# Hertie Coding Club

## Session 2: (Re) Introduction to R

Jorge Roa

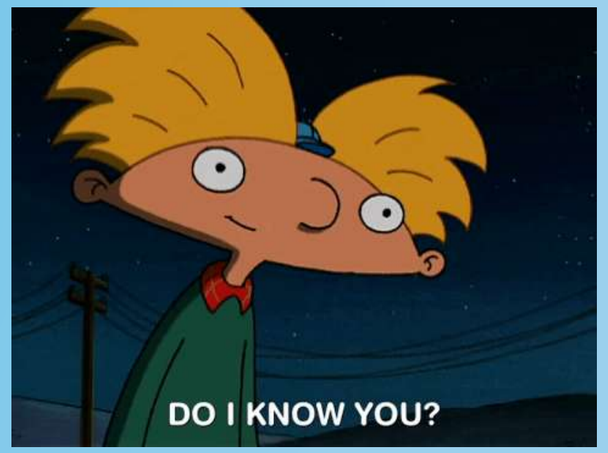
# Agenda for today

# Agenda for today

- Little introduction
- Recap from the first session
- Tidyverse and packages
- Base R and Tidyverse
- Wrangle dataframes

# Little Introduction

# Little Introduction




I want to know you:

I want you to answer four questions:

- Your name
- What are you studying?
- What would you sing at Karaoke night?
- Any unpopular opinion?

# Recap from the first session

# Recap from the first session

Objects:

To create an object, give it a name followed by the assignment operator, followed by the value.

- Assignment operator `<-`
- Can also use `=` but not recommended
- Shortcut: `Alt + -` on PC, `Option + -` on Mac

```
x <- 2 + 2

x
```

```
[1] 4
```

# More type of objects

There are 5 basic types of objects in the R language:

- **Atomic vectors** are one of the basic types of objects in R programming. Atomic vectors can store homogeneous data types such as character, doubles, integers, raw, logical, and complex.
- **List** is another type of object in R programming. List can contain heterogeneous data types such as vectors or another lists.

```r
#Numeric vector
numbers <- c(1, 2, 3, 4)

#String vector
characters <- c("a", "b", "c", "d")

#Numeric value
value <- 5

#List
my_list <- list(c(1, 2, 3, 4), list("a", "b", "c"))
```

```r
print(numbers)
```

```
[1] 1 2 3 4
```

```r
print(characters)
```

```
[1] "a" "b" "c" "d"
```

```r
print(value)
```

```
[1] 5
```

```r
print(my_list)
```

```
[[1]]
[1] 1 2 3 4

[[2]]
[[2]][[1]]
[1] "a"

[[2]][[2]]
[1] "b"

[[2]][[3]]
[1] "c"
```

# More type of objects

- **Matrices**: To store values as 2-Dimensional array, matrices are used in R. Data, number of rows and columns are defined in the matrix() function.
- **Factors**: Factor object encodes a vector of unique elements (levels) from the given data vector.
- **Arrays**: array() function is used to create n-dimensional array. This function takes dim attribute as an argument and creates required length of each dimension as specified in the attribute.

```r
x <- c(1, 2, 3, 4, 5, 6)

# Matrix
mat <- matrix(x, nrow = 2)

# array
arr <- array(c(1, 2), dim = c(3, 3))
```

```r
print(mat)
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```r
print(arr)
```

```
     [,1] [,2] [,3]
[1,]    1    2    1
[2,]    2    1    2
[3,]    1    2    1
```

# Finally: dataframes

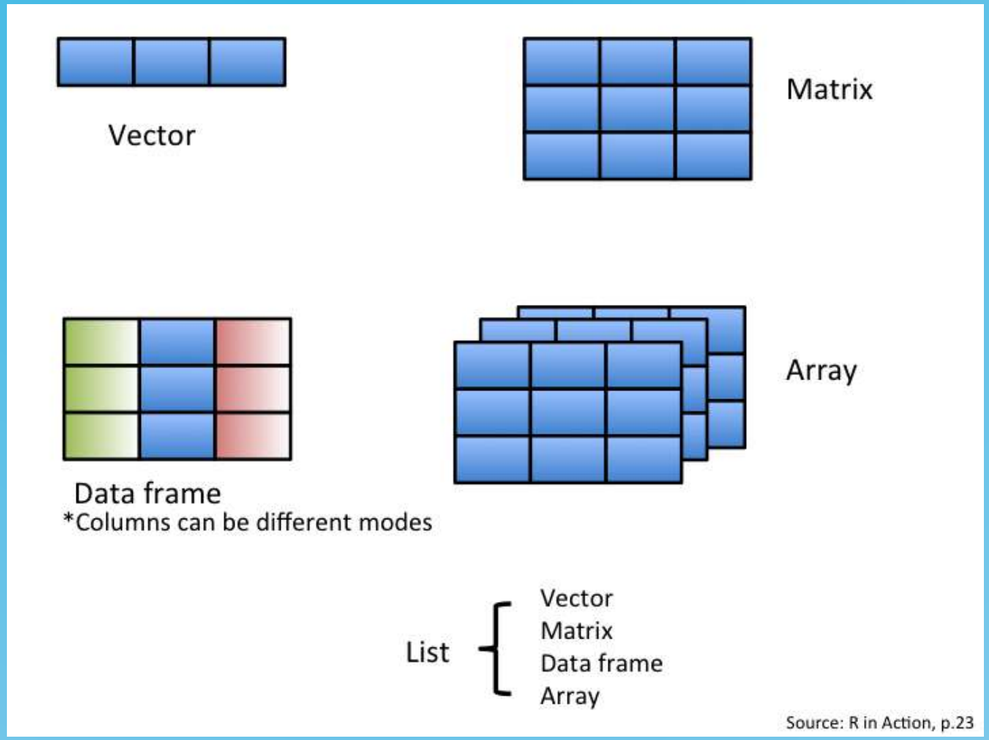
- Data frames are 2-dimensional tabular data object in R programming.
- Data frames consists of multiple columns and each column represents a vector.
- Columns in data frame can have different modes of data unlike matrices.

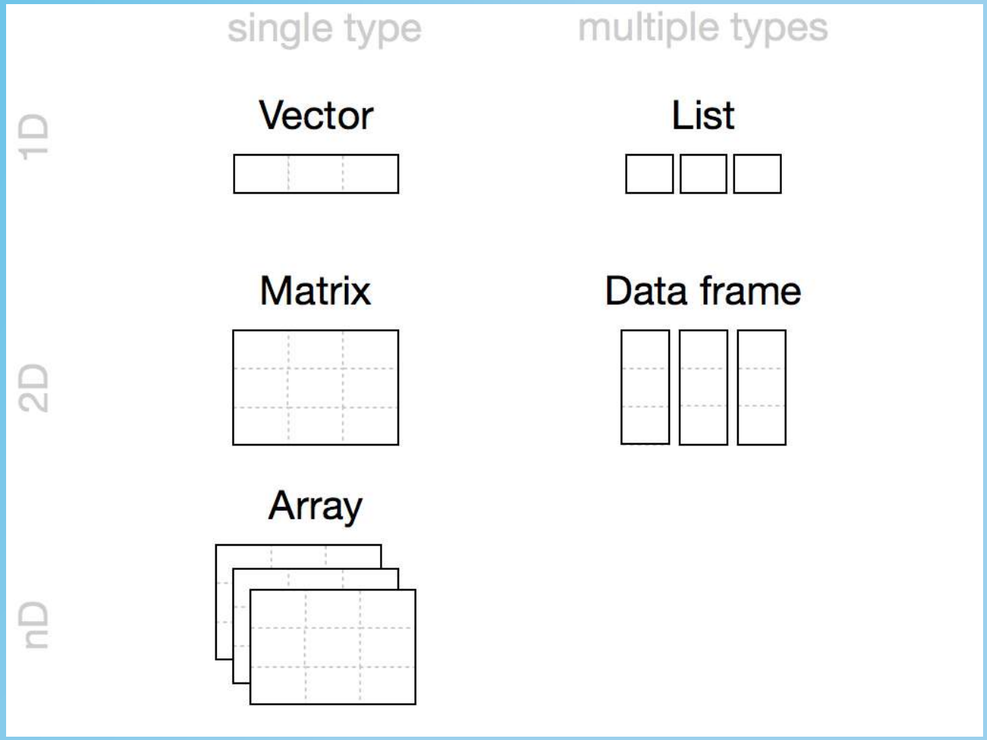
```r
# Create vectors
who <- c("Mom", "Sister", "Myself", "Dad", "Brother", "Brother", "Our dog (:")
age <- c(58, 17, 25,60, 29, 27, 5)
names <- c("Carmen", "Fernanda", "Jorge", "Arturo", "Ale", "Eduardo", "Rocky")

# Create data frame of vectors
df_my_family <- data.frame(who, age, names)
```

```r
print(df_my_family)
```

```
         who age    names
1        Mom  58   Carmen
2     Sister  17 Fernanda
3     Myself  25    Jorge
4        Dad  60   Arturo
5    Brother  29      Ale
6    Brother  27  Eduardo
7 Our dog (:   5    Rocky
```

# Objects:summary



# Operations of vectors

# Operations of numeric vectors

- length(x): how many elements you have in your vector.
- sort(x, decreasing = F): sort your numerical values.
- sum(x): returns the sum of your values.
- min(x): minimum value of your numeric vector.
- mean(x): mean of your numeric vector.
- median(x): median
- sd(x): Standard deviation.
- var(x): variance of your numeric vector.
- summary(x): summary of your numeric vector.

```r
v_age <- c(22, 25, 36, 60, 15, 25, 20, 10)

length(v_age)
```

```
[1] 8
```

```r
sort(v_age, decreasing = F)
```

```
[1] 10 15 20 22 25 25 36 60
```

```r
sum(v_age)
```

```
[1] 213
```

```r
min(v_age)
```

```
[1] 10
```

```r
mean(v_age)
```

```
[1] 26.625
```

```r
median(v_age)
```

```
[1] 23.5
```

```r
sd(v_age)
```

```
[1] 15.50979
```

```r
var(v_age)
```

```
[1] 240.5536
```

```r
summary(v_age)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  10.00   18.75   23.50   26.62   27.75   60.00 
```

# Exercises

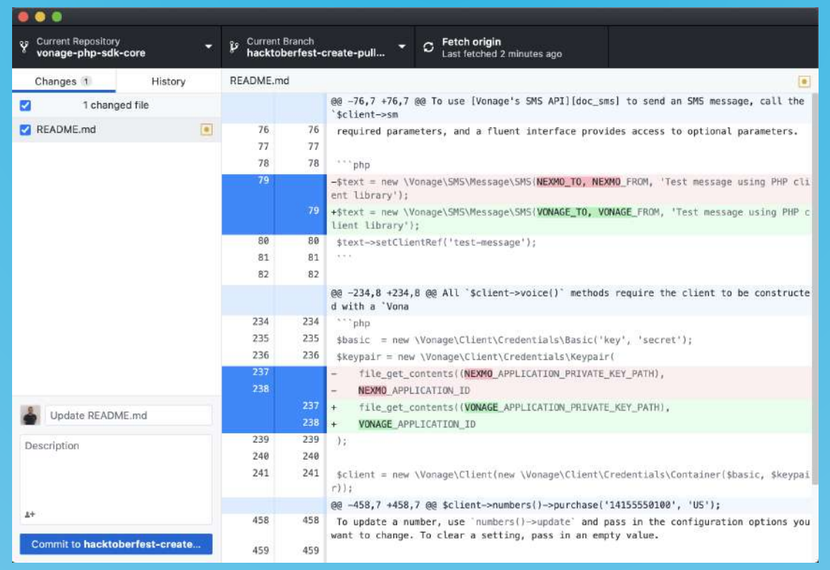
# Exercises

```r
#1.-Create a list containing strings, numbers, vectors and a logical values.

#2.-Create a dataframe of 5 variables
#Hint: Remember the length of the vectors

#3.- Create a vector with numerical values and strings with a length of 10

#4.- Assign the following vectors to a meaningful variable name:
#Hint: Remember the assignment operator.

c(2, 4, 6, 8, 10, 12, 14, 16, 20)
0
3.141593
c(1, 10, 100, 1000, 10000, 100000)

#5.- Create vectors that correspond to the following variables names:

yourage
days_of_the_week
firstFivePrimeNumbers
```

```r
#1.-Create a list containing strings, numbers, vectors and a logical values.
list <- list(c("Coding", "Club"), c(1,2,3), 7)

#2.-Create a dataframe of 5 variables
#Hint: Remember the length of the vectors
df_my_family <- data.frame(number = c(1,2,3),
                           age = c(17,18,19),
                           name = c("Alex", "Eduardo", "Jorge"),
                           favorite_color = c("blue", "orange", "black"),
                           favorite_number = c(20, "5", 50))

#3.- Create a vector with numerical values and strings with a length of 10
vector <- c(1,2,3,"number",99, 100, "yes", "hi", 9, 10)

length(vector)

#4.- Assign the following vectors to a meaningful variable name:
#Hint: Remember the assignment operator.
vector <- c(2, 4, 6, 8, 10, 12, 14, 16, 20)
value <- 0
value_2 <- 3.141593
num_vector <- c(1, 10, 100, 1000, 10000, 100000)

#5.- Create vectors that correspond to the following variables names:
yourage <- c(25)
days_of_the_week <- c("Monday", "Tuesday", "Wednesday",
                      "Thursday", "Friday", "Saturday",
                      "Sunday")
firstFivePrimeNumbers <- c(2,3,5,7,11)
```

# GitHub

# GitHub



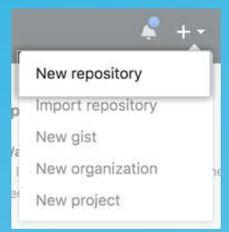

GitHub

- Website and cloud-based service to store and manage code
- Git IDE: used in the programming world. It is used for tracking changes in the source code during software development.
- It makes it easier for individuals and teams to use Git for version control and collaboration.

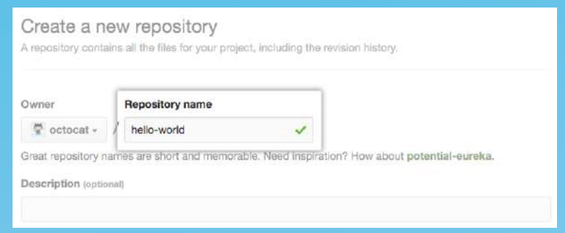
# Create our first repo

# Create our first repo

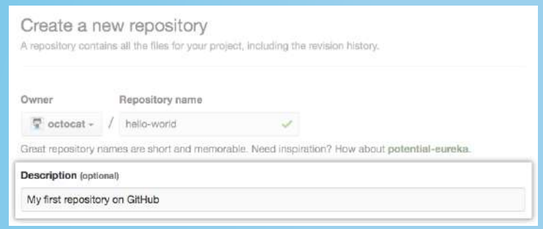
1. In the upper-right corner of any page, use the drop-down menu, and select New repository.



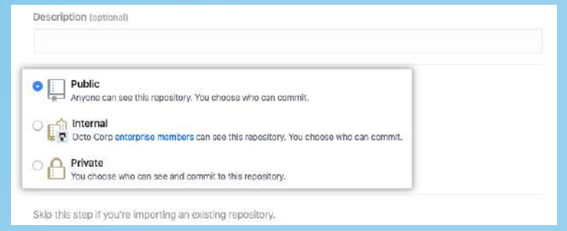

2. Type a short, memorable name for your repository. For example, “intro-to-r”.




3. Optionally, add a description of your repository. For example, “My first repository on GitHub.”

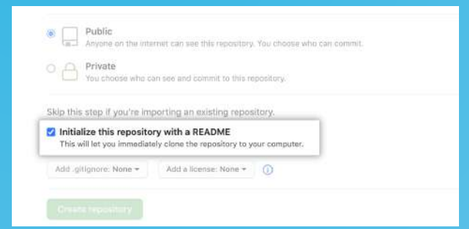



4. Choose a repository visibility.

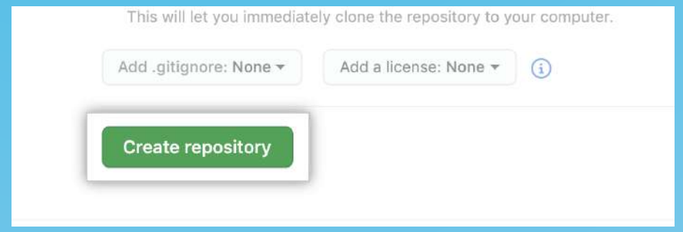


# Create our first repo

5. Select Initialize this repository with a README.




6. Click Create repository.





CONGRATULATIONS: YOU CREATED YOUR FIRST REPO

# We will explain how it works

# Tidyverse and packages

# Tidyverse and packages

- R packages are a collection of R functions, complied code and sample data.
- **Why**: There are millions of functions. If they were all preloaded, there wouldn't be enough RAM to work with. There are packages of such varied disciplines that we likely use relatively few.
- They are stored under a directory called "library" in the R environment.
- By default, R installs a set of packages during installation. More packages are added later, when they are needed for some specific purpose.
- When we start the R console, only the default packages are available by default.
- Other packages which are already installed have to be loaded explicitly to be used by the R program that is going to use them.
- We can also generate our functions and even create an R package!

# Tidyverse

Package set for: Import, Clean, Transform, Process, Analyze and Visualize

# File Import/Export

readr

Package set for: load plain text files (txt, csv, tsv)

# File Import/Export

readxl

Package set for: load excel files (xls, xlsx)

# File Import/Export

haven

Package set for: Display proprietary formats (dta, sav). Like STATA and other formats.

# Wrangling data

tidyr

Package set for: transform dataframe structures

# Wrangling data

lubridate

Package set for: wrangling dates. Tools that make working with dates and times easier.

# Wrangling data

stringr

Package set for: wrangling string or characters.

# Wrangling data

dplyr

Package set for: wrangling dataframes. facilitates several functions for the data frames in R. dplyr package is for data wrangling and data analysis purposes.

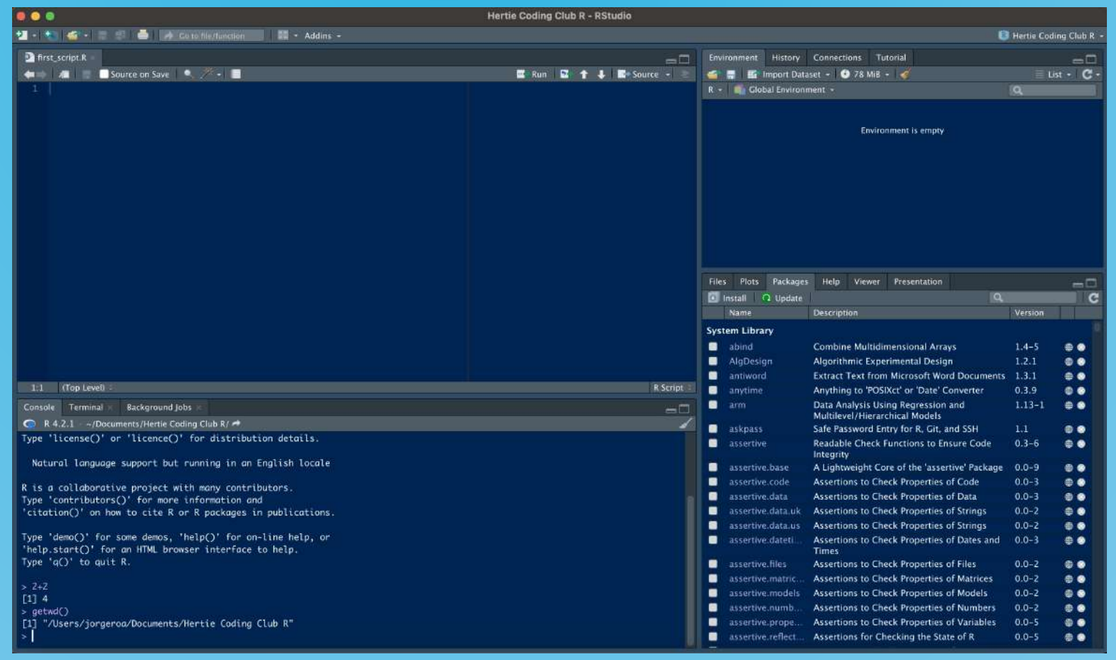
# Analysis and visualization

ggplot

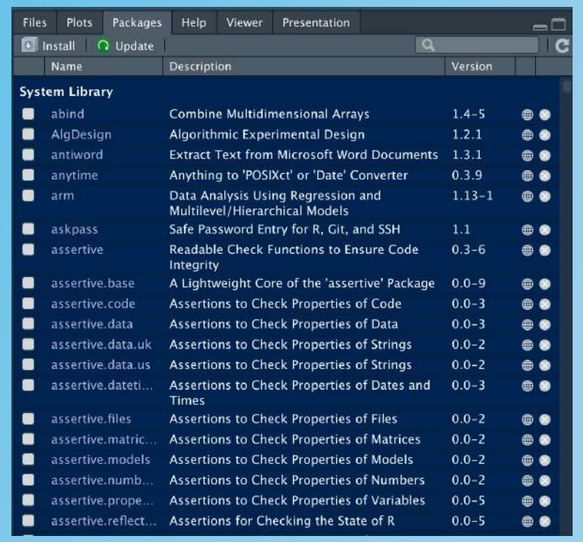
Package set for: plots and maps. One of the most popular visualization package in R.

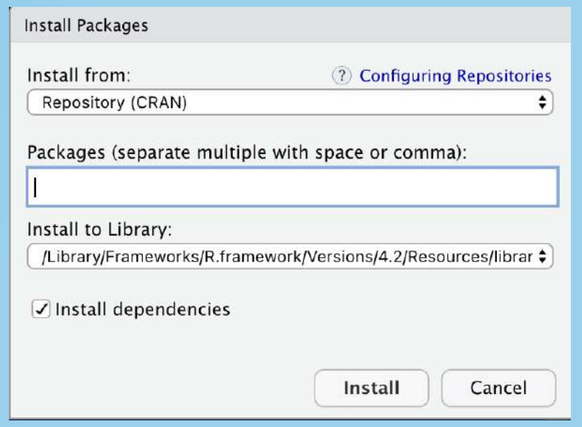
# How we install packages

The easy way



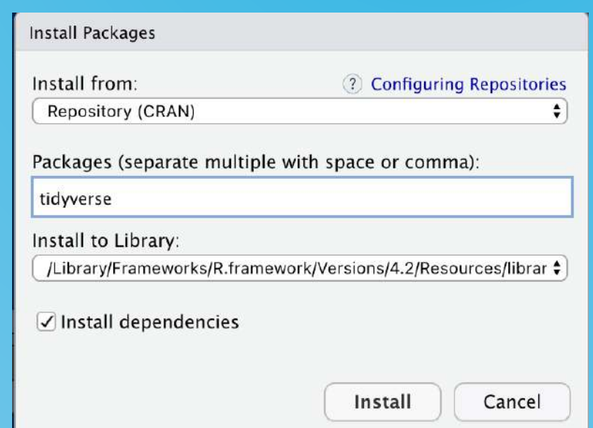

- Go to the “Packages” tab
- Press the “Install” button



# How we install packages

The easy way




- Other way is type in the console
- install.packages("tidyverse")

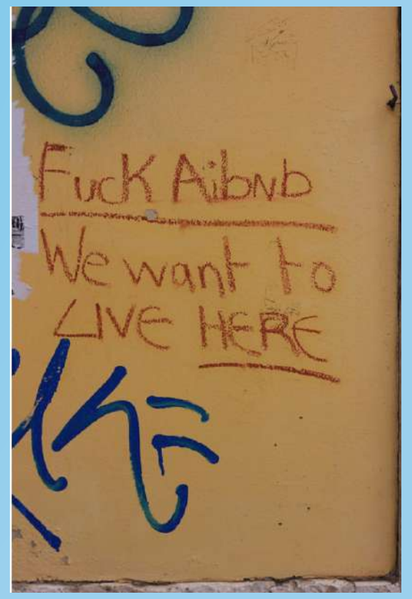
```
> install.packages("tidyverse")
trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.2/tidyverse_1.3.2.tgz'
Content type 'application/x-gzip' length 420896 bytes (411 KB)
==================================================
downloaded 411 KB


The downloaded binary packages are in
        /var/folders/q3/ztffc4r10tv5h4pftjr2qw0r0000gn/T//RtmpKwSpEV/downloaded_packages
>
```

# Dataframes

# Dataframes

Example

- We will work with Airbnb accommodation data in Berlin as of September 15, 2022. They are open data available at Airbnb: get the data.
- They are open data licensed under the Creative Commons CC0 1.0 Universal “Public Domain Dedication.
- Those who stay can choose between entire houses/apartments, only private rooms, or shared rooms (room_type).
- After the stay, they must leave an evaluation (review).
- Accommodations vary in price, a minimum number of days of stay, days available, etc.



# Import dataframes from your computer

## How we import data?

### Excel files

Quite frequently, the sample data is in Excel format, and needs to be imported into R prior to use.

```r
library(readxl)

df_listings <- read_xlsx("data/listings.xlsx")
```

| id | listing_url | scrape_id | last_scraped |
|---|---|---|---|
| <dbl> | <chr> | <dbl> | <dttm> |
| 6.528688e+17 | https://www.airbnb.com/rooms/652868795892201022 | 2.022092e+13 | 2022-09-15 |
| 2.907769e+07 | https://www.airbnb.com/rooms/29077694 | 2.022092e+13 | 2022-09-16 |
| 2.708061e+07 | https://www.airbnb.com/rooms/27080612 | 2.022092e+13 | 2022-09-15 |
| 6.656484e+17 | https://www.airbnb.com/rooms/665648367391379321 | 2.022092e+13 | 2022-09-15 |
| 3.176000e+03 | https://www.airbnb.com/rooms/3176 | 2.022092e+13 | 2022-09-15 |
| 3.773800e+07 | https://www.airbnb.com/rooms/37738004 | 2.022092e+13 | 2022-09-16 |
| 3.309000e+03 | https://www.airbnb.com/rooms/3309 | 2.022092e+13 | 2022-09-15 |
| 7.071000e+03 | https://www.airbnb.com/rooms/7071 | 2.022092e+13 | 2022-09-16 |
| 2.019132e+06 | https://www.airbnb.com/rooms/2019132 | 2.022092e+13 | 2022-09-16 |
| 9.991000e+03 | https://www.airbnb.com/rooms/9991 | 2.022092e+13 | 2022-09-16 |

1-10 of 10,000 rows | 1-4 of 75 columns  Previous 1 2 3 4 5 6 ... 1000 Next

# What do we want to find out from this data?

First, I ask myself questions, then think about the code that answers them.

- What are the variables? How many?
- How many observations do you have?
- What values do these variables take?
- Is there missing data? Are there duplicate cases?

Explore the data

```
View(df_listings)
```

OR (Best option: only works with dataframes)

CTRL + click on your object (IN YOUR SCRIPT).

Filter Cols: « < 1 - 50 > »

| | id | listing_url | scrape_id | last_scraped | source | name |
|---|---|---|---|---|---|---|
| 1 | 6.528688e+17 | https://www.airbnb.com/rooms/652868795892201 0... | 2.022092e+13 | 2022-09-15 | city scrape | Kleine Auszeit? Oder Business-Trip? Alles m√∂glich! |
| 2 | 2.907769e+07 | https://www.airbnb.com/rooms/29077694 | 2.022092e+13 | 2022-09-16 | previous scrape | Wohnung im Gr√º nen nah an der Metropole |
| 3 | 2.708061e+07 | https://www.airbnb.com/rooms/27080612 | 2.022092e+13 | 2022-09-15 | city scrape | Apartment with Living/Sleeping Room & own Kitchen |
| 4 | 6.656484e+17 | https://www.airbnb.com/rooms/665648367391379 3... | 2.022092e+13 | 2022-09-15 | city scrape | Sch√∂ne Eigentumswohnung mit Balkon |
| 5 | 3.176000e+03 | https://www.airbnb.com/rooms/3176 | 2.022092e+13 | 2022-09-15 | city scrape | Fabulous Flat in great Location |
| 6 | 3.773800e+07 | https://www.airbnb.com/rooms/37738004 | 2.022092e+13 | 2022-09-16 | previous scrape | Remise Villa Erica Superior Apartment |
| 7 | 3.309000e+03 | https://www.airbnb.com/rooms/3309 | 2.022092e+13 | 2022-09-15 | city scrape | BerlinSpot Sch√∂neberg near KaDeWe |
| 8 | 7.071000e+03 | https://www.airbnb.com/rooms/7071 | 2.022092e+13 | 2022-09-16 | previous scrape | BrightRoom with sunny greenview! |
| 9 | 2.019132e+06 | https://www.airbnb.com/rooms/2019132 | 2.022092e+13 | 2022-09-16 | previous scrape | Villa am Berliner Stadtrand f√º r bis zu 24 G√§ste |
| 10 | 9.991000e+03 | https://www.airbnb.com/rooms/9991 | 2.022092e+13 | 2022-09-16 | city scrape | Geourgeous flat - outstanding views |
| 11 | 1.432500e+04 | https://www.airbnb.com/rooms/14325 | 2.022092e+13 | 2022-09-15 | city scrape | Studio Apartment in Prenzlauer Berg |
| 12 | 3.257390e+07 | https://www.airbnb.com/rooms/32573899 | 2.022092e+13 | 2022-09-16 | previous scrape | Modern House in Berlin/Teltow-Seehof |
| 13 | 1.664400e+04 | https://www.airbnb.com/rooms/16644 | 2.022092e+13 | 2022-09-16 | city scrape | In the Heart of Berlin - Kreuzberg |
| 14 | 1.791020e+05 | https://www.airbnb.com/rooms/179102 | 2.022092e+13 | 2022-09-15 | city scrape | The Special Place |
| 15 | 1.804400e+05 | https://www.airbnb.com/rooms/180440 | 2.022092e+13 | 2022-09-16 | previous scrape | CITY STUDIO WEST@ Kurf√º rstendamm(monthly disc... |
| 16 | 1.811600e+05 | https://www.airbnb.com/rooms/181160 | 2.022092e+13 | 2022-09-15 | city scrape | Berliner Flair |
| 17 | 1.790400e+04 | https://www.airbnb.com/rooms/17904 | 2.022092e+13 | 2022-09-16 | city scrape | Beautiful Kreuzberg studio - 3 months minimum |
| 18 | 1.828160e+05 | https://www.airbnb.com/rooms/182816 | 2.022092e+13 | 2022-09-15 | city scrape | Cozy 90sqm in Prenzlauer Berg |
| 19 | 2.085800e+04 | https://www.airbnb.com/rooms/20858 | 2.022092e+13 | 2022-09-16 | city scrape | Designer Loft in Berlin Mitte |
| 20 | 1.839880e+05 | https://www.airbnb.com/rooms/183988 | 2.022092e+13 | 2022-09-15 | city scrape | Luxury Ku'damm Studio + Bikes 4you |

# Explore the data

The dim(), names(), and str() functions take a data frame as an argument.

```
dim(df_listings) # dimension of the dataframe
```

```
[1] 16680    75
```

```
nrow(df_listings) # rows
```

```
[1] 16680
```

```
ncol(df_listings) # columns (variables)
```

```
[1] 75
```

```
names(df_listings)[1:10] # variable names
```

```
 [1] "id"                    "listing_url"           "scrape_id"
 [4] "last_scraped"          "source"                "name"
 [7] "description"           "neighborhood_overview" "picture_url"
[10] "host_id"
```

```
#str(df_listings)
```

Explore the data

```
max(df_listings$price) #maximum value.
```

```
[1] 4375
```

```
median(df_listings$price) #median.
```

```
[1] 65
```

```
min(df_listings$price) #minimum value.
```

```
[1] 0
```

```
mean(df_listings$price) #mean.
```

```
[1] 96.30809
```

```
median(df_listings$price) #median.
```

```
[1] 65
```

```
var(df_listings$price) #variance.
```

```
[1] 13638.87
```

```
sd(df_listings$price) #Standard deviation.
```

```
[1] 116.7856
```

# Exercise

- Upload the Barcelona listings.xlsx file.
- Calculate the mean, median, and variance of the variables minimum_nights, number_of_reviews, and last_review. Then, store those results in different vectors and create a dataframe with them. Name the dataframe with `df_exercise_1`

# Exercise

- Upload the Berlin listings.csv file.
- Calculate the mean, median, and variance of the variables minimum_nights, number_of_reviews, and last_review. Then, store those results in different vectors and create a dataframe with them. Name the dataframe with `df_exercise_1`
- Why get NA in the variance of the last_review variable?
  - It's a string. We can't apply numerical function to strings.

```r
v_mean_minimum_nights <- mean(df_listings$minimum_nights)
v_mean_number_of_reviews <- mean(df_listings$number_of_reviews)
v_mean_last_review <- mean(df_listings$last_review)

v_med_minimum_nights <- median(df_listings$minimum_nights)
v_med_number_of_reviews <- median(df_listings$number_of_reviews)
v_med_last_review <- median(df_listings$last_review)

v_var_minimum_nights <- var(df_listings$minimum_nights)
v_var_number_of_reviews <- var(df_listings$number_of_reviews)
v_var_last_review <- var(df_listings$last_review)


df_exercise_1 <- data.frame(mean = c(v_mean_minimum_nights,
                                     v_mean_number_of_reviews,
                                     v_mean_last_review),
                            median = c(v_med_minimum_nights,
                                       v_med_number_of_reviews,
                                       v_med_last_review),
                            variance = c(v_var_minimum_nights,
                                         v_var_number_of_reviews,
                                         v_var_last_review))


df_exercise_1
```

```
      mean median variance
1 12.11559      3 1530.456
2 27.76787      6 3881.410
3       NA     NA       NA
```

# Thanks for your time

Hertie Coding Club

Remember that everybody can learn how to code!!

1,100 lines of code where created for this presentation.