Introduction to R

Ernesto Cuadra Foy
Insert Presentation Date

Outline

- What is `R`
- Setting up `RStudio` in your computer
- Examples
- Basics of `R`
- Some `R` Coding
- Importing a dataset in `RStudio`

What is R?

- R is a programming language environment for statistical computing

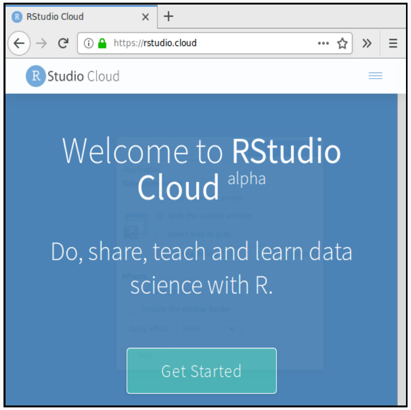
- Widely used among statisticians, data scientists and is currently ranked 9th in the TIOBE index

- R includes a wide variety of statistical and graphical techniques

- Many free resources to learn `R`
- **It's free!**

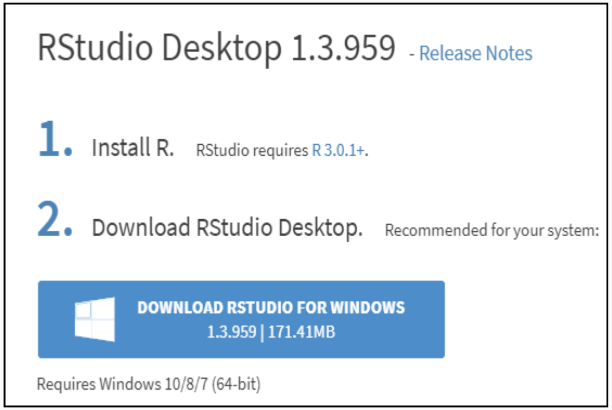
Setting up R

- R can be accessed through two methods
- Downloading and installing RStudio Desktop and R from the following link
  - https://rstudio.com/products/rstudio/download
- Using the cloud version of RStudio. No install required, can be run from the browser
  - https://rstudio.cloud/



Welcome to RStudio Cloud alpha

Do, share, teach and learn data science with R.



RStudio Desktop 1.3.959 - Release Notes

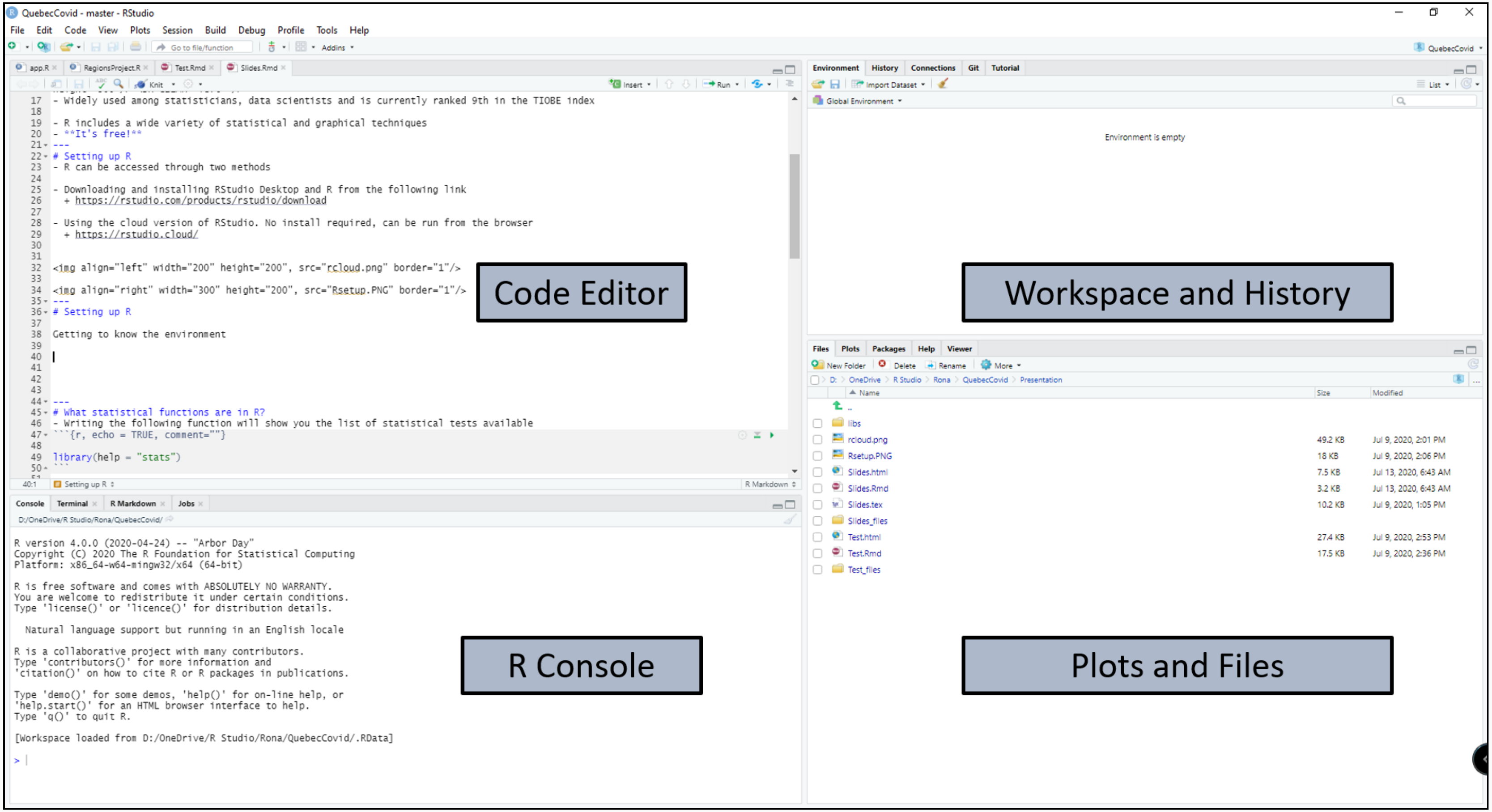
1. Install R. RStudio requires R 3.0.1+.

2. Download RStudio Desktop. Recommended for your system:

Requires Windows 10/8/7 (64-bit)

Working to RStudio

- Getting to know the environment



~~~
- Widely used among statisticians, data scientists and is currently ranked 9th in the TIOBE index

- R includes a wide variety of statistical and graphical techniques
- **It's free!**
---
# Setting up R
- R can be accessed through two methods

- Downloading and installing RStudio Desktop and R from the following link
  + https://rstudio.com/products/rstudio/download

- Using the cloud version of RStudio. No install required, can be run from the browser
  + https://rstudio.cloud/


<img align="left" width="200" height="200", src="rcloud.png" border="1"/>

<img align="right" width="300" height="200", src="Rsetup.PNG" border="1"/>
---
# Setting up R

Getting to know the environment





---
# What statistical functions are in R?
- Writing the following function will show you the list of statistical tests available
```{r, echo = TRUE, comment=""}

library(help = "stats")
```
~~~

D:/OneDrive/R Studio/Rona/QuebecCovid/

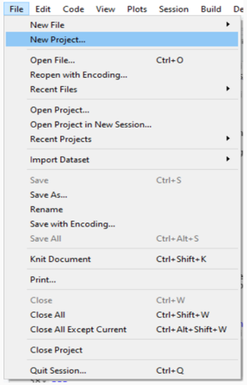
```
R version 4.0.0 (2020-04-24) -- "Arbor Day"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

[Workspace loaded from D:/OneDrive/R Studio/Rona/QuebecCovid/.RData]

> |
```

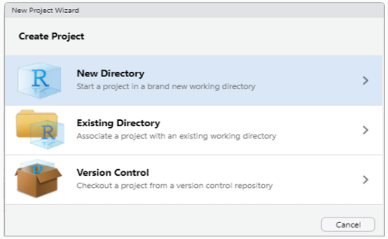
D: > OneDrive > R Studio > Rona > QuebecCovid > Presentation

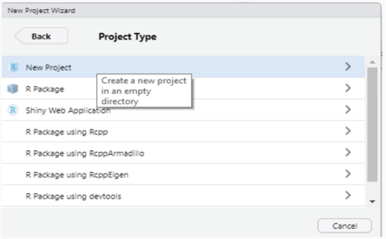
Name	Size	Modified
..		
libs		
rcloud.png	49.2 KB	Jul 9, 2020, 2:01 PM
Rsetup.PNG	18 KB	Jul 9, 2020, 2:06 PM
Slides.html	7.5 KB	Jul 13, 2020, 6:43 AM
Slides.Rmd	3.2 KB	Jul 13, 2020, 6:43 AM
Slides.tex	10.2 KB	Jul 9, 2020, 1:05 PM
Slides_files		
Test.html	27.4 KB	Jul 9, 2020, 2:53 PM
Test.Rmd	17.5 KB	Jul 9, 2020, 2:36 PM
Test_files		

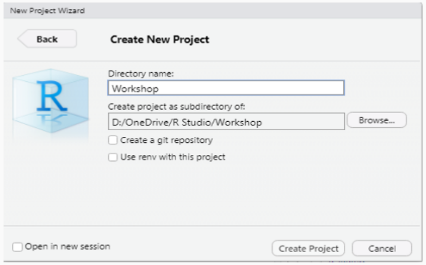
Working with RStudio

- Before starting any kind of work it is **very important** to have your environment set
- An environment is the space where all your variables, formulas, plots and everything related to your project will be saved




New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel





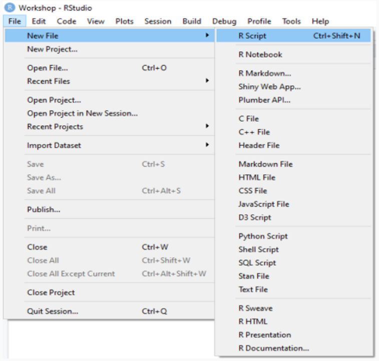
Directory name: Workshop

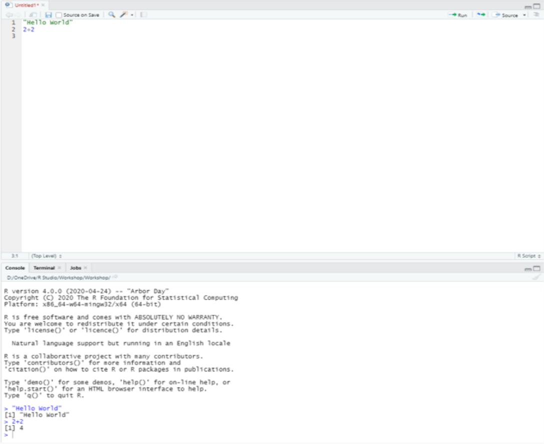
Create project as subdirectory of: D:/OneDrive/R Studio/Workshop

☐ Create a git repository

☐ Use renv with this project

☐ Open in new session





```
"Hello World"
2+2
```

```
R version 4.0.0 (2020-04-24) -- "Arbor Day"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

> "Hello World"
[1] "Hello World"
> 2+2
[1] 4
> 
```

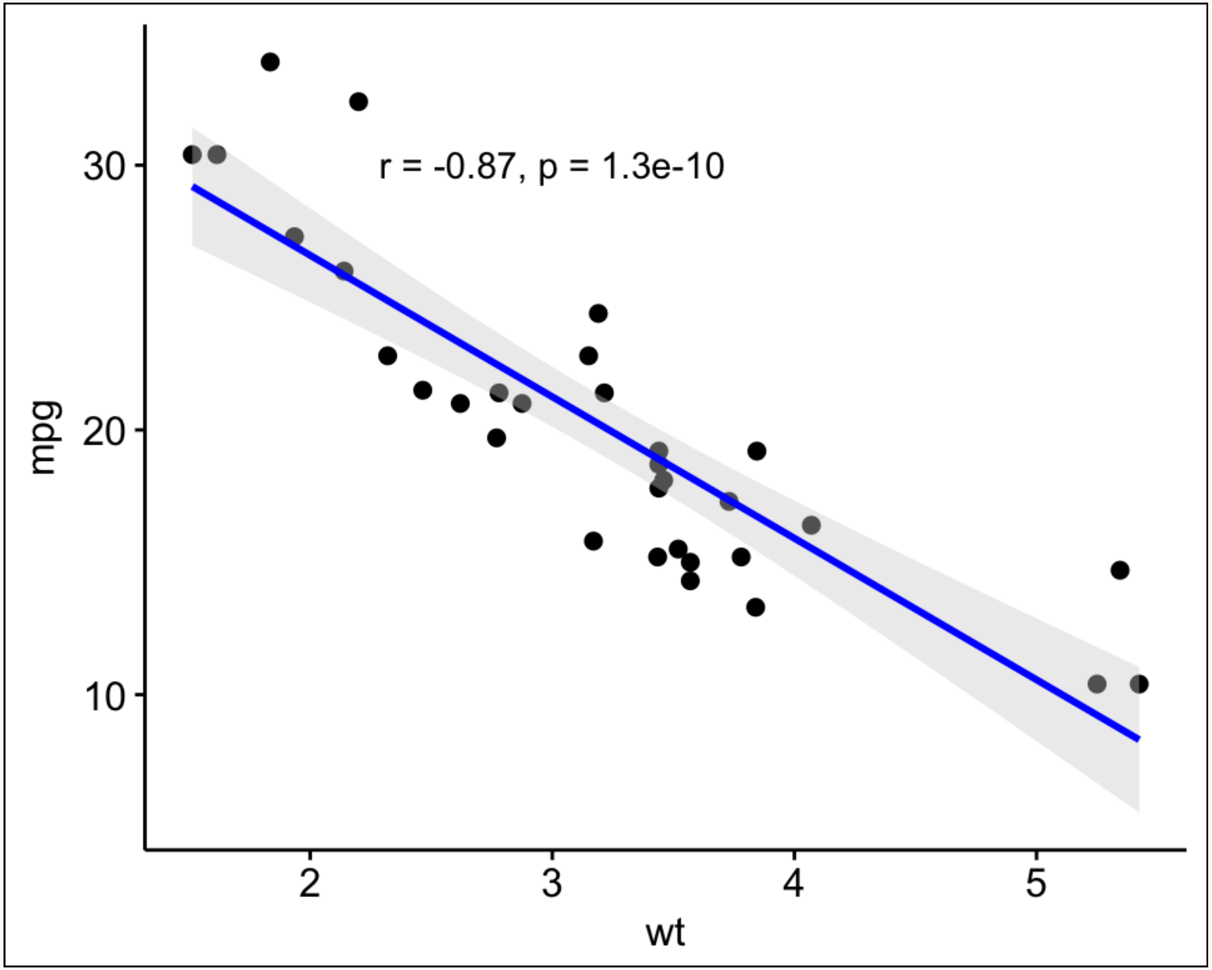
What statistical functions are in R?

- Writing the following function will show you the list of statistical tests available

```
library(help = "stats")
```

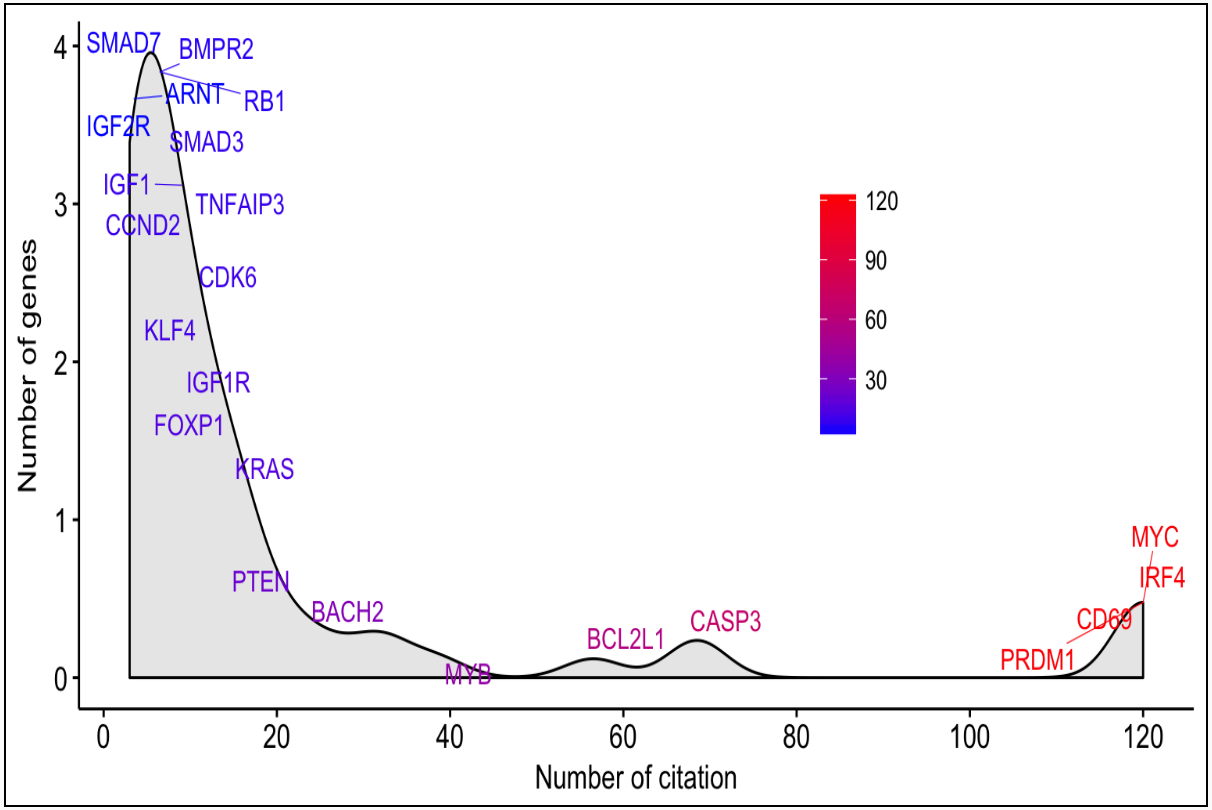
Examples of R Graphics

Scatterplot with statistical notation



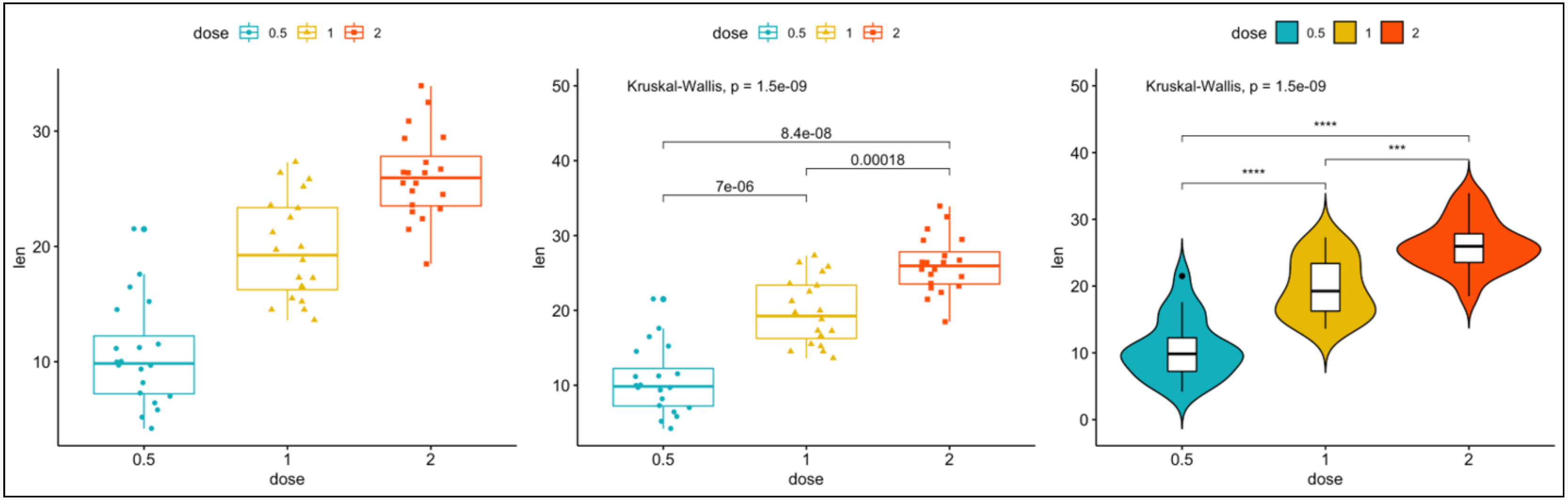
Examples of R Graphics

Histograms



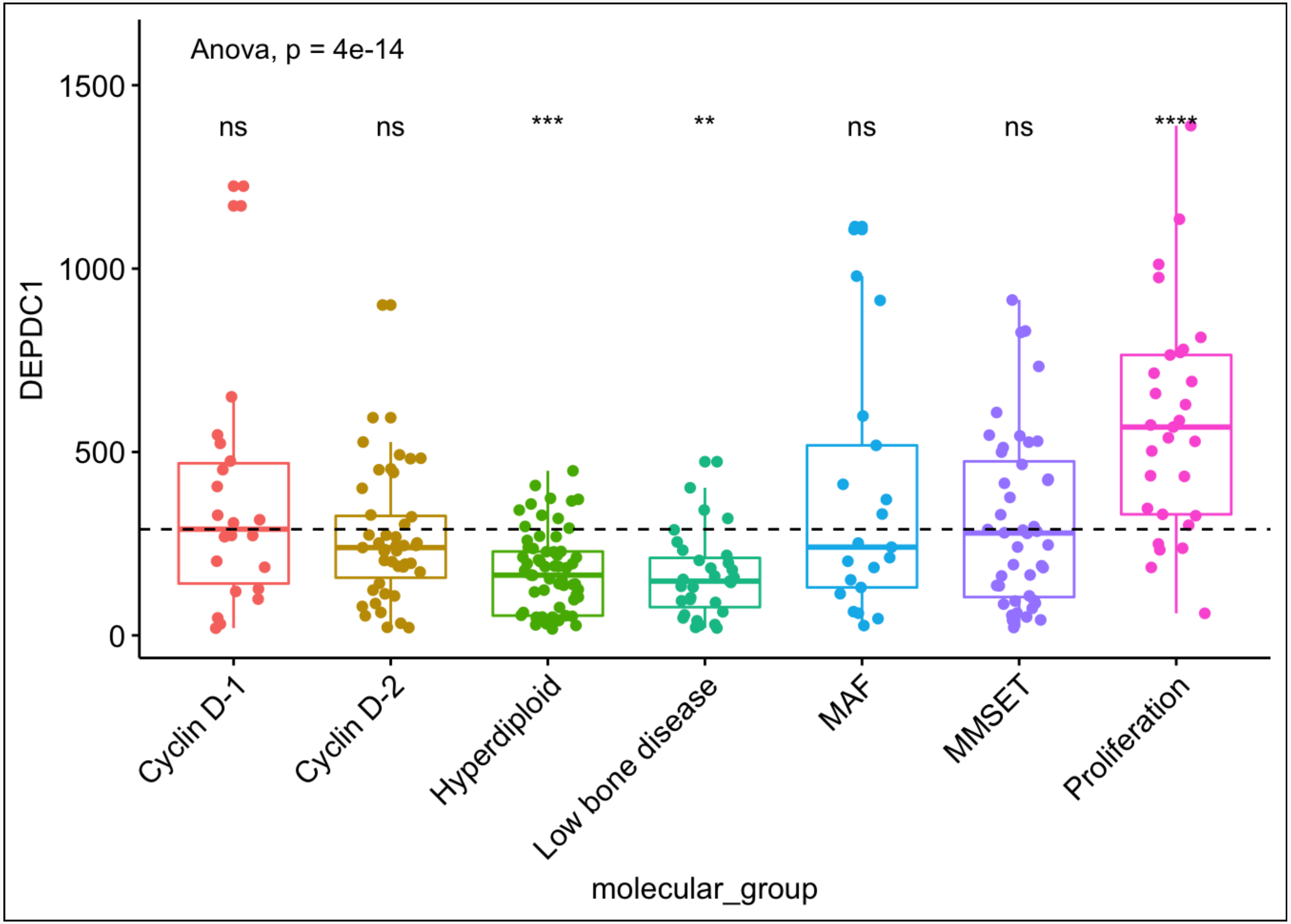
Examples of R Graphics

Boxplot



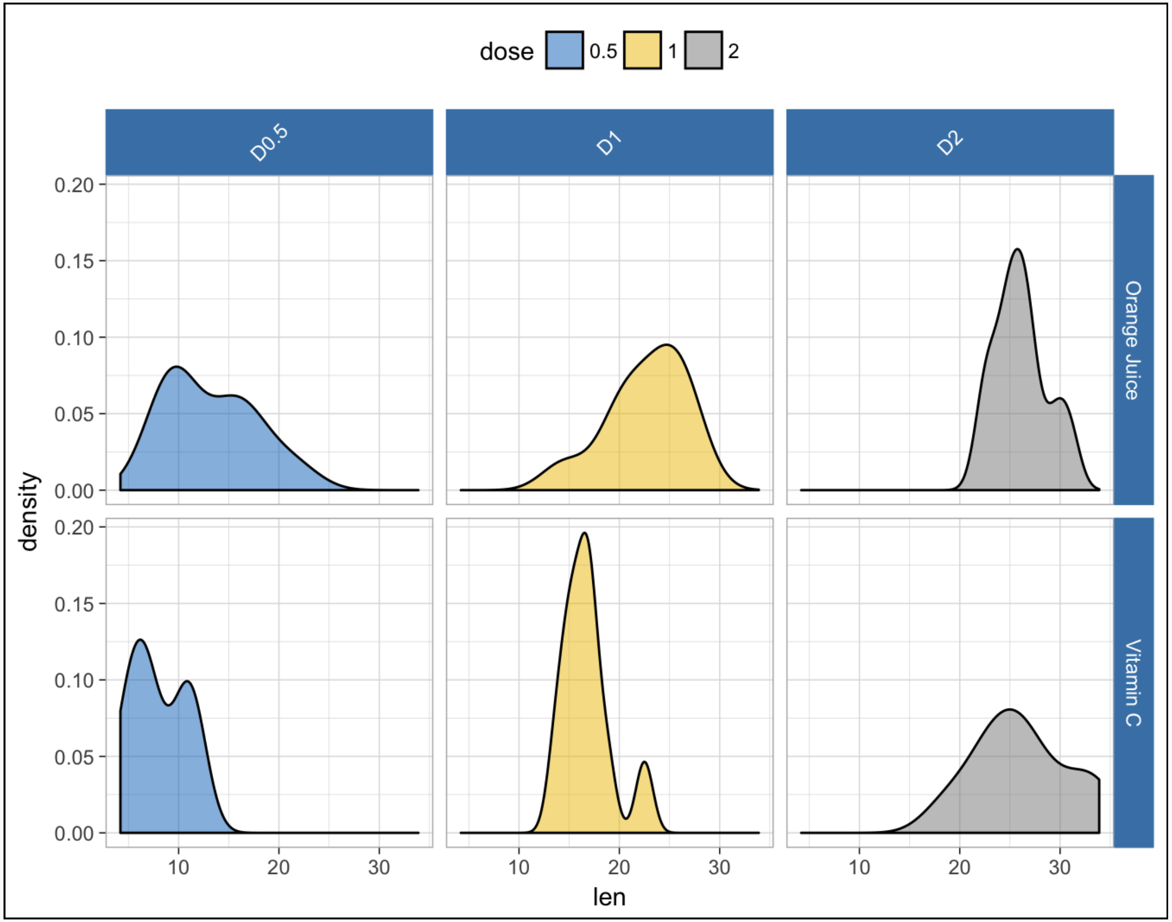
Examples of R Graphics

Boxplot



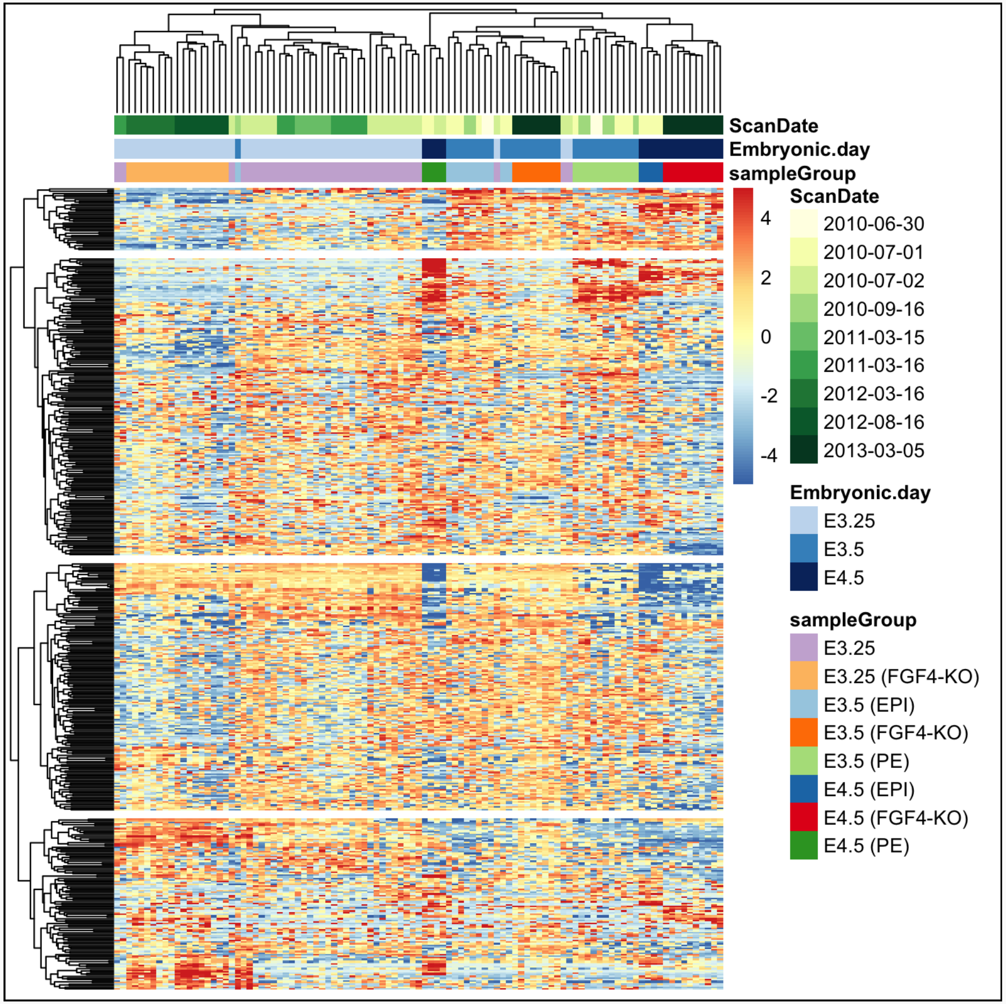
Examples of R Graphics

Faceting graphs



Examples of R Graphics

Heatmaps



Specialized tools for Bioinformatics

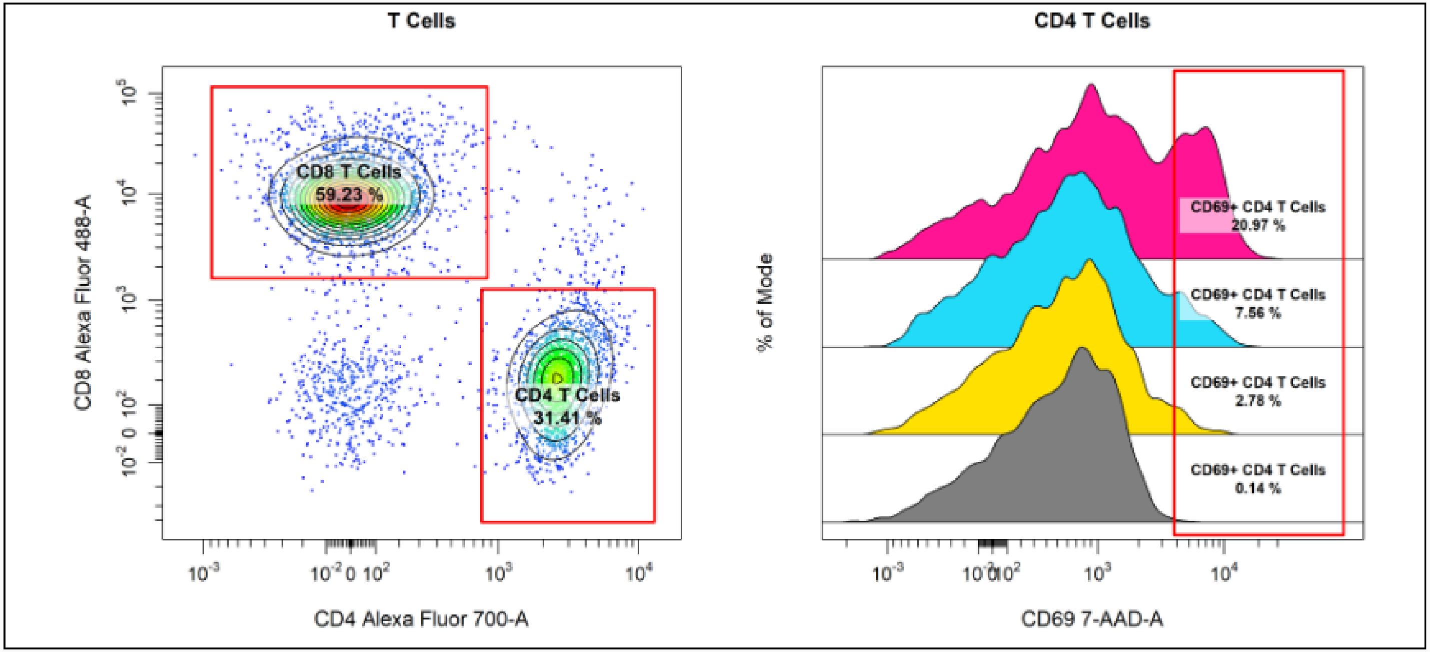
Bioconductor is a project that focuses in providing `R` tools for analysis of high-throughput biological assays. Some of the project's goals include:

- Provide widespread access to a borad range of powerful statistical and graphical methods for analysis of genomic data
- Facilitate he inclusion of biological metadata in the analysis of genomic data from different sources
- Provide a common software platform that enables rapid development of interoperable software
- Train researchers on computational and statistical methods

Specialized tools for Bioinformatics

CytoExploreR

- Specifically designed to integrate all existing cytometry analysis techniques (e.g. manual gating, automated gating and dimension reduction)



Specialized tools for Bioinformatics

CytoExploreR

Basic R Operations

```
1+1  #Simple Arithmetic
```

```
[1] 2
```

```
2 + 3 * 4 #PEMDAS also applies
```

```
[1] 14
```

```
3 ^ 2 #Exponentiation
```

```
[1] 9
```

```
exp(1) #Basic mathematical functions are available
```

```
[1] 2.718282
```

Basic R Operations

```r
sqrt(10)
```

```
[1] 3.162278
```

```r
pi #constants are also defined within R
```

```
[1] 3.141593
```

```r
2+pi*43 #Some random operation
```

```
[1] 137.0885
```

Variables in R

- Variables are names for values. Created variables can be found in the workspace pane.
- Numeric
  - Store floating point values
- Boolean (True or False)
  - TRUE or FALSE are reserved values, can not be used to name variables
- Strings
  - Essentially text, sequences of characters
- A variable can be determined with the `←` operator

Variables in R

- Shortcut for the `←` operator, `Alt + -`
- `x ← 1` Assigns 1 to a variable called `x`
- `y ← 3` Assigns 3 to a variable called `y`
- `z ← 4` Assigns 4 to a variable called `z`
- `x*y*x` We can perform operations with variables, if allowed

```
[1] 12
```

Variables in R

- Variables under the same name can overwrite each other when called later

```
x <- 1
x <- 3

x == 1 #Is x still 1?
```

```
[1] FALSE
```

```
x == 3 #Is x now 3?
```

```
[1] TRUE
```

Variables in R

- Certain rules must be respected when naming variables
- `letters_123. ← 4` Letters, numbers, dots, and underscores are allowed
  - `letters.123_awesome`
  - Can start with a dot but it should not be followed by a number `.letters123`
- Variables can't start with numbers `1illegalvariable`
- The `%` character isn't allowed `another%badvariable`
- Starting with `_` isn't valid `_lastbadvariable`
- Variables can hold Numeric, Boolean and String values the same way
  - `x ← 23`
  - `y ← "I love this variable"`

Vectors

- In `R` a vector is a collection of numbers, individual numbers are elements of the vector

```
myvector <- c(10,20,30,40,50)

myvector + 1
```

```
[1] 11 21 31 41 51
```

```
myvector + myvector
```

```
[1]  20  40  60  80 100
```

```
length(myvector)
```

```
[1] 5
```

```
c(myvector, myvector)
```

```
 [1] 10 20 30 40 50 10 20 30 40 50
```

Vectors

- We can create and index vectors with `:`
- Indexing means accessing specific elements in the vector

```
autovector ← 1:10
autovector
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
autovector[1]
```

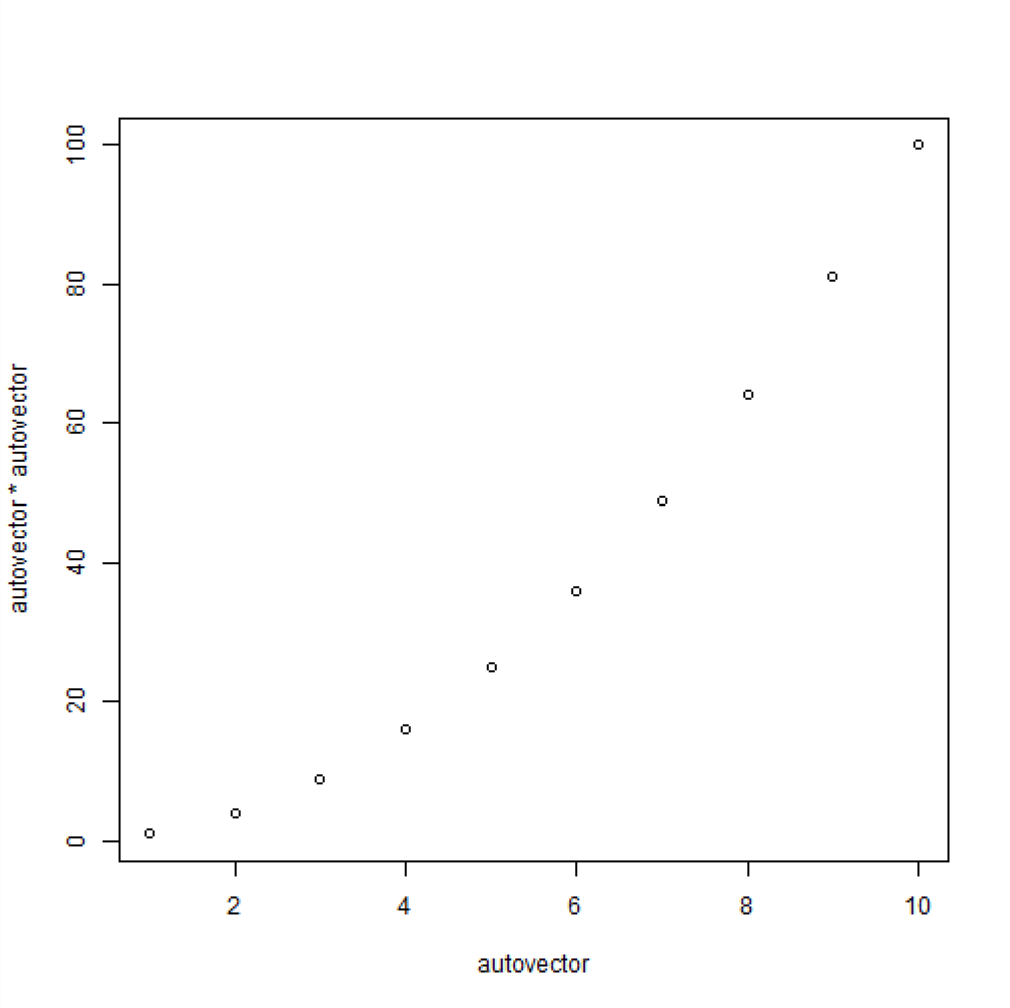
```
[1] 1
```

```
items ← c("spam", "eggs", "beans", "bacon", "sausage")
items[1:4]
```

```
[1] "spam"  "eggs"  "beans" "bacon"
```

Vectors

```
plot(autovector, autovector*autovector)
```



Functions

- Functions are things that `R` do for us: calculate, manipulate data, read and write to files, produce plots.
- `R` contains many functions and we can also build our own (more advanced stuff)

```
myvector
```

```
[1] 10 20 30 40 50
```

```
sum(myvector)
```

```
[1] 150
```

```
rep(42,10) #Repeats the number 42, 10 times
```

```
 [1] 42 42 42 42 42 42 42 42 42 42
```

```
rep(c(1,2,3),10) #Functions work from the inner brackets to the outer ones. In this ca
```

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

Working with Data

Data frames and the Tidyverse

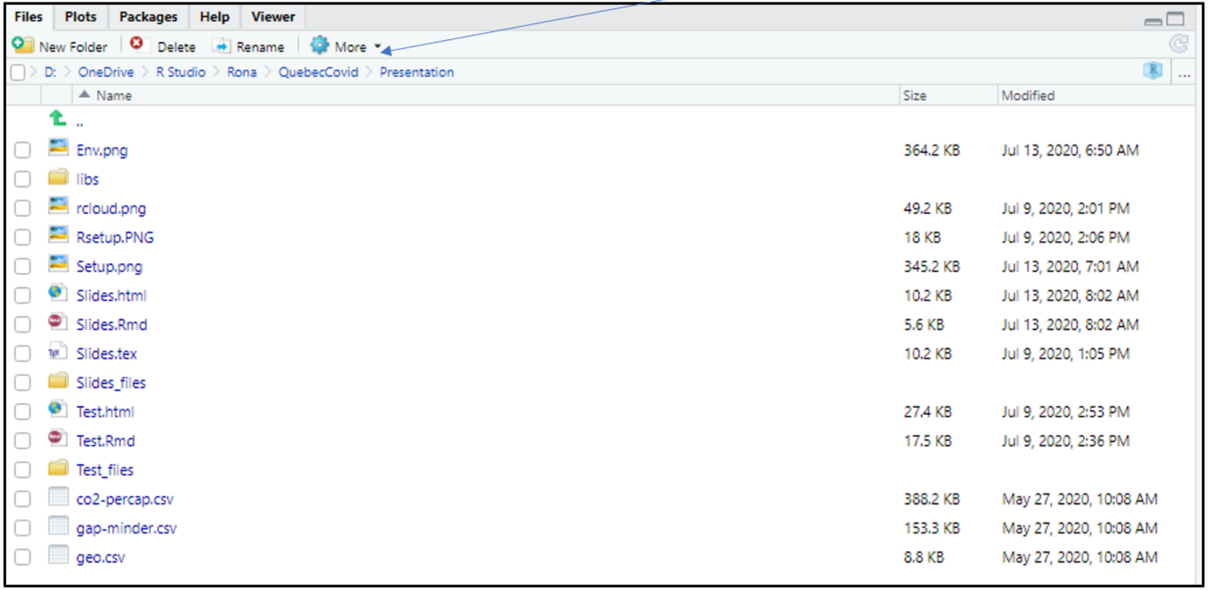
- Data frames is the `R` name for tabular data. Essentially tables.
- Generally we want each row in a data frame to represent a unit of observation and each column to contain a different type of information
- As a beginner, it is advisable to prepare your tabular data in Excel prior to working in `RStudio`
- We will work with data using a package called `Tidyverse` which contains many popular tools used for data analysis in `R`
- To install `Tidyverse` in your computer write `install.packages("tidyverse")`
- To load `Tidyverse` write `library(tidyverse)`
- Whenever you close `RStudio` make sure to run the `library(tidyverse)` function the first time you log back in

Working with Data

Opening a csv file

- Save the file you wish to work with in the directory where you saved your project
- This will ensure you don't have you search everywhere in your computer and makes it easy for `RStudio` to access them

Address where I saved my project



D: > OneDrive > R Studio > Rona > QuebecCovid > Presentation

Name	Size	Modified
..		
Env.png	364.2 KB	Jul 13, 2020, 6:50 AM
libs		
rcloud.png	49.2 KB	Jul 9, 2020, 2:01 PM
Rsetup.PNG	18 KB	Jul 9, 2020, 2:06 PM
Setup.png	345.2 KB	Jul 13, 2020, 7:01 AM
Slides.html	10.2 KB	Jul 13, 2020, 8:02 AM
Slides.Rmd	5.6 KB	Jul 13, 2020, 8:02 AM
Slides.tex	10.2 KB	Jul 9, 2020, 1:05 PM
Slides_files		
Test.html	27.4 KB	Jul 9, 2020, 2:53 PM
Test.Rmd	17.5 KB	Jul 9, 2020, 2:36 PM
Test_files		
co2-percap.csv	388.2 KB	May 27, 2020, 10:08 AM
gap-minder.csv	153.3 KB	May 27, 2020, 10:08 AM
geo.csv	8.8 KB	May 27, 2020, 10:08 AM

All the files that I will be working with

Working with Data

Opening a csv file

```r
geo ← readr::read_csv("geo.csv")
geo
```

```
# A tibble: 196 x 7
   name                region   oecd  g77     lat   long income2017
   <chr>               <chr>    <lgl> <lgl> <dbl>  <dbl> <chr>
 1 Afghanistan         asia     FALSE TRUE   33     66    low
 2 Albania             europe   FALSE FALSE  41     20    upper_mid
 3 Algeria             africa   FALSE TRUE   28      3    upper_mid
 4 Andorra             europe   FALSE FALSE  42.5    1.52 high
 5 Angola              africa   FALSE TRUE  -12.5   18.5  lower_mid
 6 Antigua and Barbuda americas FALSE TRUE   17.0  -61.8  high
 7 Argentina           americas FALSE TRUE  -34    -64    upper_mid
 8 Armenia             europe   FALSE FALSE  40.2   45    lower_mid
 9 Australia           asia     TRUE  FALSE -25    135    high
10 Austria             europe   TRUE  FALSE  47.3   13.3  high
# … with 186 more rows
```

Working with Data

Exploring the data

- We can extract details from the dataset

```
nrow(geo)  #Number of observations
```

```
[1] 196
```

```
ncol(geo)  #Number of columns
```

```
[1] 7
```

```
colnames(geo)  #Names of the columns
```

```
[1] "name"       "region"     "oecd"       "g77"        "lat"       
[6] "long"       "income2017"
```

Working with Data

Exploring the data

```
summary(geo)  #Descriptive statistics (if available) for each column
```

```
     name              region              oecd              g77         
 Length:196         Length:196         Mode :logical   Mode :logical  
 Class :character   Class :character   FALSE:165       FALSE:65       
 Mode  :character   Mode  :character   TRUE :31        TRUE :131      
                                                                      
                                                                      
                                                                      
      lat              long            income2017       
 Min.   :-42.00   Min.   :-175.000   Length:196        
 1st Qu.:  4.00   1st Qu.:  -5.625   Class :character  
 Median : 17.42   Median :  21.875   Mode  :character  
 Mean   : 19.03   Mean   :  23.004                     
 3rd Qu.: 39.82   3rd Qu.:  51.892                     
 Max.   : 65.00   Max.   : 179.145                     
```

Working with Data

Indexing data frames

- The syntax to subset (selecting parts of a table) data frames is `[row,column]`

```
geo[4, 2]  #Here were are taking row 4, column 2.
```

```
# A tibble: 1 x 1
  region
  <chr>
1 europe
```

```
geo[4, "region"]  #We can also use the name of the column instead. Be aware that this
```

```
# A tibble: 1 x 1
  region
  <chr>
1 europe
```

Working with Data

Indexing data frames

```r
geo[4, ]  #We can choose a row and all the columns that are associated with it by leav
```

```
# A tibble: 1 x 7
  name    region oecd  g77     lat  long income2017
  <chr>   <chr>  <lgl> <lgl> <dbl> <dbl> <chr>
1 Andorra europe FALSE FALSE  42.5  1.52 high
```

```r
head(geo[, "region"])  #Similarly we can choose an entire column alone with all its ro
```

```
# A tibble: 6 x 1
  region
  <chr>
1 asia
2 europe
3 africa
4 europe
5 africa
6 americas
```

Working with Data

Indexing data frames

```
geo[c(1, 50, 180), ]  #We can index different rows in one line
```

```
# A tibble: 3 x 7
  name               region   oecd  g77     lat  long income2017
  <chr>              <chr>    <lgl> <lgl> <dbl> <dbl> <chr>
1 Afghanistan        asia     FALSE TRUE   33    66   low
2 Dominican Republic americas FALSE TRUE   19   -70.7 upper_mid
3 Turkmenistan       asia     FALSE TRUE   39.8  59.7 upper_mid
```

```
head(geo[2:8, ])  #Selecting a sequence of rows is called slicing
```

```
# A tibble: 6 x 7
  name                region   oecd  g77     lat   long income2017
  <chr>               <chr>    <lgl> <lgl> <dbl>  <dbl> <chr>
1 Albania             europe   FALSE FALSE  41     20   upper_mid
2 Algeria             africa   FALSE TRUE   28      3   upper_mid
3 Andorra             europe   FALSE FALSE  42.5    1.52 high
4 Angola              africa   FALSE TRUE  -12.5   18.5  lower_mid
5 Antigua and Barbuda americas FALSE TRUE   17.0 -61.8  high
6 Argentina           americas FALSE TRUE  -34    -64   upper_mid
```

Working with Data

Logical indexing

- Instead of specifying rows and columns we can use logic to determine `TRUE` values that fit our request
- The `tidyverse` package provides the function `filter()` which allows this to happen in a much easier way than using pure `R`
- Some operators available include
    - `x==y` Equal to
    - `x!=y` Not equal to
    - `x<y` Less than
    - `x>y` Greater than
    - `x<=y` Less than or equal to
    - `x>=y` Greater than or equal to

Working with Data

Logical indexing

- We want countries that are positioned on latitudes that are negative, meaning southern

```
dplyr::filter(geo, lat < 0)  #Select the dataframe first, then the condition, negative
```

```
# A tibble: 40 x 7
   name             region   oecd  g77     lat  long income2017
   <chr>            <chr>    <lgl> <lgl> <dbl> <dbl> <chr>
 1 Angola           africa   FALSE TRUE  -12.5  18.5 lower_mid
 2 Argentina        americas FALSE TRUE  -34   -64   upper_mid
 3 Australia        asia     TRUE  FALSE -25   135   high
 4 Bolivia          americas FALSE TRUE  -17   -65   lower_mid
 5 Botswana         africa   FALSE TRUE  -22    24   upper_mid
 6 Brazil           americas FALSE TRUE  -10   -55   upper_mid
 7 Burundi          africa   FALSE TRUE   -3.5  30   low
 8 Chile            americas TRUE  TRUE  -33.5 -70.6 high
 9 Comoros          africa   FALSE TRUE  -12.2  44.4 low
10 Congo, Dem. Rep. africa   FALSE TRUE   -2.5  23.5 low
#  ... with 30 more rows
```

Simple Graph Creation

- `Tidyverse` contains the package `ggplot2` which can assist in creating custom made plots
- Three things to consider
  - A dataframe
  - How the columns of the dataframe can be translated into positions, colors, size, and shapes of graphical elements. The "aesthetics"
  - The actual graphical element to display (scatterplot, lineplot, barplots)

```
## # A tibble: 6 x 11
##   name   year population gdp_percap life_exp region oecd  g77     lat   long
##   <chr> <dbl>      <dbl>      <dbl>    <dbl> <chr>  <lgl> <lgl> <dbl>  <dbl>
## 1 Afgh~  1800    3280000        603     28.2 asia   FALSE TRUE   33    66
## 2 Alba~  1800     410445        667     35.4 europe FALSE FALSE  41    20
## 3 Alge~  1800    2503218        715     28.8 africa FALSE TRUE   28     3
## 4 Ando~  1800       2654       1197     NA   europe FALSE FALSE  42.5   1.52
## 5 Ango~  1800    1567028        618     27.0 africa FALSE TRUE  -12.5  18.5
## 6 Anti~  1800      37000        757     33.5 ameri~ FALSE TRUE   17.0 -61.8
## # … with 1 more variable: income2017 <chr>
```

Simple Graphs

```
ggplot2::ggplot(gap_geo, ggplot2::aes(x = year, y = life_exp)) + ggplot2::geom_point()
```

Simple Graphs

Let's trying adding more `Aesthetics(R)`

```
ggplot2::ggplot(gap_geo, ggplot2::aes(x = year, y = life_exp, color = region, size = p
    ggplot2::geom_point()
```

Simple Graphs

-Boxplots look cool in publications (not these ones though) -The year aesthetic separates the boxplots by year, otherwise it would be one single boxplot

```
ggplot2::ggplot(gap_geo, ggplot2::aes(x = year, y = life_exp, color = region, group =
    ggplot2::geom_boxplot()
```