y)) +
  geom_line() +
  # Add vertical reference lines at standard deviations
  geom_vline(xintercept = c(-3, -2, -1, 0, 1, 2, 3), linetype = "dashed", colour = "gray50", alpha
= 0.7) +
  geom_area(data = subset(df, x ⩾ -1 & x ⩽ 1), fill = "darkblue", alpha = 0.3) +
  geom_area(data = subset(df, (x ⩾ -2 & x < -1) | (x > 1 & x ⩽ 2)), fill = "darkgreen", alpha =
0.3) +
  geom_area(data = subset(df, (x ⩾ -3 & x < -2) | (x > 2 & x ⩽ 3)), fill = "darkred", alpha =
0.3) +
  annotate("text", x = 0, y = 0.2, label = "68%", colour = "darkblue") +
  annotate("text", x = 1.5, y = 0.1, label = "95%", colour = "darkgreen") +
  annotate("text", x = 2.5, y = 0.05, label = "99.7%", colour = "darkred") +
```

```
  labs(title = "Normal Distribution: Empirical Rule",
       x = "Standard Deviations from Mean",
       y = "Density")
```



Why normal distributions matter in data exploration

When exploring data, understanding distributions helps you:

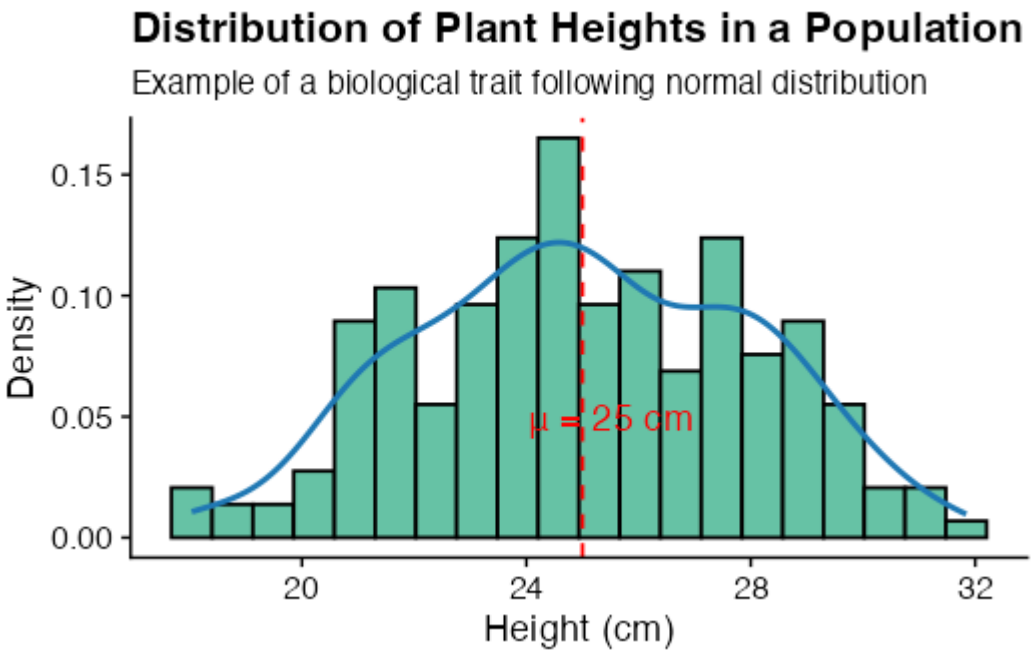
1. **Identify patterns and anomalies**
   - Is your data normally distributed as expected?
   - Are there unexpected skews or outliers?
2. **Choose appropriate analysis methods**
   - Many statistical tests assume normality
   - Non-normal data may require different approaches
3. **Interpret results correctly**
   - Context for understanding how unusual a value is
   - Framework for making statistical inferences

Example

Many biological traits follow normal distributions. For example, plant heights within a species:

```
# Simulate plant height data
set.seed(456)
plant_heights ← rnorm(200, mean = 25, sd = 3)  # Heights in cm

# Plot the distribution
ggplot(data.frame(height = plant_heights), aes(x = height)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 20,
                 fill = "#66c2a5",
                 colour = "black") +
  geom_density(colour = "#1f78b4", linewidth = 1) +
  geom_vline(xintercept = 25, linetype = "dashed", colour = "red") +
  annotate("text", x = 25.5, y = 0.05, label = "μ = 25 cm", colour = "red") +
  labs(title = "Distribution of Plant Heights in a Population",
       subtitle = "Example of a biological trait following normal distribution",
       x = "Height (cm)",
       y = "Density")
```



This example shows how plant heights cluster around the mean (25 cm) following a normal distribution pattern. This helps researchers identify outliers, establish experimental categories, and detect environmental effects on growth patterns.

Skewness

What is skewness?

- Measure of asymmetry in a distribution
- Indicates which side of the distribution has a longer tail
- Important for selecting appropriate statistical tests

Positive skew (right-skewed)

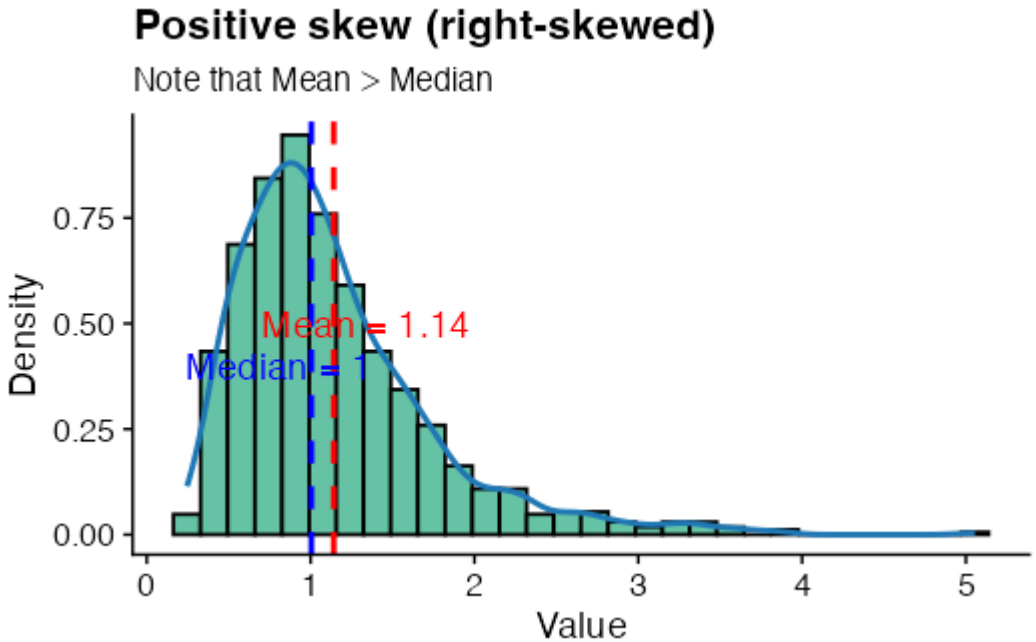
- Long tail on right side
- Mean > median

```
# Generate positive skewed distribution
set.seed(123)
right_skewed ← exp(rnorm(1000, 0, 0.5))

# Calculate statistics
mean_val ← mean(right_skewed)
median_val ← median(right_skewed)

# Plot positive skewed distribution
ggplot(data.frame(x = right_skewed), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 30,
                 fill = "#66c2a5", # colourblind-friendly green
                 colour = "black") +
  geom_density(colour = "#1f78b4", linewidth = 1) +
  # Add vertical lines for mean and median
```

```
  geom_vline(xintercept = mean_val, colour = "red", linetype = "dashed", linewidth = 1) +
  geom_vline(xintercept = median_val, colour = "blue", linetype = "dashed", linewidth = 1) +
  # Add annotations
  annotate("text", x = mean_val + 0.2, y = 0.5, label = paste("Mean =", round(mean_val, 2)), colour
= "red") +
  annotate("text", x = median_val - 0.2, y = 0.4, label = paste("Median =", round(median_val, 2)),
colour = "blue") +
  labs(title = "Positive skew (right-skewed)",
       subtitle = "Note that Mean > Median",
       x = "Value",
       y = "Density")
```


Positive skew (right-skewed)
Note that Mean > Median
Mean = 1.14
Median = 1
Density
0.00
0.25
0.50
0.75
0
1
2
3
4
5
Value

Negative skew (left-skewed)

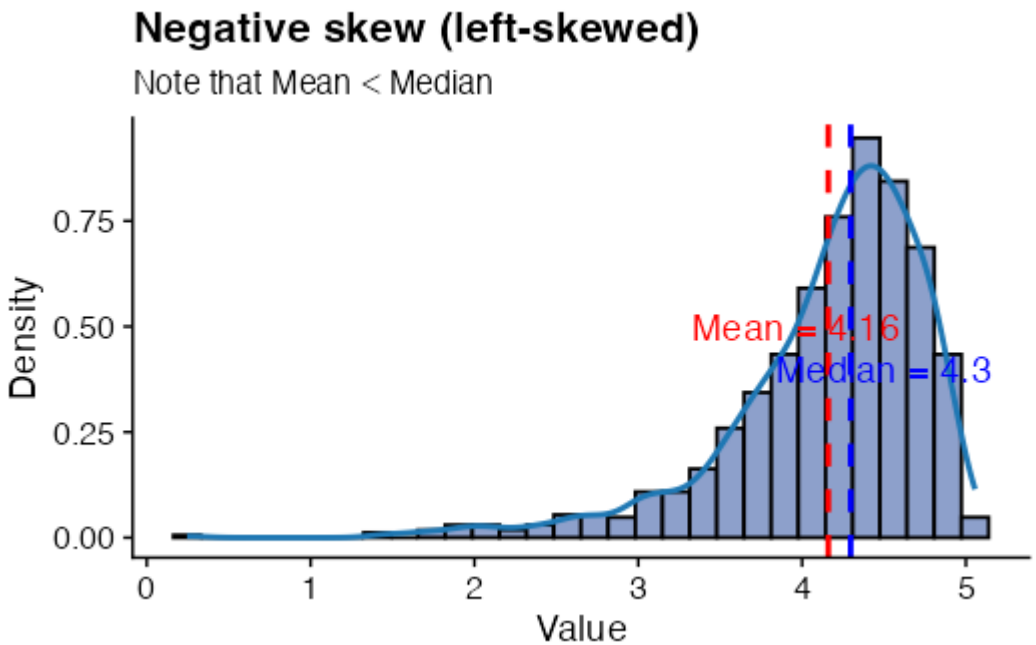
- Long tail on left side
- Mean < median

```
# Generate skewed distributions
set.seed(123)
left_skewed ← max(right_skewed) - right_skewed + min(right_skewed)

# Calculate statistics
mean_val ← mean(left_skewed)
median_val ← median(left_skewed)

# Plot negative skewed distribution
ggplot(data.frame(x = left_skewed), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)),
                 bins = 30,
                 fill = "#8da0cb", # colourblind-friendly blue
                 colour = "black") +
  geom_density(colour = "#1f78b4", linewidth = 1) +
  # Add vertical lines for mean and median
```

```
  geom_vline(xintercept = mean_val, colour = "red", linetype = "dashed", linewidth = 1) +
  geom_vline(xintercept = median_val, colour = "blue", linetype = "dashed", linewidth = 1) +
  # Add annotations
  annotate("text", x = mean_val - 0.2, y = 0.5, label = paste("Mean =", round(mean_val, 2)), colour
= "red") +
  annotate("text", x = median_val + 0.2, y = 0.4, label = paste("Median =", round(median_val, 2)),
colour = "blue") +
  labs(title = "Negative skew (left-skewed)",
       subtitle = "Note that Mean < Median",
       x = "Value",
       y = "Density")
```


Negative skew (left-skewed)
Note that Mean < Median
Density
0.75
0.50
0.25
0.00
Mean = 4.16
Median = 4.3
0
1
2
3
4
5
Value

Kurtosis

- Measure of “tailedness” of a distribution
- Describes the shape of a distribution’s tails relative to its overall shape
- Affects the choice of statistical methods

Types of kurtosis

- **Mesokurtic**: Normal distribution (kurtosis = 3)
- **Leptokurtic**: Sharper peak, heavier tails (kurtosis > 3)
- **Platykurtic**: Flatter peak, thinner tails (kurtosis < 3)

Visualising kurtosis

```r
# Generate distributions with different kurtosis
set.seed(123)
normal ← rnorm(1000, 0, 1)  # Mesokurtic
leptokurtic ← rt(1000, df = 5)  # t-distribution with 5 df is leptokurtic
platykurtic ← runif(1000, -3, 3)  # Uniform distribution is platykurtic

# Calculate kurtosis values (using e1071 package)
library(e1071)
```

```
Attaching package: 'e1071'
```

```
The following object is masked from 'package:ggplot2':

    element
```

```r
k_normal ← kurtosis(normal)
k_lepto ← kurtosis(leptokurtic)
k_platy ← kurtosis(platykurtic)

# Plot distributions
p1 ← ggplot(data.frame(x = normal), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)), bins = 30,
                 fill = "#a6cee3", # colourblind-friendly blue
                 colour = "black") +
  geom_density(colour = "#1f78b4", linewidth = 1) + # Darker blue
  annotate("text", x = 2, y = 0.3,
           label = paste("Kurtosis =", round(k_normal, 2)),
           colour = "#1f78b4") +
  labs(title = "Mesokurtic (normal)",
       subtitle = "Normal distribution with balanced tails",
       x = "Value",
       y = "Density")

p2 ← ggplot(data.frame(x = leptokurtic), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)), bins = 30,
                 fill = "#fb9a99", # colourblind-friendly pink
```

```
                 colour = "black") +
  geom_density(colour = "#e31a1c", linewidth = 1) + # Darker red
  annotate("text", x = 2, y = 0.3,
           label = paste("Kurtosis =", round(k_lepto, 2)),
           colour = "#e31a1c") +
  labs(title = "Leptokurtic (heavy-tailed)",
       subtitle = "Sharper peak, heavier tails",
       x = "Value",
       y = "Density")

p3 ← ggplot(data.frame(x = platykurtic), aes(x = x)) +
  geom_histogram(aes(y = after_stat(density)), bins = 30,
                 fill = "#b2df8a", # colourblind-friendly green
                 colour = "black") +
  geom_density(colour = "#33a02c", linewidth = 1) + # Darker green
  annotate("text", x = 0, y = 0.15,
           label = paste("Kurtosis =", round(k_platy, 2)),
           colour = "#33a02c") +
  labs(title = "Platykurtic (light-tailed)",
       subtitle = "Flatter peak, thinner tails",
       x = "Value",
```

```
       y = "Density")

# Display plots vertically
library(patchwork)
```

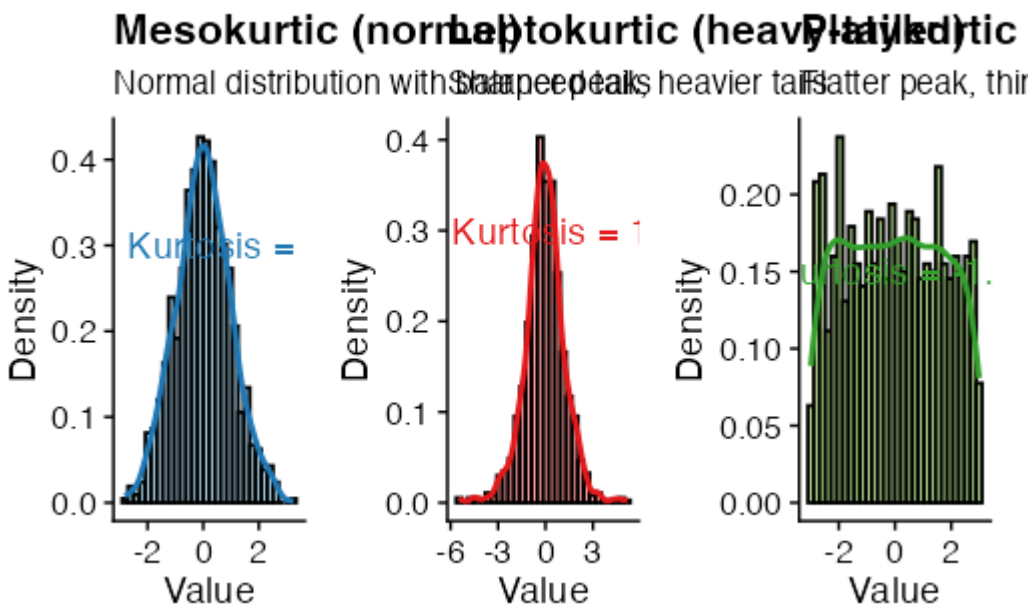
```

Attaching package: 'patchwork'
```

```
The following object is masked from 'package:cowplot':

    align_plots
```

```
p1 + p2 + p3
```


Mesokurtic (normal)
Normal distribution with
Density
Value
Kurtosis =
Kurtosis = 1
Value
Value

Exploring data in R

Data structures in R

Data is stored in R in various structures, each with specific purposes:

Vectors: 1-dimensional collection of elements

```
# Vector - 1-dimensional collection of elements
heights ← c(1.65, 1.70, 1.75, 1.80, 1.85)
heights
```

```
[1] 1.65 1.70 1.75 1.80 1.85
```

Data frames: 2-dimensional tables with rows and columns

```
# Data frame - 2-dimensional table
df ← data.frame(
  species = c("A", "B", "C", "A", "B"),
  height = c(1.65, 1.70, 1.75, 1.80, 1.85),
  weight = c(60, 65, 70, 75, 80)
)
df
```

```
  species height weight
1       A   1.65     60
2       B   1.70     65
3       C   1.75     70
4       A   1.80     75
5       B   1.85     80
```

Other data structures include lists, matrices, arrays, and factors, but these are less common at your level.

Common functions

Use these essential functions to understand your data structure and summary statistics:

```
# Core function 1: Structure overview
str(df)
```

```
'data.frame':   5 obs. of  3 variables:
 $ species: chr  "A" "B" "C" "A" ...
 $ height : num  1.65 1.7 1.75 1.8 1.85
 $ weight : num  60 65 70 75 80
```

```
# Core function 2: Statistical summary
summary(df)
```

```
   species              height         weight
 Length:5           Min.   :1.65   Min.   :60
 Class :character   1st Qu.:1.70   1st Qu.:65
 Mode  :character   Median :1.75   Median :70
                    Mean   :1.75   Mean   :70
```

```
                  3rd Qu.:1.80   3rd Qu.:75
                  Max.   :1.85   Max.   :80
```

The `summary()` function provides a quick overview of your data and can help identify skewness, outliers, and missing values...but it isn't always enough.

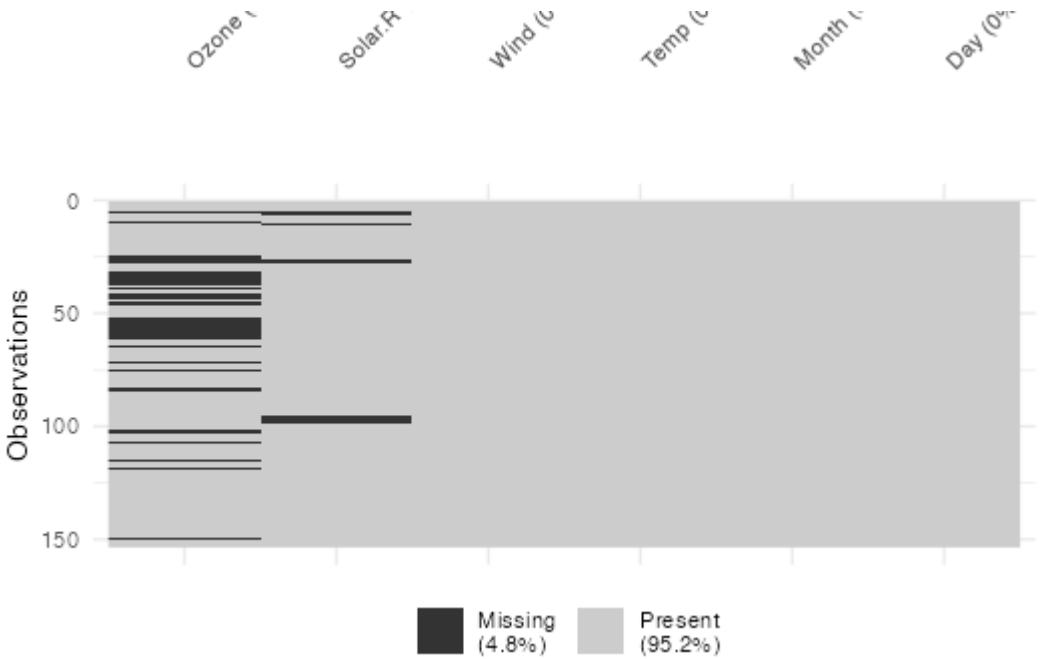
Your options are endless (almost)

There are many specialised functions for exploring different aspects of your data:

```
# Check for unique values in categorical variables
unique(df$species)
```

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

```
# Visualise missing data patterns
library(naniar)
vis_miss(airquality)
```


Ozone
Solar.R
Wind
Temp
Month
Day
0
50
100
150
Observations
Missing
(4.8%)
Present
(95.2%)

The value of data visualisation

The output of `vis_mis()` cleary demonstrates the advantage of a **visual** approach to data exploration.

Compare the visualisation to looking at the raw data or a summary of the raw data

```
airquality$Ozone
```

```
  [1]  41  36  12  18  NA  28  23  19   8  NA   7  16  11  14  18  14  34   6
 [19]  30  11   1  11   4  32  NA  NA  NA  23  45 115  37  NA  NA  NA  NA  NA
 [37]  NA  29  NA  71  39  NA  NA  23  NA  NA  21  37  20  12  13  NA  NA  NA
 [55]  NA  NA  NA  NA  NA  NA  NA 135  49  32  NA  64  40  77  97  97  85  NA
 [73]  10  27  NA   7  48  35  61  79  63  16  NA  NA  80 108  20  52  82  50
 [91]  64  59  39   9  16  78  35  66 122  89 110  NA  NA  44  28  65  NA  22
[109]  59  23  31  44  21   9  NA  45 168  73  NA  76 118  84  85  96  78  73
[127]  91  47  32  20  23  21  24  44  21  28   9  13  46  18  13  24  16  13
[145]  23  36   7  14  30  NA  14  18  20
```

```
summary(airquality$Ozone)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   1.00   18.00   31.50   42.13   63.25  168.00      37
```

Common plot types and their applications

Different types of data require different visualisation approaches. Let's explore the most common plot types and when to use them.

Histograms

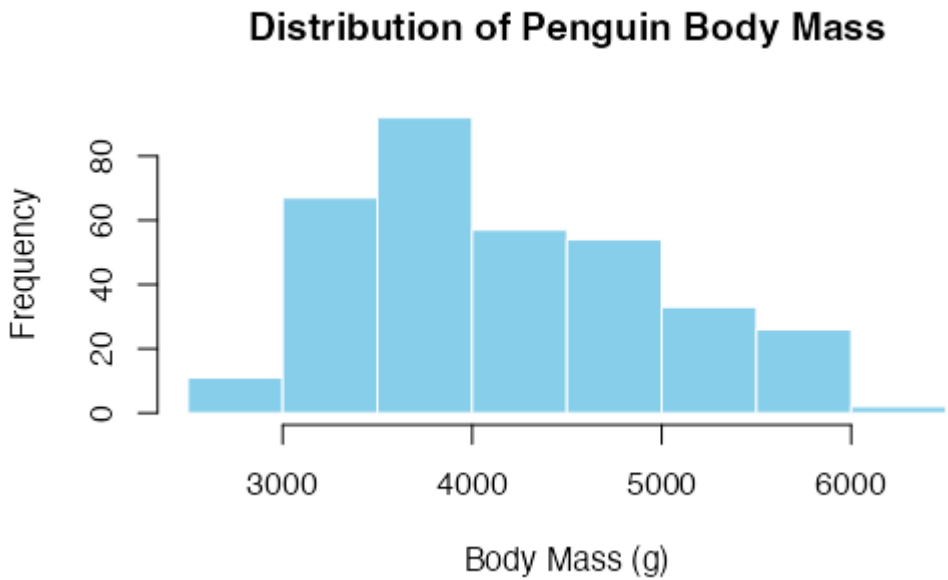
Purpose and applications:

- Visualise distribution of continuous data
- Identify central tendency, spread, outliers, and skewness
- Examine distributions of measurements in biological data

When to use:

- For continuous variables (interval or ratio data)
- When you want to understand the shape of a distribution
- Examples: heights, weights, temperatures, measurements

```r
# Example using base R with palmerpenguins data
hist(penguins$body_mass_g,
     main = "Distribution of Penguin Body Mass",
     xlab = "Body Mass (g)",
     col = "skyblue",
     border = "white")
```


Distribution of Penguin Body Mass
Frequency
0
20
40
60
80
3000
4000
5000
6000
Body Mass (g)

Bar plots

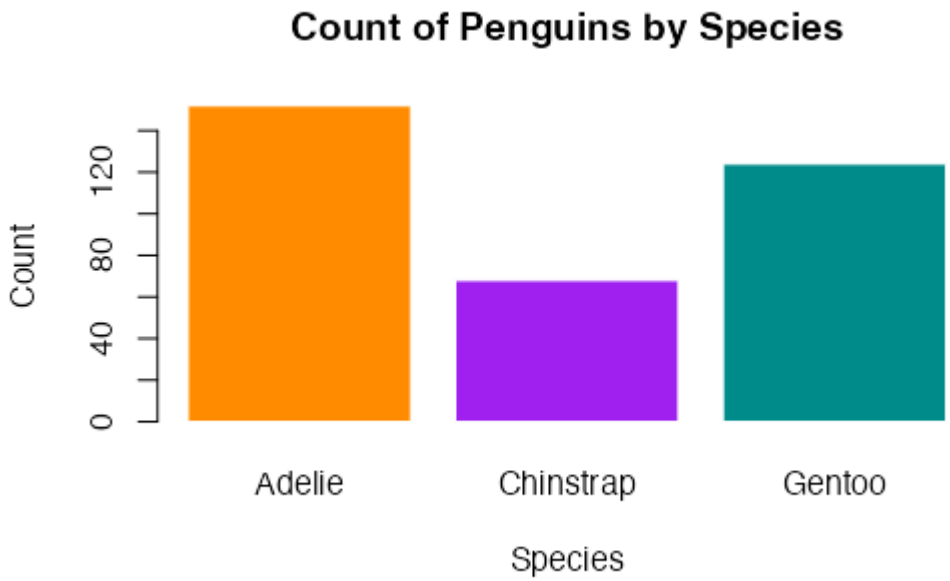
Purpose and applications:

- Compare values across categories
- Show proportions or counts in categorical data
- Visualise species abundance, treatment effects

When to use:

- For categorical variables (nominal or ordinal data)
- When comparing frequencies or counts across groups
- Examples: species counts, treatment groups, survey responses

```r
# Example using base R with palmerpenguins data
species_counts ← table(penguins$species)
barplot(species_counts,
        main = "Count of Penguins by Species",
        xlab = "Species",
        ylab = "Count",
        col = c("darkorange", "purple", "cyan4"),
        border = "white")
```


Count of Penguins by Species
Count
0
40
80
120
Adelie
Chinstrap
Gentoo
Species

Scatterplots

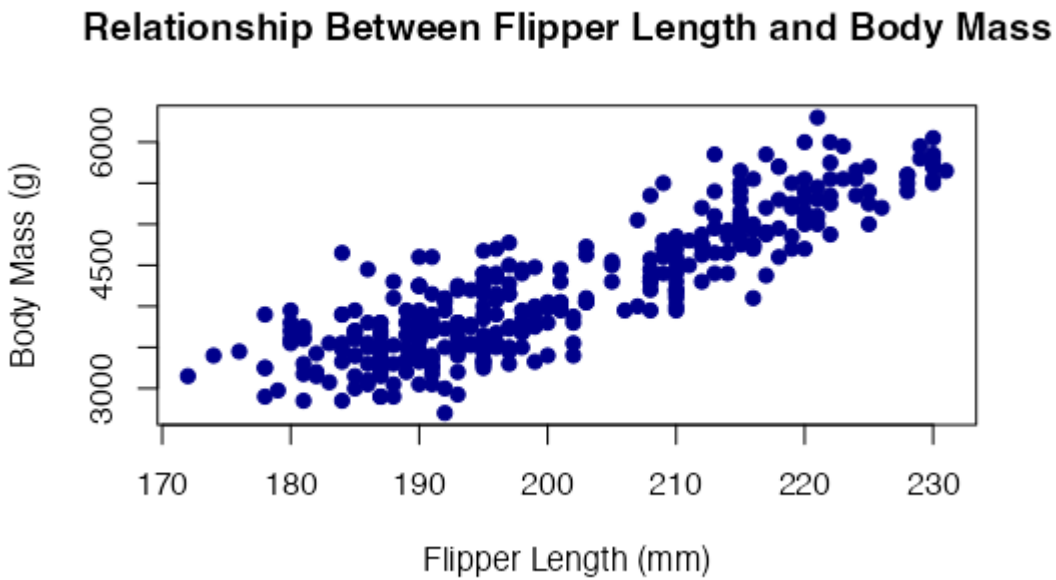
Purpose and applications:

- Examine relationships between continuous variables
- Identify correlations, patterns, and outliers
- Explore relationships between measurements

When to use:

- When examining relationships between two continuous variables
- When looking for correlations or patterns
- Examples: height vs. weight, temperature vs. growth rate

```r
# Example using base R with palmerpenguins data
# Remove NA values for this example
penguins_clean ← na.omit(penguins[, c("flipper_length_mm", "body_mass_g")])
plot(penguins_clean$flipper_length_mm, penguins_clean$body_mass_g,
     main = "Relationship Between Flipper Length and Body Mass",
     xlab = "Flipper Length (mm)",
     ylab = "Body Mass (g)",
     pch = 19,
     col = "darkblue")
```


Relationship Between Flipper Length and Body Mass
Body Mass (g)
3000
4500
6000
170
180
190
200
210
220
230
Flipper Length (mm)

Boxplots

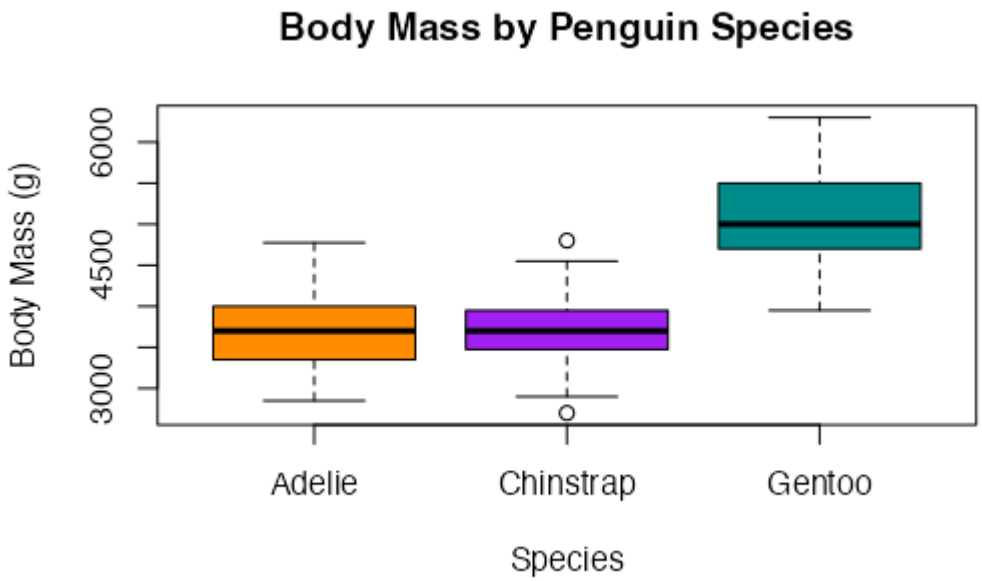
Purpose and applications:

- Compare distributions across groups
- Visualise median, quartiles, and outliers
- Compare measurements across treatments

When to use:

- When comparing a continuous variable across categorical groups
- When you need to show the spread and central tendency
- Examples: comparing heights across species, measurements across treatments

```r
# Example using base R with palmerpenguins data
boxplot(body_mass_g ~ species, data = penguins,
        main = "Body Mass by Penguin Species",
        xlab = "Species",
        ylab = "Body Mass (g)",
        col = c("darkorange", "purple", "cyan4"),
        border = "black")
```


Body Mass by Penguin Species
Body Mass (g)
3000
4500
6000
Adelie
Chinstrap
Gentoo
Species

Introduction to ggplot2

The Grammar of Graphics

ggplot2 is based on the Grammar of Graphics, a systematic approach to creating visualisations by combining different components:

1. **Data**: The dataset you want to visualise
2. **Aesthetics**: Mapping variables to visual properties (position, colour, size, etc.)
3. **Geometries**: The shapes used to represent the data (points, lines, bars, etc.)
4. **Scales**: How values are mapped to visual properties
5. **Facets**: How to split the data into subplots
6. **Coordinates**: The coordinate system to use
7. **Themes**: Visual styling of the plot

This grammar allows you to build complex visualisations layer by layer.

Why use ggplot2?

- **Consistent syntax** across different plot types
- **Layered approach** makes it easy to build complex visualisations
- **Excellent defaults** that produce publication-quality graphics
- **Highly customisable** with extensive options for fine-tuning
- **Large community** with extensive documentation and examples

Building a plot: Step 1 - Start with data

Let's build a scatterplot of penguin flipper length vs. body mass using the palmerpenguins dataset.

First, we need to load the ggplot2 package and prepare our data:

```
library(ggplot2)
# Remove missing values for this example
penguins_clean ← na.omit(penguins)

# Look at the first few rows of our data
head(penguins_clean[, c("species", "flipper_length_mm", "body_mass_g")])
```

```
# A tibble: 6 × 3
  species flipper_length_mm body_mass_g
  <fct>               <int>       <int>
1 Adelie                181        3750
2 Adelie                186        3800
3 Adelie                195        3250
4 Adelie                193        3450
```

```
5 Adelie                 190         3650
6 Adelie                 181         3625
```

Building a plot: Step 2 - Create a blank canvas

The `ggplot()` function initialises a plot with data:

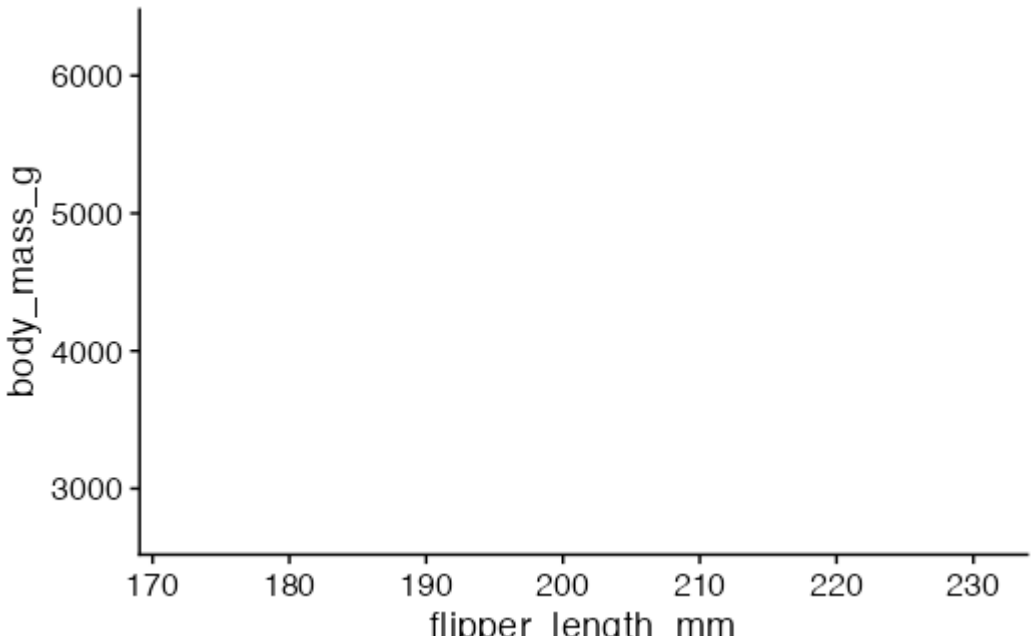
```
# Create a blank canvas with our data
p ← ggplot(penguins_clean)
p
```

This creates an empty plot. We need to add layers to visualise our data.

Building a plot: Step 3 - Add aesthetics

Aesthetics map variables in the data to visual properties:

```
# Add aesthetics mapping
p ← ggplot(penguins_clean, aes(x = flipper_length_mm, y = body_mass_g))
p
```

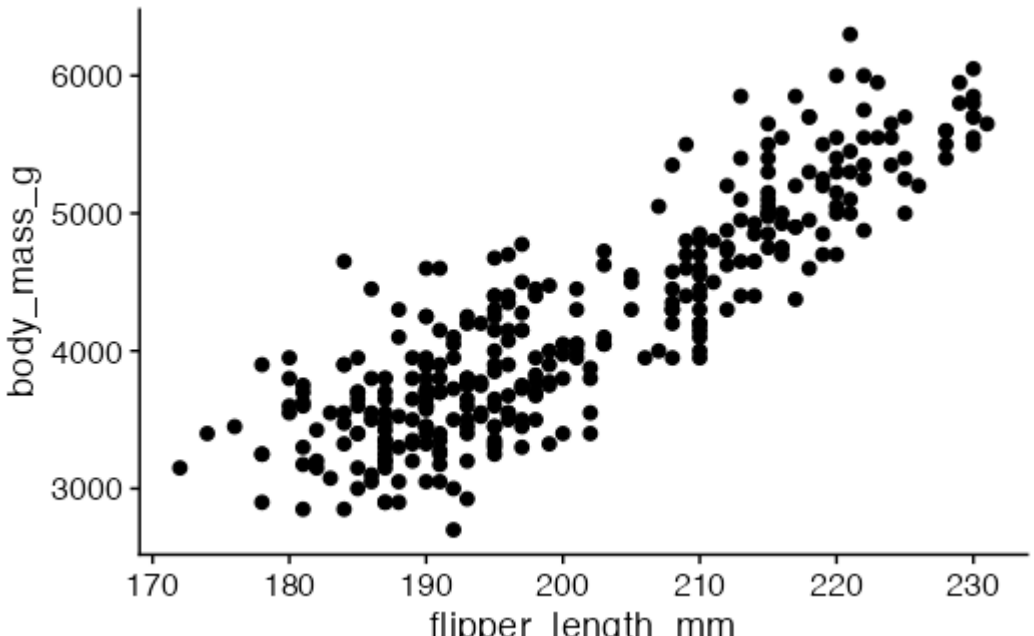


We've defined which variables go on which axes, but we still need to specify how to represent the data.

Building a plot: Step 4 - Add a geometry

Geometries define how the data is represented visually:

```
# Add points geometry
p ← ggplot(penguins_clean, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
p
```

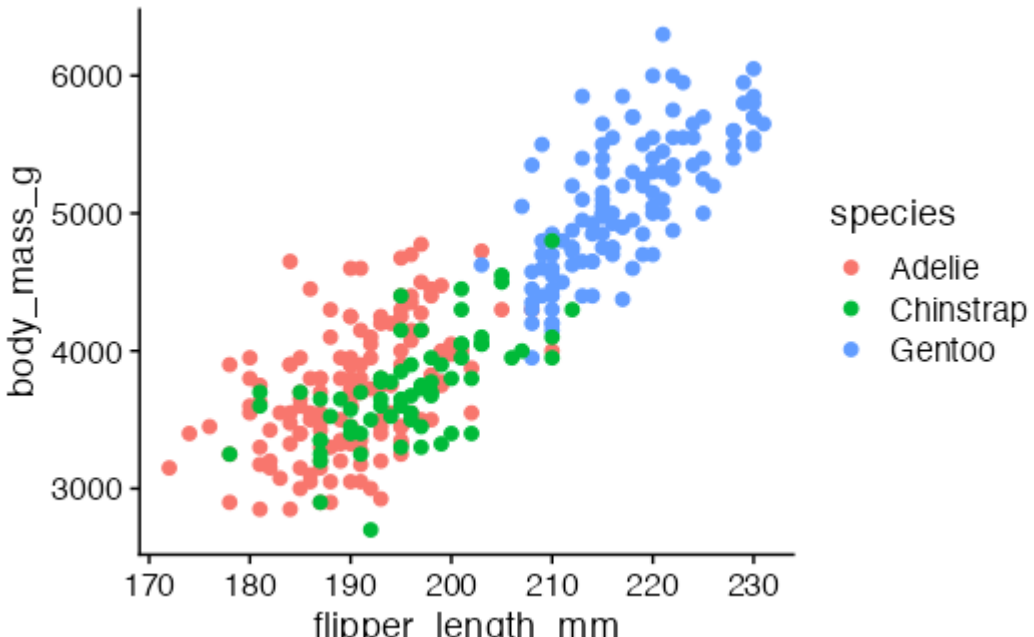


Now we can see the relationship between flipper length and body mass!

Building a plot: Step 5 - Add colour by species

We can map the species variable to the colour aesthetic:

```
# colour points by species
p ← ggplot(penguins_clean, aes(x = flipper_length_mm, y = body_mass_g, colour = species)) +
  geom_point()
p
```

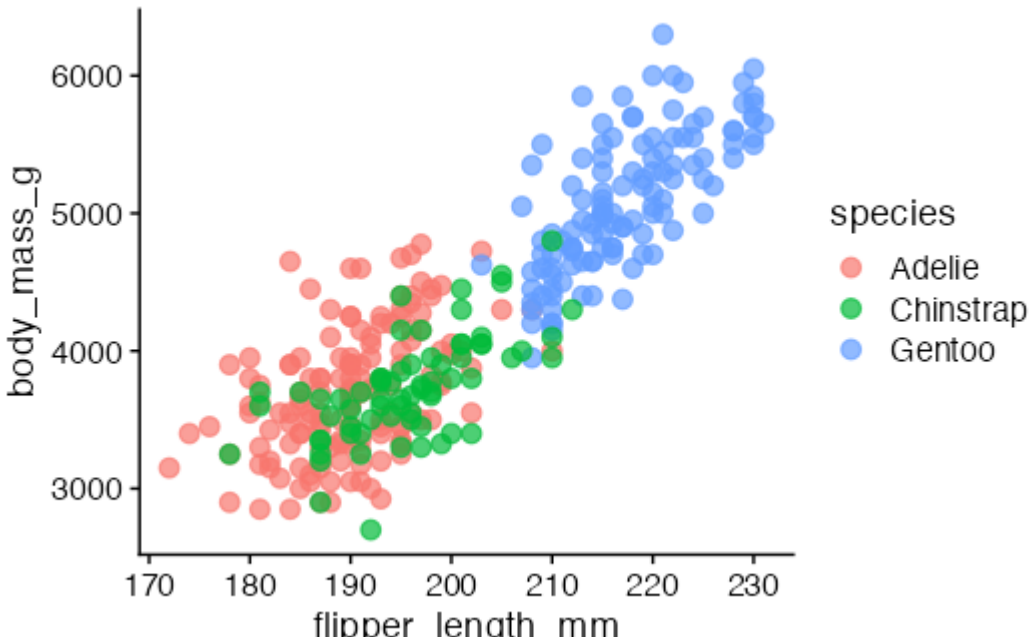


Notice how ggplot2 automatically creates a legend for the species colours.

Building a plot: Step 6 - Customise point appearance

We can adjust the size and transparency of points:

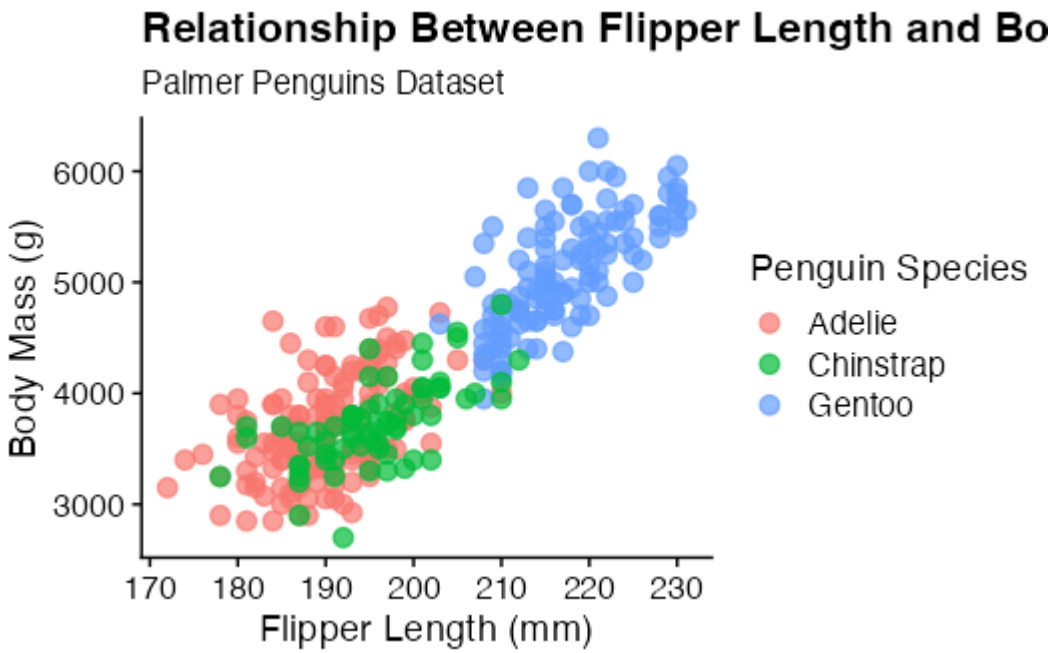
```
# Customize point appearance
p ← ggplot(penguins_clean, aes(x = flipper_length_mm, y = body_mass_g, colour = species)) +
  geom_point(size = 3, alpha = 0.7)
p
```


body_mass_g
6000
5000
4000
3000
170
180
190
200
210
220
230
flipper_length_mm
species
Adelie
Chinstrap
Gentoo

Building a plot: Step 7 - Add labels and title

Let's add informative labels and a title:

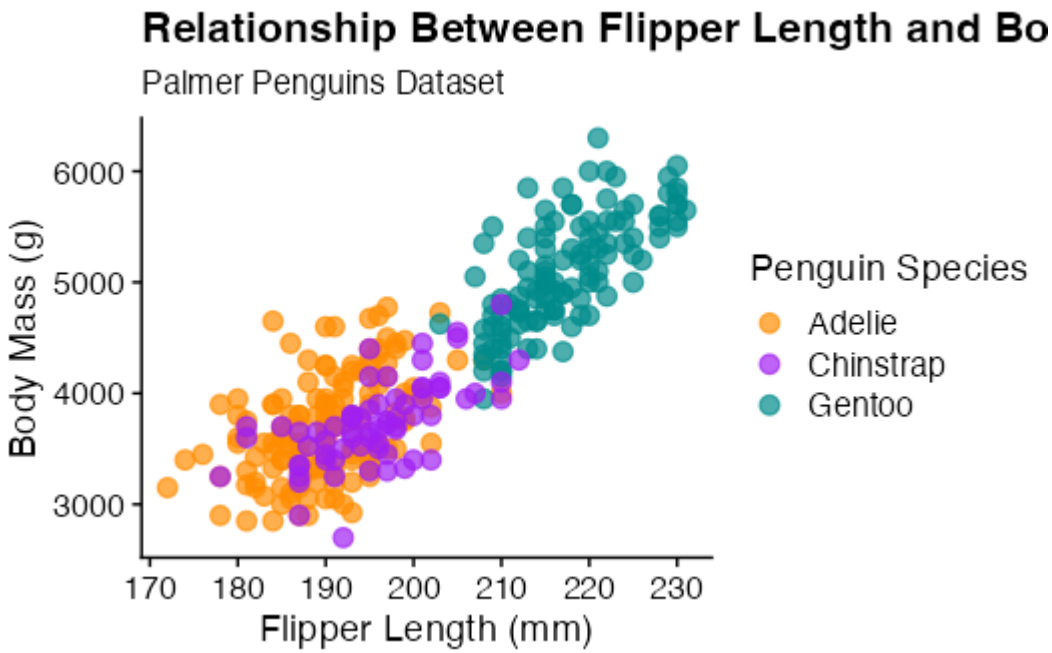
```
# Add labels and title
p ← ggplot(penguins_clean, aes(x = flipper_length_mm, y = body_mass_g, colour = species)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(
    title = "Relationship Between Flipper Length and Body Mass",
    subtitle = "Palmer Penguins Dataset",
    x = "Flipper Length (mm)",
    y = "Body Mass (g)",
    colour = "Penguin Species"
  )
p
```


Relationship Between Flipper Length and Bo
Palmer Penguins Dataset
Body Mass (g)
6000
5000
4000
3000
170
180
190
200
210
220
230
Flipper Length (mm)
Penguin Species
Adelie
Chinstrap
Gentoo

Building a plot: Step 8 - Customise colours

We can use a custom colour palette:

```
# Customize colours
p ← ggplot(penguins_clean, aes(x = flipper_length_mm, y = body_mass_g, colour = species)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_colour_manual(values = c("darkorange", "purple", "cyan4")) +
  labs(
    title = "Relationship Between Flipper Length and Body Mass",
    subtitle = "Palmer Penguins Dataset",
    x = "Flipper Length (mm)",
    y = "Body Mass (g)",
    colour = "Penguin Species"
  )
p
```

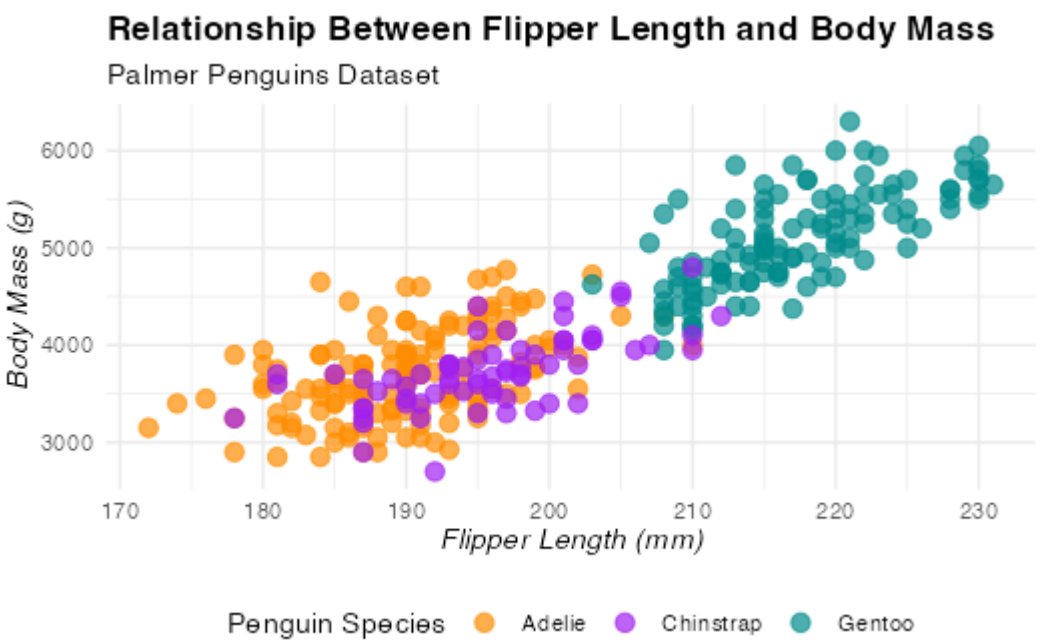

Relationship Between Flipper Length and Bo
Palmer Penguins Dataset
Body Mass (g)
6000
5000
4000
3000
170
180
190
200
210
220
230
Flipper Length (mm)
Penguin Species
Adelie
Chinstrap
Gentoo

Building a plot: Step 9 - Apply a theme

Finally, let's apply a theme to change the overall appearance:

```
# Apply a theme
p ← ggplot(penguins_clean, aes(x = flipper_length_mm, y = body_mass_g, colour = species)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_colour_manual(values = c("darkorange", "purple", "cyan4")) +
  labs(
    title = "Relationship Between Flipper Length and Body Mass",
    subtitle = "Palmer Penguins Dataset",
    x = "Flipper Length (mm)",
    y = "Body Mass (g)",
    colour = "Penguin Species"
  ) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    plot.title = element_text(face = "bold"),
    axis.title = element_text(face = "italic")
```

```
  )
p
```

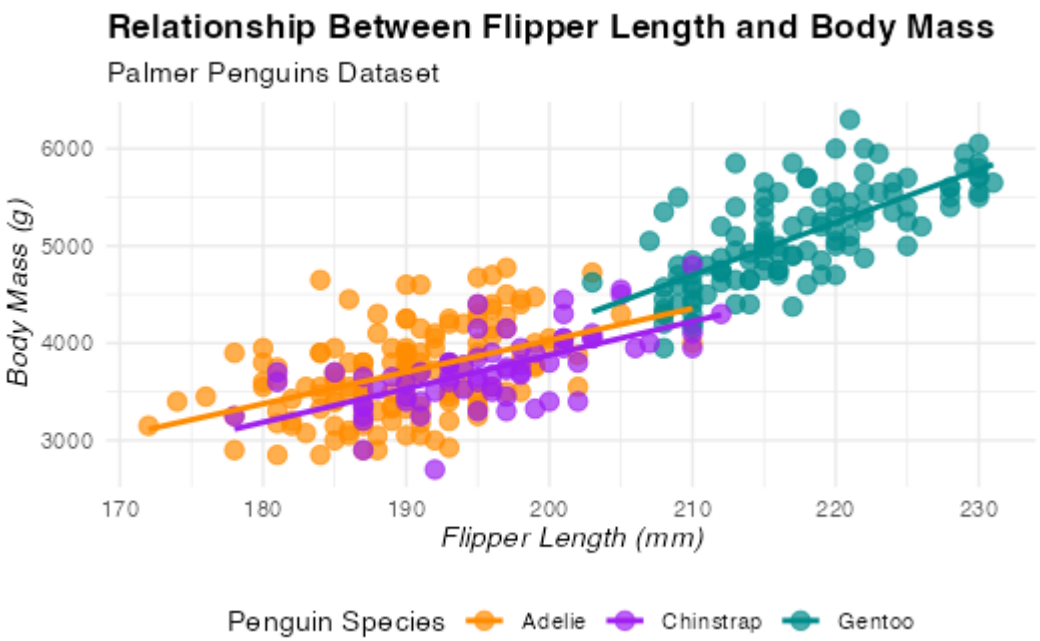

Relationship Between Flipper Length and Body Mass
Palmer Penguins Dataset
Body Mass (g)
6000
5000
4000
3000
170
180
190
200
210
220
230
Flipper Length (mm)
Penguin Species
Adelie
Chinstrap
Gentoo

Adding more layers: Trend lines

One of the strengths of ggplot2 is the ability to add multiple layers:

```
# Add trend lines for each species
p + geom_smooth(method = "lm", se = FALSE)
```

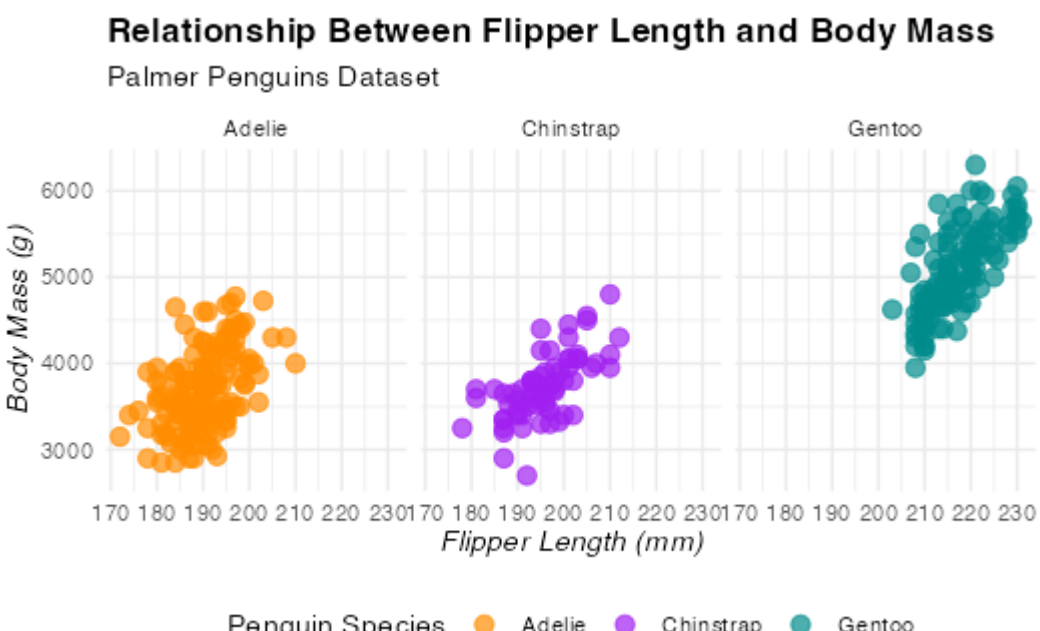
```
`geom_smooth()` using formula = 'y ~ x'
```


Relationship Between Flipper Length and Body Mass
Palmer Penguins Dataset
Body Mass (g)
6000
5000
4000
3000
170
180
190
200
210
220
230
Flipper Length (mm)
Penguin Species
Adelie
Chinstrap
Gentoo

Faceting: Split by species

We can also split the plot into facets by species:

```
# Create facets by species
p + facet_wrap(~ species)
```



The complete ggplot2 code

Here's the complete code for our final plot:

```
ggplot(penguins_clean, aes(x = flipper_length_mm, y = body_mass_g, colour = species)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_colour_manual(values = c("darkorange", "purple", "cyan4")) +
  labs(
    title = "Relationship Between Flipper Length and Body Mass",
    subtitle = "Palmer Penguins Dataset",
    x = "Flipper Length (mm)",
    y = "Body Mass (g)",
    colour = "Penguin Species"
  ) +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    plot.title = element_text(face = "bold"),
    axis.title = element_text(face = "italic")
  )
```

Resources for further learning

- R Graphics Cookbook
- ggplot2 documentation
- R for Data Science

References and resources

Core reading

- Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer.
- Chang, W. (2018). *R Graphics Cookbook*. O'Reilly Media.
- Wickham, H., & Grolemund, G. (2017). *R for Data Science*. O'Reilly Media.

Online resources

- ggplot2 documentation
- R for Data Science - Data Visualization chapter
- The R Graph Gallery
- Cookbook for R - Graphs

Thanks!