# Lecture 02: Introduction to statistical programming

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. Navigate and use the RStudio interface effectively
2. Execute basic R functions and understand their syntax
3. Feel confident explaining basic statistical concepts like **samples** and **populations**
4. Understand and explain measures of **central tendency** (mean, median, mode) clearly and without mathematical jargon
5. Master different measures of **spread** (range, IQR, variance, standard deviation) through practical examples
6. Calculate statistical measures using **both R and Excel**
7. Choose appropriate statistical measures for your biological data and explain your choices

## Quick checklist

By now you should have...

- ⍰ Installed **R**
- ⍰ Installed **RStudio**
- ⍰ Created one (or two) documents in Markdown using Quarto

# History of statistical programming

# From calculators to computers


Figure 1: 1800s: Mechanical calculators. Source

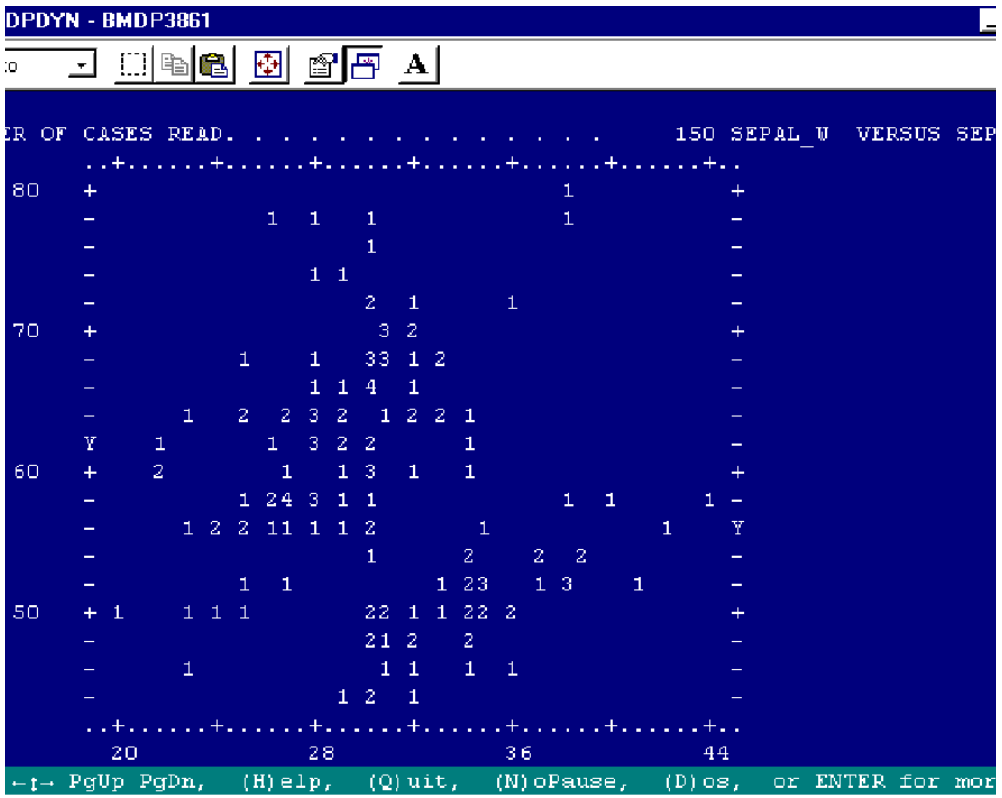



Figure 2: 1960s: Statistical software BMDP and SPSS (not in image). Source

# Statistical software in the 1970s

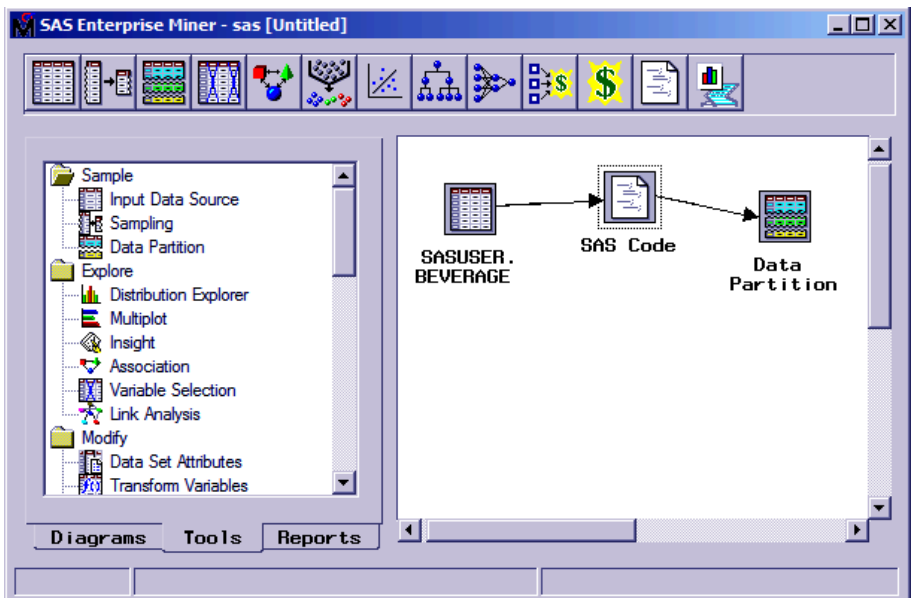



Figure 3: 1970s: SAS (Statistical Analysis System) Source

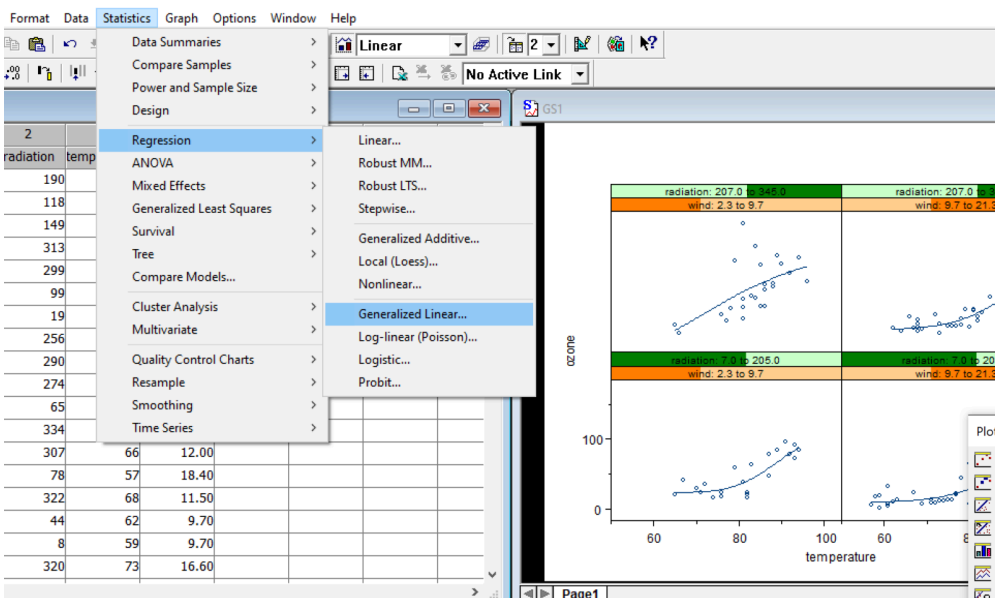
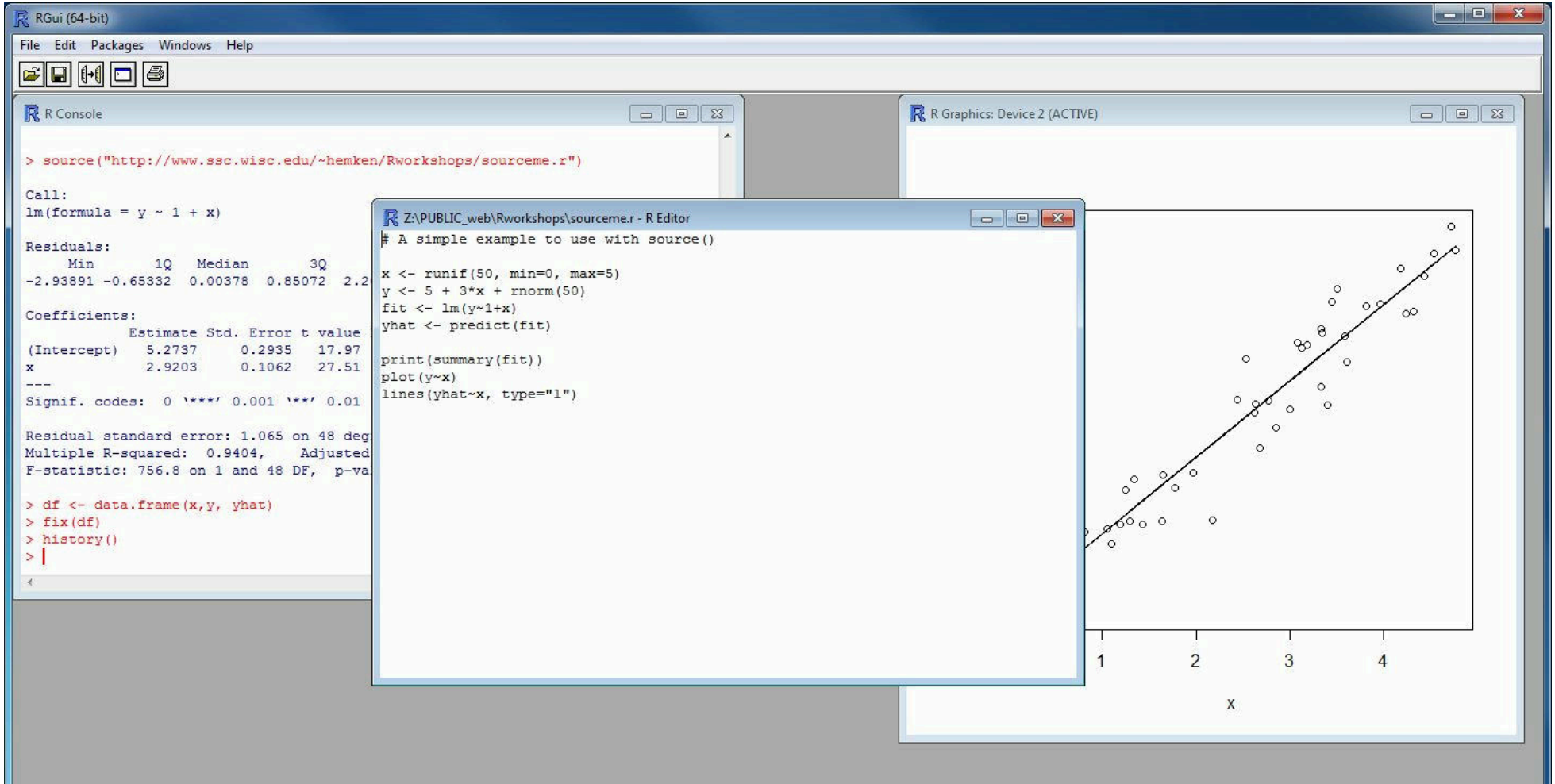



Figure 4: 1976: Birth of S at Bell Labs. S-PLUS debuts in 1988. Source

- Created at University of Auckland, New Zealand in 1993
- Named after creators (Ross & Robert) – and inspired by S programming language
- Developed rapidly in the 2000s
- Designed specifically for statistical computing and graphics, but now used in many fields



## R in today's world

- Leading tool in data science and statistics (although Python leads in majority of machine learning workflows)
- Over 22,000 packages on CRAN – extensive statistical capabilities
- Integration with other modern tools: Python, HTML, Javascript, Excel, AJAX...
- **Meets modern academic standards of reproducibility and increasingly preferred by statisticians**

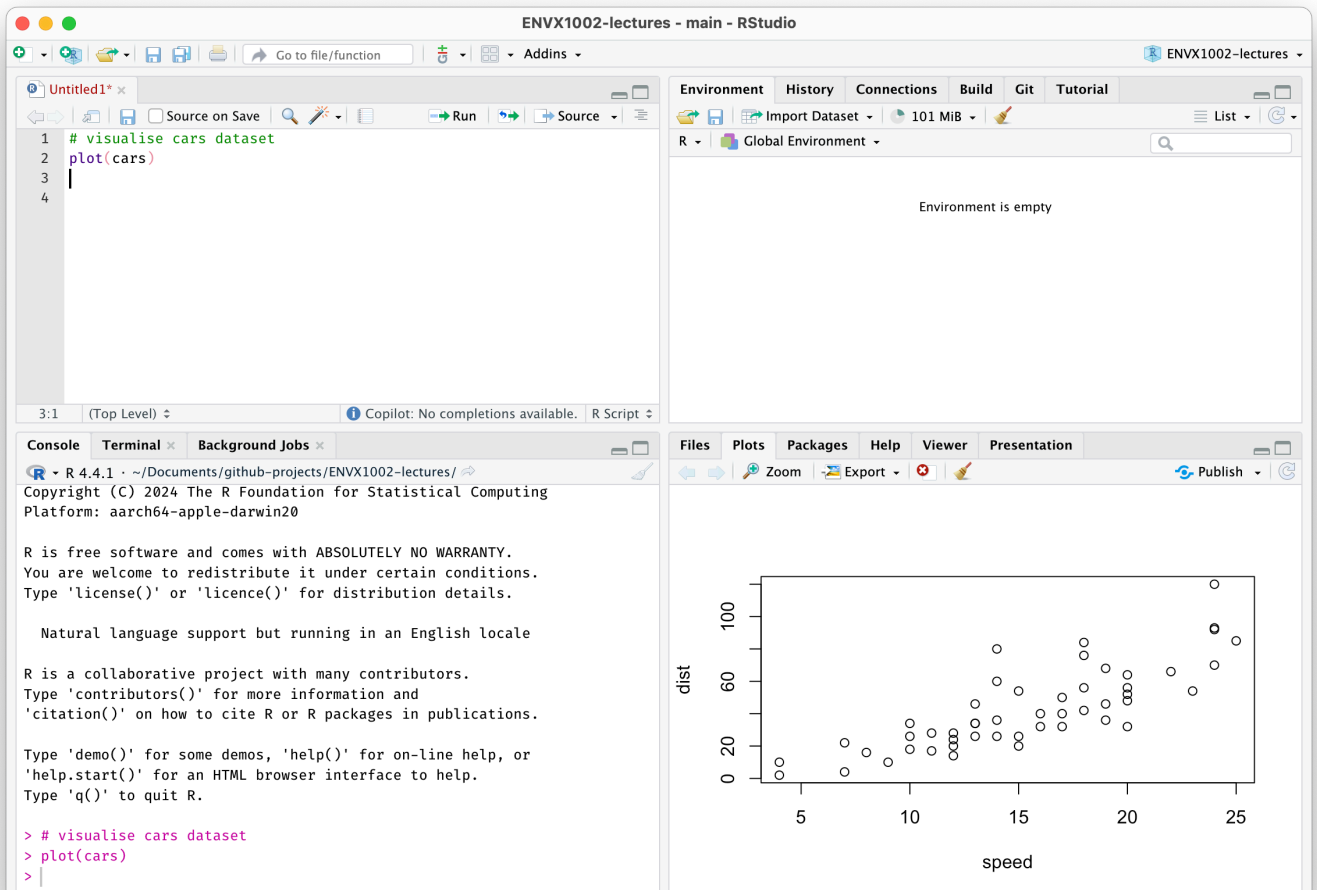
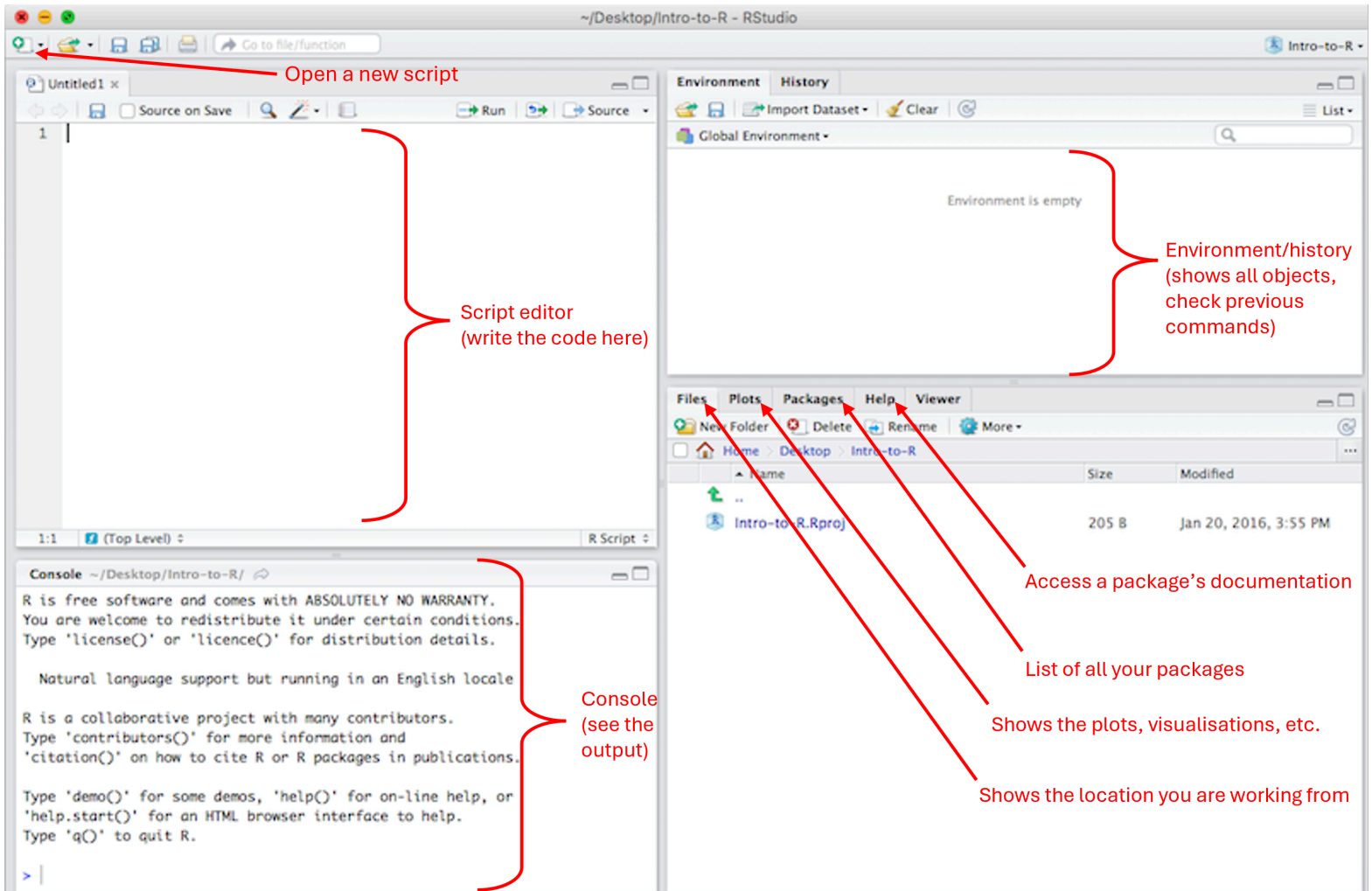



Figure 2: RStudio IDE. Source: Januar Harianto

# Getting Started with R

# Your RStudio workspace


~/Desktop/Intro-to-R - RStudio
Open a new script
Untitled1
Source on Save
Run
Source
Script editor
(write the code here)
Environment
History
Import Dataset
Clear
Global Environment
Environment is empty
Environment/history
(shows all objects,
check previous
commands)
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Desktop
Intro-to-R
Name
Size
Modified
Intro-to-R.Rproj
205 B
Jan 20, 2016, 3:55 PM
Access a package's documentation
List of all your packages
Shows the plots, visualisations, etc.
Shows the location you are working from
Console ~/Desktop/Intro-to-R/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
Console
(see the
output)

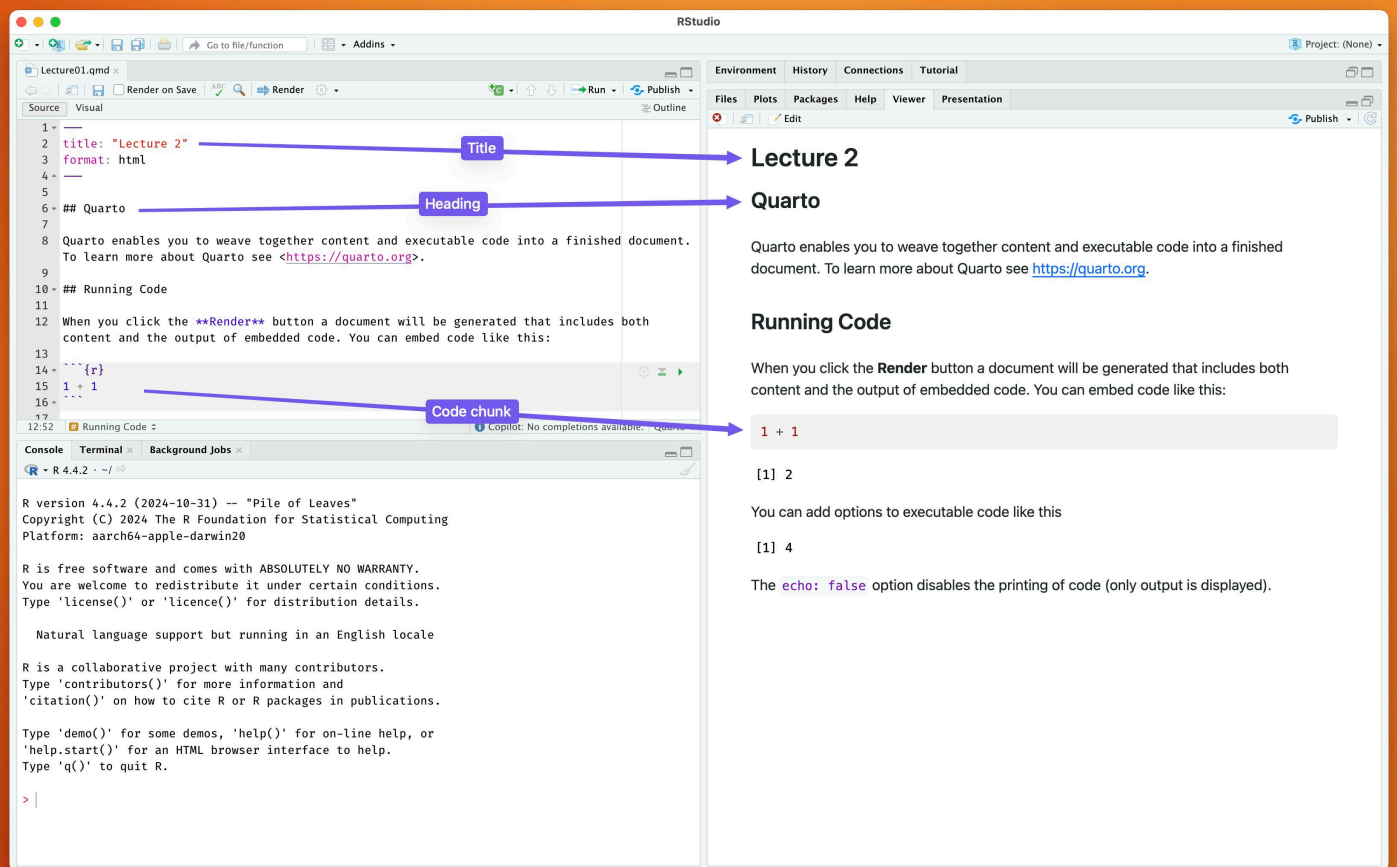
**Left**: all input, **Right**: all output

## We will always work in Quarto

- A **Markdown-based** authoring tool
- Allows you to write **reproducible documents** with code and text
- Outputs to various formats: HTML, PDF, Word, and more
- Quarto gallery

## How Quarto works

- Write your content in **Markdown** (left panel)
- View the output in **Preview** (right panel)
- Quick demo (only available in the live lecture)

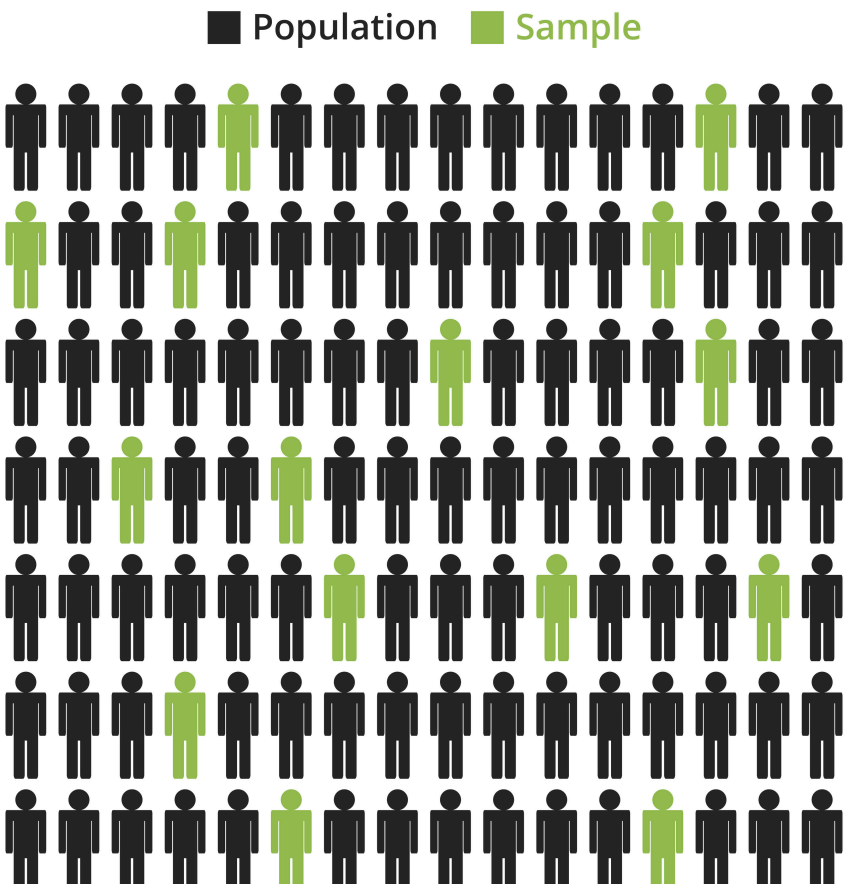

RStudio
Lecture01.qmd
Render on Save
Render
Run
Publish
Source
Visual
Outline
1 ---
2 title: "Lecture 2"
3 format: html
4 ---
5
6 ## Quarto
7
8 Quarto enables you to weave together content and executable code into a finished document.
To learn more about Quarto see <https://quarto.org>.
9
10 ## Running Code
11
12 When you click the **Render** button a document will be generated that includes both
content and the output of embedded code. You can embed code like this:
13
14 ```{r}
15 1 + 1
16 ```
Title
Heading
Code chunk
12:52
Running Code
Copilot: No completions available.
Console
Terminal
Background Jobs
R 4.4.2 · ~/
R version 4.4.2 (2024-10-31) -- "Pile of Leaves"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
Tutorial
Files
Plots
Packages
Help
Viewer
Presentation
Edit
Publish
Lecture 2
Quarto
Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.
Running Code
When you click the Render button a document will be generated that includes both content and the output of embedded code. You can embed code like this:
1 + 1
[1] 2
You can add options to executable code like this
[1] 4
The echo: false option disables the printing of code (only output is displayed).

## Essential Resources

- Read “A brief R guide for surviving ENVX1002” by Dr. Geoffrey Mazue
  - Available in the Tool Kit section on Canvas
  - Contains essential R programming tips and best practices for this unit
- Use the Help panel in RStudio

# Key statistical concepts

## Population vs Sample




**Population**

- All possible observations
- Usually too large to measure
- Example: All trees in a forest

**Sample**

- **Subset of the population**
- What we *actually* measure
- Example: 100 trees measured in a forest

**Most (if not all) statistical analyses are based on samples, not populations.**

## Populations and their samples

| Population | Sample |
|---|---|
| All koalas in Australia | 150 koalas studied in NSW |
| Every fish in Sydney Harbour | 300 fish caught in specific locations |
| All soil bacteria in a forest | Bacteria from 50 soil cores |
| All cookies in a bakery | Tasting 3 cookies to judge quality |
| All students at the university | The 200 students in this course |
| Water quality in the entire ocean | Water samples from specific locations |
| All trees in a national park | 75 trees measured in random plots |
| All possible blood test results | Blood samples from 100 patients |

*Disclaimer: use of GenAI for content generation in this slide.*

## How well does a sample represent the population?

It depends:

- **Sample size**: Larger samples are more likely to represent the population
- **Sampling method**: Random samples are more likely to be representative
- **Population variability**: More variability means larger samples are needed

In reality, we often have to balance these factors due to time, cost, and practical constraints.

## Samples vary

Different samples give different results – suppose we have a population of **1000 trees** and we randomly sample 6 tree heights. If this is done 3 times, it is likely that the samples will be different.

We have code to demonstrate this but just focus on the results for now:

```
set.seed(258)
population ← rnorm(1000, mean = 12, sd = 5)

# create samples
sample1 ← round(sample(population, size = 6), 1)
sample2 ← round(sample(population, size = 6), 1)
sample3 ← round(sample(population, size = 6), 1)
# show samples
for (i in 1:3) {
   cat(sprintf("Sample %d: ", i), get(paste0("sample", i)), "\n")
}
```

```
Sample 1:  13.7 14.6 14.8 9.6 6.5 10
Sample 2:  7.6 6.1 9.9 10.1 12.5 14.9
Sample 3:  9.8 7.9 18.4 19.1 7 26.1
```

Are the samples different? *How* different are they?

## Descriptive statistics

We use **descriptive statistics** to summarise and describe data, helping us compare and contrast.

1. **Measures of central tendency** – describe the "typical" value in a sample
   - mean, median, mode
2. **Measures of spread** – describe how much the data varies
   - standard deviation, variance (commonly used)
   - range, quartiles, IQR (for unique cases)
3. **Measures of uncertainty** – describe how confident we are in our estimates
   - standard error
   - confidence intervals

# Measures of central tendency

## Mean – also known as the *average*

- Add up all your numbers
- Divide by how many numbers you have

**i** Mathematical notation

- Population mean: $\mu = \frac{\sum_{i=1}^{N} x_i}{N}$
- Sample mean: $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

Where $x_i$ is each individual value, $N$ is population size, and $n$ is sample size.

# Mean in Excel

Excel offers several ways to calculate the mean:

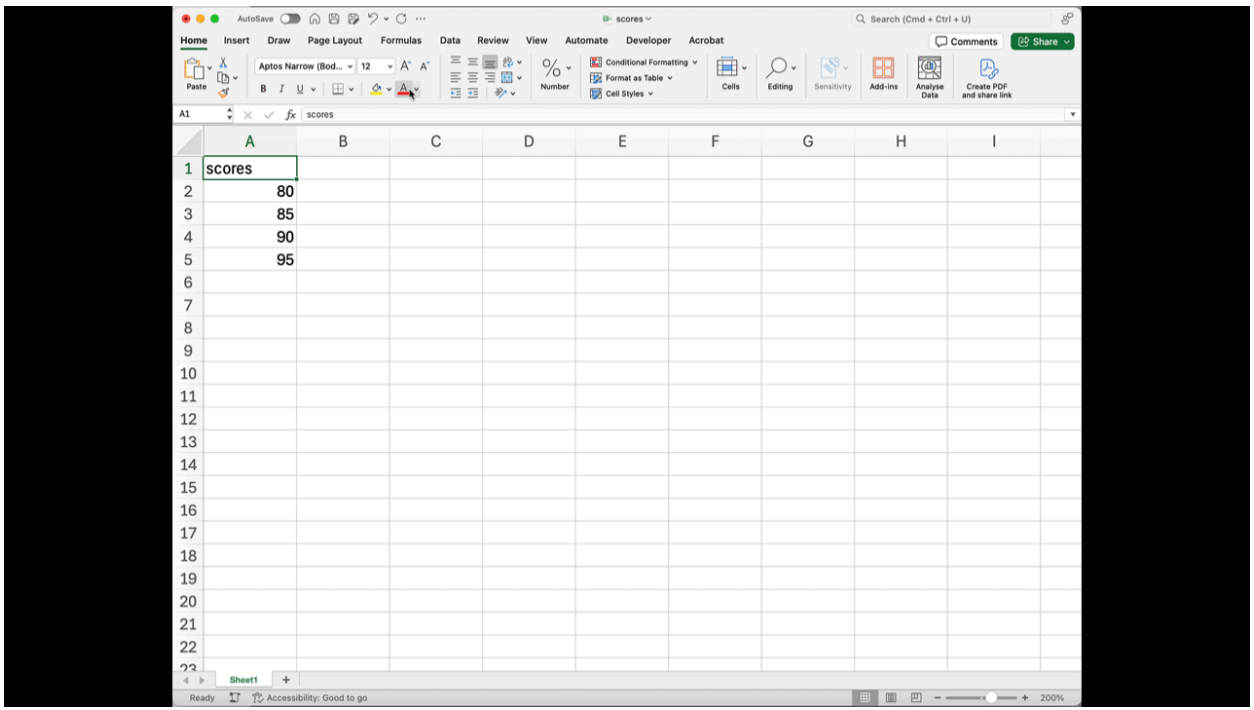
1. **Using AVERAGE function**

```
=AVERAGE(A1:A4)
```

   - Type `=AVERAGE(`
   - Select cells with your data
   - Press Enter

2. **Using AutoCalculate**

   - Select your data cells
   - Look at bottom right
   - Average shown automatically



## Mean in R

I'll show you how to do this in RStudio.

## Step 1: Create a vector of values

In R, we store data in **vectors** (lists of values):

```
# Create a vector of test scores
scores ← c(80, 85, 90, 95)

# Display the scores
scores
```

```
[1] 80 85 90 95
```

## Step 2: Calculate

Either calculate the mean manually:

Or use the `mean()` function:

```r
# Sum divided by count
sum(scores) / length(scores)
```

```
[1] 87.5
```

```r
# The mean() function does the work for us
mean(scores)
```

```
[1] 87.5
```

## Median – the middle value

The median is the middle number when your data is in order:

1. First, put your numbers in order
2. Find the middle value
3. If you have an even number of values, take the average of the two middle numbers

Example: House prices ($'000s): 450, 1100, 480, 460, 470, 420, 1400, 450, 470

Order: 450, 450, 420, 460, **470**, 470, 480, 1100, 1400

### How is it useful?

## Median in Excel

Excel provides two main ways to find the median:

1. **Using MEDIAN function**

```
=MEDIAN(A1:A9)
```

   - Type `=MEDIAN(`
   - Select your data range
   - Press Enter

2. **Alternative method**

   - Sort your data first (use the Sort functionality in the Data tab)
   - Find middle value(s)
   - If even number of values, average the middle two

## Median in R

R does all the ordering and finding the middle for us:

```
# House prices
prices ← c(450, 1100, 480, 460, 470, 420, 1400, 450, 470)

# Find median
median(prices)
```

```
[1] 470
```

Comparing the mean and median:

```
# Compare with mean
mean(prices)
```

```
[1] 633.3333
```

**Which is a better measure for house prices?**

## Mode – most frequent value

The mode is the value that appears most frequently in your data. It's particularly useful for:

- Categorical data (like blood types, eye colors)
- Finding the most common item in a group
- Data that has clear repeated values

Calculating the mode can be tricky, especially if there are multiple modes or no mode at all. This is why the mode is not commonly used in statistics.

## Questions that the mode can answer

- What is the most common blood type in a population?
- What is the most common eye color in a group of people?

## Mode in Excel

Excel provides several methods to find the mode but the simplest is to use the MODE function:

```
=MODE(A1:A10)
```

- Type `=MODE(`
- Select your data range
- Press Enter

## Using frequencies

There is no built-in function to calculate the mode, so we use the `modeest` package:

```
if(!require("modeest")) install.packages("modeest")
```

```
Loading required package: modeest
```

```
library(modeest)

df ← c(1, 2, 3, 3, 4, 5, 5, 5, 6)
mlv(df, method = "mfv")  # most frequent value
```

```
[1] 5
```

If you were to do it yourself, how would you do it in R?

# Using frequencies

Use the `table()` function to count frequencies:

```
freq_table ← table(df) # Count frequencies of each value
# Find which value(s) appear most often
modes ← as.numeric(names(freq_table[freq_table == max(freq_table)]))
modes
```

```
[1] 5
```

## Using run-length encoding

Use run-length encoding after sorting:

```
sorted_df ← sort(df) # Sort the vector first
runs ← rle(sorted_df) # Use run-length encoding to find sequences
modes ← runs$values[runs$lengths == max(runs$lengths)] # Find the value(s) with max length
modes
```

```
[1] 5
```

# Using loops

Loop through the vector and count occurrences:

```
unique_vals ← factor(df) # Create a factor of unique values
counts ← tapply(df, unique_vals, length) # Count occurrences using tapply
modes ← as.numeric(names(counts[counts == max(counts)])) # Find which values have the maximum
count
modes
```

```
[1] 5
```

**The point is that it doesn't matter how you calculate the mode, as long as you are able to do it.** Also – if you needed this – aren't you glad R has a package for it?

# Measures of spread

# A biological example

Figure 3: Source: Adobe Stock # 85659279

Imagine sampling seagrass blade lengths from two different sites in a marine ecosystem, and they have the same mean length of 15.2 cm. Are both sites the same?

- **Site A (Protected Bay)**: 15.2, 15.0, 15.3, 15.1, 15.2 centimetres
- **Site B (Wave-exposed Coast)**: 12.0, 18.0, 14.5, 16.5, 15.0 centimetres

## Comparing Different Measures

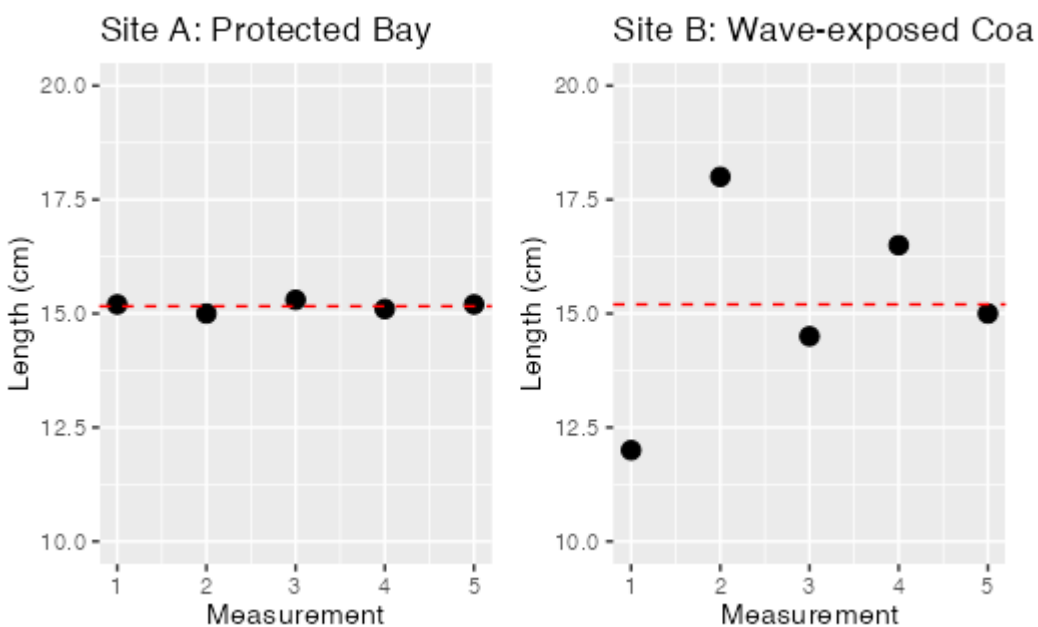
```
# Plot seagrass lengths
library(ggplot2)
library(patchwork)

seagrass_protected ← c(15.2, 15.0, 15.3, 15.1, 15.2)
seagrass_exposed ← c(12.0, 18.0, 14.5, 16.5, 15.0)

# Create plots for both sites
p1 ← ggplot() +
   geom_point(aes(x = 1:5, y = seagrass_protected), size = 3) +
   geom_hline(yintercept = mean(seagrass_protected), linetype = "dashed", color = "red") +
   labs(title = "Site A: Protected Bay", x = "Measurement", y = "Length (cm)") +
   ylim(10, 20)

p2 ← ggplot() +
   geom_point(aes(x = 1:5, y = seagrass_exposed), size = 3) +
   geom_hline(yintercept = mean(seagrass_exposed), linetype = "dashed", color = "red") +
   labs(title = "Site B: Wave-exposed Coast", x = "Measurement", y = "Length (cm)") +
   ylim(10, 20)
```

```
# Combine plots side by side
p1 + p2
```



## Why do we need measures of spread?

- Central tendency (mean, median, mode) only tells part of the story
- Spread tells us how much variation exists in our data
- Different measures of spread tell us different things:
  - **Range**: Overall spread of data
  - **IQR**: Spread of middle 50% of data
  - **Variance**: Average squared deviation from mean
  - **Standard deviation**: Average deviation in original units

## Range – The simplest measure of spread

```
# Create our seagrass data
seagrass_protected ← c(15.2, 15.0, 15.3, 15.1, 15.2)  # Protected bay
seagrass_exposed ← c(12.0, 18.0, 14.5, 16.5, 15.0)    # Wave-exposed coast

# Calculate ranges
cat("Protected bay range:", diff(range(seagrass_protected)), "cm\n")
```

```
Protected bay range: 0.3 cm
```

```
cat("Wave-exposed range:", diff(range(seagrass_exposed)), "cm\n")
```

```
Wave-exposed range: 6 cm
```

**i** Note

The range shows us that seagrass lengths are much more variable in the wave-exposed site!

## Interquartile range (IQR): The middle 50%

The IQR tells us how spread out the middle 50% of our data is:

```
# Get quartiles for protected bay
quantile(seagrass_protected)
```

```
  0%  25%  50%  75% 100%
15.0 15.1 15.2 15.2 15.3
```

- 25% of data below Q1 (1st quartile)
- 75% of data below Q3 (3rd quartile)
- IQR = Q3 - Q1

## Why use IQR?

- Ignores extreme values
- Works with skewed data
- More stable than range

## Comparing Sites Using IQR

```
# Compare IQRs
pbay ← IQR(seagrass_protected)
pbay
```

```
[1] 0.1
```

```
exbay ← IQR(seagrass_exposed)
exbay
```

```
[1] 2
```

- Protected bay IQR: 0.1 cm
- Wave-exposed IQR: 2 cm

> **i** Note
>
> The larger IQR in the wave-exposed site shows more spread in the typical seagrass lengths

## Variance: a detailed measure of spread

Variance measures how far data points are spread from their mean by:

1. Finding how far each point is from the mean
2. Squaring these distances (to handle negative values)
3. Taking the average of these squared distances

## Why use variance?

- Uses **all** data points (unlike IQR)
- Less sensitive to outliers than range
- Shows total spread in both directions

## Key points

- Measured in squared units ($cm^2$)
- Larger variance = more spread

## Calculating Variance in R

```
# Calculate variance for both sites
cat("Protected bay variance:", var(seagrass_protected), "cm²\n")
```

```
Protected bay variance: 0.013 cm²
```

```
cat("Wave-exposed variance:", var(seagrass_exposed), "cm²\n")
```

```
Wave-exposed variance: 5.075 cm²
```

> **Note**
>
> The larger variance in wave-exposed site shows more spread from the mean!

## Standard deviation: a more interpretable measure

Standard deviation (SD, or $\sigma$ for population, $s$ for sample) is the square root of variance:

- Tells us the "typical distance" from the mean
- Easy to understand - similar to saying "± value" after a mean
- Small SD means values cluster closely around mean
- Large SD means values are more spread out

## When and why to use it

- Values are in the **same units** as your data (unlike variance)
- Perfect for describing natural variation (height, weight, temperature)
- Used in many statistical tests
- Great for comparing different groups or datasets

## Interpreting standard deviation (with R)

We can describe our seagrass lengths using mean ± standard deviation:

```
# Protected bay
mean_p ← mean(seagrass_protected)
sd_p ← sd(seagrass_protected)
cat("Protected bay:", round(mean_p, 1), "±", round(sd_p, 2), "cm\n")
```

```
Protected bay: 15.2 ± 0.11 cm
```

```
# Wave-exposed
mean_e ← mean(seagrass_exposed)
sd_e ← sd(seagrass_exposed)
cat("Wave-exposed:", round(mean_e, 1), "±", round(sd_e, 2), "cm\n")
```

```
Wave-exposed: 15.2 ± 2.25 cm
```

Tip

The ± tells us about the typical variation around the mean. Larger values indicate more spread!

## Comparing spread measures

| Measure | Protected Bay | Wave-exposed Coast | What it Tells Us |
|---|---|---|---|
| Range | 0.3 cm | 6 cm | Overall spread (sensitive to outliers) |
| IQR | 0.1 cm | 2 cm | Middle 50% spread (ignores extremes) |
| Variance | 0.01 $cm^2$ | 5.07 $cm^2$ | Average squared distance from mean |
| SD | 0.11 cm | 2.25 cm | Average distance from mean (in original units) |

## Key Observations

- Wave-exposed site shows consistently more variation
- Each measure gives a different perspective
- Choose based on your data and goals
- Standard deviation is most commonly used in research papers

## Range and IQR in Excel

Common Excel functions for measuring spread:

1. **Range**: Use `MAX()` and `MIN()`

```
=MAX(A1:A10) - MIN(A1:A10)
```

2. **Quartiles and IQR**: Use `QUARTILE.INC()`

```
For Q1: =QUARTILE.INC(A1:A10, 1)
For Q3: =QUARTILE.INC(A1:A10, 3)
For IQR: =QUARTILE.INC(A1:A10, 3) - QUARTILE.INC(A1:A10, 1)
```

## Variance and standard deviation in Excel

Statistical functions for variance and standard deviation:

1. **Sample Variance**: Use `VAR.S()`

```
=VAR.S(A1:A10)
```

2. **Sample Standard Deviation**: Use `STDEV.S()`

```
=STDEV.S(A1:A10)
```

> Tip
>
> Use `.P` instead of `.S` for population measures:
>
> - `VAR.P()` for population variance
> - `STDEV.P()` for population standard deviation

# References and Resources

## Core Reading

- Quinn & Keough (2024). *Experimental Design and Data Analysis for Biologists*. Cambridge University Press. **Chapter 2**: Things to know before proceeding.
- Canvas site for lecture notes and additional resources

# Thanks!