Lab 03 – Exploring and visualising data

ENVX1002 Handbook

Semester 1, 2025

Learning outcomes

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

- ☐ Import and prepare data for visualisation
- ☐ Create basic plot types using ggplot2 (histograms, boxplots, bar plots, scatterplots)
- ☐ Customise plots with appropriate labels, colours, and themes
- ☐ Identify and visualise distribution properties (normality, skewness, kurtosis)
- ☐ Interpret visualisations to draw meaningful conclusions
- ☐ Export and save plots for reports

Before we begin

For this lab, you will need to:

1. Create a new Quarto document in your **project folder** (or create a new RStudio project) to simplify file management and reproducibility
2. Download the data files from the links provided in Exercise 1
3. Make sure you have the following packages installed and loaded:
   - `tidyverse`: For data manipulation and visualization (includes `ggplot2`)
   - `moments`: For calculating skewness and kurtosis
   - `patchwork`: For combining multiple plots

You can install packages using `install.packages("package_name")` if needed, and load them with `library(package_name)`.

Exercise 1: Dataset exploration

Exploring the `pie_crab` dataset

In this lab, we'll work with two environmental datasets. They can be downloaded from the links below:

- pie_crab.csv: Crab size measurements across different sites and latitudes
- hbr_maples.csv: Maple seedling measurements from different watersheds

By now, you should know how to use `read_csv()` to read these datasets into R. If you need a refresher, refer to the previous labs or ask for help from a demonstrator (if available).

Let's start by exploring the `pie_crab` dataset. We've done this before, but it's always good to refresh our memory as these functions are extremely common to use.

- The `str()` function shows us the structure of the dataset, including variable names, types, and a preview of the values.
- The `head()` function shows the first six rows of the dataset, giving us a quick look at the data.
- The `summary()` function provides descriptive statistics for each variable, including minimum, maximum, mean, and quartiles for numeric variables.

```
str(pie_crab)
```

```
tibble [392 × 9] (S3: tbl_df/tbl/data.frame)
 $ date         : Date[1:392], format: "2016-07-24" "2016-07-24" ...
 $ latitude     : num [1:392] 30 30 30 30 30 30 30 30 30 30 ...
 $ site         : chr [1:392] "GTM" "GTM" "GTM" "GTM" ...
 $ size         : num [1:392] 12.4 14.2 14.5 12.9 12.4 ...
 $ air_temp     : num [1:392] 21.8 21.8 21.8 21.8 21.8 ...
 $ air_temp_sd  : num [1:392] 6.39 6.39 6.39 6.39 6.39 ...
 $ water_temp   : num [1:392] 24.5 24.5 24.5 24.5 24.5 ...
 $ water_temp_sd: num [1:392] 6.12 6.12 6.12 6.12 6.12 ...
 $ name          : chr [1:392] "Guana Tolomoto Matanzas NERR" "Guana Tolomoto
Matanzas NERR" "Guana Tolomoto Matanzas NERR" "Guana Tolomoto Matanzas NERR" ...
```

```
head(pie_crab)
```

```
# A tibble: 6 × 9
  date       latitude site   size air_temp air_temp_sd water_temp water_temp_sd
  <date>        <dbl> <chr> <dbl>    <dbl>       <dbl>      <dbl>         <dbl>
1 2016-07-24       30 GTM    12.4     21.8        6.39       24.5          6.12
2 2016-07-24       30 GTM    14.2     21.8        6.39       24.5          6.12
3 2016-07-24       30 GTM    14.5     21.8        6.39       24.5          6.12
4 2016-07-24       30 GTM    12.9     21.8        6.39       24.5          6.12
5 2016-07-24       30 GTM    12.4     21.8        6.39       24.5          6.12
6 2016-07-24       30 GTM    13.0     21.8        6.39       24.5          6.12
# ℹ 1 more variable: name <chr>
```

```
summary(pie_crab)
```

```
      date               latitude         site                size
 Min.   :2016-07-24   Min.   :30.00   Length:392         Min.   : 6.64
 1st Qu.:2016-07-28   1st Qu.:34.00   Class :character   1st Qu.:12.02
```

```
Median :2016-08-01   Median :39.10   Mode  :character   Median :14.44
Mean   :2016-08-02   Mean   :37.69                      Mean   :14.66
3rd Qu.:2016-08-09   3rd Qu.:41.60                      3rd Qu.:17.34
Max.   :2016-08-13   Max.   :42.70                      Max.   :23.43
   air_temp      air_temp_sd      water_temp    water_temp_sd
Min.   :10.29   Min.   :6.391   Min.   :13.98   Min.   :4.838
1st Qu.:12.05   1st Qu.:8.110   1st Qu.:14.33   1st Qu.:6.567
Median :13.93   Median :8.410   Median :17.50   Median :6.998
Mean   :15.20   Mean   :8.654   Mean   :17.65   Mean   :7.252
3rd Qu.:18.63   3rd Qu.:9.483   3rd Qu.:20.54   3rd Qu.:7.865
Max.   :21.79   Max.   :9.965   Max.   :24.50   Max.   :9.121
    name
Length:392
Class :character
Mode  :character
```

Question 1

What variables are in the dataset? What are the data types of each variable? Are there any missing values? What are the ranges of the numeric variables?

⚠ Answer 1

From our exploration of the pie_crab dataset, we can see that:

- **What variables are in the dataset?** The dataset contains 9 variables: date, latitude, site, size, air_temp, air_temp_sd, water_temp, water_temp_sd, and name
- **What are the data types of each variable?** Data types:
  - `date`: Date format
  - `latitude`, `size`, `air_temp`, `air_temp_sd`, `water_temp`, `water_temp_sd`: numeric variables
  - `site`, `name`: character variables (which we'll later convert to factors)
- **Are there any missing values?** Based on the summary output there are no missing values
- **What are the ranges of the numeric variables?**

```
# We can answer this question by viewing the numbers in `summmary()`
# but, we can compute it too. See ?sapply for more information
sapply(pie_crab[, sapply(pie_crab, is.numeric)], range)
```

```
     latitude  size air_temp air_temp_sd water_temp water_temp_sd
[1,]     30.0  6.64   10.293       6.391     13.976         4.838
[2,]     42.7 23.43   21.792       9.965     24.502         9.121
```

- Ranges of numeric variables:
  - latitude: 30.00 to 42.70 degrees
  - size: 6.64 to 23.43 mm
  - air_temp: 10.29 to 21.79 ℃
  - water_temp: 13.98 to 24.50 ℃
  - air_temp_sd: 6.39 to 9.97
  - water_temp_sd: 4.84 to 9.12

From our exploration, we can see that the pie_crab dataset contains information about:

- `date`: When the crabs were measured
- `latitude`: The latitude where the crabs were collected
- `site`: The specific collection site
- `size`: The size of the crabs in millimeters
- `air_temp`: Air temperature in degrees Celsius
- `water_temp`: Water temperature in degrees Celsius

Identifying factors in the data

One of the first steps in data exploration is to determine if your data types are recognised correctly by R. Looking at the output of `str()` can help you identify variables that might be better represented as different data types.

In our dataset, `site` and `name` are character variables with repeating values. When categorical variables like these have a limited number of possible values that repeat throughout the dataset,

they're good candidates to be converted to factors. Factors are R's way of representing categorical data efficiently.

We can check how many unique values these variables have using the `unique()` function:

```
unique(pie_crab$site)
```

```
 [1] "GTM" "SI"  "NIB" "ZI"  "RC"  "VCR" "DB"  "JC"  "CT"  "NB"  "CC"  "BC"
[13] "PIE"
```

```
unique(pie_crab$name)
```

```
 [1] "Guana Tolomoto Matanzas NERR"     "Sapelo Island NERR"
 [3] "North Inlet Winyah Bay NERR"      "Zeke's Island NERR"
 [5] "Rachel Carson NERR"               "Virginia Coastal Reserve LTER"
 [7] "Delaware Bay NERR"                "Jacques Cousteau NERR"
 [9] "Sixpenny Island - Connecticut"    "Narragansett Bay NERR"
[11] "Cape Cod"                         "Bare Cove Park"
[13] "Plum Island Estuary - West Creek"
```

For a dataset with 392 observations, having only a few unique sites and names suggests that these variables should be factors. Let's count exactly how many unique values we have:

```
length(unique(pie_crab$site))
```

```
[1] 13
```

```
length(unique(pie_crab$name))
```

```
[1] 13
```

We can convert these character variables to factors using the `factor()` function:

```
# Convert site and name to factors
pie_crab$site <- factor(pie_crab$site)
pie_crab$name <- factor(pie_crab$name)

# Check the structure again to confirm the conversion
str(pie_crab)
```

```
tibble [392 × 9] (S3: tbl_df/tbl/data.frame)
 $ date         : Date[1:392], format: "2016-07-24" "2016-07-24" ...
 $ latitude     : num [1:392] 30 30 30 30 30 30 30 30 30 30 ...
 $ site         : Factor w/ 13 levels "BC","CC","CT",..: 5 5 5 5 5 5 5 5 5 5 ...
 $ size         : num [1:392] 12.4 14.2 14.5 12.9 12.4 ...
 $ air_temp     : num [1:392] 21.8 21.8 21.8 21.8 21.8 ...
 $ air_temp_sd  : num [1:392] 6.39 6.39 6.39 6.39 6.39 ...
 $ water_temp   : num [1:392] 24.5 24.5 24.5 24.5 24.5 ...
 $ water_temp_sd: num [1:392] 6.12 6.12 6.12 6.12 6.12 ...
 $ name          : Factor w/ 13 levels "Bare Cove Park",..: 4 4 4 4 4 4 4 4
4 4 ...
```

Notice how we've converted the specific variables to factors and used the assignment operator `<-` to update the dataset. This is a common pattern in R, where we update the dataset in place. The output of `str()` now shows these variables as factors with their levels (unique values) listed.

Converting to factors is important because:

1. It makes R treat the data appropriately in statistical analyses
2. It can make visualizations more informative
3. It improves memory efficiency for large datasets

Exercise 2: Building visualizations with `ggplot2`

The grammar of graphics

The ggplot2 package is based on the "grammar of graphics," a framework that breaks visualisations into components, similar to how grammar breaks language into parts of speech. This approach makes it possible to create complex visualisations by combining simple elements.

The key components (or layers) include:

- **Data**: The dataset being visualised
- **Aesthetics**: Mappings from data variables to visual properties
- **Geometries**: The shapes used to represent the data
- **Facets**: Subplots that show different subsets of the data
- **Statistics**: Transformations of the data (e.g., counts, means)
- **Coordinates**: The space in which the data is plotted
- **Themes**: Visual styling of non-data elements

Let's learn how to create visualisations using this approach by building a plot step by step, explaining each component along the way. **As you follow along, you are improving the code that produces the plot – not re-writing it – at each step.**

Step 1: The canvas

Every ggplot2 visualisation starts with a blank canvas:

```
# Start with an empty canvas
ggplot()
```


This creates an empty plotting space. It doesn't show anything yet because we haven't specified any data or how to visualise it.

Step 2: Adding data

Next, we tell ggplot2 what data to use. Ideally the data is a `data.frame` or `tibble` object (which you will know by using `str()` previously):

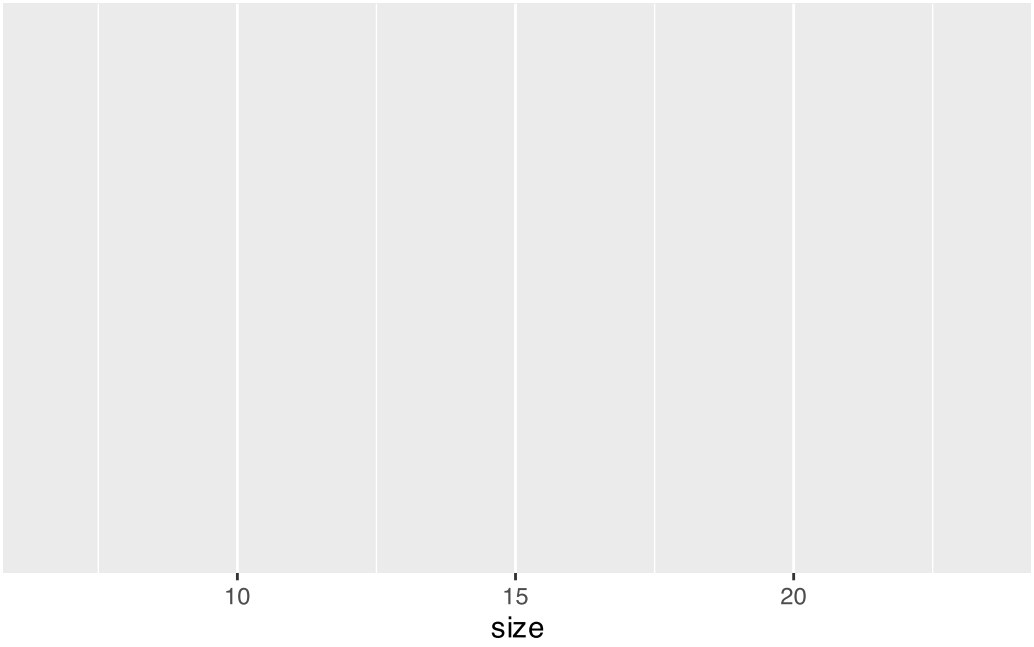
```
# Add data to the plot
ggplot(data = pie_crab)
```

We've now told ggplot2 to use the `pie_crab` dataset, but we still cannot see anything because we have not specified which variables to plot – or how to represent them.

Step 3: Mapping aesthetics

Aesthetics map variables in your data to visual properties in the plot. They are a function `aes()` on their own. Think of aesthetics as how you woul define `x`, `y` and other dimensions in the plot.

```
# Map variables to visual properties
ggplot(data = pie_crab, mapping = aes(x = size))
```



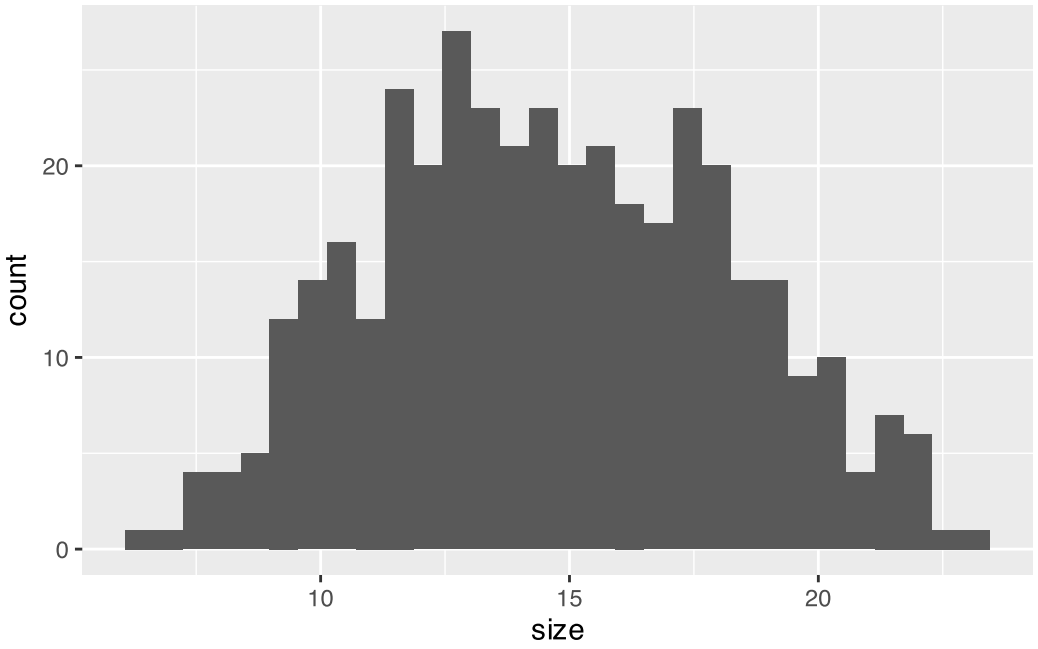
Here, we've mapped the `size` variable to the x-axis. This variable is something that exists in the `pie_crab` data frame. We still cannot see any data points because we haven't specified how to represent the data (e.g., as points, bars, or lines).

Step 4: Adding a Geometry

Geometries determine how the data is represented visually. They are functions that are prefixed by `geom_*` so that users know what they are meant to do. In most cases it is clear what type of plot is being create by reading the name of the geometry function(s) in the plot code.

```
# Add a histogram geometry
ggplot(data = pie_crab, mapping = aes(x = size)) +
  geom_histogram()
```

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



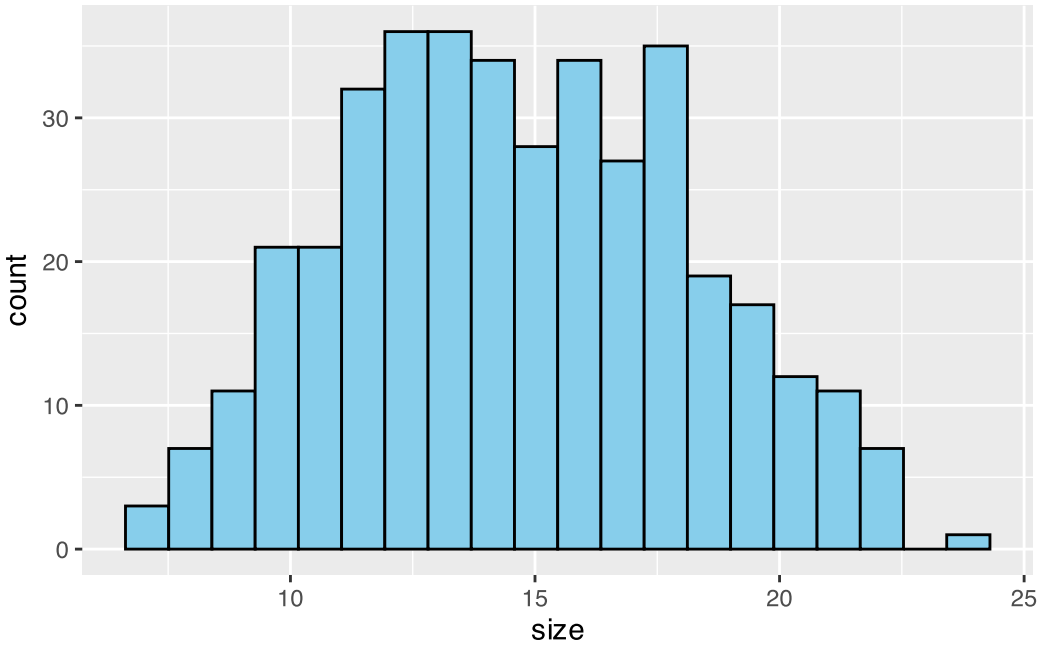
Now we can see the data! We've added a histogram geometry (`geom_histogram()`), which counts the number of observations falling into bins along the x-axis. The + operator adds layers to the plot.

Notice the message about the default bin width. ggplot2 automatically chose 30 bins, but we can adjust this.

Step 5: Customizing the Geometry

Let's customize our histogram:

```
# Customize the histogram
ggplot(data = pie_crab, mapping = aes(x = size)) +
  geom_histogram(bins = 20, fill = "skyblue", color = "black")
```



We've made several changes:

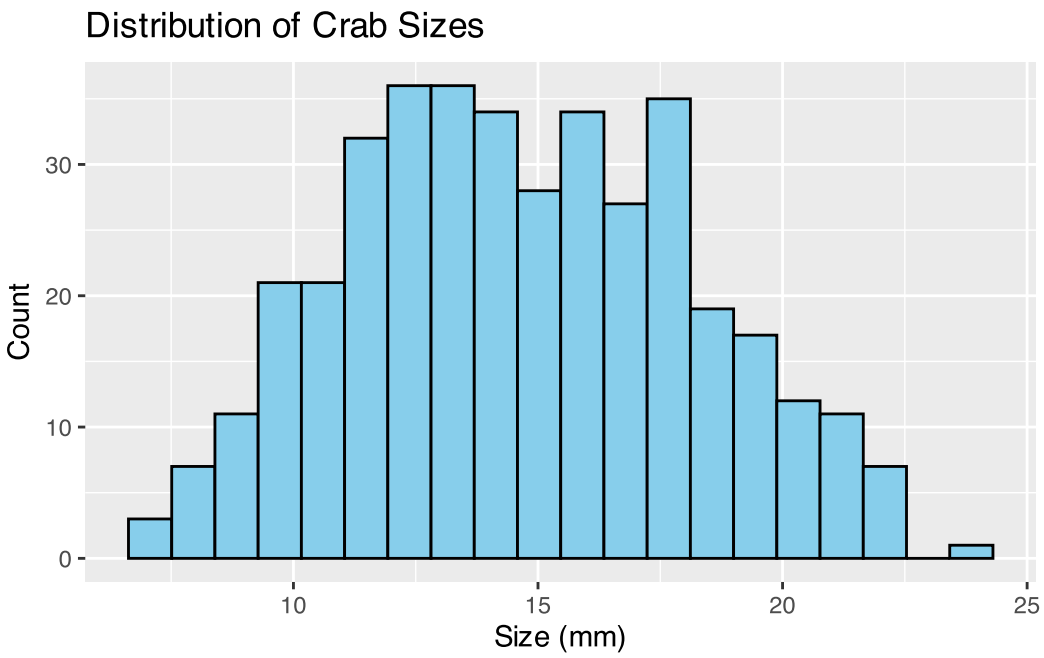
- `bins = 20`: Changed the number of bins to 20
- `fill = "skyblue"`: Set the fill color of the bars to sky blue
- `color = "black"`: Set the outline color of the bars to black

These are fixed properties applied to all bars, not mappings from data variables. For colours you may search "r colours" in your web browser to see what available colours you can use (it's a *lot*).

Step 6: Adding Labels and Titles

Good visualisations have clear labels. Titles are optional – but the option is available if you need it. We can add all of these in the next layer using the `labs()` function:

```
# Add informative labels
ggplot(data = pie_crab, mapping = aes(x = size)) +
  geom_histogram(bins = 20, fill = "skyblue", color = "black") +
  labs(
    title = "Distribution of Crab Sizes",
    x = "Size (mm)",
    y = "Count"
  )
```



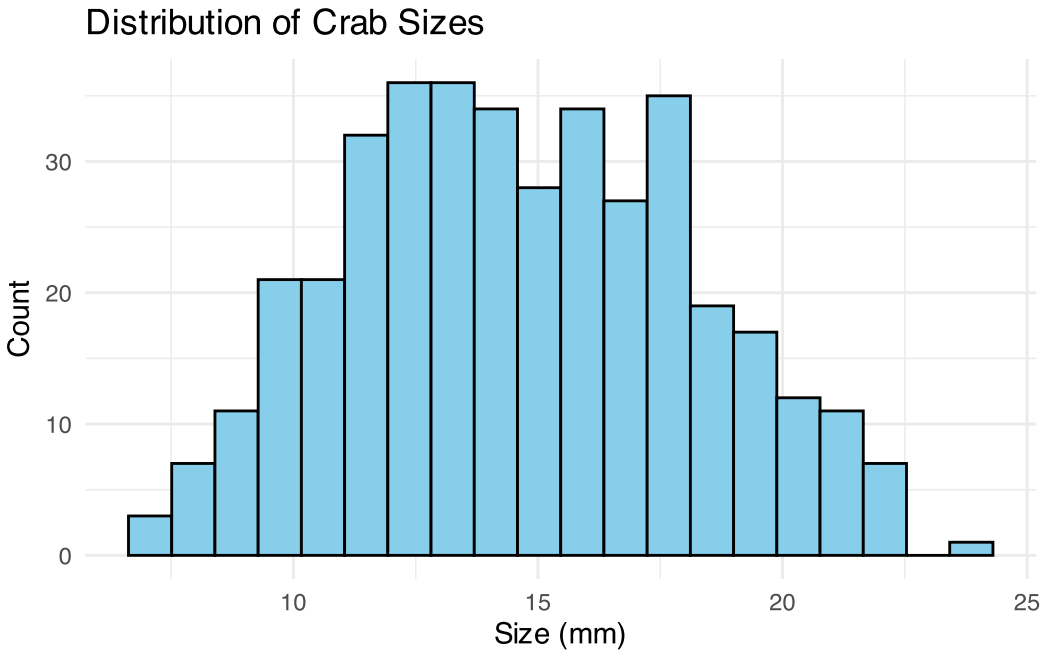
The `labs()` function adds various text elements to the plot:

- `title`: The main title of the plot
- `x`: The x-axis label
- `y`: The y-axis label

Step 7: Applying a Theme

Themes control the overall appearance of the plot:

```
# Add a theme for consistent styling
ggplot(data = pie_crab, mapping = aes(x = size)) +
  geom_histogram(bins = 20, fill = "skyblue", color = "black") +
  labs(
    title = "Distribution of Crab Sizes",
    x = "Size (mm)",
    y = "Count"
  ) +
  theme_minimal()
```



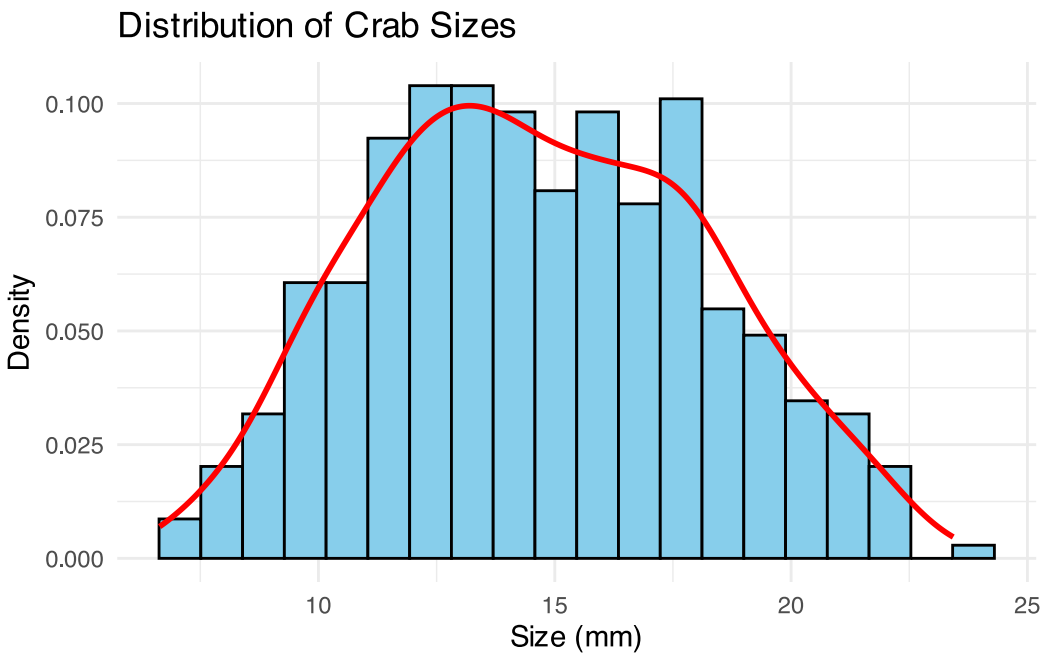
The `theme_minimal()` function applies a minimalist theme with a white background and subtle grid lines. Other common themes include:

- `theme_classic()`: No grid lines, simple axes
- `theme_light()`: Light background with subtle grid lines
- `theme_dark()`: Dark background for presentations

Bonus: Adding multiple geometries

One of the powerful features of ggplot2 is the ability to layer multiple geometries:

```
# Add a density curve on top of the histogram
ggplot(data = pie_crab, mapping = aes(x = size)) +
  geom_histogram(aes(y = after_stat(density)),
    bins = 20,
    fill = "skyblue", color = "black"
  ) +
  geom_density(color = "red", linewidth = 1) +
  labs(
    title = "Distribution of Crab Sizes",
    x = "Size (mm)",
    y = "Density"
  ) +
  theme_minimal()
```



In this plot:

- We've changed the y-axis of the histogram to show density instead of count using `aes(y = after_stat(density))`
- We've added a density curve with `geom_density()`
- We've set the density curve color to red and increased its line width

> Question 2
>
> 1. What's the difference between setting a fixed property (like `fill = "blue"`) and mapping a variable to an aesthetic (like `aes(fill = site)`)?
> 2. How would you modify the histogram to have more or fewer bins? Use `?geom_histogram` to help you think of an answer.
> 3. What would happen if you changed the order of the `geom_histogram()` and `geom_density()` layers?

These questions can typically be answered by making changes to the code to view the differences.

⚠ Answer 2

1. **Fixed property vs. aesthetic mapping**:
   - Fixed property (e.g., `fill = "blue"`): Applies the same value to all elements, regardless of data values
   - Aesthetic mapping (e.g., `aes(fill = site)`): Maps a variable in your data to a visual property, creating different values based on the data
   - Fixed properties are used for consistent styling, while aesthetic mappings are used to represent data values visually
2. **Modifying histogram bins**:
   - To change the number of bins: Use the `bins` parameter (e.g., `geom_histogram(bins = 30)`)
   - To specify bin width directly: Use the `binwidth` parameter (e.g., `geom_histogram(binwidth = 2)`)
   - Fewer bins show broader patterns but less detail, while more bins show more detail but may be noisier
3. **Changing layer order**:
   - If `geom_density()` came before `geom_histogram()`, the histogram would be drawn on top of the density curve
   - This would make the density curve partially or completely hidden behind the histogram
   - Layer order matters because layers are drawn in the order they are added, with later layers appearing on top

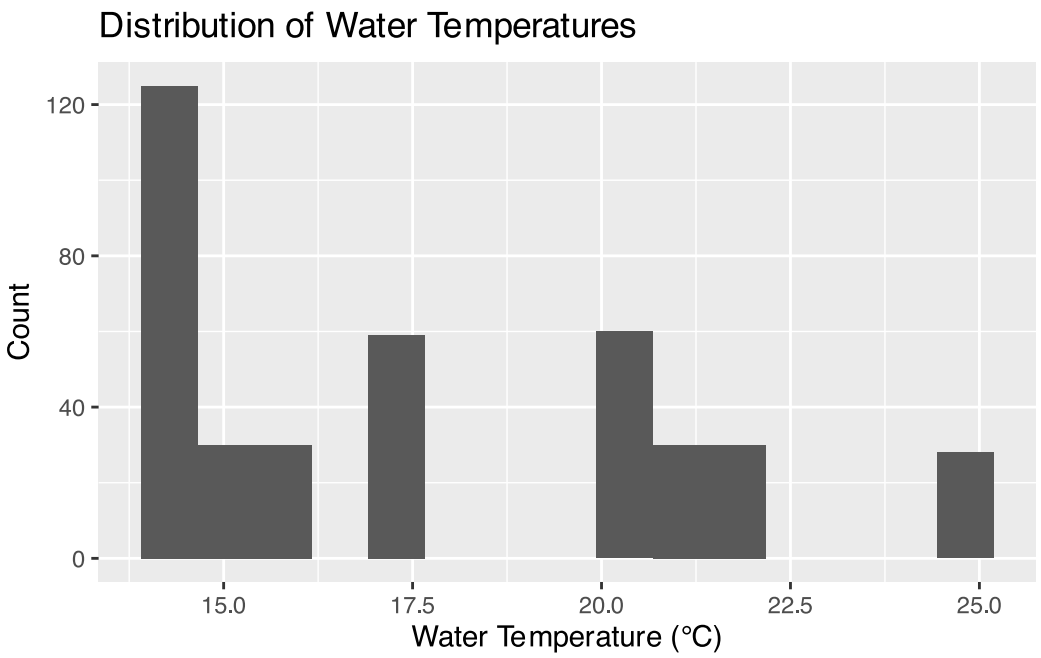
Exercise 3: Analysing environmental variables

Now that we understand the grammar of graphics approach, let's analyse a different variable in our dataset.

Examining water temperature distribution

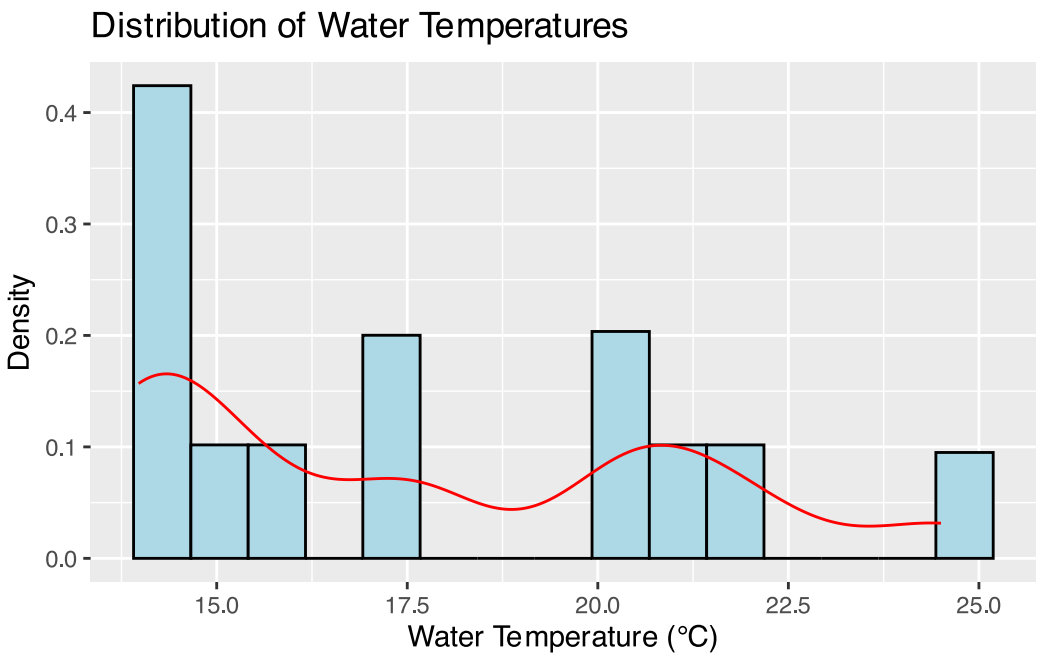
Let's examine the distribution of water temperatures across our sampling sites:

```
# Create a basic histogram of water temperatures
ggplot(pie_crab, aes(x = water_temp)) +
  geom_histogram(bins = 15) +
  labs(
    title = "Distribution of Water Temperatures",
    x = "Water Temperature (°C)",
    y = "Count"
  )
```



The histogram shows us the frequency distribution of water temperatures. We can see the shape of the distribution, including any skewness or unusual patterns.

```
# Add a density curve
ggplot(pie_crab, aes(x = water_temp)) +
  geom_histogram(aes(y = after_stat(density)),
    bins = 15,
    fill = "lightblue", colour = "black"
  ) +
  geom_density(colour = "red") +
  labs(
    title = "Distribution of Water Temperatures",
    x = "Water Temperature (°C)",
    y = "Density"
  )
```



Adding a density curve helps us see the overall shape of the distribution more clearly.

Question 3

What is the shape of the distribution of water temperatures? Does the distribution appear to be normal? Are there any outliers?

Answer 3

Based on the histogram and density plot of water temperatures:

- **Shape of the distribution**: The distribution appears to be right-skewed (positively skewed), with a longer tail extending toward higher temperatures. The data also appears to be bimodal, although there are gaps that make it a bit harder to assert tha this is the case.
- **Normality**: The distribution does not appear to be perfectly normal. A normal distribution would be symmetric around the mean, but this distribution shows some asymmetry. The density curve helps visualize this deviation from normality.
- **Outliers**: There appear to be a few potential outliers on the right side of the distribution, representing unusually warm water temperatures. Outliers are (often) isolated bars in the histogram at the right end of the distribution.

Skewness and Kurtosis

To quantify the shape of the water temperature distribution, we can calculate skewness and kurtosis:

```
# Calculate skewness and kurtosis for water temperature
skewness_value <- skewness(pie_crab$water_temp)
kurtosis_value <- kurtosis(pie_crab$water_temp)

# Print the values
cat("Skewness of water temperature:", skewness_value, "\n")
```

```
Skewness of water temperature: 0.4750277
```

```
cat("Kurtosis of water temperature:", kurtosis_value, "\n")
```

```
Kurtosis of water temperature: 1.888369
```

Interpreting these values:

- **Skewness** measures the asymmetry of the distribution:
  - 0 = symmetric (like a normal distribution)
  - More than, > 0 = right-skewed (tail extends to the right)
  - Less than, < 0 = left-skewed (tail extends to the left)
- **Kurtosis** measures the "tailedness" of the distribution:
  - 3 = normal distribution (in the `moments` package, this is sometimes normalized to 0)
  - More than, > 3 = leptokurtic (heavy-tailed, more outliers)
  - Less than, < 3 = platykurtic (light-tailed, fewer outliers)

The skewness value of approximately 0.5 confirms our visual observation that the water temperature distribution is moderately right-skewed. The kurtosis value of approximately 2.5 indicates the distribution has slightly lighter tails than a normal distribution.

These numerical measures help us quantify what we observe visually in the histograms and density plots. Now that we understand the overall distribution of our data, let's explore how it varies across different groups.

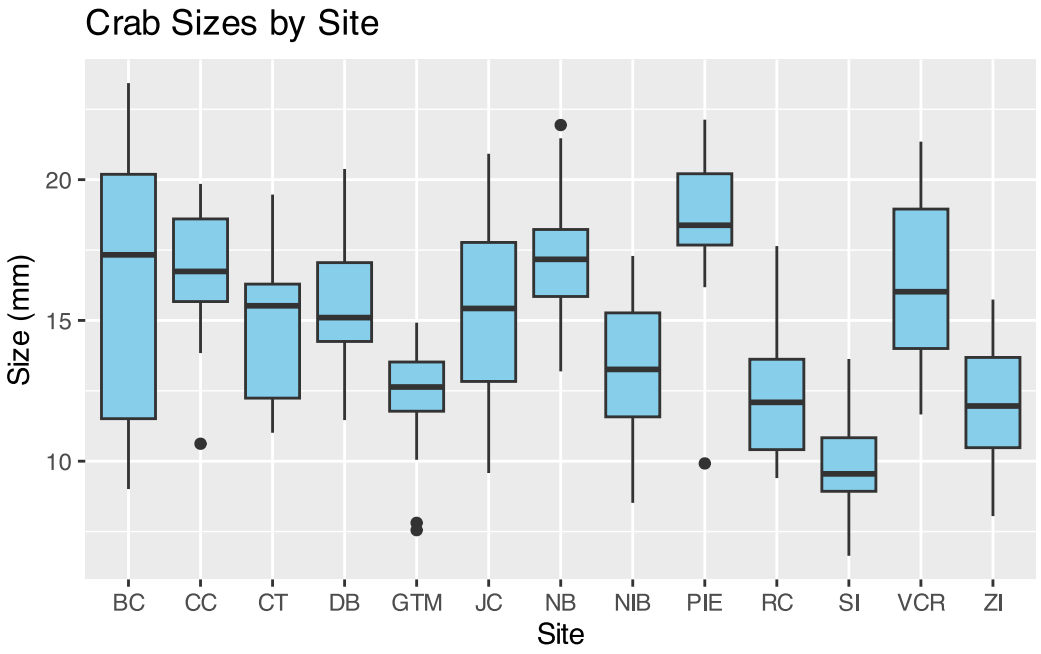
Exercise 4: Comparing groups

Now that we've examined the overall distribution of crab sizes, let's compare sizes across different groups.

Creating boxplots to compare sites

Boxplots are excellent for comparing distributions between two or more groups:

```r
# Create boxplots of crab sizes by site
ggplot(pie_crab, aes(x = site, y = size)) +
  geom_boxplot(fill = "skyblue") +
  labs(
    title = "Crab Sizes by Site",
    x = "Site",
    y = "Size (mm)"
  )
```



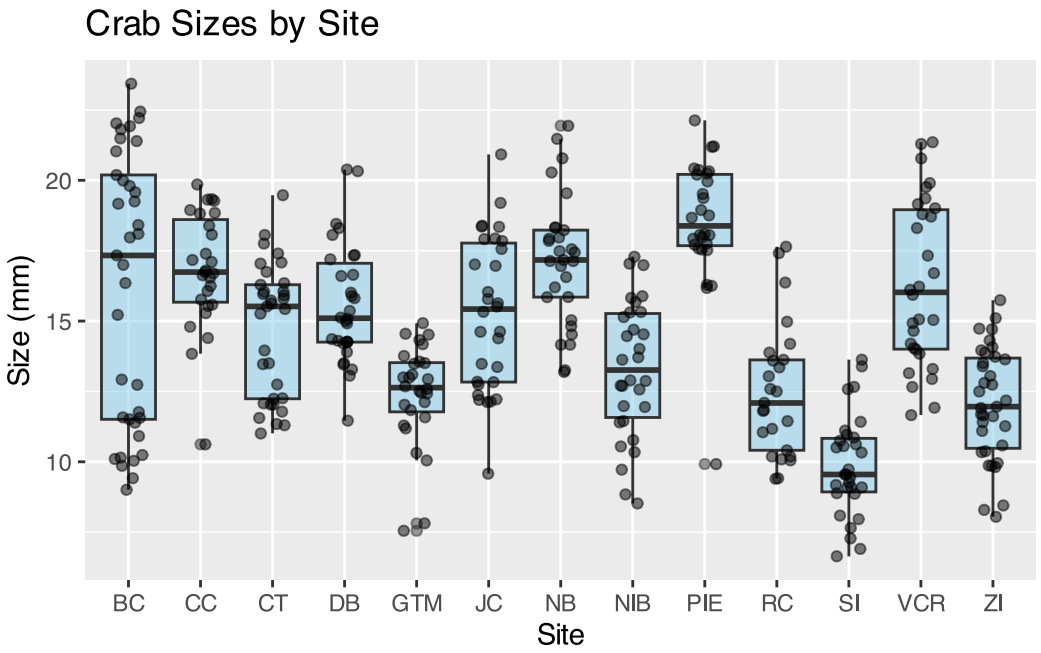
A boxplot shows:

- The median (middle line)
- The interquartile range (IQR) from the 25th to 75th percentile (the box)
- The whiskers (typically extend to 1.5 × IQR)
- Outliers (points beyond the whiskers)

To see the actual data points alongside the boxplots we can add another geometric layer. You can see how it may or may not be useful to readers. In most cases adding the data points is **not** necessary, but in some cases it could be useful in the exploration phase (i.e. not the final plot for publication).

```r
# Add points to see the actual data
ggplot(pie_crab, aes(x = site, y = size)) +
  geom_boxplot(fill = "skyblue", alpha = 0.5) +
  geom_jitter(width = 0.2, alpha = 0.5) +
  labs(
```

```
    title = "Crab Sizes by Site",
    x = "Site",
    y = "Size (mm)"
  )
```



We've added:

- `geom_jitter()` to add individual data points with a slight horizontal jitter to avoid overplotting
- `alpha = 0.5` to make both the boxplots and points semi-transparent
- `width = 0.2` to control the amount of horizontal jittering

Question 4

How do crab sizes vary across different sites? Which site has the largest median crab size? Which site shows the most variability in crab sizes? Are there any outliers at specific sites? Answer these questions in a descriptive manner using the plot.

Here we are exercising our ability to see patterns from data visualisations and using them to make certain observations about the data.

⚠️ Answer 4

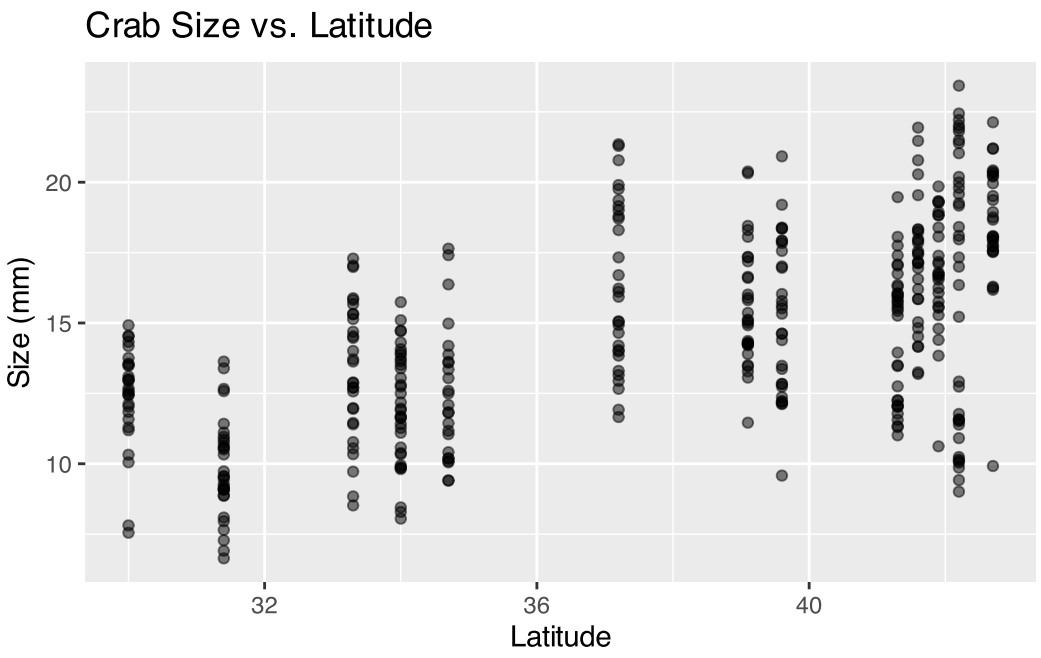
Based on the boxplots comparing crab sizes across different sites:

- **Variation across sites**: There is considerable variation in crab sizes between sites.
- **Largest median size**: PIE (Plum Island Estuary) has the largest median crab size since the median line is the highest among the different boxplots.
- **Most variability**: BC (Bare Cove Park) shows the most variability in crab sizes. This is evidenced by its wider box and longer whiskers in the boxplot, indicating a broader spread of sizes.
- **Outliers**: Yes, there are outliers at specific sites. Several sites show individual points above or below their whiskers, representing unusually large or small crabs for those locations. These outliers are important to note as they may represent unique environmental conditions or genetic factors affecting growth.

Exploring the relationship between latitude and size

Let's examine if there's a relationship between latitude and crab size:

```
# Create a scatterplot of size vs. latitude
ggplot(pie_crab, aes(x = latitude, y = size)) +
  geom_point(alpha = 0.5) +
  labs(
    title = "Crab Size vs. Latitude",
    x = "Latitude",
    y = "Size (mm)"
  )
```

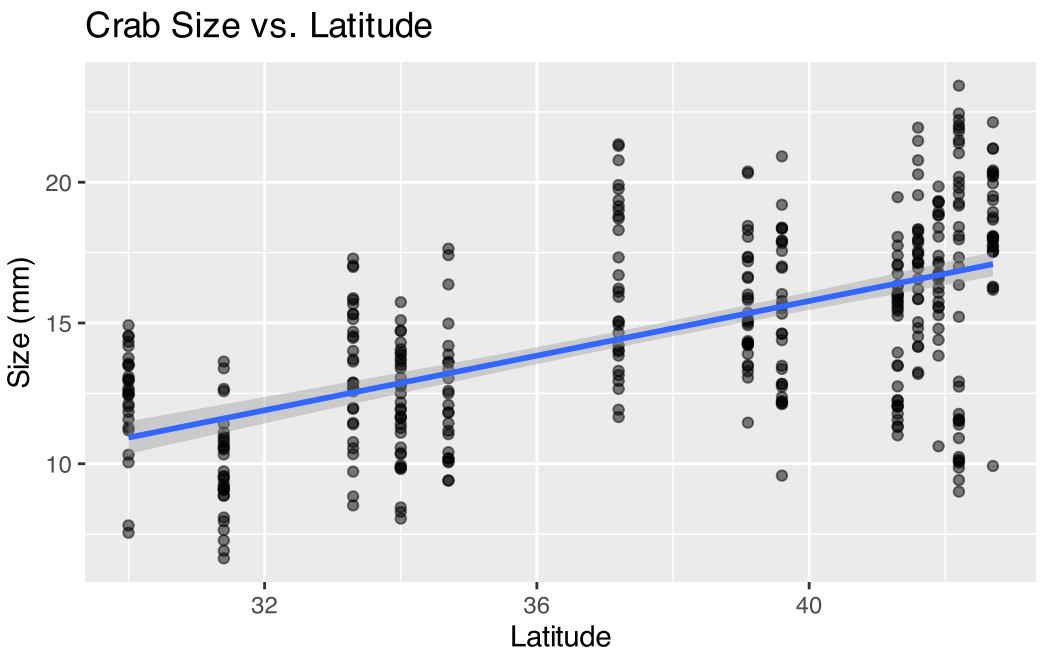


Scatterplots show the relationship between two continuous variables. Each point represents a single observation.

To help visualise the trend, we can add a trend line (something that we will cover in greater detail once we look at linear models in Week 7):

```
# Add a trend line
ggplot(pie_crab, aes(x = latitude, y = size)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = TRUE) +
  labs(
    title = "Crab Size vs. Latitude",
    x = "Latitude",
    y = "Size (mm)"
  )
```

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



We've added:

- `geom_smooth(method = "lm")` to add a linear regression line
- `se = TRUE` to include the standard error as a shaded confidence band

> Question 5
>
> Is there a relationship between latitude and crab size?

> Answer 5
>
> Based on the scatterplot it looks like as latitude increases, size also increases, because the trend line shows a general positive trend.

Exercise 5: Faceting and grouping

So far, we've created separate plots for different analyses. Now, let's explore techniques for comparing multiple groups or variables within a single plot.

Exploring the `hbr_maples` dataset

Let's switch to our second dataset, which contains measurements of maple seedlings from different watersheds:

```
# Examine the structure of the maples dataset
str(hbr_maples)
```

```
tibble [359 × 11] (S3: tbl_df/tbl/data.frame)
 $ year               : num [1:359] 2003 2003 2003 2003 2003 ...
 $ watershed          : Factor w/ 2 levels "Reference","W1": 1 1 1 1 1 1 1 1 1
1 ...
 $ elevation          : Factor w/ 2 levels "Low","Mid": 1 1 1 1 1 1 1 1 1 1 ...
 $ transect           : Factor w/ 12 levels "R1","R2","R3",..: 1 1 1 1 1 1 1 1
1 1 ...
 $ sample             : Factor w/ 20 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8
9 10 ...
 $ stem_length        : num [1:359] 86.9 114 83.5 68.1 72.1 77.7 85.5 81.6 92.9
59.6 ...
 $ leaf1area          : num [1:359] 13.84 14.57 12.45 9.97 6.84 ...
 $ leaf2area          : num [1:359] 12.13 15.27 9.73 10.07 5.48 ...
 $ leaf_dry_mass       : num [1:359] 0.0453 0.0476 0.0423 0.0397 0.0204 0.0317
0.0382 0.0179 0.0286 0.0125 ...
 $ stem_dry_mass     : num [1:359] 0.03 0.0338 0.0248 0.0194 0.018 0.0246 0.0316
0.015 0.0291 0.0149 ...
 $ corrected_leaf_area: num [1:359] 29.1 33 25.3 23.2 15.5 ...
```

```
# View the first few rows
head(hbr_maples)
```

```
# A tibble: 6 × 11
   year watershed elevation transect sample stem_length leaf1area leaf2area
  <dbl> <fct>     <fct>     <fct>    <fct>        <dbl>     <dbl>     <dbl>
1  2003 Reference Low       R1       1             86.9     13.8      12.1
2  2003 Reference Low       R1       2            114       14.6      15.3
3  2003 Reference Low       R1       3             83.5     12.5       9.73
4  2003 Reference Low       R1       4             68.1      9.97     10.1
5  2003 Reference Low       R1       5             72.1      6.84      5.48
6  2003 Reference Low       R1       6             77.7      9.66      7.64
# ℹ 3 more variables: leaf_dry_mass <dbl>, stem_dry_mass <dbl>,
#   corrected_leaf_area <dbl>
```

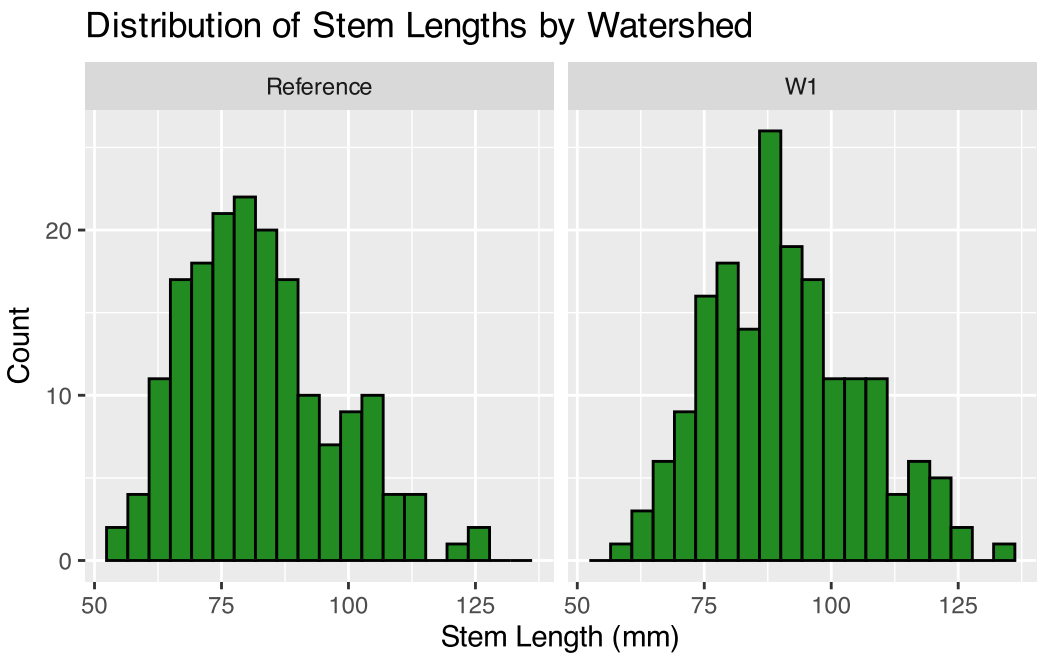
```
# Get a summary of the variables
summary(hbr_maples)
```

```
      year          watershed   elevation     transect       sample
 Min.   :2003   Reference:179   Low :120   R1     : 40   1      : 18
 1st Qu.:2003   W1       :180   Mid :120   R2     : 40   2      : 18
 Median :2003                  NA's:119   W1-1   : 40   3      : 18
 Mean   :2003                             W1-2   : 40   4      : 18
 3rd Qu.:2004                             W1-3   : 40   5      : 18
 Max.   :2004                             R3     : 39   6      : 18
```

```
                                          (Other):120   (Other):251
 stem_length       leaf1area        leaf2area     leaf_dry_mass
Min.   : 52.70   Min.   : 2.480   Min.   : 3.40   Min.   :0.01170
1st Qu.: 75.65   1st Qu.: 8.818   1st Qu.: 8.95   1st Qu.:0.03975
Median : 85.70   Median :11.636   Median :11.28   Median :0.05590
Mean   : 86.86   Mean   :11.800   Mean   :11.75   Mean   :0.06368
3rd Qu.: 97.05   3rd Qu.:14.016   3rd Qu.:14.30   3rd Qu.:0.07855
Max.   :132.30   Max.   :26.952   Max.   :25.79   Max.   :0.38700
                 NA's   :119      NA's   :119
stem_dry_mass     corrected_leaf_area
Min.   :0.00790   Min.   : 9.597
1st Qu.:0.02360   1st Qu.:21.180
Median :0.03320   Median :25.950
Mean   :0.04506   Mean   :26.687
3rd Qu.:0.05820   3rd Qu.:31.751
Max.   :0.15000   Max.   :55.874
                  NA's   :119
```

Now, let's create histograms of stem length by watershed using faceting:

```
# Create histograms of stem length by watershed
ggplot(hbr_maples, aes(x = stem_length)) +
  geom_histogram(bins = 20, fill = "forestgreen", colour = "black") +
  facet_wrap(~watershed) +
  labs(
    title = "Distribution of Stem Lengths by Watershed",
    x = "Stem Length (mm)",
    y = "Count"
  )
```



The `facet_wrap()` function creates separate panels for each value of the specified variable. This allows us to compare distributions across groups.

Question 6

How do stem lengths differ between watersheds? Which watershed shows more variability in stem lengths? Are the distributions similarly shaped? Use only the visualisations to answer these questions.

Answer 6

TBC

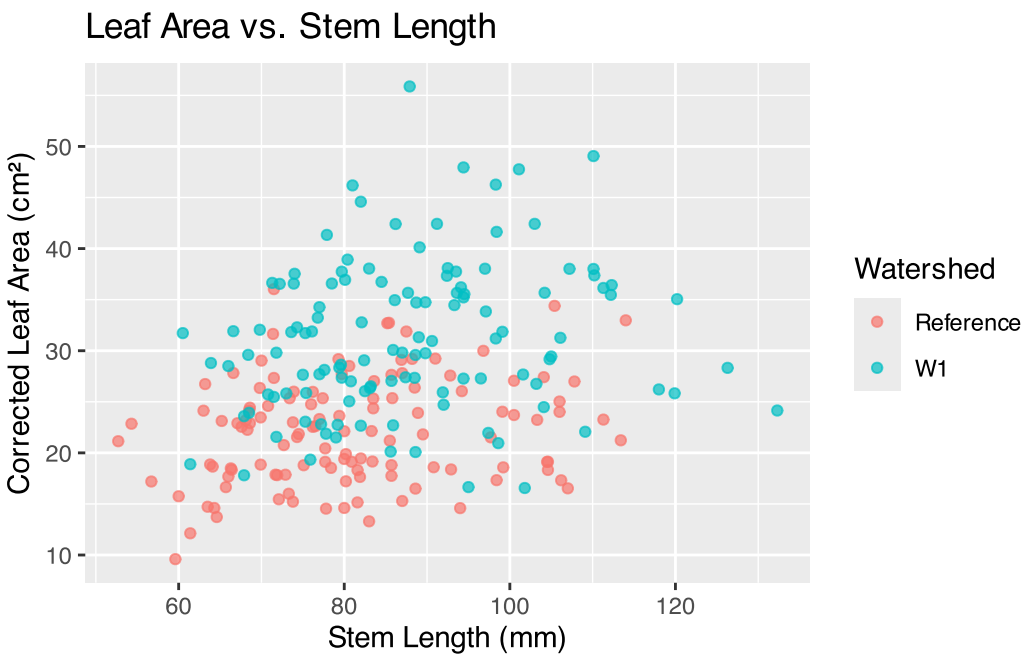
Comparing leaf area and stem length

Let's examine the relationship between leaf area and stem length, comparing across watersheds:

```
# Create a scatterplot of leaf area vs. stem length, coloured by watershed
ggplot(hbr_maples, aes(x = stem_length, y = corrected_leaf_area, colour =
watershed)) +
  geom_point(alpha = 0.7) +
  labs(
    title = "Leaf Area vs. Stem Length",
    x = "Stem Length (mm)",
    y = "Corrected Leaf Area (cm²)",
```

```
    colour = "Watershed"
  )
```

```
Warning: Removed 119 rows containing missing values or values outside the scale
range
(`geom_point()`).
```



Here, we've mapped the `watershed` variable to the `colour` aesthetic, which automatically creates a color-coded legend.

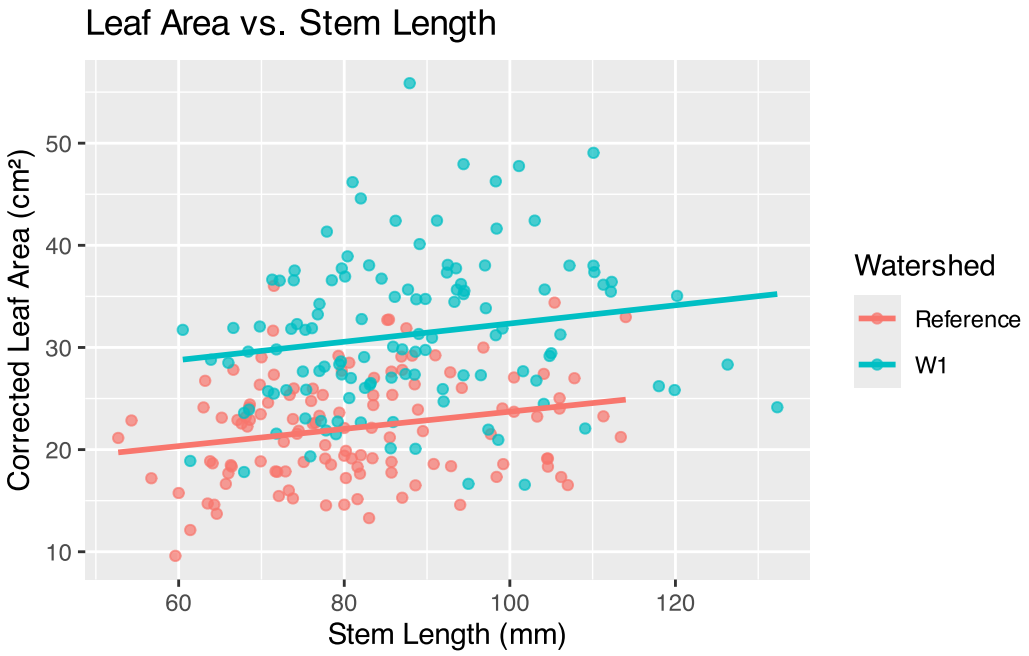
Let's add separate trend lines for each watershed:

```
# Add separate trend lines for each watershed
ggplot(hbr_maples, aes(x = stem_length, y = corrected_leaf_area, colour =
watershed)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    title = "Leaf Area vs. Stem Length",
    x = "Stem Length (mm)",
    y = "Corrected Leaf Area (cm²)",
    colour = "Watershed"
  )
```

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

```
Warning: Removed 119 rows containing non-finite outside the scale range
(`stat_smooth()`).
```

```
Warning: Removed 119 rows containing missing values or values outside the scale
range
(`geom_point()`).
```



When we include `colour = watershed` in the global aesthetics, ggplot2 automatically applies this grouping to all geometries, including `geom_smooth()`. This creates separate trend lines for each watershed.

Question 7

Is there a relationship between stem length and leaf area? Does this relationship differ between watersheds? What might explain these differences from an ecological perspective?

⚠ Answer 7

Based on the scatterplot examining the relationship between stem length and leaf area:

- **Relationship existence**: Yes, there is a clear positive relationship between stem length and leaf area. As stem length increases, leaf area tends to increase as well. This suggests that larger seedlings generally have both longer stems and larger leaves.
- **Differences between watersheds**: The relationship does differ between watersheds. The trend lines for each watershed have different slopes, indicating that the rate at which leaf area increases with stem length varies. Watershed 6 appears to have a steeper slope than Watershed 1, suggesting that for each unit increase in stem length, seedlings in Watershed 6 gain more leaf area than those in Watershed 1.
- **Ecological explanation**: These differences could be explained by several ecological factors:
  1. **Resource availability**: Watershed 6 may have more favorable growing conditions (e.g., better soil nutrients, more water, or optimal light conditions) that allow seedlings to allocate more resources to leaf production relative to stem growth.
  2. **Adaptation to local conditions**: Seedlings in different watersheds may have adapted different growth strategies in response to local environmental pressures.
  3. **Competition**: Different levels of competition in each watershed might influence how seedlings allocate resources between stem and leaf growth.
  4. **Genetic differences**: There could be genetic differences between maple populations in different watersheds that influence their growth patterns.

These findings highlight the importance of considering environmental context when studying plant growth relationships, as the same species can show different growth patterns in different habitats.

Exercise 6: Bonus take-home

These exercises are designed for you to practice the visualisation techniques we've covered in this lab. You can complete them during the lab if you finish early, or at home for additional practice.

Basic visualisation practice

1. Create a histogram of the `water_temp` variable in the crabs dataset. Calculate and interpret its skewness and kurtosis.
2. Create boxplots comparing the `leaf_dry_mass` between watersheds in the maples dataset. What do you observe?
3. Create a scatterplot examining the relationship between `stem_dry_mass` and `leaf_dry_mass` in the maples dataset, with points coloured by watershed.

Advanced challenge: `patchwork`

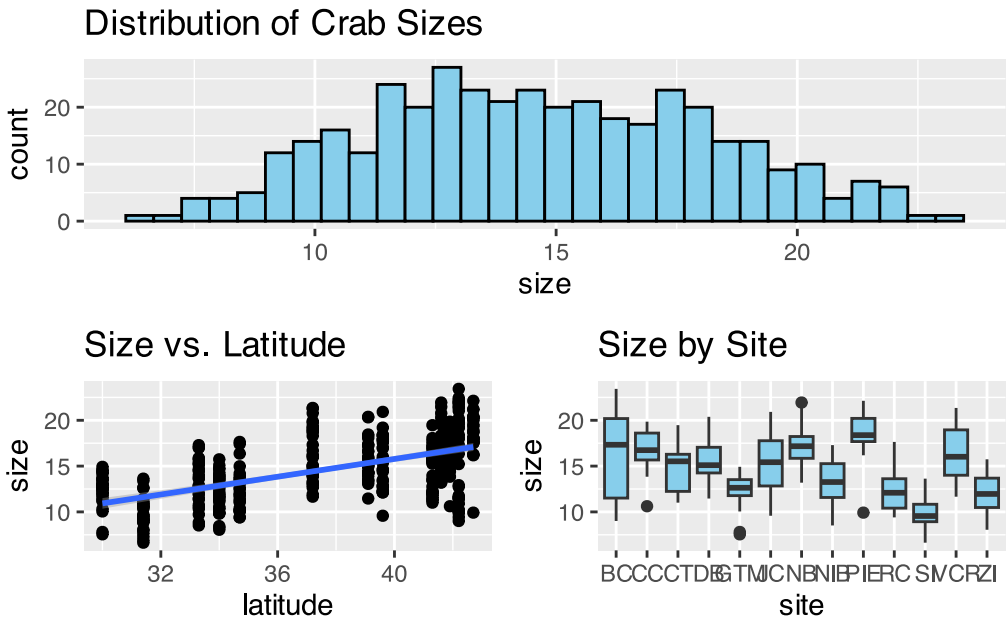
The patchwork package allows you to combine multiple plots into a single figure. This is useful for creating complex visualisations that tell a story about your data.

```
# Load the patchwork package
library(patchwork)
```

Let's create a few plots and then combine them:

```
# Create multiple plots
p1 <- ggplot(pie_crab, aes(x = size)) +
  geom_histogram(fill = "skyblue", colour = "black") +
  labs(title = "Distribution of Crab Sizes")

p2 <- ggplot(pie_crab, aes(x = latitude, y = size)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Size vs. Latitude")

p3 <- ggplot(pie_crab, aes(x = site, y = size)) +
  geom_boxplot(fill = "skyblue") +
  labs(title = "Size by Site")

# Combine plots (this is patchwork in action!)
p1 / (p2 + p3)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`geom_smooth()` using formula = 'y ~ x'
```



The patchwork syntax is intuitive:

- `/` arranges plots vertically (one above the other)
- `+` arranges plots horizontally (side by side)
- You can use parentheses to control the layout

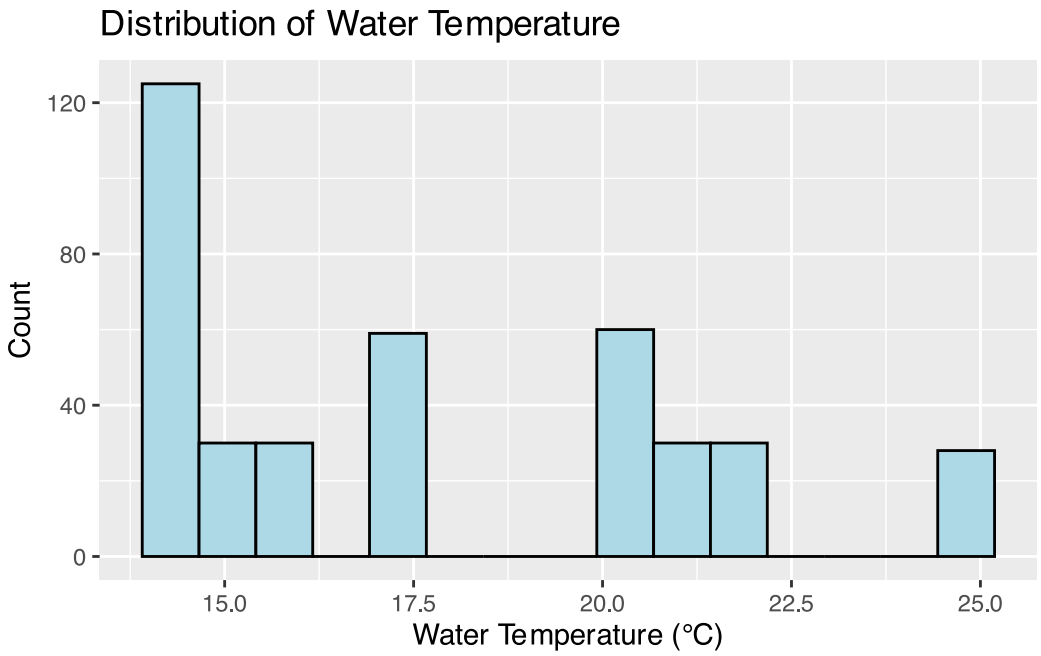
Now, try these exercises:

4. Create a combined plot using patchwork that shows:
   - A histogram of stem lengths
   - A scatterplot of stem length vs. leaf area
   - Boxplots of stem lengths by watershed
   - Arrange these plots in a 2x2 grid
5. Create a combined plot that tells a story about the relationship between temperature and crab size:
   - A scatterplot of air temperature vs. crab size
   - A scatterplot of water temperature vs. crab size
   - A boxplot of crab sizes by site
   - Arrange the scatterplots side by side and the boxplot below them

⚠ Solutions to Take-home Exercises

1. Histogram of water temperature with skewness and kurtosis:

```
# Create histogram of water temperature
ggplot(pie_crab, aes(x = water_temp)) +
  geom_histogram(bins = 15, fill = "lightblue", color = "black") +
  labs(
    title = "Distribution of Water Temperature",
    x = "Water Temperature (°C)",
    y = "Count"
  )
```

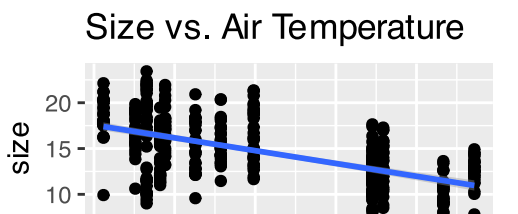


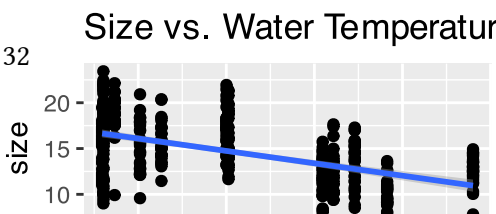

```
# Calculate skewness and kurtosis
skew_water <- skewness(pie_crab$water_temp)
kurt_water <- kurtosis(pie_crab$water_temp)

cat("Skewness of water temperature:", skew_water, "\n")
```

```
Skewness of water temperature: 0.4750277
```

```
cat("Kurtosis of water temperature:", kurt_water, "\n")
```

```

```





0.5, he tely = dis-

Summary

In this lab, we've explored how to create and customize various types of visualisations using ggplot2. We've learned:

1. The grammar of graphics approach to building visualisations layer by layer
2. How to create and interpret histograms, density plots, boxplots, and scatterplots
3. How to quantify and interpret distribution properties like skewness and kurtosis
4. How to compare groups using boxplots and faceting
5. How to examine relationships between variables using scatterplots and trend lines
6. How to combine multiple plots using the patchwork package

These skills will be valuable for exploring and presenting data in future labs and assignments.

Additional Resources

- R for Data Science - Data Visualisation chapter
- ggplot2 documentation
- patchwork package documentation
- R Graph Gallery - Examples of various visualisations in R
- Cookbook for R - Graphs - Recipes for common visualisation tasks