

Introduction to R and Rstudio

AUTHORS

Camila Pacheco

Katrín Björnsdóttir

Our mission

- Get you familiar with the R environment
- Teach you how to import and work with datasets
- Get you familiar with tidy data and [Tidyverse](#)
- Show you how to make simple plots in [ggplot](#)

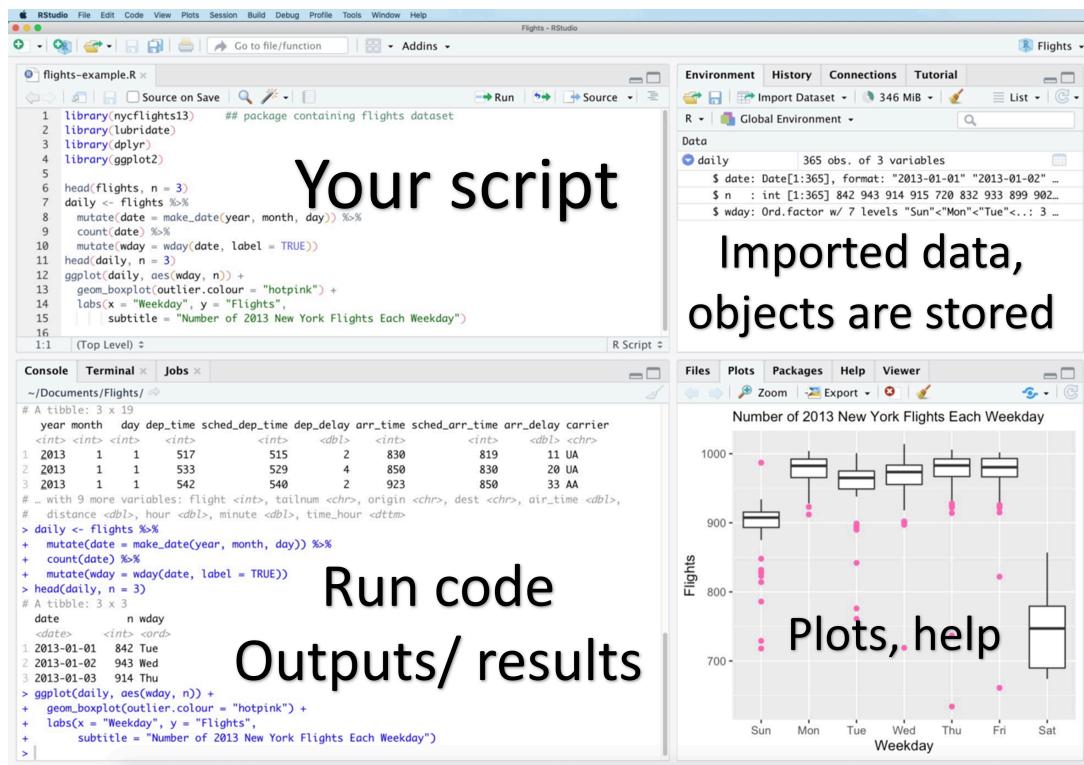
What is R ?



- Originally developed for statistical computing and graphics
- R has evolved into a versatile programming language

What is RStudio?

RStudio is a user-friendly interface for R



Your script
Run code
Outputs/ results

Imported data,
objects are stored

: Plots, help

R is prone to errors

- ● Such as typos, using the wrong letter case, forgetting a quote, bracket or comma. Such mistakes will break your code and throw an error.
- ❗ These type of errors are the most common, so always double-check your code whenever R is unhappy 😞.

Course material

<https://github.com/Lacapary/BIO503>

R studio Projects

- It is a convenient way to organize your work in RStudio -it creates a dedicated directory (folder) on your computer where you can store all the files related to your project, including R scripts, data files, documentation, and more.
- To create a new project, go to:
 - **File > New Project... > New Directory (or Existing Directory)**
- If you want to create your project from an existing folder:

- o > **New Project** and choose a **Directory name** for your project.

Work directory

```
setwd("/Your directory")
```

Code versus comment

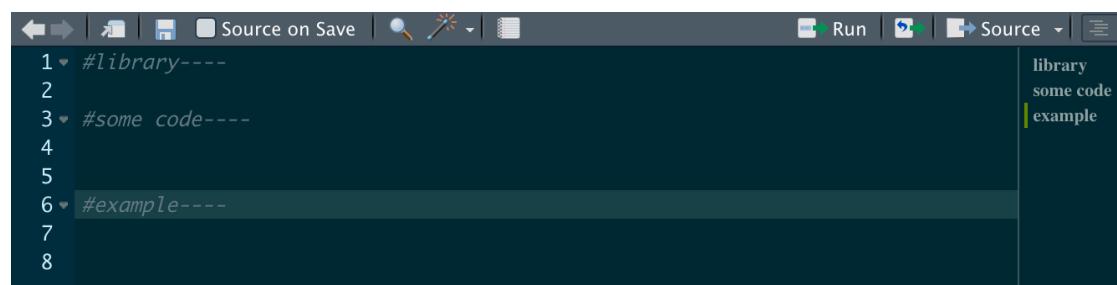
There are two types of lines: those that start with the symbol **#**, and those that do not.

```
1 # This is a comment in R
2 # Comments are used to provide explanations or annotate the code
3
4 x <- 5 # Assigning the value 5 to the variable x
```

Comments to index your scripts

```
#Library----
#Some code----
#Example----
```

Script outline in Rstudio



R package

- Package is a collection of R functions, data sets, and other resources bundled together for specific purposes.
- To install a package, type:

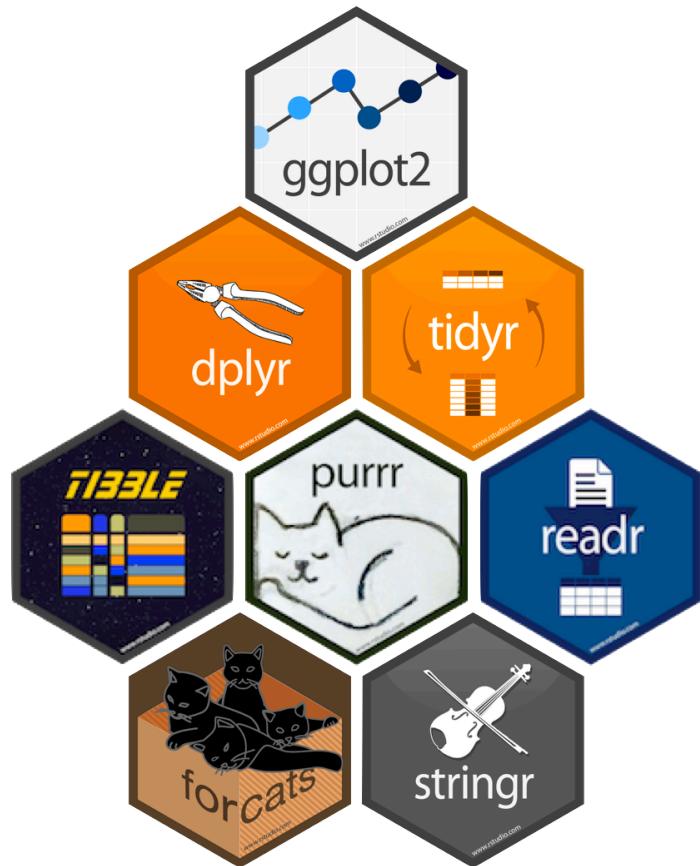
- `install.packages("package-name")`
- **You only need to install packages once**
- Load the packages, type:
 - `library(package-name)`

Example

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

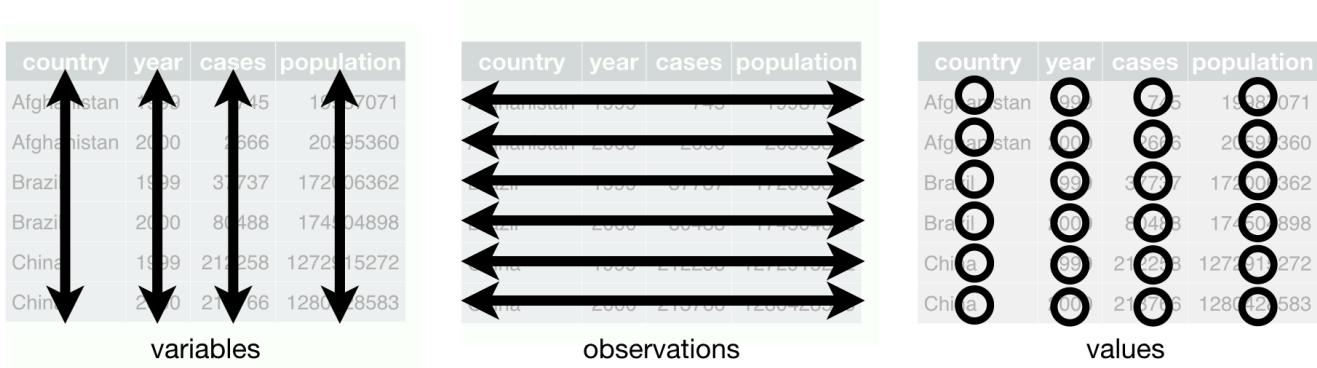
Tidyverse

- It is a collection of R packages designed to make data science tasks easier and more efficient.



About tidy data

Tidy data is a standard, consistent way to organize tabular data. Briefly, tidy data follows a short series of rules:



Source: R for Data Science. Grolemund and Wickham.

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 via a `%>%` or `|>`, making the code easy to read and write.
- **The pipe basically means “and then”.**
- **Note:** the keyboard shortcut for `|>` is `Ctrl+Shift+M` (Windows & Linux) or `Cmd+Shift+M` (Mac).

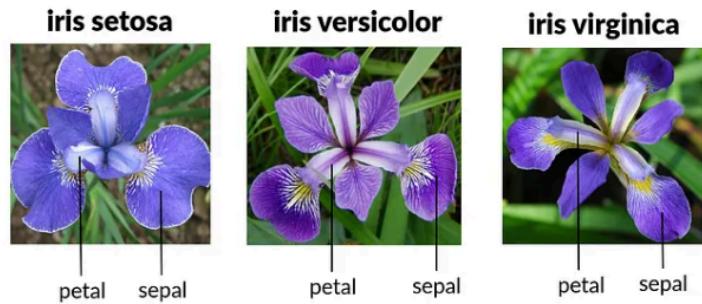
Pipe operator

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

```
get("coffee_grounds", "23") |>  
  fill_up("coffee_maker", "water") |>  
  on(start_button) |>  
  put(cup)
```

Iris dataset

The iris [flower dataset](#) was collected by Edgar Anderson, an American botanist, in the 1920s. This data was used by statistician Ronald Fisher to demonstrate statistical methods of classification.



Importing data in R

R can import data from files in many different formats. For example:

- 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

Formats

Function	Value Separator	Decimal Separator
<code>read_csv()</code>	,	.
<code>read_csv2()</code>	;	,
<code>read_tsv()</code>	tab	.
<code>read_delim()</code>	custom character	.
<code>read_table2()</code>	space	.

Reading data

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

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

```
— Attaching core tidyverse packages ————— tidyverse 2.0.0 —
✓ dplyr     1.1.4     ✓ readr     2.1.5
✓ forcats   1.0.0     ✓ stringr   1.5.1
✓ ggplot2   3.5.0     ✓ tibble    3.2.1
✓ lubridate 1.9.3     ✓ tidyr    1.3.1
✓ purrr    1.0.2

— Conflicts ————— tidyverse_conflicts() —
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()

i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

Exploring the data

Check that your data was imported without any mistakes

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

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5         1.4         0.2  setosa
2          4.9         3.0         1.4         0.2  setosa
3          4.7         3.2         1.3         0.2  setosa
4          4.6         3.1         1.5         0.2  setosa
5          5.0         3.6         1.4         0.2  setosa
6          5.4         3.9         1.7         0.4  setosa
```

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

```
Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
145        6.7         3.3         5.7         2.5 virginica
146        6.7         3.0         5.2         2.3 virginica
147        6.3         2.5         5.0         1.9 virginica
148        6.5         3.0         5.2         2.0 virginica
149        6.2         3.4         5.4         2.3 virginica
150        5.9         3.0         5.1         1.8 virginica
```

Exploring the data

```
glimpse(iris) # Tells you variables types
```

```
Rows: 150
Columns: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4...
$ Sepal.Width   <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3...
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1...
```

```
$ Petal.Width <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0...
$ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, s...
```

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

```
Sepal.Length   Sepal.Width    Petal.Length   Petal.Width  
Min. :4.300   Min. :2.000   Min. :1.000   Min. :0.100  
1st Qu.:5.100 1st Qu.:2.800  1st Qu.:1.600  1st Qu.:0.300  
Median :5.800  Median :3.000  Median :4.350  Median :1.300  
Mean   :5.843  Mean   :3.057  Mean   :3.758  Mean   :1.199  
3rd Qu.:6.400 3rd Qu.:3.300  3rd Qu.:5.100  3rd Qu.:1.800  
Max.   :7.900  Max.   :4.400  Max.   :6.900  Max.   :2.500  
  
Species  
setosa      :50  
versicolor:50  
virginica :50
```

Manipulating data



From wide to long

The iris data are organized in "wide" format. Let's transform it 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  
<fct>  <chr>       <dbl>  
1 setosa Sepal.Length    5.1  
2 setosa Sepal.Width     3.5  
3 setosa Petal.Length    1.4  
4 setosa Petal.Width     0.2  
5 setosa Sepal.Length    4.9  
6 setosa Sepal.Width     3
```

Group_by and summarize

```
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  
<fct>     <dbl>   <dbl>  
1 setosa     5.01  0.0498  
2 versicolor 5.94  0.0730  
3 virginica  6.59  0.0899
```

Filter - subset rows

```
iris_versicolor <- iris |> filter( Species == "versicolor")  
  
iris_no_versicolor <- iris |> filter( Species != "versicolor")  
  
iris_pl<-iris |> filter(Petal.Length > 2)
```

Select - subset columns

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

Mutate

```
iris_log <-iris |> mutate(log.Sepal.length = log(Sepal.Length))
```

```
head(iris_log)
```

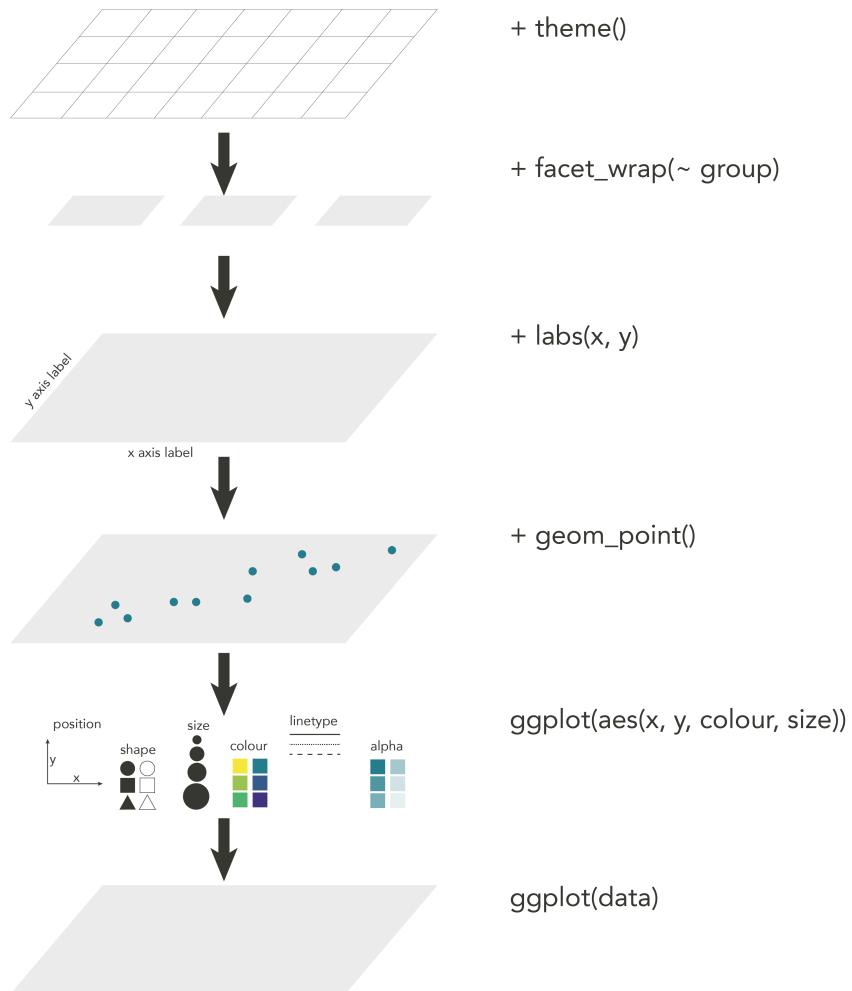
```
Sepal.Length Sepal.Width Petal.Length Petal.Width Species log.Sepal.length
1          5.1        3.5       1.4      0.2   setosa     1.629241
2          4.9        3.0       1.4      0.2   setosa     1.589235
3          4.7        3.2       1.3      0.2   setosa     1.547563
4          4.6        3.1       1.5      0.2   setosa     1.526056
5          5.0        3.6       1.4      0.2   setosa     1.609438
6          5.4        3.9       1.7      0.4   setosa     1.686399
```

Plotting data

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

Note: `ggplot` functions are chained using a `+` sign. This is because `ggplot` does not pass an object to a function but add different layers on top of each other.

ggplot layers



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

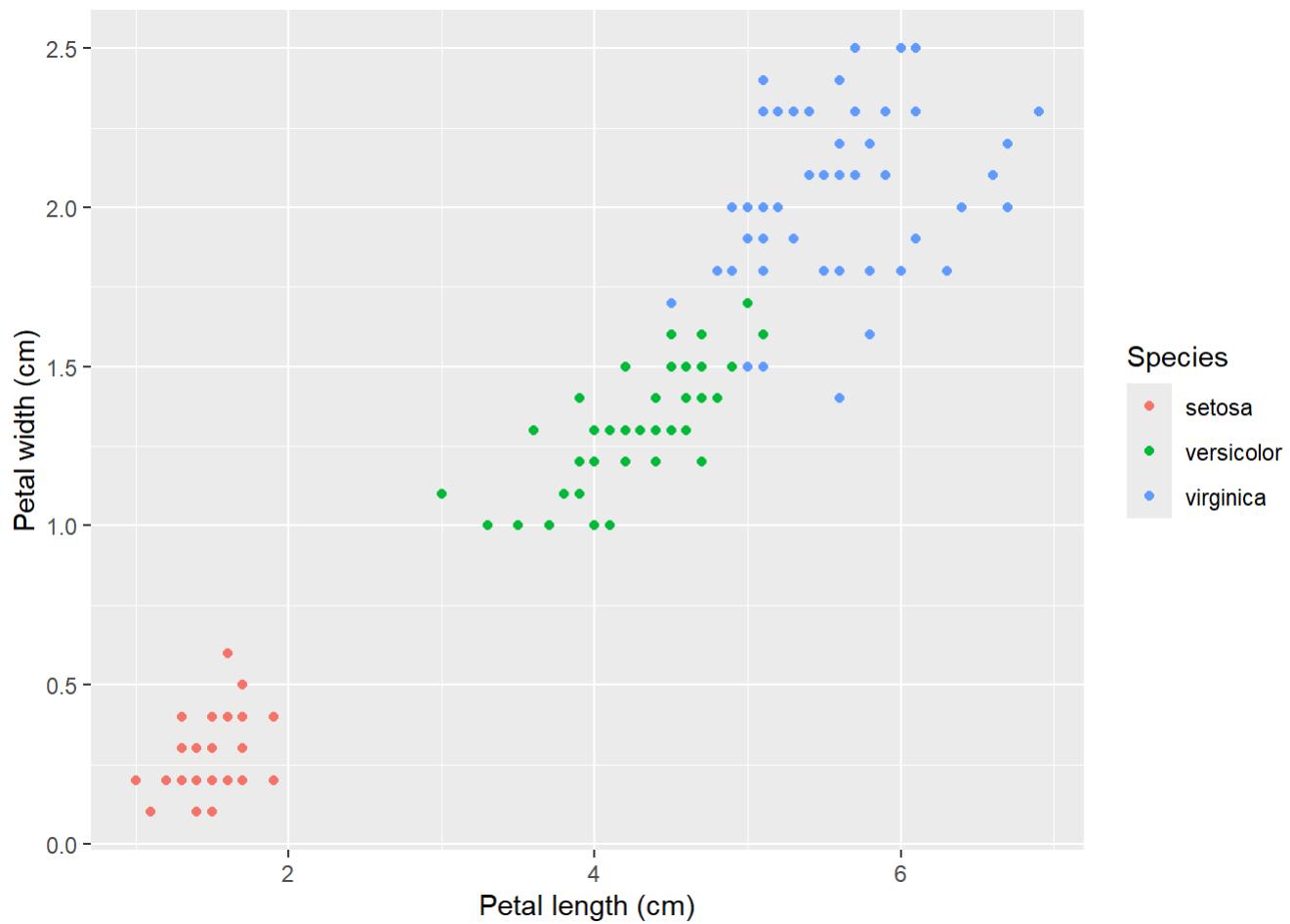
Plotting Iris

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

You can visualize such relationship by plotting petal length against petal width in a scatter plot:

```
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

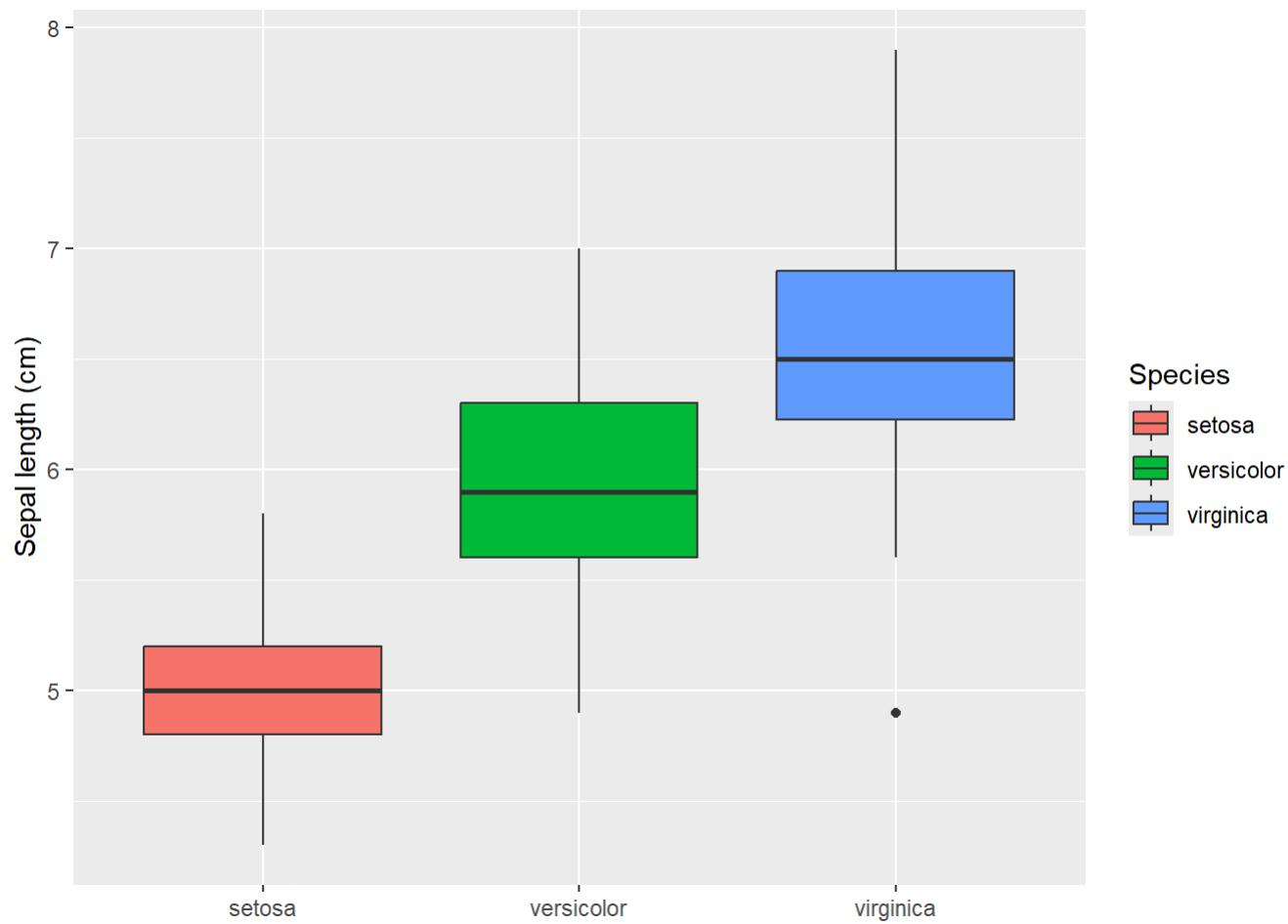


Plotting Iris

Now lets say you would like to make a boxplot to compare the sepal length between the three species.

```
p<-iris %>%
  ggplot(aes(
    x = Species,
    y = Sepal.Length,
    fill = Species
  )) +
  geom_boxplot() + # to plot a boxplot
  labs(
    x = NULL, # to remove the label of the x axis
    y = "Sepal length (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`.

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

Modelling

Now we want to test if there is a relationship between petal length and petal width. For that we run a simple linear model.

```
iris.m1 <- lm(Petal.Length ~ Petal.Width, data = iris)
summary(iris.m1)
```

Call:

```
lm(formula = Petal.Length ~ Petal.Width, data = iris)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.33542	-0.30347	-0.02955	0.25776	1.39453

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)							
(Intercept)	1.08356	0.07297	14.85	<2e-16 ***							
Petal.Width	2.22994	0.05140	43.39	<2e-16 ***							

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'. '	0.1	' '	1

Residual standard error: 0.4782 on 148 degrees of freedom

Multiple R-squared: 0.9271, Adjusted R-squared: 0.9266

F-statistic: 1882 on 1 and 148 DF, p-value: < 2.2e-16

The summary function gives you the output from the model. We can see that petal width significantly effects petal length (as you might have expected). Our model did very good, explaining about 93% of the variation in petal length.

Modelling

But do we expect there to be a difference between species Lets add species as an interaction to make it a bit more complicated.

```
iris.m2 <- lm(Petal.Length ~ Petal.Width * Species, data = iris)
summary(iris.m2)
```

Call:

```
lm(formula = Petal.Length ~ Petal.Width * Species, data = iris)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.84099	-0.19343	-0.03686	0.16314	1.17065

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
--	----------	------------	---------	----------

```

(Intercept)           1.3276    0.1309   10.139 < 2e-16 ***
Petal.Width          0.5465    0.4900   1.115   0.2666
Speciesversicolor   0.4537    0.3737   1.214   0.2267
Speciesvirginica    2.9131    0.4060   7.175  3.53e-11 ***
Petal.Width:Speciesversicolor 1.3228    0.5552   2.382   0.0185 *
Petal.Width:Speciesvirginica  0.1008    0.5248   0.192   0.8480
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3615 on 144 degrees of freedom
Multiple R-squared:  0.9595,    Adjusted R-squared:  0.9581
F-statistic: 681.9 on 5 and 144 DF,  p-value: < 2.2e-16

```

Modelling

Now lets visualize the relationship using ggplot (notice how we only included stat`smooth to our original scatter plot).

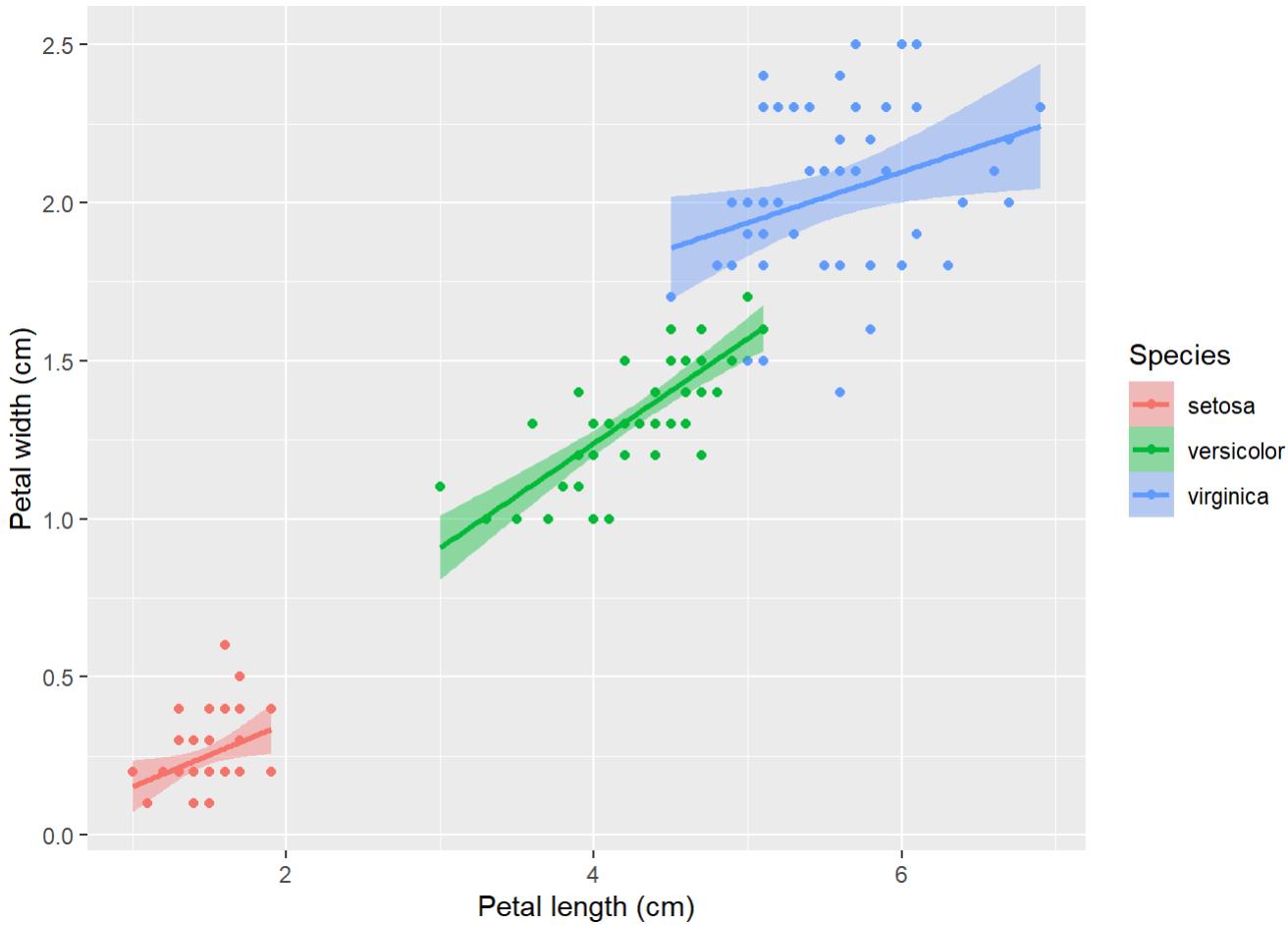
```

p<-iris %>%
  ggplot(aes(
    x = Petal.Length,
    y = Petal.Width,
    color = Species)) + # to assign a color to each group
  geom_point() +
  stat_smooth(method = "lm", aes(fill = Species, colour = Species)) +
  labs(
    x = "Petal length (cm)",
    y = "Petal width (cm)")

```

Modelling

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



Modelling

In the previous plot we just made we used the default ggplot settings. Although this is a great looking plot, we can make it even better since ggplot has a lot of power in customization, here is one example but feel free to play around with this, it can actually be quite satisfying when you get the hang of it.

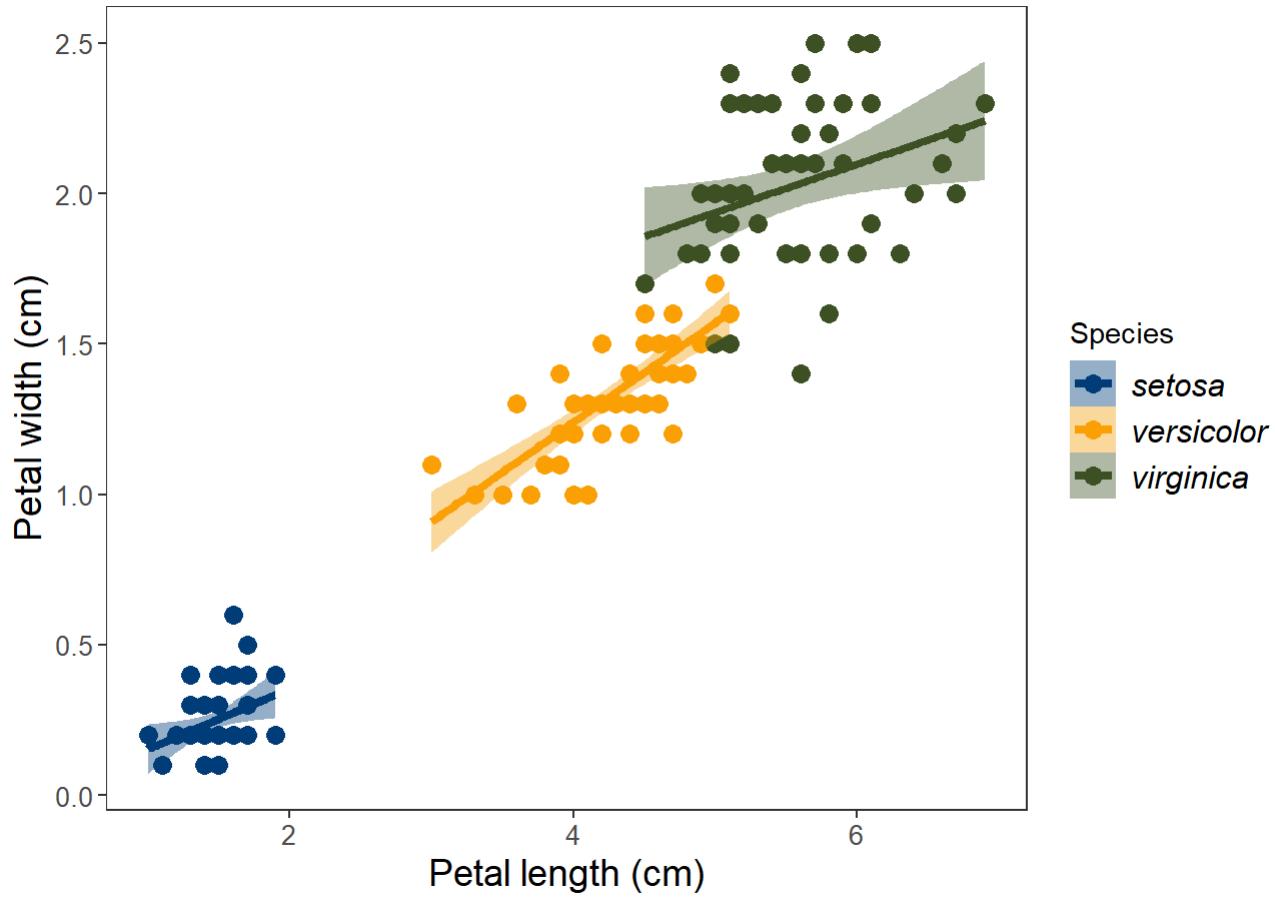
```
p<-iris %>%
  ggplot(aes(
    x = Petal.Length,
    y = Petal.Width,
    color = Species # to assign a color to each group
  )) +
  geom_point(size = 3) + # to plot a scatter plot
  stat_smooth(method = "lm", aes(fill = Species, colour = Species), linewidth = 1.5) + #
  scale_color_manual(name = "Species", values = c("#023d79", "#faa307", "#3f5426")) + #
  scale_fill_manual(name = "Species", values = c("#023d79", "#faa307", "#3f5426")) + #
  labs(
    x = "Petal length (cm)", #x axis label
    y = "Petal width (cm)" #y axis label
  ) +
  theme_bw() + #theme within the ggplot2 package (different themes available, f.ex. theme_minimal())
  theme(axis.text.x = element_text(size = 10), #use the theme function to customize axis
        axis.text.y = element_text(size = 10))
```

```
axis.text.y = element_text(size = 10),
axis.title.x = element_text(size = 14, face = "plain"),
axis.title.y = element_text(size = 14, face = "plain"),
panel.grid.major.x = element_blank(),
panel.grid.minor.x = element_blank(),
panel.grid.minor.y = element_blank(),
panel.grid.major.y = element_blank(),
plot.margin = unit(c(0.5, 0.5, 0.5, 0.5), units = , "cm"),
plot.title = element_text(size = 20, vjust = 1, hjust = 0.5),
legend.text = element_text(size = 12, face = "italic"),
legend.position = "right")
```

Modelling

p

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



To go further

- Visit: [Coding Club](#)
- EDGE coding club: Every 15 days, we have a coding club on Mondays at 10:30 in the Supernova room (Natrium)

What is next

- 15 of April 13:15-17:00
- *2125 Marelden*
- [Linear mixed models](#)
- On **Tuesday May 21th**, there will be a dedicated session for the projects.