AN INTRO TO R, RSTUDIO & CAUSAL INFERENCE

Camila Pacheco Katrín Björnsdóttir

WELCOME!

SESSION GOALS

By the end of this session, you'll be able to:

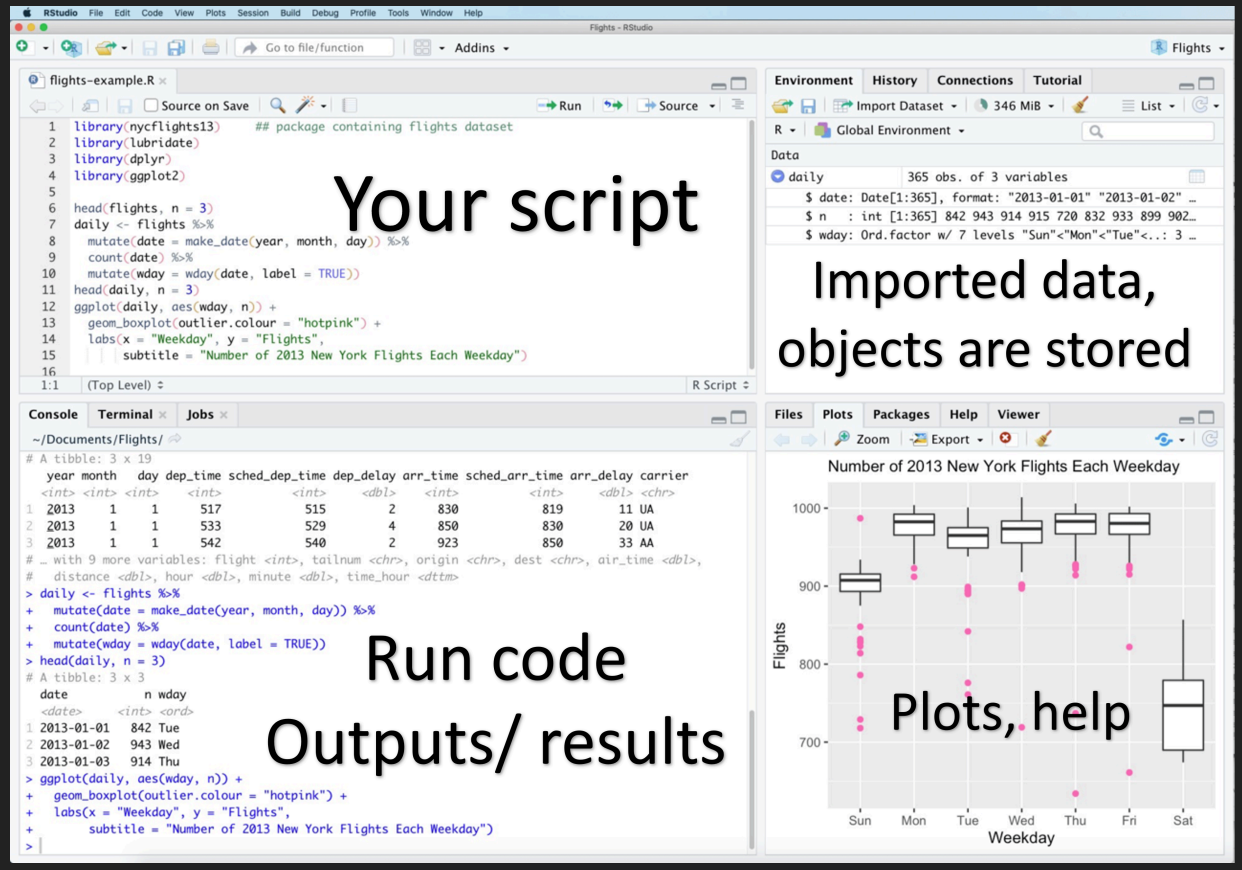
- ✅ Navigate the RStudio IDE
- ✅ Import and clean datasets using `tidyverse`
- ✅ Create simple plots using `ggplot2`
- ✅ Understand the basics of tidy data
- ✅ Introduced to causal inference

WHAT IS R ?

- Originally developed for statistical computing and graphics
- Evolved into a full-featured programming language for data science
- Especially powerful in ecological and environmental science 🐛 🌍

WHAT IS RSTUDIO?

- RStudio is a user-friendly interface (IDE) for R that makes your work more efficient.



R IS PRONE TO ERRORS ⚠️

> 💡 **Tip**
>
> Use `Ctrl + Enter` to run code in RStudio

- 💥 Typos, wrong letter case, missing commas or brackets = broken code
- 🧠 These are common—don’t panic!
- 🔍 Double-check syntax, use the error message as a clue

R STUDIO PROJECTS 📁

- Keeps your files organized in one place
- Stores scripts, data, outputs, and more
- To create a new project:
- Go to `File` → `New Project…` → `New Directory` (or `Existing Directory`)
- If using an existing folder: `New Project` → `Existing Directory`

> **Tip**
>
> Use projects to avoid broken file paths and messy folders.

COURSE MATERIAL 📘

> **Important**
>
> https://github.com/Lacapary/BIO503

CODING ETIQUETTE

USE COMMENTS TO ORGANIZE YOUR SCRIPT

```
1 #Library----
2
3 #Some code----
4
5 #Example----
```

COMMENTS FOR YOURSELF AND FOR OTHERS

There are two types of lines: those that start with the symbol #, and those that do not.

```r
# This is a comment in R
# Comments are used to provide explanations or annotate

x <- 9  # Assigning the value to the variable x
```

R PACKAGE

- Package is a collection of R functions, data sets, and other resources bundled together for specific purposes.
- To install a package, type:
  - `install.packages("package-name")`
- **You only need to install packages once**
- Load the packages, type:
  - `library(package-name)`

EXAMPLE

```
1 install.packages("dplyr")
2 library(dplyr)
```

INTRO TO tidyverse 📦

PIPE OPERATOR

- One of the key features of tidyverse is the possibility to chain functions in an effective way using the pipe operator `%>%` or `|>`.
- Pipes pass the results from one function directly into the next function connected to each other
- **The pipe basically means “and then”.**

> **Note**
>
> the keyboard shorcut for `|>` is `Ctrl+Shift+M` (Windows & Linux) or `Cmd+Shift+M` (Mac).

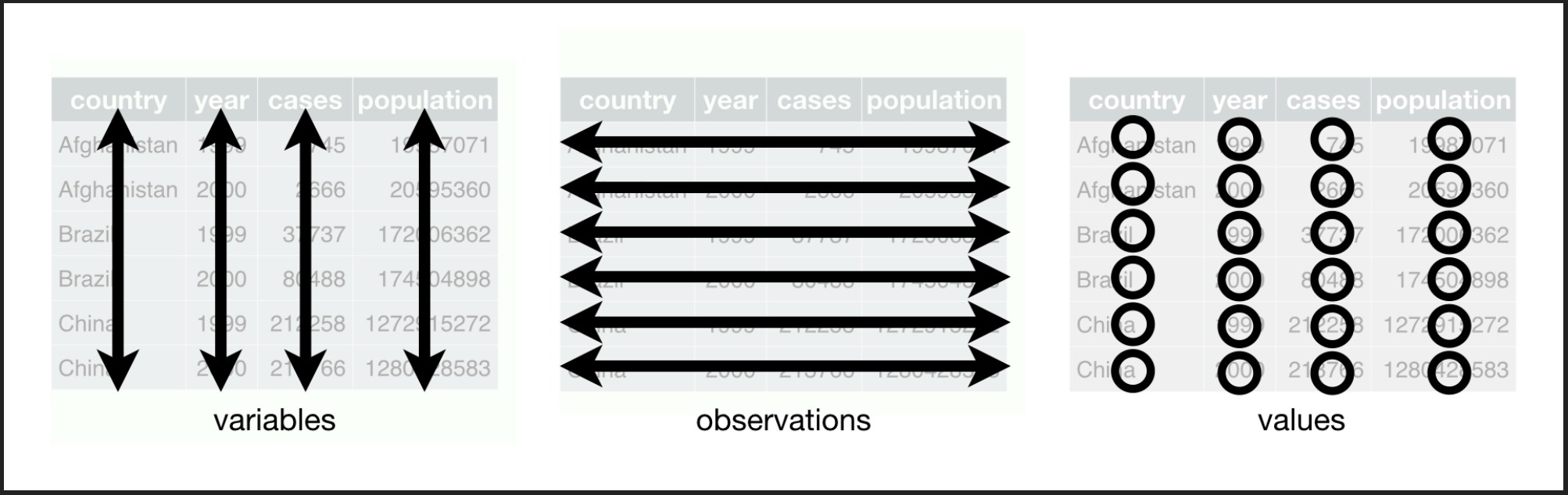
PIPE OPERATOR

First, I'll grab the coffee grounds, then I'll fill up the coffee maker with water, hit the start button, wait for it to brew, and finally pour myself a cup

```
1 get("coffee_groundS","23")   |>
2   fill_up("coffee_maker","water")    |>
3   on(start_button)   |>
4   put(cup)
```

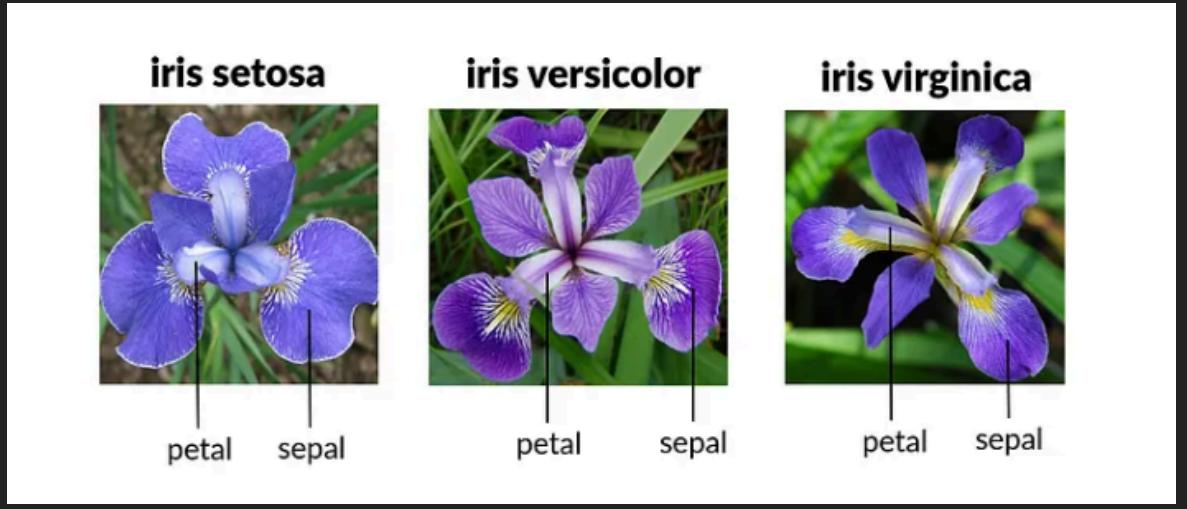
TIDY DATA CONCEPT

- Each variable forms a column
- Each observation forms a row
- Each type of observational unit forms a table



USING THE `iris` DATASET

- The iris flower dataset was collected by Edgar Anderson, an American botanist, in the 1920s.
- It includes measurements of 150 flowers across 3 species: *setosa*, *versicolor*, and *virginica*.



IMPORTING DATA IN R

R can import data from files in many different formats.

- csv files with the `readr` package
- excel files with the `readxl` package
- xlm files with the `xml2` package
- netcdf files with the `ncdf4` package
- shapefiles with the `sf` package

READING DATA

```
1 library(tidyverse)
2 iris <- read_csv("Data/iris.csv")
```

```
1 # R basic function
2 iris <-read.csv("Data/iris.csv")
```

EXPLORING THE DATA

Check that your data was imported without any mistakes

```r
head(iris)     # Displays the first rows of a df
```

```
# A tibble: 6 × 6
   ...1 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
  <dbl>        <dbl>       <dbl>        <dbl>       <dbl> <chr>
1     1          5.1         3.5          1.4         0.2 setosa
2     2          4.9         3            1.4         0.2 setosa
3     3          4.7         3.2          1.3         0.2 setosa
4     4          4.6         3.1          1.5         0.2 setosa
5     5          5           3.6          1.4         0.2 setosa
6     6          5.4         3.9          1.7         0.4 setosa
```

```r
tail(iris)   # Displays the last rows of a df
```

```
# A tibble: 6 × 6
   ...1 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
  <dbl>        <dbl>       <dbl>        <dbl>       <dbl> <chr>
1   145          6.7         3.3          5.7         2.5 virginica
2   146          6.7         3            5.2         2.3 virginica
3   147          6.3         2.5          5           1.9 virginica
4   148          6.5         3            5.2         2   virginica
5   149          6.2         3.4          5.4         2.3 virginica
6   150          5.9         3            5.1         1.8 virginica
```

EXPLORING THE DATA

```r
summary(iris) # Gives you a summary of the data
```

```
     ...1         Sepal.Length    Sepal.Width     Petal.Length
Min.   :  1.00   Min.   :4.300   Min.   :2.000   Min.   :1.000
1st Qu.: 38.25   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
Median : 75.50   Median :5.800   Median :3.000   Median :4.350
Mean   : 75.50   Mean   :5.843   Mean   :3.057   Mean   :3.758
3rd Qu.:112.75   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
Max.   :150.00   Max.   :7.900   Max.   :4.400   Max.   :6.900
 Petal.Width      Species
Min.   :0.100   Length:150
1st Qu.:0.300   Class :character
Median :1.300   Mode  :character
Mean   :1.199
3rd Qu.:1.800
Max.   :2.500
```

MANIPULATING DATA

FROM WIDE TO LONG

The iris data are organized is “wide” format. Let’s transform in “long” format

```
iris_long <- iris |> pivot_longer(
                       cols = -Species,
                       names_to = "trait",
                       values_to = "measurement")
head(iris_long)
```

```
# A tibble: 6 × 3
  Species trait        measurement
  <chr>   <chr>              <dbl>
1 setosa  ...1                   1
2 setosa  Sepal.Length         5.1
3 setosa  Sepal.Width          3.5
4 setosa  Petal.Length         1.4
5 setosa  Petal.Width          0.2
6 setosa  ...1                   2
```

GROUP_BY AND SUMMARIZE

```r
iris_means <- iris |>
  group_by(Species) |>
  summarize(SL_mean = mean(Sepal.Length),
            SL_se = sd(Sepal.Length)/sqrt(n()))

head(iris_means)
```

```
# A tibble: 3 × 3
  Species    SL_mean  SL_se
  <chr>        <dbl>  <dbl>
1 setosa        5.01 0.0498
2 versicolor    5.94 0.0730
3 virginica     6.59 0.0899
```

FILTER - SUBSET ROWS

```r
iris_versicolor <- iris |>  filter( Species == "versicol
iris_pl<-iris |>  filter(Petal.Length > 2)
```

SELECT - SUBSET COLUMNS

```
1 iris_subset<-iris |> select(Species, Petal.Width, Petal.
```

MUTATE

```
1  iris_log <-iris |> mutate(log.Sepal.length = log(Sepal.L
2
3  head(iris_log)
```

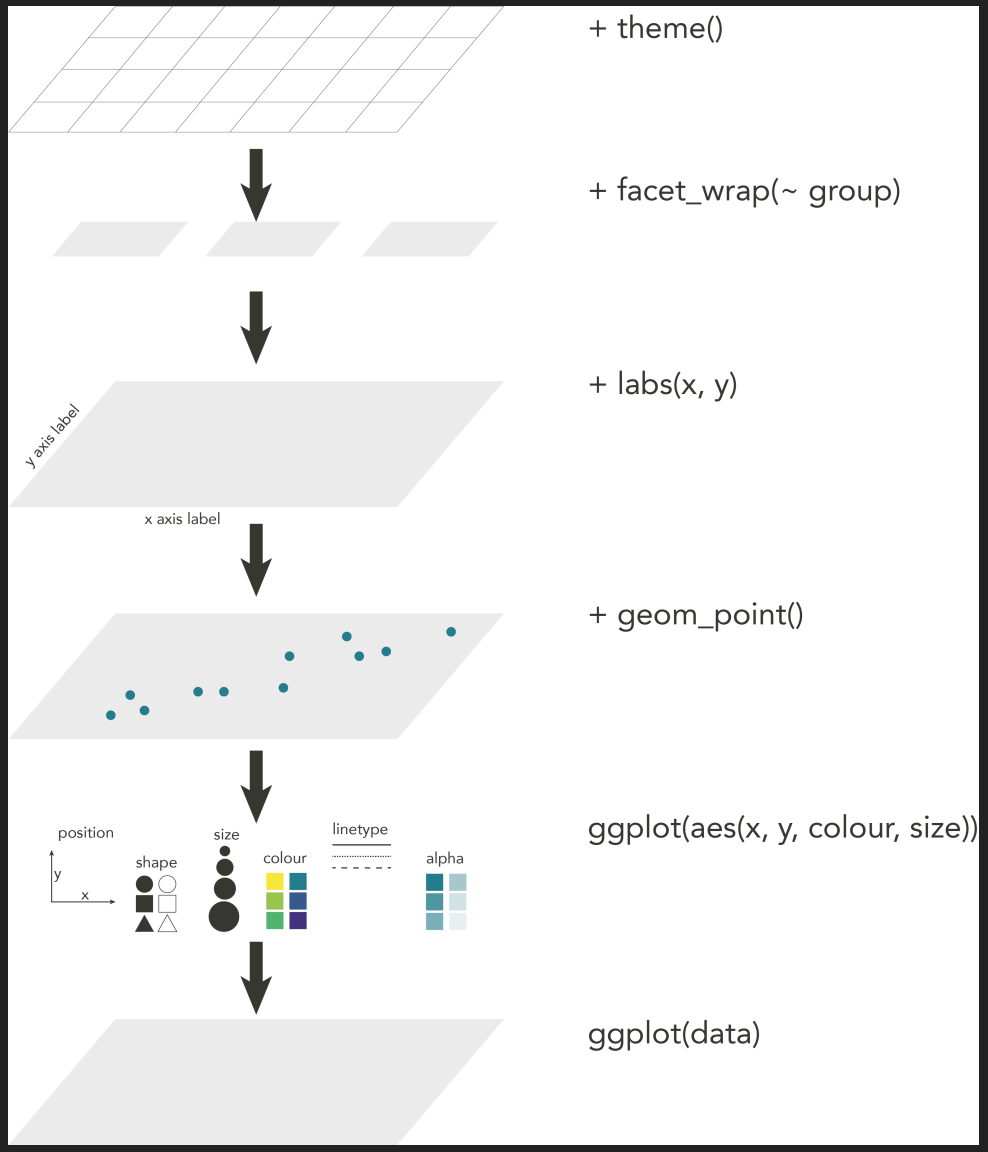
```
# A tibble: 6 × 7
   ...1 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
  <dbl>        <dbl>       <dbl>        <dbl>       <dbl> <chr>
1     1          5.1         3.5          1.4         0.2 setosa
2     2          4.9         3            1.4         0.2 setosa
3     3          4.7         3.2          1.3         0.2 setosa
4     4          4.6         3.1          1.5         0.2 setosa
5     5          5           3.6          1.4         0.2 setosa
6     6          5.4         3.9          1.7         0.4 setosa
# ℹ 1 more variable: log.Sepal.length <dbl>
```

PLOTTING DATA

You can plot graphs using the `ggplot2` package (part of the tidyverse).

> **Note**
>
> `ggplot` functions are chained using a `+`

GGPLOT LAYERS



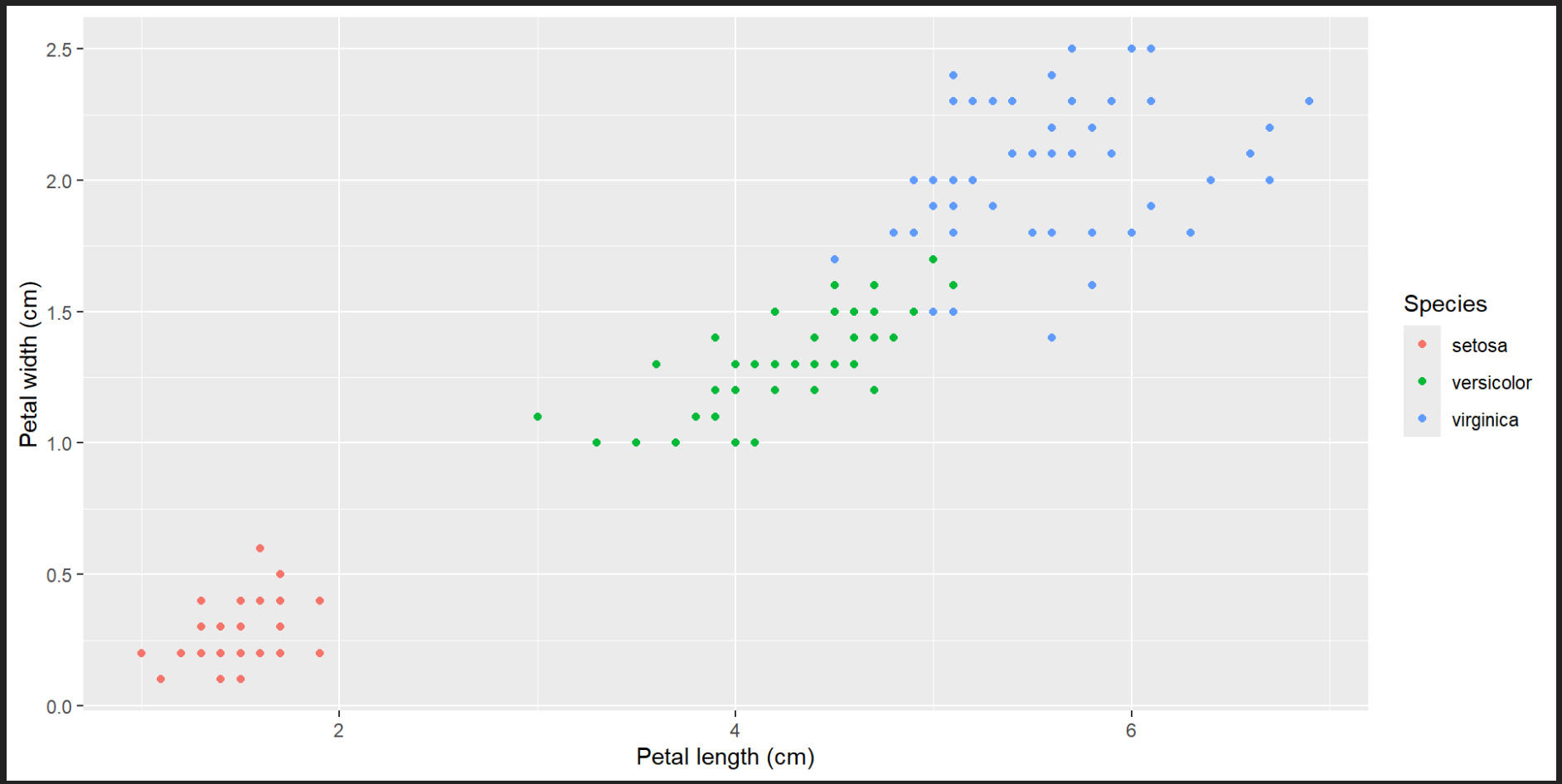
Source: https://biostats-r.github.io/

PLOTTING IRIS

Lets say we want to plot the relationship between petal length and petal width for each species.

```r
p<-iris %>%
  ggplot(aes(
    x = Petal.Length,
    y = Petal.Width,
    color = Species # to assign a color to each group
  )) +
  geom_point() + # to plot a scatter plot
  labs(
    x = "Petal length (cm)",
    y = "Petal width (cm)"
  )
```

PLOTTING IRIS



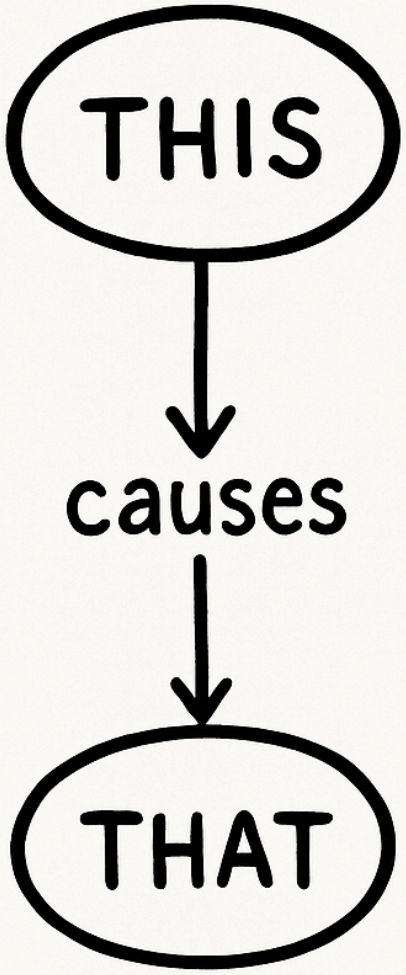
`ggplot` offers many plotting possibilities which we will go further into later.
If you are eager, you can learn more here or here.

SAVING PLOTS

You can save a plot by clicking on the `Export` button in the `Plots` window (bottom right window by default).

Save your plots as `.svg` if your text editor supports it and if you are not limited by file sizes. Otherwise, save your plots as `.png`.

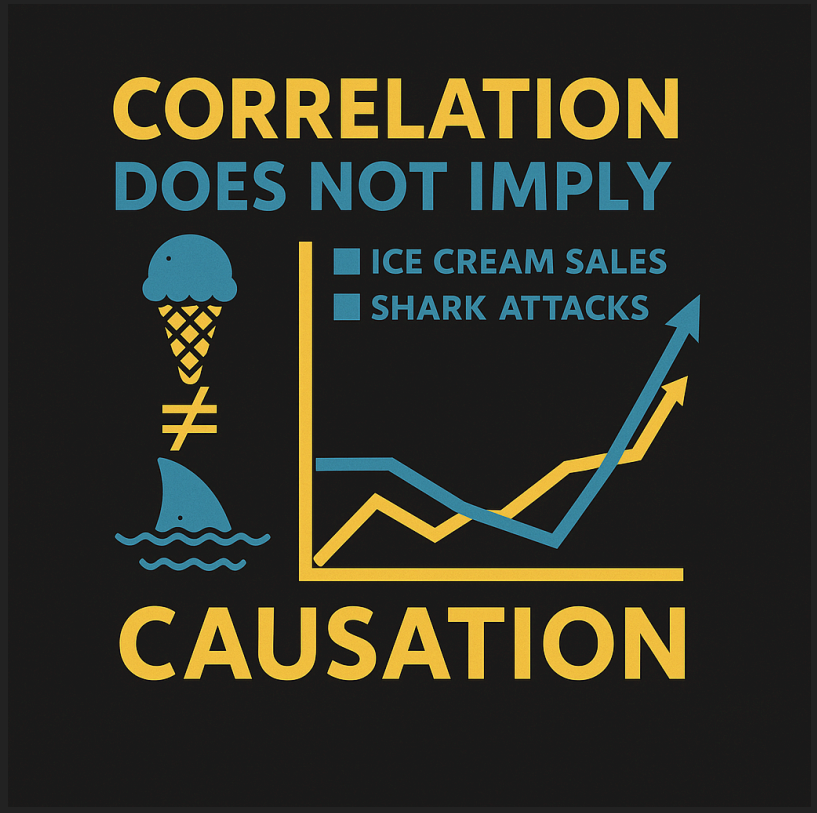
```
1 p |> ggsave("plot.png")
```



Causal Inference

WHAT IS CAUSAL INFERENCE? 🤔

It’s the process of determining cause-and-effect relationship.



Moving beyond **correlation** to understand **causal relationships**

Causal inference is about answering questions like: **“Does X cause Y?”**

WHY IS THIS IMPORTANT IN ECOLOGY?

- Ecological systems are complex
- Many things can affect each other at the same time
- We need tools to figure out which relationships are **real causes**, not just coincidences

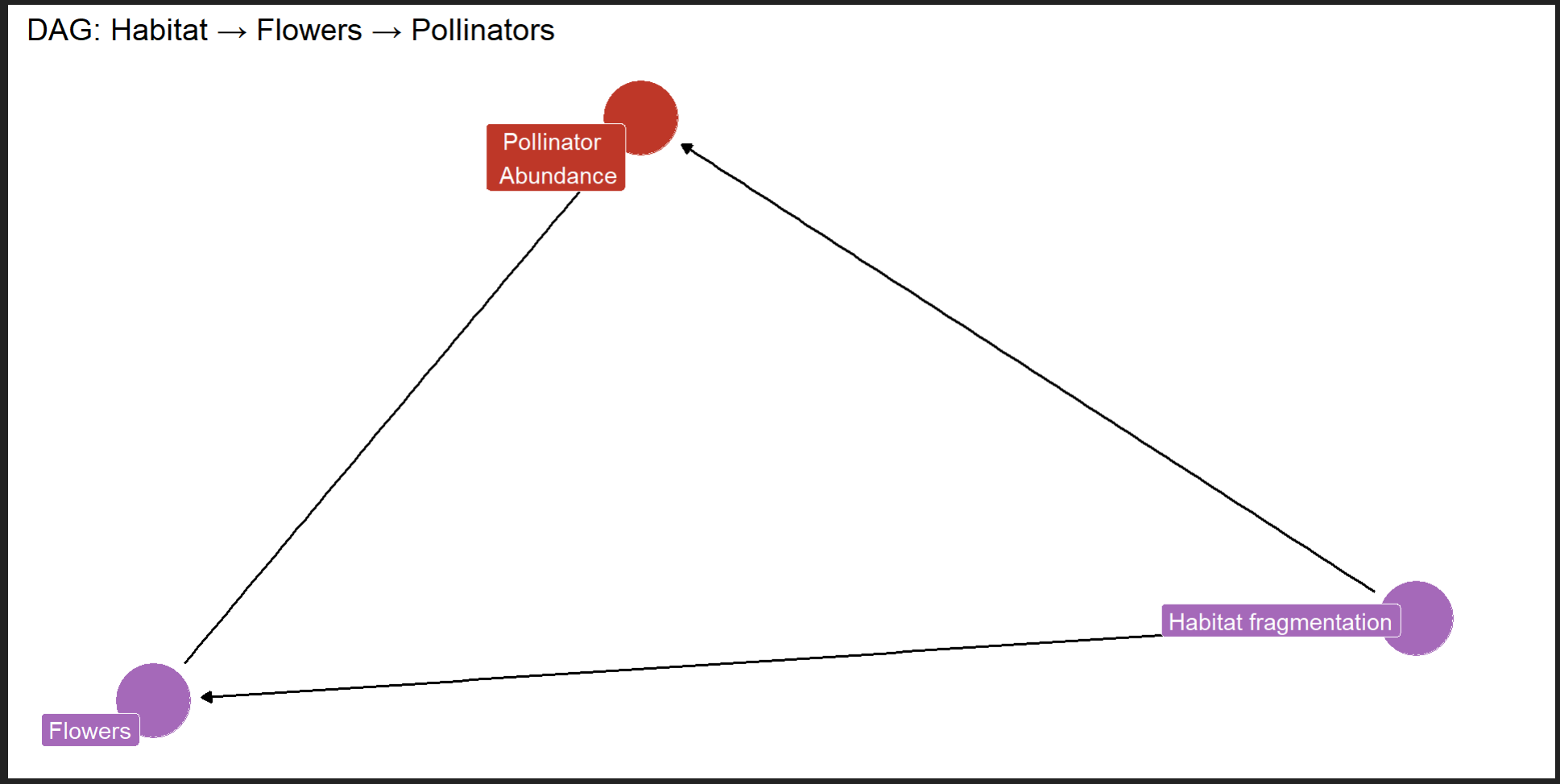
A SIMPLE TOOL: DAGS

A **DAG** (Directed Acyclic Graph) is a simple diagram to help us think clearly about cause and effect.

- Circles = variables (things we measure)
- Arrows = “causes”

DAG EXAMPLE

Fragmentation affects pollinators **directly** and **indirectly** (by reducing flowers).



HOW DAGS HELP

- They help us ask better research questions
- They show us what to measure and control for
- They help us avoid confusing correlation with causation

ANY QUESTIONS

TO GO FURTHER

- Visit: Coding Club
- EDGE coding club: Every 15 days, we have a coding club on Thursdays at 1:00 pm in the 5405 Rosshavet room (Natrium)

WHAT IS NEXT

- 15 of April 8:15am -12m *2125 Marelden*
- Practice using R (self-study)
  - Reviewing and practicing the material we covered in today's lecture
  - Linear models