Introduction to R and RStudio

Welcome to this session!

University of Auckland, 2021

ResBaz

Behind the scenes!

Laura Duntsch

Javiera Benavente

About me

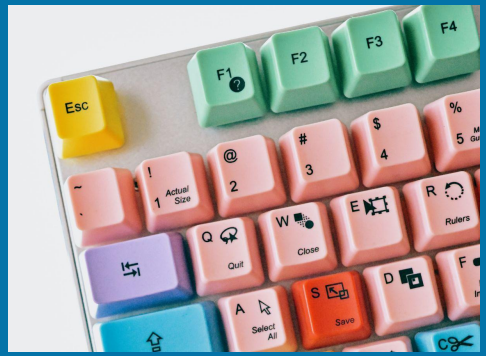
What people think I do:

What I actually do:

ResBaz

Session housekeeping

- Please be on “mute” for the duration of the session
- Raise your hand if you’d like a helper to get back to you
- Camera on or off - that’s totally up to you :)
- Safe and supportive environment



Spot this image in a slide? It means we will demonstrate what’s on the slide live in R.

Session housekeeping

- Use Etherpad for your in-session questions: https://pad.riseup.net/p/UoA_ResBaz2021_Intro_to_R
- Please open a www.menti.com tab in your browser
- Want to connect? ResBaz2021 slack channel!

Our session today

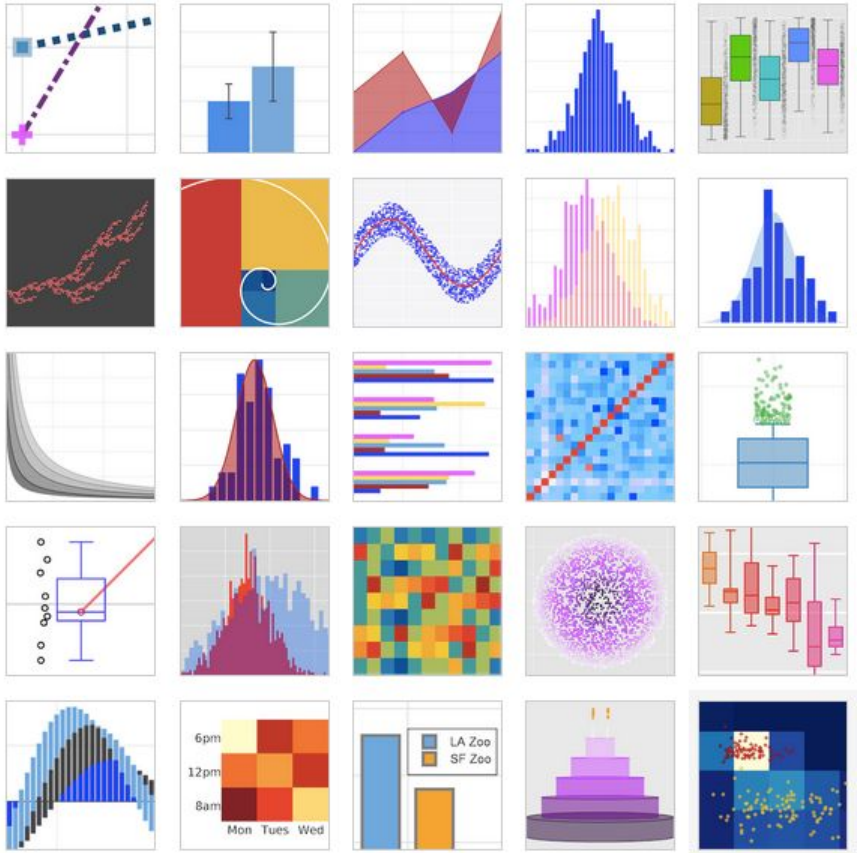
Photo by Steinar Engeland on Unsplash

Thanks to:

Ross Ihaka

Robert Gentleman

R can do a lot!

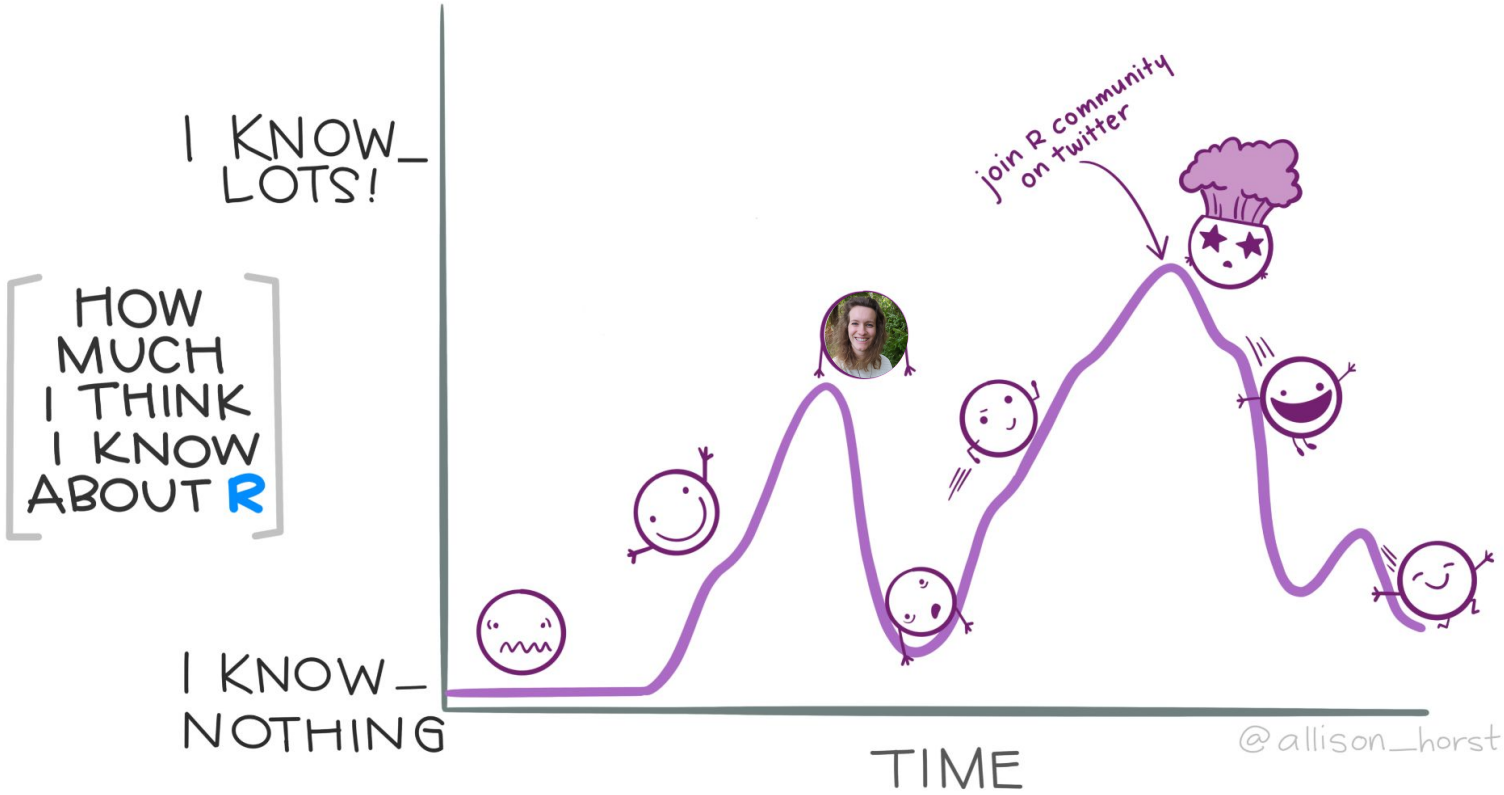


www.datasciencecentral.com

Learning Objectives

1. Understand how to access and navigate R and RStudio
2. Distinguish between install.packages() and library()
3. Load, view and subset datasets (e.g. csv)
4. Interpret warnings and errors & where to get help
5. Plot a graph and adjust the labels

Please go to
www.menti.com
via the code
in our etherpad



Artwork by @allison_horst

Demonstration 1: Basics

- Understand how to access and navigate R and RStudio
- Getting started: working directories and projects
- Assignments, basic functions and their arguments

R terminology: you find definitions in the Etherpad (bottom)!

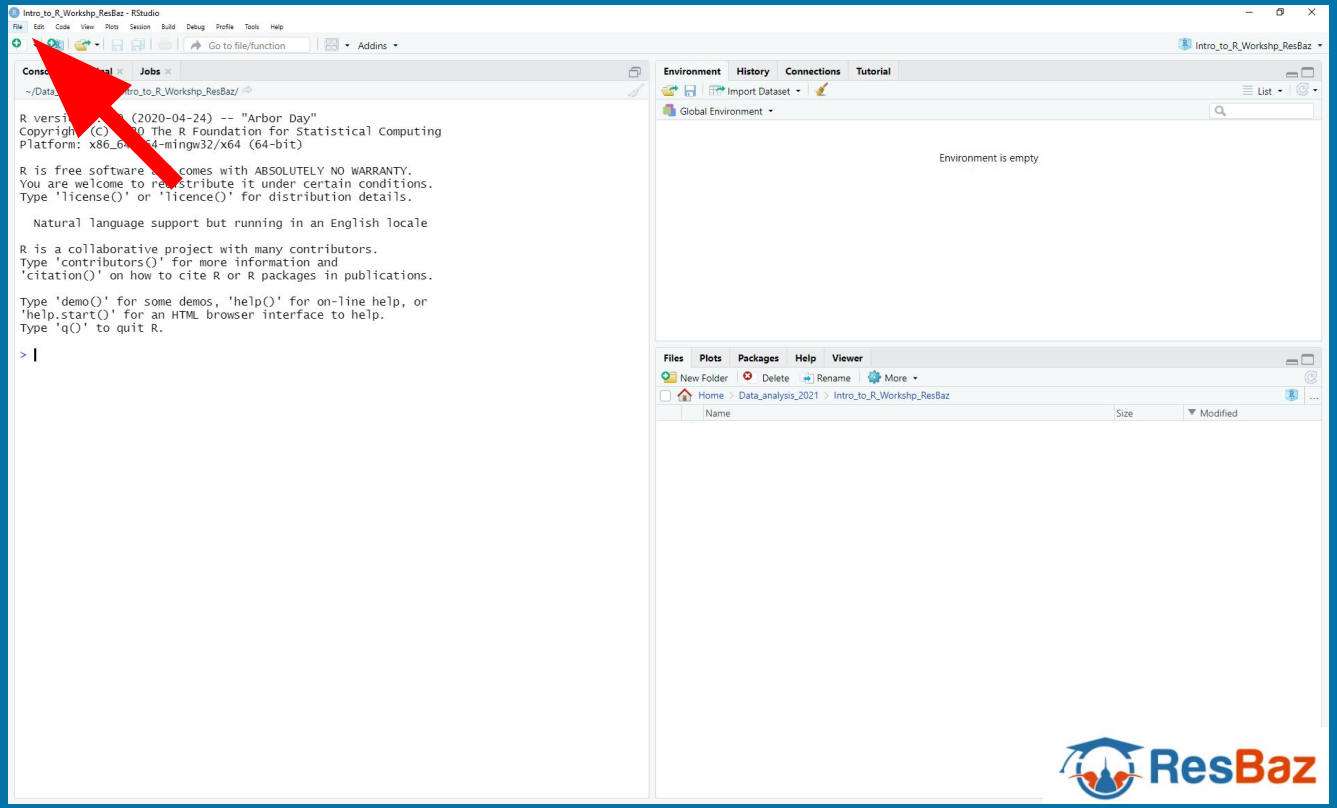
R is a programming language you can use to ask your computer to carry out certain computations.

- It's free
- It's open source
- A general-purpose of programming language
- It's available for all operating systems (Windows, Linux, and Mac)
- There is a huge online support network
- It's extremely flexible!

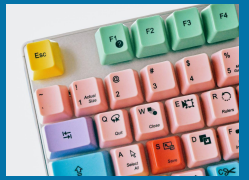
It is an interface that makes it easier to communicate with your computer in the language R.

- Speaks nicely to R
- Tab completion
- Debugging capabilities
- There is a huge online support network
- Offers many other features and tools to make your workflow with R easier
- It facilitates reproducibility

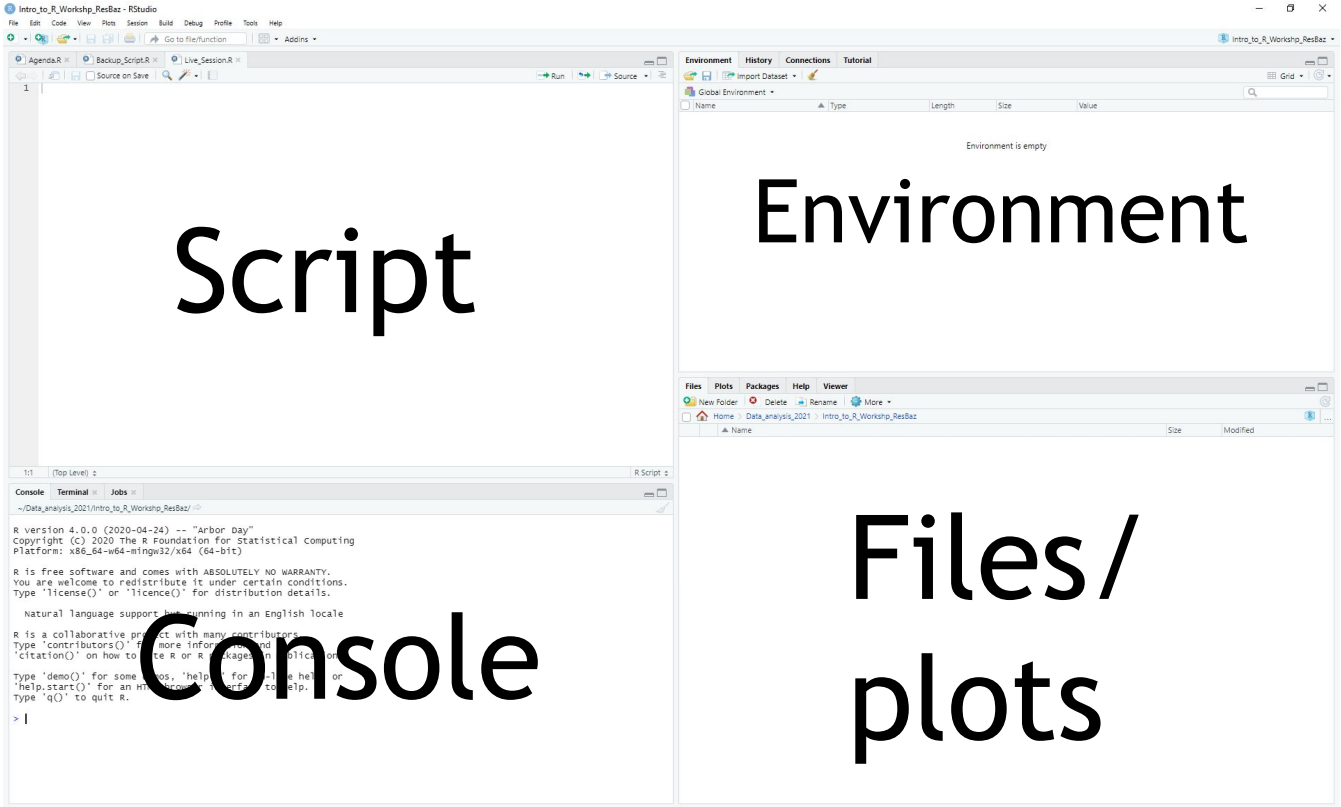
Let's open up RStudio together!



Please let the helpers in the chat know if you need assistance :)

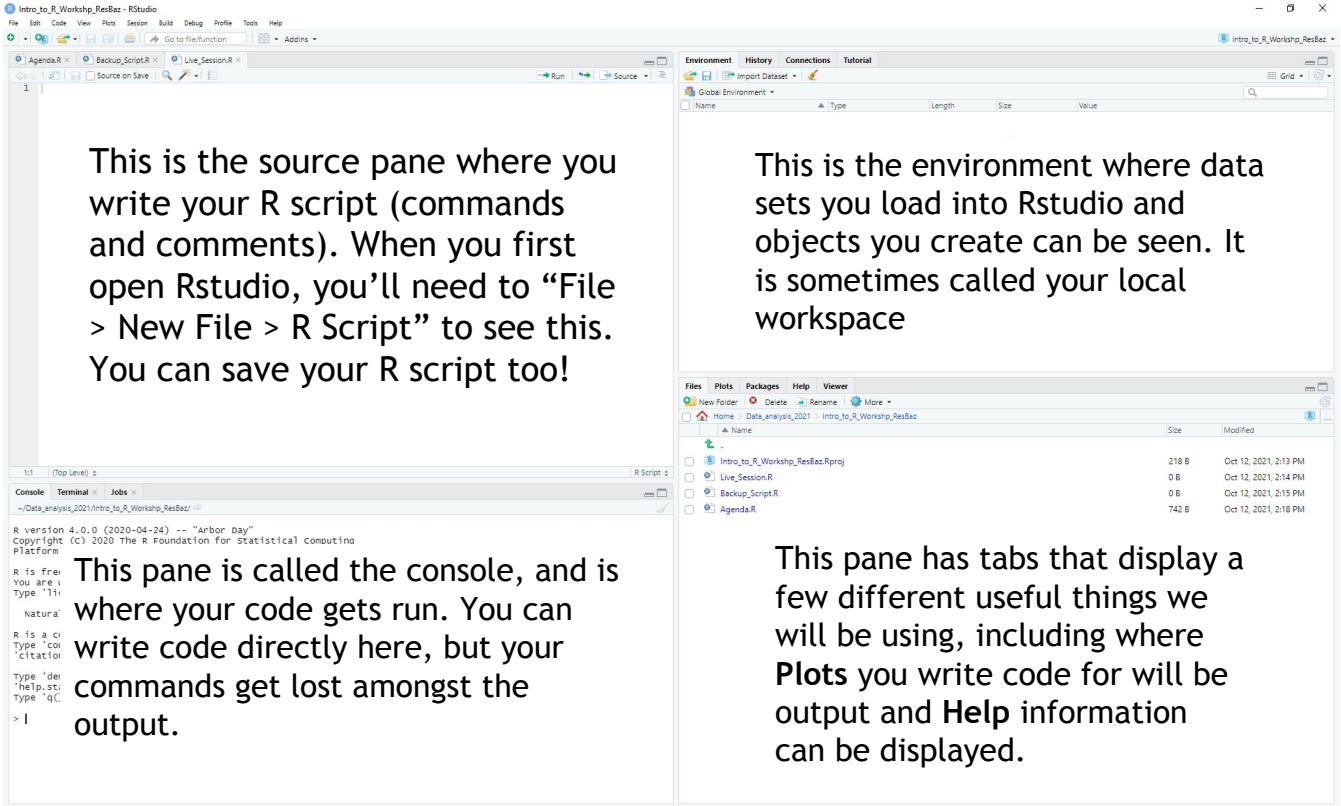


RStudio Layout



ResBaz

RStudio Layout



This is the source pane where you write your R script (commands and comments). When you first open Rstudio, you'll need to "File > New File > R Script" to see this. You can save your R script too!

This is the environment where data sets you load into Rstudio and objects you create can be seen. It is sometimes called your local workspace

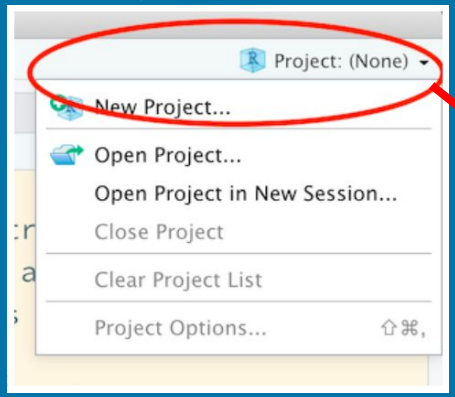
This pane is called the console, and is where your code gets run. You can write code directly here, but your commands get lost amongst the output.

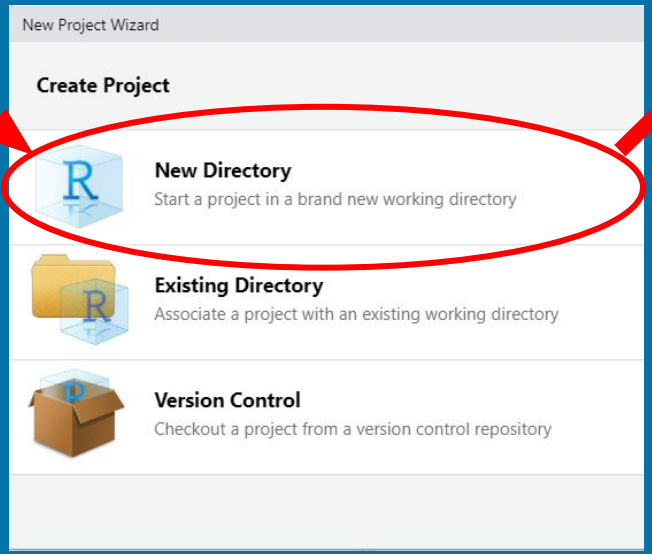
This pane has tabs that display a few different useful things we will be using, including where **Plots** you write code for will be output and **Help** information can be displayed.

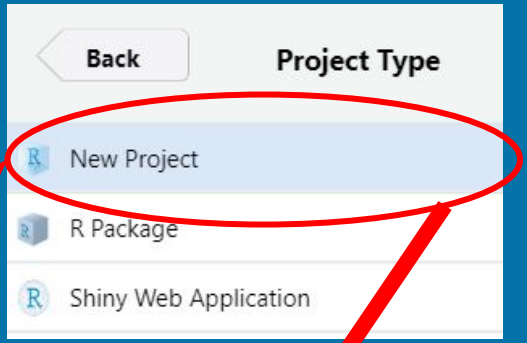
ResBaz

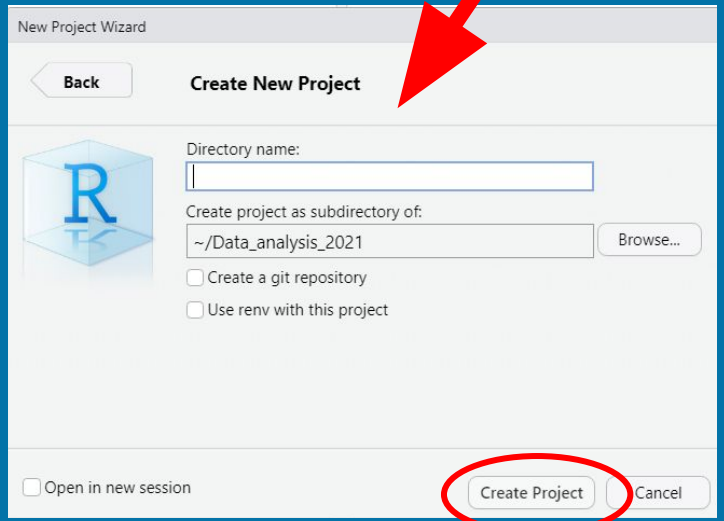
Demonstration 1: Set up your project

Top right corner:









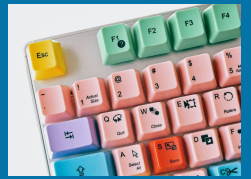

here: find your PATH!
"/NOT/HERE"
here!
seriously, just here.
IT'S ALL RIGHT HERE!
Horst '18

Demonstration 1: Your RStudio project

Your project is managed just like any other folder on your computer.

To see where your project is located, type:

```
> getwd()   # stands for get your working directory
```



Demonstration 1: Comments vs code

```
## IGNORE ME
## I'm a comment
## I repeat I'm a comment
## I am not a cat
## OK let's run some code
2 + 2
```

```
## [1] 4
```

Demonstration 1: Data types

- **Integers** are whole values like 1, 0, 220.
- **Numeric** values are a larger set of values containing integers but also fractions and decimal values, for example -56.94 and 1.3.
- **Logicals** are either TRUE or FALSE (also called BOOLEAN).
- **Characters** are text such as “Laura”, “ResBaz”, and “Statistics is the greatest subject ever”. Note that characters are denoted with the quotation marks around them and can also be called “integers”.

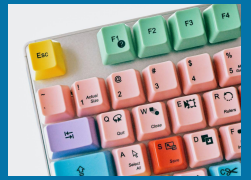
Demonstration 1: Creating objects

```
my_name <- "Laura"                ## assignment operator: <-
```

So now the Object my_name ‘contains’ the value "Laura". Another assignment to the same object will overwrite the content.

```
my_name <- "Susanne"
```

```
> my_name
[1] "Susanne"
```



Demonstration 1: R functions

- **Functions** (or commands) perform tasks in R.
- They take in inputs called **arguments** and return **outputs**.
- Example: `seq()` `# generates regular sequences`

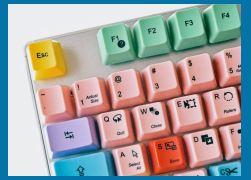
Demonstration 1: R functions

```
> seq()    ## same as seq(from = 1, to = 1, by =1)
[1] 1                                          #intervall
```

If you don't pass in different values for from and to to change this behaviour, your computer just assumes all you want is the number 1. Let's change it!

```
> seq(from = 2, to = 6, by = 2)
[1] 2 4 6
```

Don't know how to use a function?
Type:
`> ?seq()`
or
`> help(seq)`
into your console!



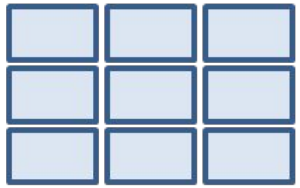
Data in R can take different forms!

Vector

- 1 column or row of data
- 1 type

Vectors can be created using the c() function.

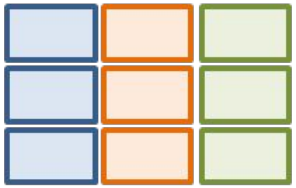
Matrix



- multiple columns and/or rows of data
- 1 type

Vectors can be created using the matrix() function.

Data Frame



- multiple columns and/or rows of data
- multiple types

A data frame looks similar to a table in Excel!

	A	B
1	**Name**	**Height**
2	Lisa	167
3	Bob	179
4	Cecile	180
5		

Image: https://libguides.wellesley.edu/r/basics

Time to practise!

What you could practise:

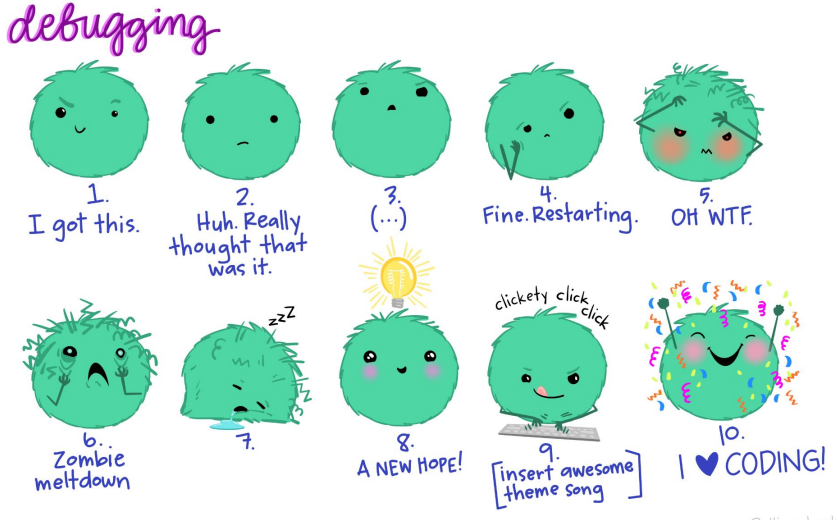
```
## I'm a comment

class("Name")

class(2021)

4+7

4*5
```

```
my_name <- "Name"
my_name


some_numbers <- c(1,4,5,13,45,90)
length(some_numbers)
mean(some_numbers)
str(some_numbers)
```

Remember: Functions (or commands) perform tasks in R. They take in inputs called arguments and return outputs.

Have a break!

See you at 2:30pm.



Please go to
www.menti.com
via the code
in our etherpad

R packages and libraries

Photo by Gabriel Sollmann on Unsplash

R packages and libraries

- An R **package** is simply a collection of functions - like a book that contains facts
- To access the contents of a **package** (book) you first need to **install it** onto (request it for) your local computer (your local library):

R command:

install.packages()

Online repository:
e.g. CRAN is like a centralised library with thousands of books in stock.

Local computer

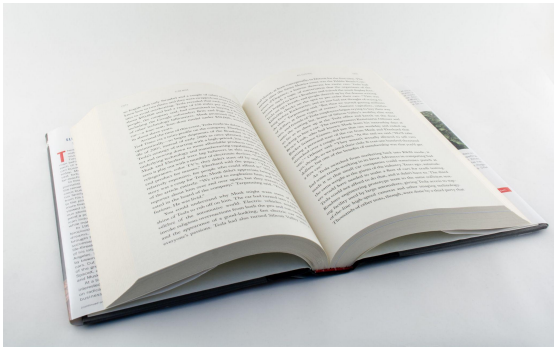
R packages and libraries

- To access the knowledge in a particular book (use the **function** in the **package**) you need to tell your computer via R to go get the book off the shelf.
- Then you have access to all the functions it contains:

Local computer

R command:

library()



Your current R session

In other words

R...

i) basic features in the base R package installed

ii) you need to install R libraries to get more functionality

```
> install.packages()
```

… is like a new phone:

i) basic features (calling, texting etc.) installed

ii) you can download apps to get more functionality.

Photo by Rahul Chakraborty on Unsplash

ResBaz

Demonstration 2: R packages

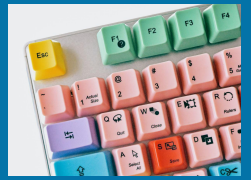
- Distinguish between `install.packages()` and `library()`
- The double ::
- Interpret R warnings and R errors
- Where to get help: e.g. stackoverflow

Demonstration 2: Installing packages

```
> install.packages("data.table")    ## install package

> library(data.table)               ## load package

> fread("filename.csv")             ## use this function
```

If you want to access a function in a specific package:

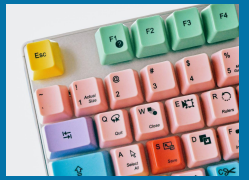
```
## package::functionname, e.g.

> data.table::fread("filename.csv")
```



Demonstration 2: Dealing with errors

```
## Error in library(fiddler): there is no package called 'fiddler'
```

```
## Warning in file(file, "rt"): cannot open file 'paua.csv': No such file or
## directory
```



Demonstration 2: Dealing with errors

```
## Error in library(fiddler): there is no package called 'fiddler'
```

```
## Warning in file(file, "rt"): cannot open file 'paua.csv': No such file or
## directory
```

Approach 1:

Read message and check your code!

Approach 2:

Approach 3:

stackoverflow

Time to practise!

What you could practise:

```
install.packages("magrittr")

library(magrittr)

??magrittr


install.packages("tidyr")

library(tidyr)

??tidyr
```

Try this, too:

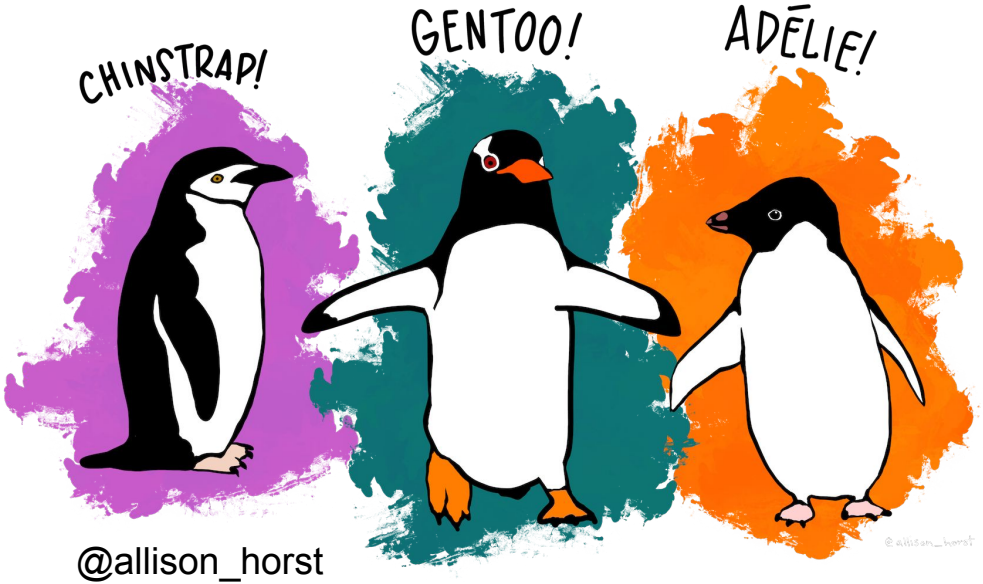
```
# all three lines of code need to be run together
if(!require(praise)) install.packages("praise")
require(praise)
print(praise())
```

And share your output in the chat 😉

```
# Make a mistake on purpose, and see what
# R tells you about the error!
```

Demonstration 3: Data wrangling

- Load datasets (e.g. csv)
- View and subset datasets
- Piping



Demonstration 3: Reading in data from a .csv file

get package ready that you're likely to use in everyday data analyses:

```
> install.packages("data.table")
> library(data.table)        # offers a natural and flexible syntax
```

one of many ways to load a data set called `filename.csv` into your R session:

```
> my_data <- fread("filename.csv")
> my_data
     Name Height
1:   Lisa    167
2:    Bob    179
3: Cecile    180
```

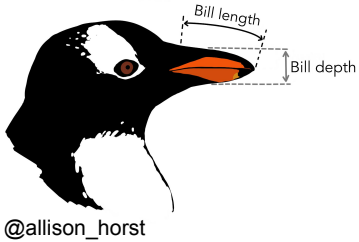
Outside R, a .csv file looks like:

	A	B
1	**Name**	**Height**
2	Lisa	167
3	Bob	179
4	Cecile	180
5		

Demonstration 3: Data within a package

get package with pre-curated dataset:

```
> install.packages("palmerpenguins")
```



load package:

```
> library(palmerpenguins)    ## contains some nice penguin data ;)
```

view data set in console:

```
> penguins
```

```
# A tibble: 344 x 8
   species island    bill_length_mm
   <fct>   <fct>              <dbl>
 1 Adelie  Torgersen           39.1
 2 Adelie  Torgersen           39.5
 3 Adelie  Torgersen           40.3
 4 Adelie  Torgersen           NA
 5 Adelie  Torgersen           36.7
```

ResBaz

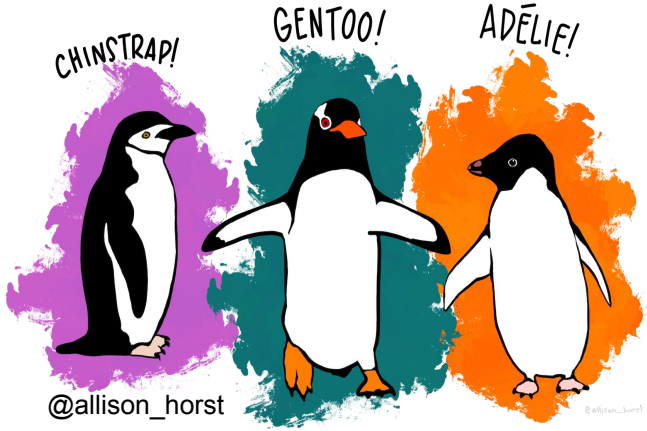
Demonstration 3: View penguin data

view data set in console:

```
> penguins
> head(penguins)
```

view data in new tab in RStudio:

```
> View(penguins)
```



penguins

Filter

	species	island	bill_length_mm	bill_depth_mm
1	Adelie	Torgersen	39.1	18.7
2	Adelie	Torgersen	39.5	17.4
3	Adelie	Torgersen	40.3	18.0
4	Adelie	Torgersen	*NA*	*NA*
5	Adelie	Torgersen	36.7	19.3

ResBaz

Demonstration 3: The pipe operator %>%

##Packages we'll need from now on:

```
> install.packages("magrittr")    # provides the pipe operator
> install.packages("tidyr")       # provides drop_na() function
```

remember to also load the `library()`

and remove the individuals with NAs (cells in table with no entry)

```
> penguins_nafree <- penguins %>% drop_na()
> penguins_nafree
```

Look! All entries with NA have been removed!

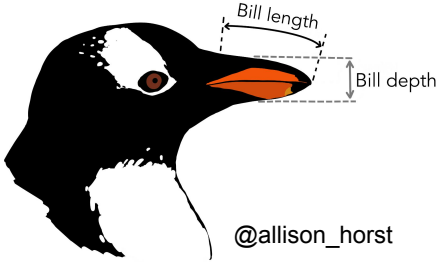
Trick: read pipe symbol as "then" or "and then"!

Demonstration 3: The pipe operator %>%

To calculate the **mean bill length** of each **species** in the penguin dataset,

we would use this code:

```
> penguins_nafree %>%
>   group_by(species) %>%
>   summarise(average_bill_length = mean(bill_length_mm))
```



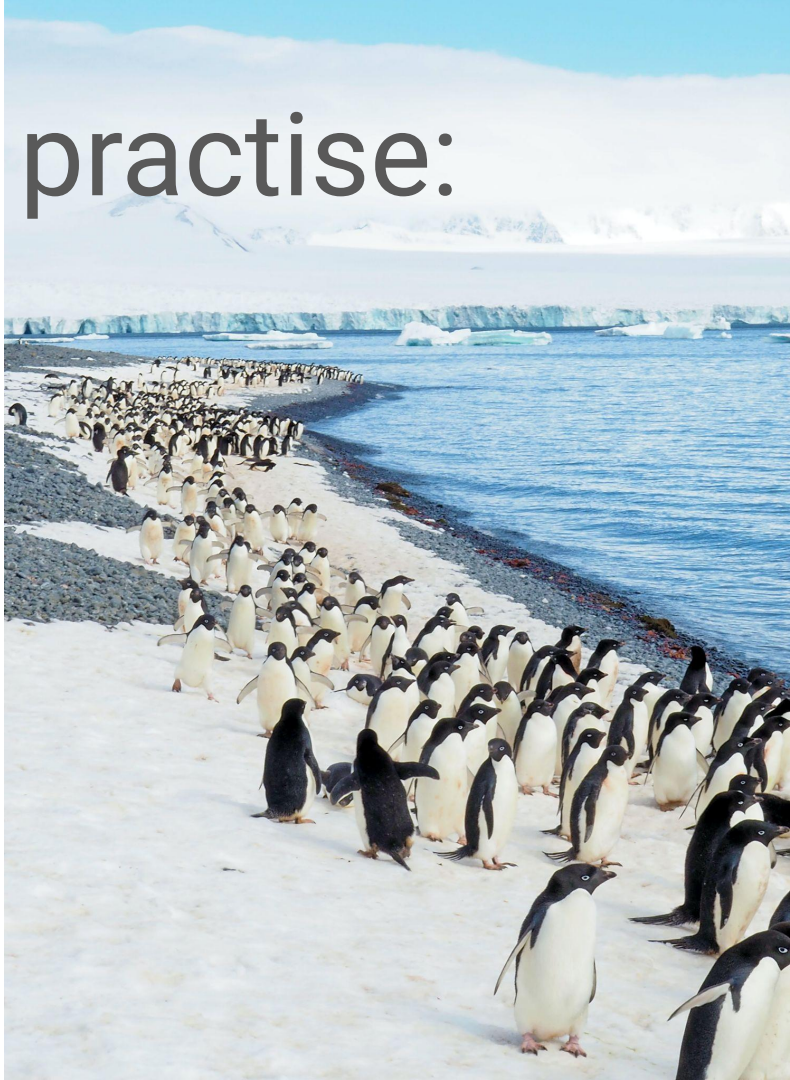

```
# Take the penguins_nafree data, then

# Use this and apply the group_by() function to group by species, then

# Use this output and apply the summarize() function to calculate the mean (using (mean()) bill          #
length (bill_length_mm) of each group (species), calling the resulting number 'average_bill_length'.
```

ResBaz

Time to practise!

What you could practise:

```
View(penguins)

mean(penguins_nafree$bill_length_mm)

group_by(penguins_nafree, species)

penguins_nafree %>%
  group_by(island,species) %>%
  summarise(average_bill_length = mean(bill_length_mm))

summarise(penguins_nafree,
                  average_bill_length = mean(bill_length_mm))

filter(penguins_nafree, island == "Torgersen" )
```



Break

See you at 4pm!

Photo by Daria Nepriakhina on Unsplash

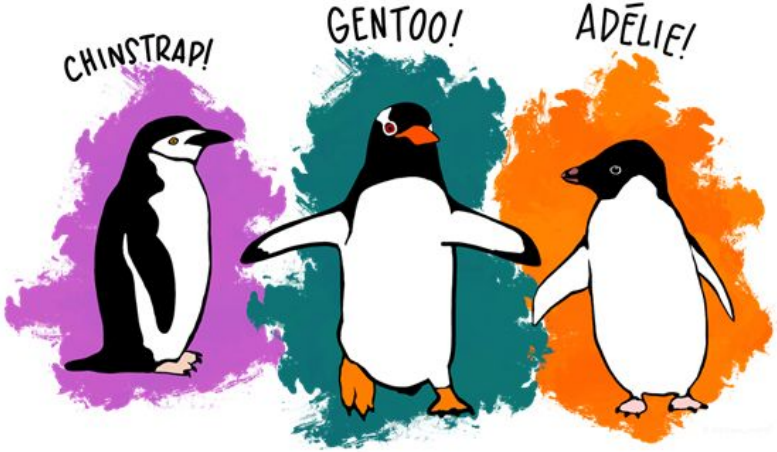
Please go to
www.menti.com
via the code
in our etherpad

Data visualization in R

Data visualization in R

3 main plotting systems:

- base plotting system
- lattice package
- ggplot2 package*



Remember!

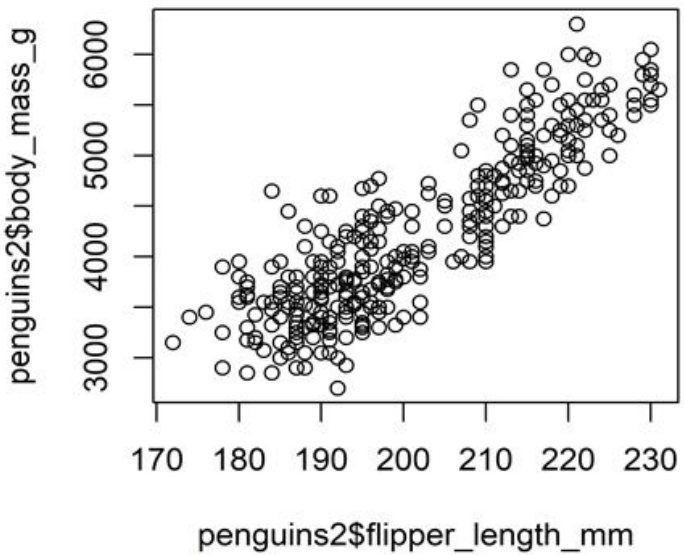
- Load magrittr, tidyr and palmerpenguins libraries
- Remove *NA* values from penguins dataframe

```
library(palmerpenguins)
library(magrittr)
library(tidyr)

#Check data
head(penguins)

#Create NA free dataframe
penguins_nafree<-penguins %>% drop_na()
```

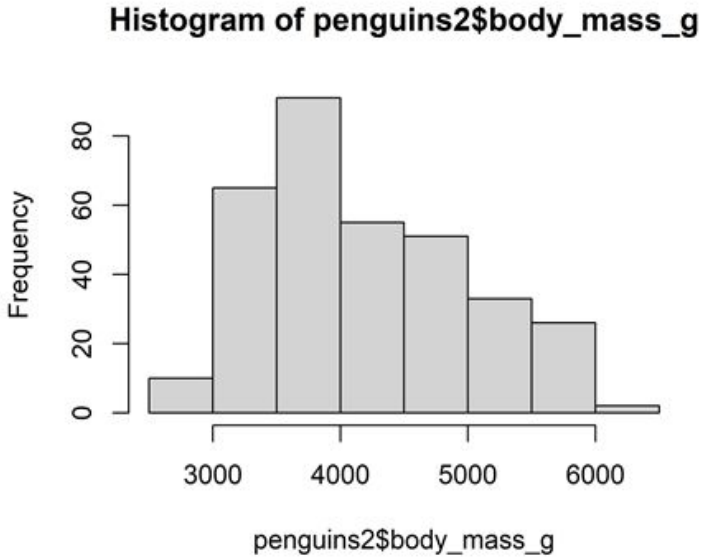
Base R plot are useful for quick plots for data exploration

Useful functions:

```
plot()
barplot()
boxplot()
hist()
density()
```




```
plot(penguins_nafree$flipper_length_mm,
     penguins_nafree$body_mass_g)
```




```
hist(penguins_nafree$body_mass_g)
```

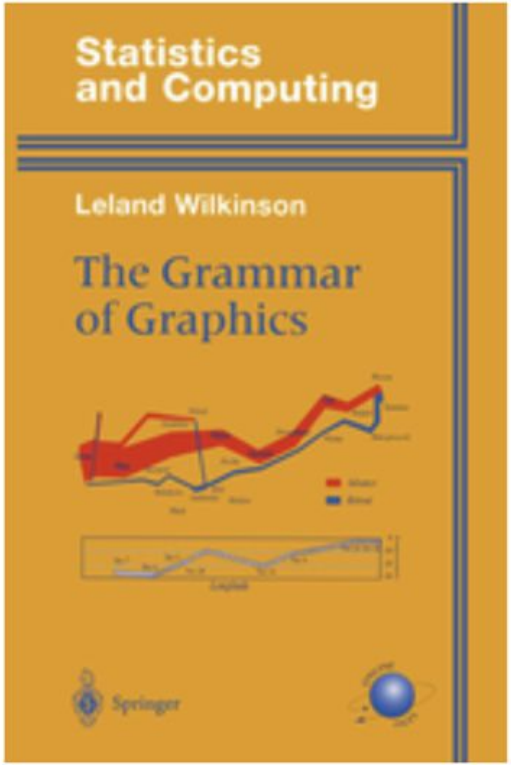
The grammar of graphics (**ggplot2**)

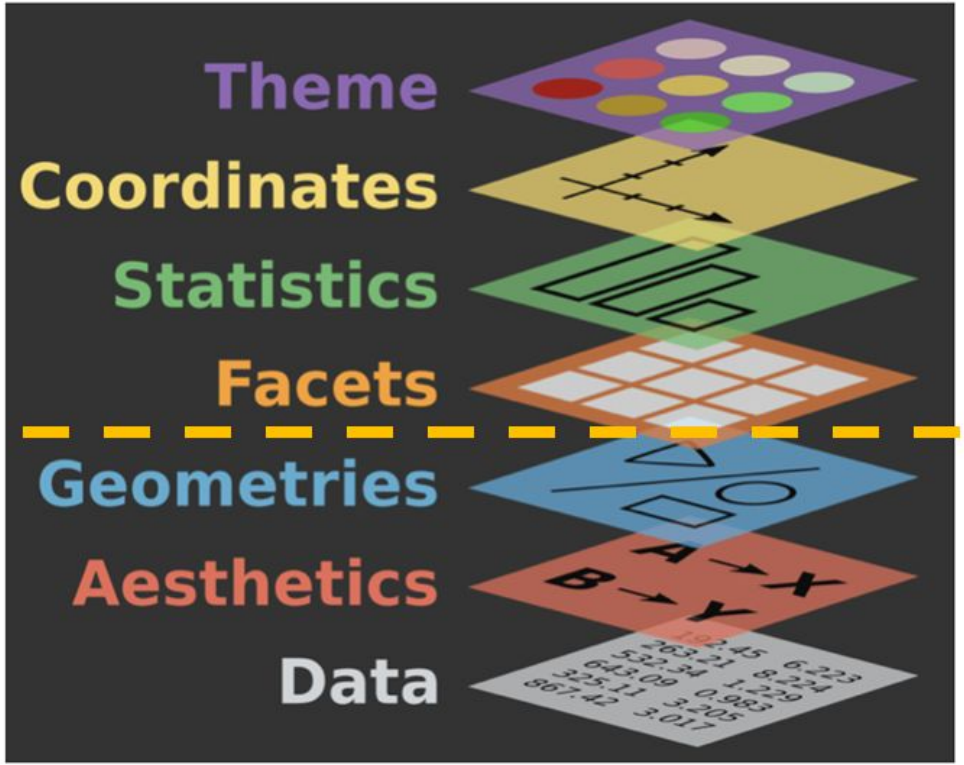
In **ggplot2** all plots are built from the same set of components:

A **dataset** (could be multiple)

The mapping or aesthetics (**aes**): The variables of the dataset that are mapped or shown in the plot

A set of geometric representations or **geoms**: visual representations of data (type of plot, there can be multiple geoms in one plot)



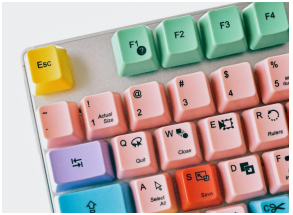


Optional components

Essential components

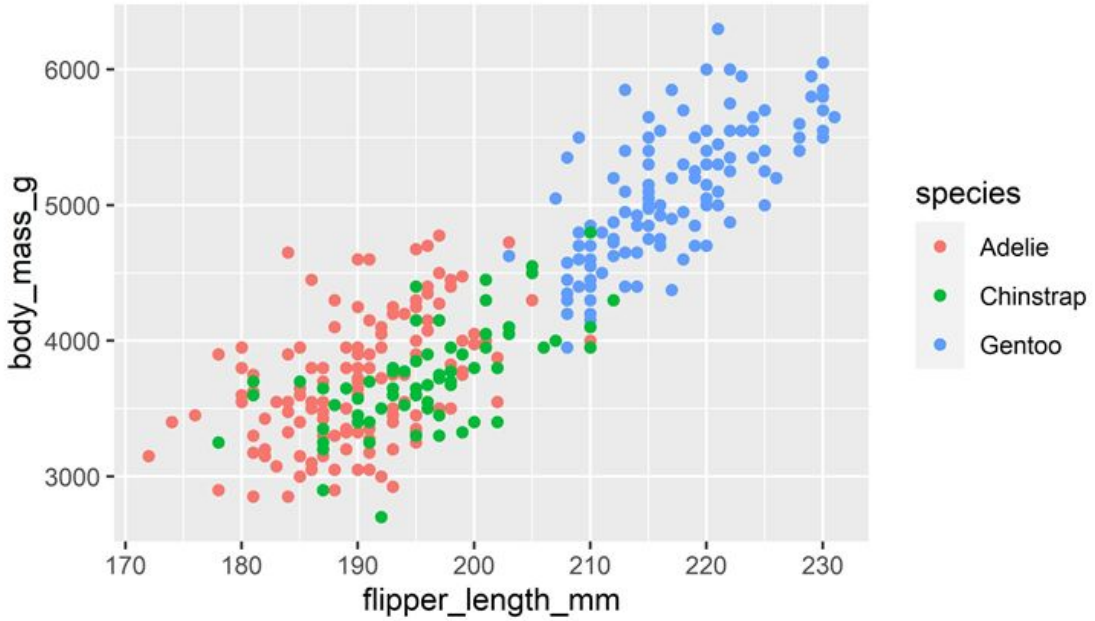
Basic ggplot call

```
install.packages("ggplot2")
library("ggplot2")
  ggplot(data = DataFrame, aes(x = VarXaxis, y = VarYaxis)) +
    geom_...()
```



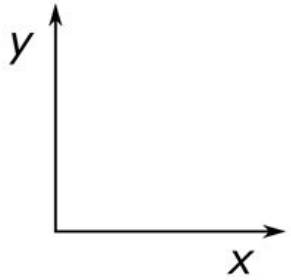
Basic ggplot call

```
ggplot(data = penguins_nafree, aes(x=flipper_length_mm, y=body_mass_g,
color=species))+
geom_point()
```



Examples aesthetics **(aes)**

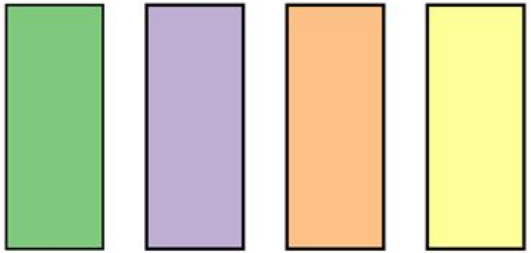
Position



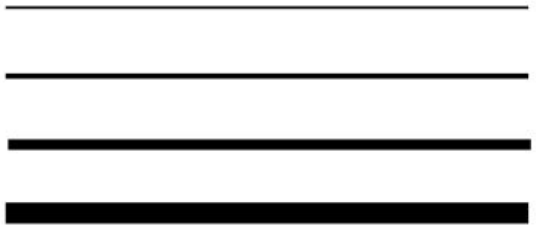
Shape

Size

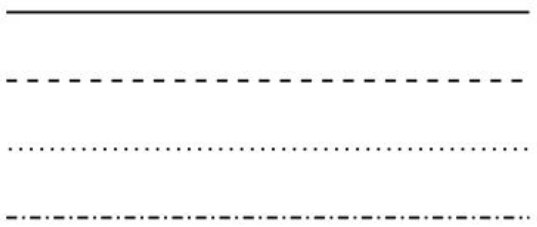
Colour



Line width



Line type



From: Wilkes 2018

Time to practise!

Challenge 1:

•Produce a **boxplot** with body_mass_g on the Y axis and species on the X axis

Challenge 2:

•Think in **layers**! You can add geoms on top of each other

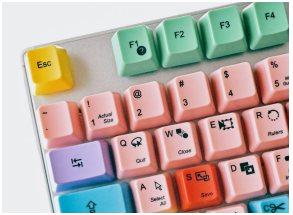
•Add **points** on top of the previous **boxplot** to visualize better the distribution of the data

Remember!

```
ggplot(data = penguins_nafree, aes(x=flipper_length_mm, y=body_mass_g,
color=species))+
geom_point()
```

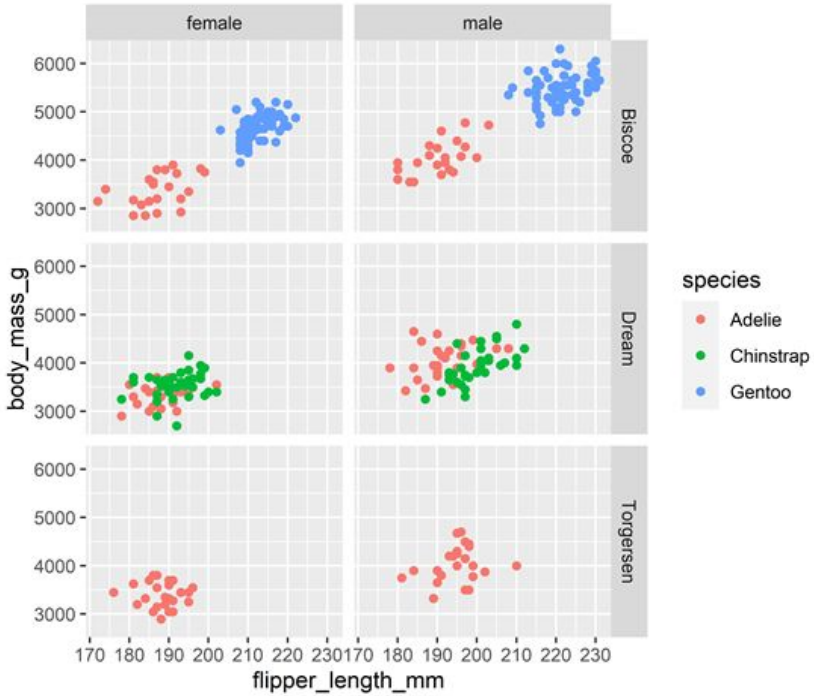
Facets

- `facet_grid(row ~ columns)`
- `facet_wrap()`



Facets

```
ggplot(penguins_nafree, aes(x=flipper_length_mm, y=body_mass_g, color=species)) +
  geom_point() +
  facet_grid(island~sex )
```



Time to practise!

Challenge 3:

- Divide the scatter plot in **3 column facets** corresponding to each penguin species

Remember the dot (.) to indicate no row faceting

Remember!

```
ggplot(penguins_nafree, aes(x=flipper_length_mm, y=body_mass_g, color=species)) +
  geom_point() +
  facet_grid(island~sex )
```

Statistics (statistical summaries)

`stat_summary()`: Allows to display means, standard deviations, and other statistics in plots without calculating them previously

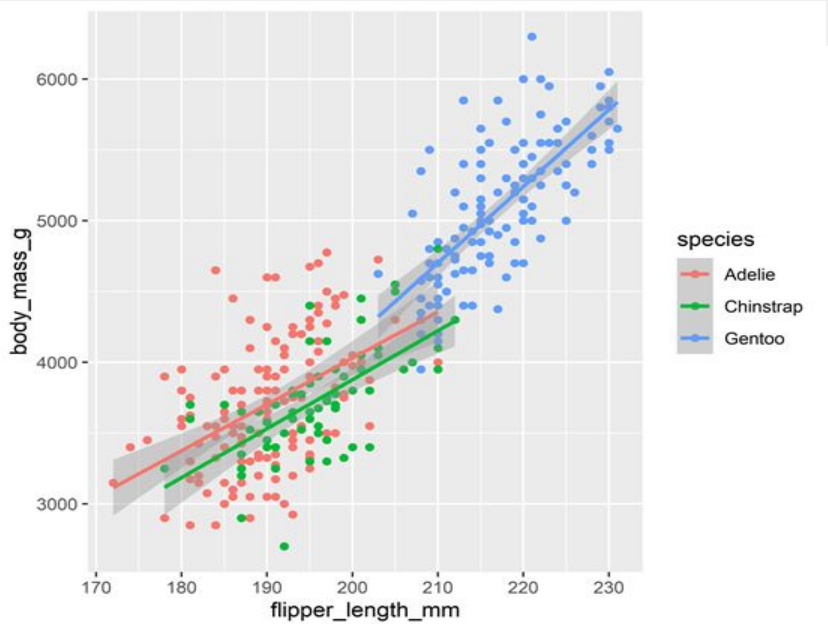
`geom_smooth()`: Fits a regression line to data

Other examples of geoms that display summarised data:

`geom_errorbar()`, `geom_pointrange()`, `geom_linerange()`, `geom_crossbar()`

Statistics (regression lines)

```
ggplot(penguins_nafree, aes(x=flipper_length_mm, y=body_mass_g, color=species))+
geom_point() +
geom_smooth(method="lm")
#This just a visual representation of a linear relationship IT DOESN'T REPLACE REAL STATS
```



Themes and labels: Make your plot pretty!

Many ways of manipulating plot labels

- `labs()`: Can be used to label all aesthetics in plot

- Labels can also be changed independently when changing scales of axes and other aesthetics (i.e., `scale_x_continuous()`, `scale_color_discrete()`, `scale_shape_manual()`, etc)

ResBaz

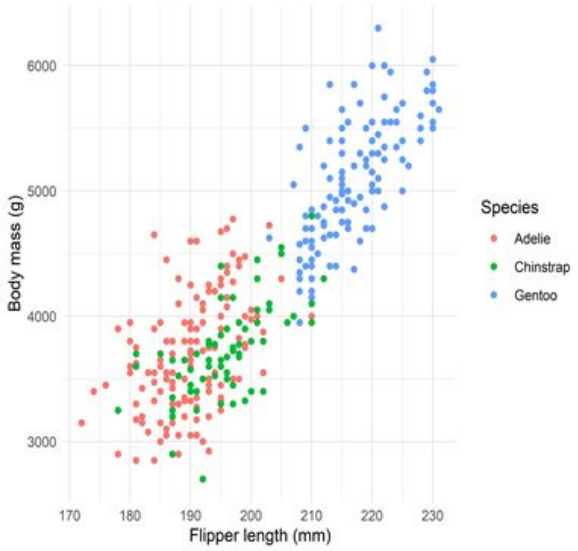
Themes

ggplot2 has many inbuilt themes you can try

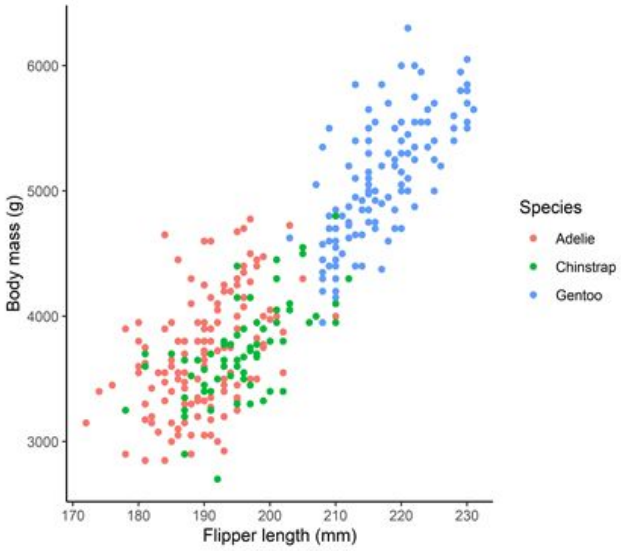
For extra premade themes you can install the package **ggthemes**

Play with different themes (start typing `theme_` to see them)!!

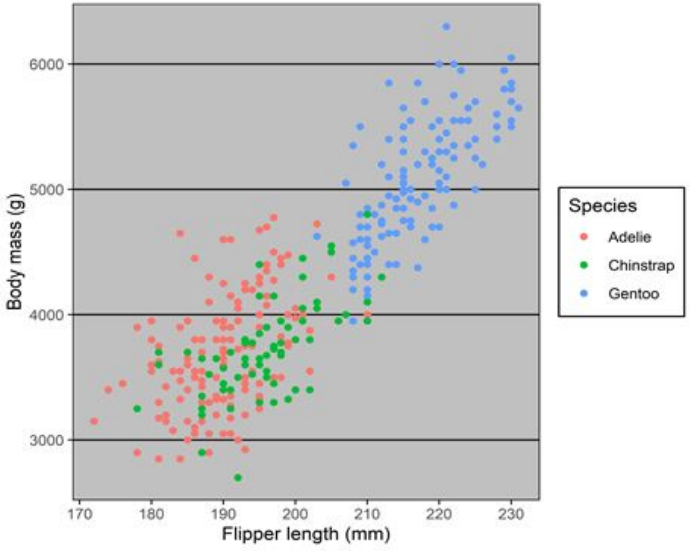
Some theme examples



`theme_minimal()`



`theme_classic()`

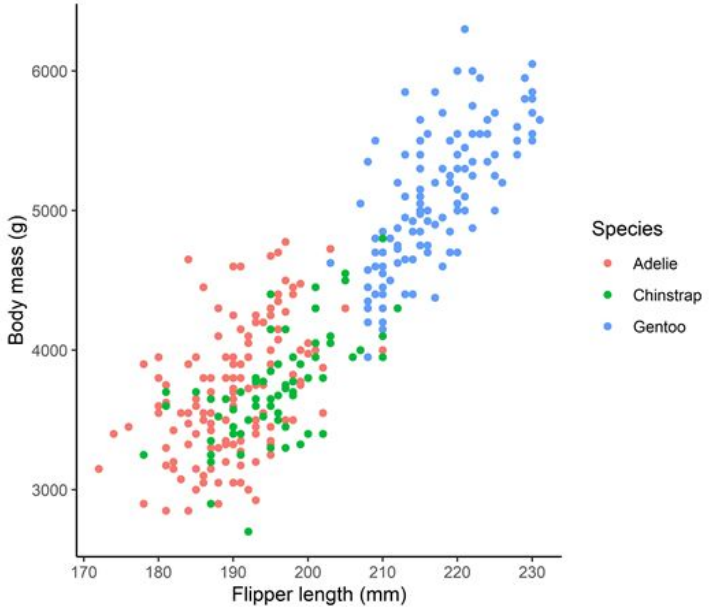


`theme_excel()`

*ggthemes library

Themes and labels

```
ggplot(penguins_nafree, aes(x=flipper_length_mm, y=body_mass_g, color=species))+
geom_point() +
labs(x = "Flipper length (mm)", y = "Body mass (g)", color= "Species") +
  theme_classic()
```



Colors and color palettes

Use your own colors using

scale_color_manual(values=
“color name or hex codes”**)**

ResBaz

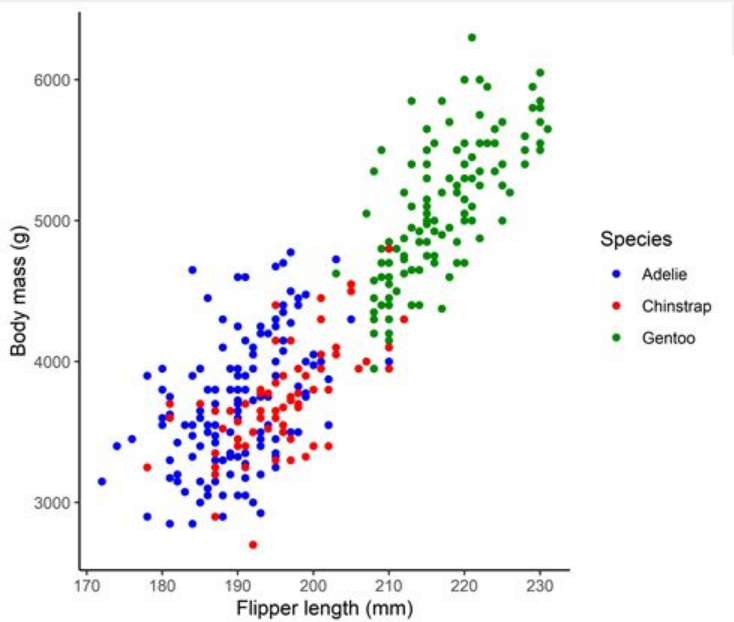
Color	Hex Code #RRGGBB
maroon	#800000
dark red	#8B0000
brown	#A52A2A
firebrick	#B22222
crimson	#DC143C
red	#FF0000
tomato	#FF6347
coral	#FF7F50
indian red	#CD5C5C
light coral	#F08080
dark salmon	#E9967A
salmon	#FA8072
light salmon	#FFA07A
orange red	#FF4500
dark orange	#FF8C00
orange	#FFA500
gold	#FFD700
dark golden rod	#B8860B
golden rod	#DAA520
pale golden rod	#EEE8AA
dark khaki	#BDB76B
khaki	#F0E68C
olive	#808000
yellow	#FFFF00
yellow green	#9ACD32
dark olive green	#556B2F
olive drab	#6B8E23
lawn green	#7CFC00
chart reuse	#7FFF00
green yellow	#ADFF2F
dark green	#006400
green	#008000
forest green	#228B22
lime	#00FF00
lime green	#32CD32
light green	#90EE90
pale green	#98FB98
dark sea green	#8FBC8F
medium spring green	#00FA9A
spring green	#00FF7F
sea green	#2E8B57
medium aqua marine	#66CDAA
medium sea green	#3CB371
light sea green	#20B2AA
dark slate gray	#2F4F4F
teal	#008080
dark cyan	#008B8B

Color	Hex Code #RRGGBB
aqua	#00FFFF
cyan	#00FFFF
light cyan	#E0FFFF
dark turquoise	#00CED1
turquoise	#40E0D0
medium turquoise	#48D1CC
pale turquoise	#AFEEEE
aqua marine	#7FFFD4
powder blue	#B0E0E6
cadet blue	#5F9EA0
steel blue	#4682B4
corn flower blue	#6495ED
deep sky blue	#00BFFF
dodger blue	#1E90FF
light blue	#ADD8E6
sky blue	#87CEEB
light sky blue	#87CEFA
midnight blue	#191970
navy	#000080
dark blue	#00008B
medium blue	#0000CD
blue	#0000FF
royal blue	#4169E1
blue violet	#8A2BE2
indigo	#4B0082
dark slate blue	#483D8B
slate blue	#6A5ACD
medium slate blue	#7B68EE
medium purple	#9370DB
dark magenta	#8B008B
dark violet	#9400D3
dark orchid	#9932CC
medium orchid	#BA55D3
purple	#800080
thistle	#D8BFD8
plum	#DDA0DD
violet	#EE82EE
magenta / fuchsia	#FF00FF
orchid	#DA70D6
medium violet red	#C71585
pale violet red	#DB7093
deep pink	#FF1493
hot pink	#FF69B4
light pink	#FFB6C1
pink	#FFC0CB
antique white	#FAEBD7

Color	Hex Code #RRGGBB
beige	#F5F5DC
bisque	#FFE4C4
blanched almond	#FFEBCD
wheat	#F5DEB3
corn silk	#FFF8DC
lemon chiffon	#FFFACD
light golden rod yellow	#FAFAD2
light yellow	#FFFFE0
saddle brown	#8B4513
sienna	#A0522D
chocolate	#D2691E
peru	#CD853F
sandy brown	#F4A460
burly wood	#DEB887
tan	#D2B48C
rosy brown	#BC8F8F
moccasin	#FFE4B5
navajo white	#FFDEAD
peach puff	#FFDAB9
misty rose	#FFE4E1
lavender blush	#FFF0F5
linen	#FAF0E6
old lace	#FDF5E6
papaya whip	#FFEFD5
sea shell	#FFF5EE
mint cream	#F5FFFA
slate gray	#708090
light slate gray	#778899
light steel blue	#B0C4DE
lavender	#E6E6FA
floral white	#FFFAF0
alice blue	#F0F8FF
ghost white	#F8F8FF
honeydew	#F0FFF0
ivory	#FFFFF0
azure	#F0FFFF
snow	#FFFAFA
black	#000000
dim gray / dim grey	#696969
gray / grey	#808080
dark gray / dark grey	#A9A9A9
silver	#C0C0C0
light gray / light grey	#D3D3D3
gainsboro	#DCDCDC
white smoke	#F5F5F5
white	#FFFFFF

Colors and color palettes

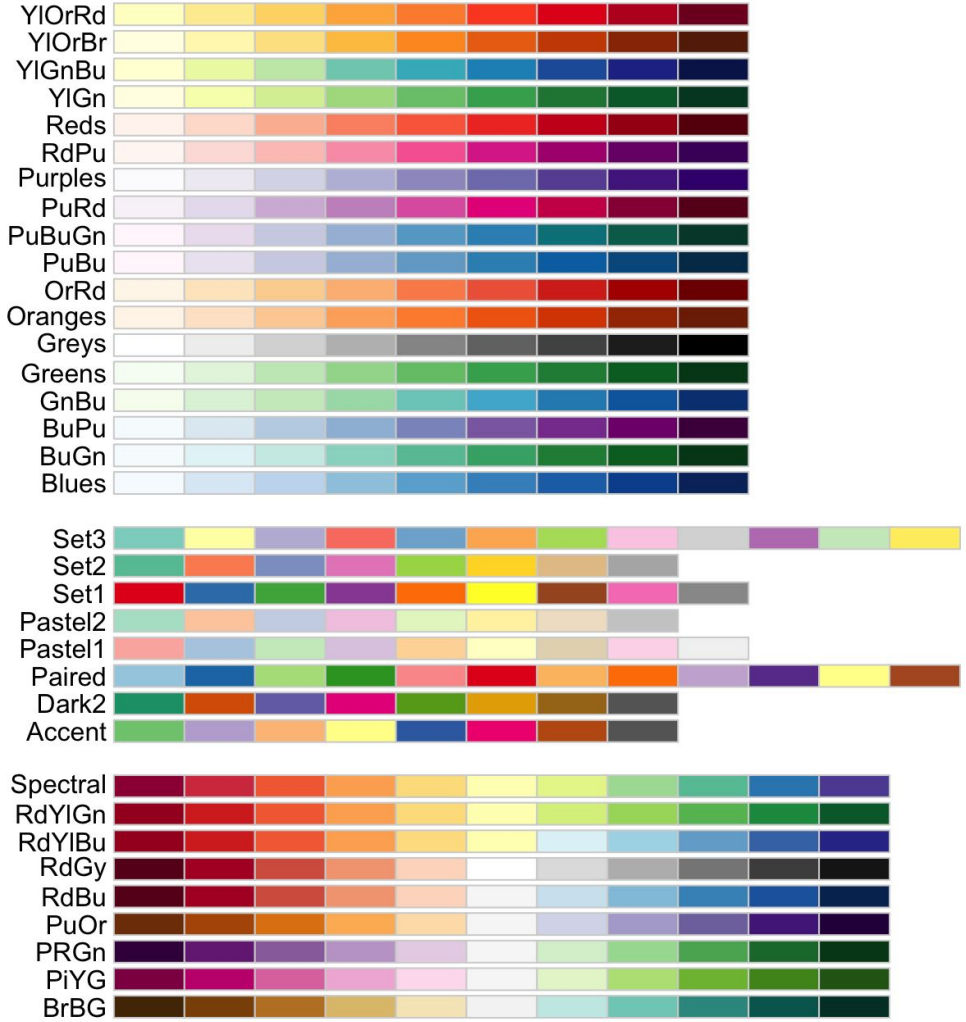
```
ggplot(penguins_nafree, aes(x=flipper_length_mm, y=body_mass_g, color=species))+
geom_point() +
labs(x = "Flipper length (mm)", y = "Body mass (g)", color= "Species") +
scale_color_manual(values=c("blue", "red" , "green4" )) +
theme_classic()
```



Colors and color palettes

Use pre-fabricated color palettes with **RColorBrewer** :

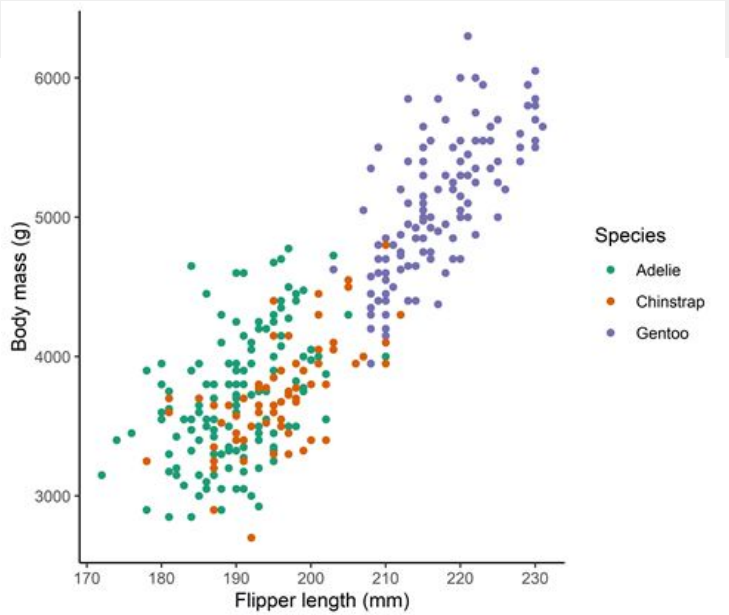
```
scale_color_brewer(values=
“palette of your choice”)
```



ResBaz

Colors and color palettes

```
ggplot(penguins_nafree, aes(x=flipper_length_mm, y=body_mass_g, color=species))+
geom_point() +
labs(x = "Flipper length (mm)", y = "Body mass (g)", color= "Species") +
scale_color_brewer(palette="Dark2") +
theme_classic()
```

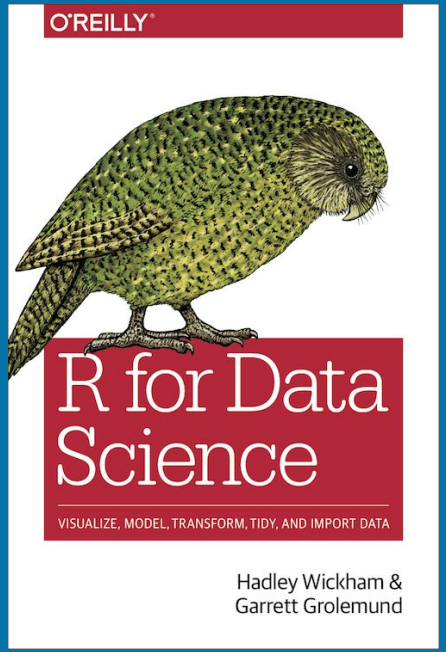


Happy plotting !!

Please go to
www.menti.com
via the code
in our etherpad

Further resources and support

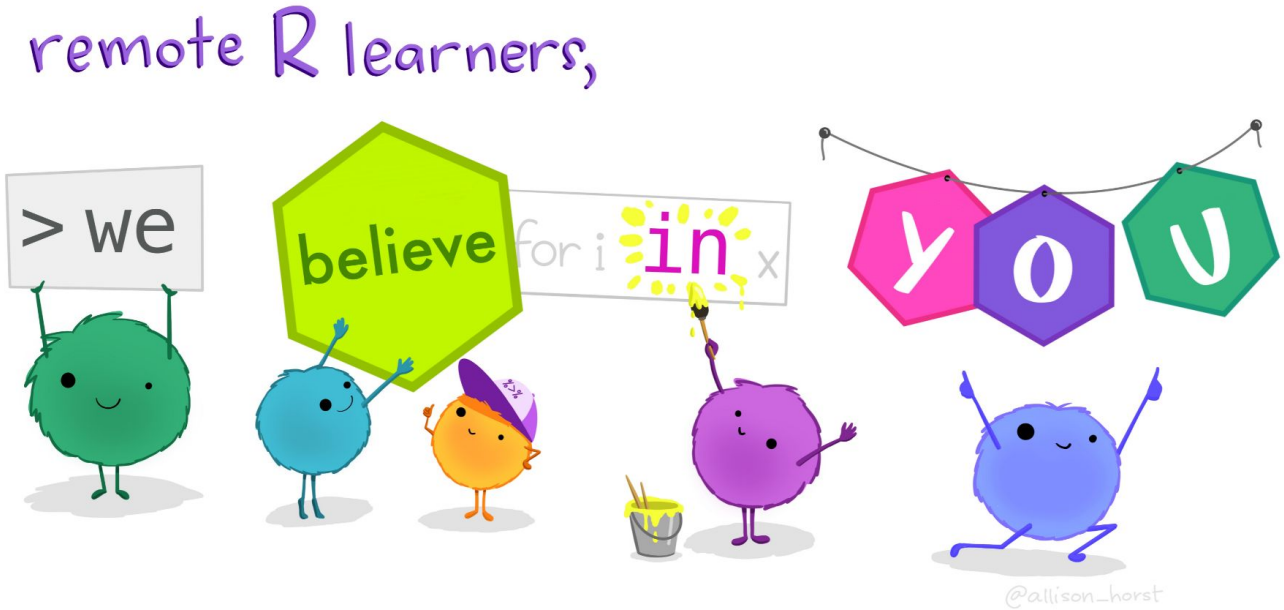
- Google is your best R friend!
- @allison_horst
- https://intro2r.com/
- https://www.r-bloggers.com/
- Hacky Hour sessions



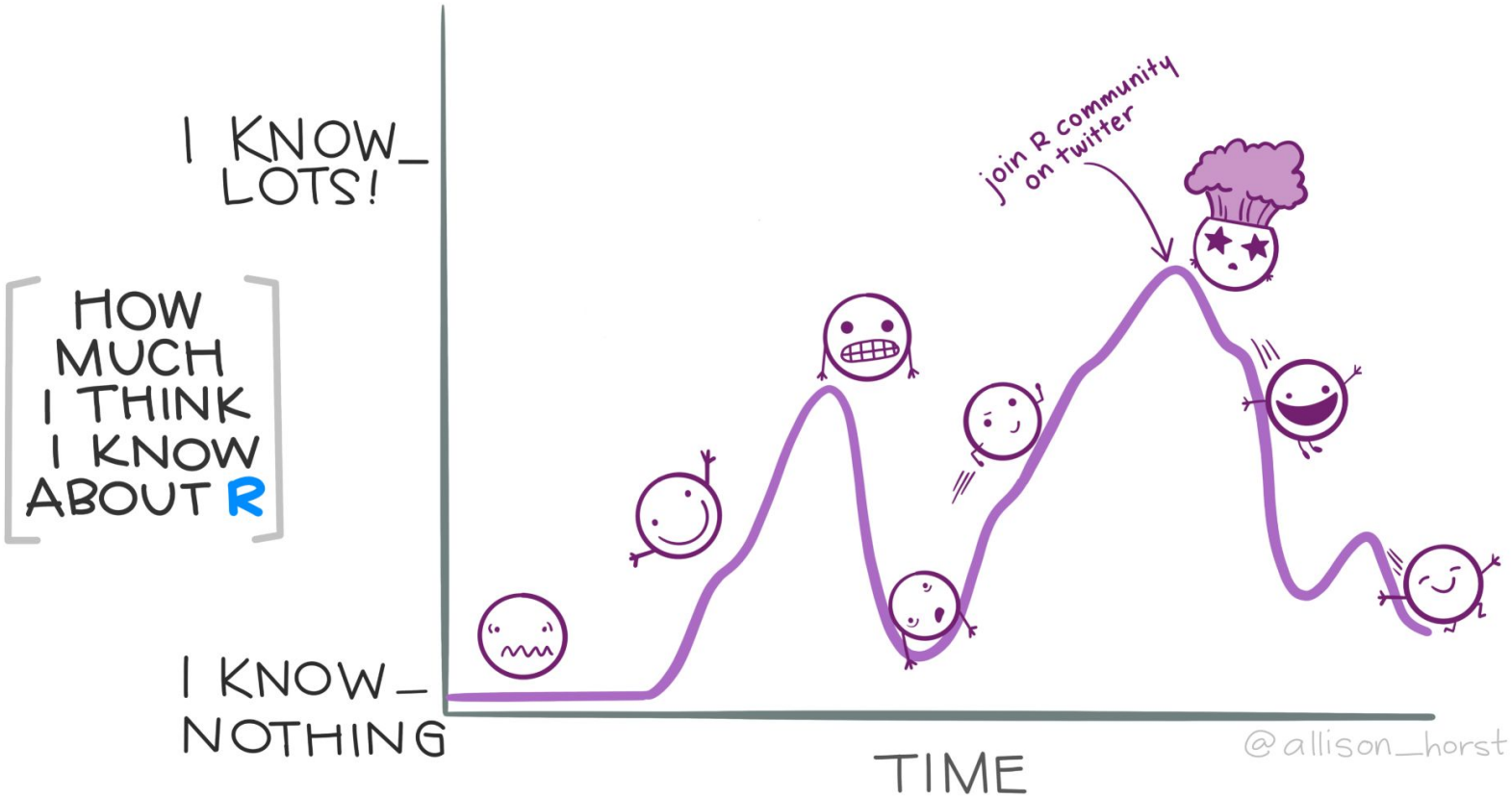
RStudio Education

stackoverflow

Well done! You have learned so much today!



Please go to
www.menti.com
via the code
in our etherpad



Artwork by @allison_horst

Want to connect? #ResBaz2021 slack channel!