An Introduction to R and RStudio for Exploratory Data Analysis

Jessica Minnier, PhD & Meike Niederhausen, PhD
OCTRI Biostatistics, Epidemiology, Research & Design (BERD) Workshop

Part 1: 2020/09/16 & Part 2: 2020/09/17

slides: bit.ly/berd_intro_part1
pdf: bit.ly/berd_intro_part1_pdf

1. Open slides: bit.ly/berd_eda

2. Install R & Rstudio: instructions bit.ly/berd_install

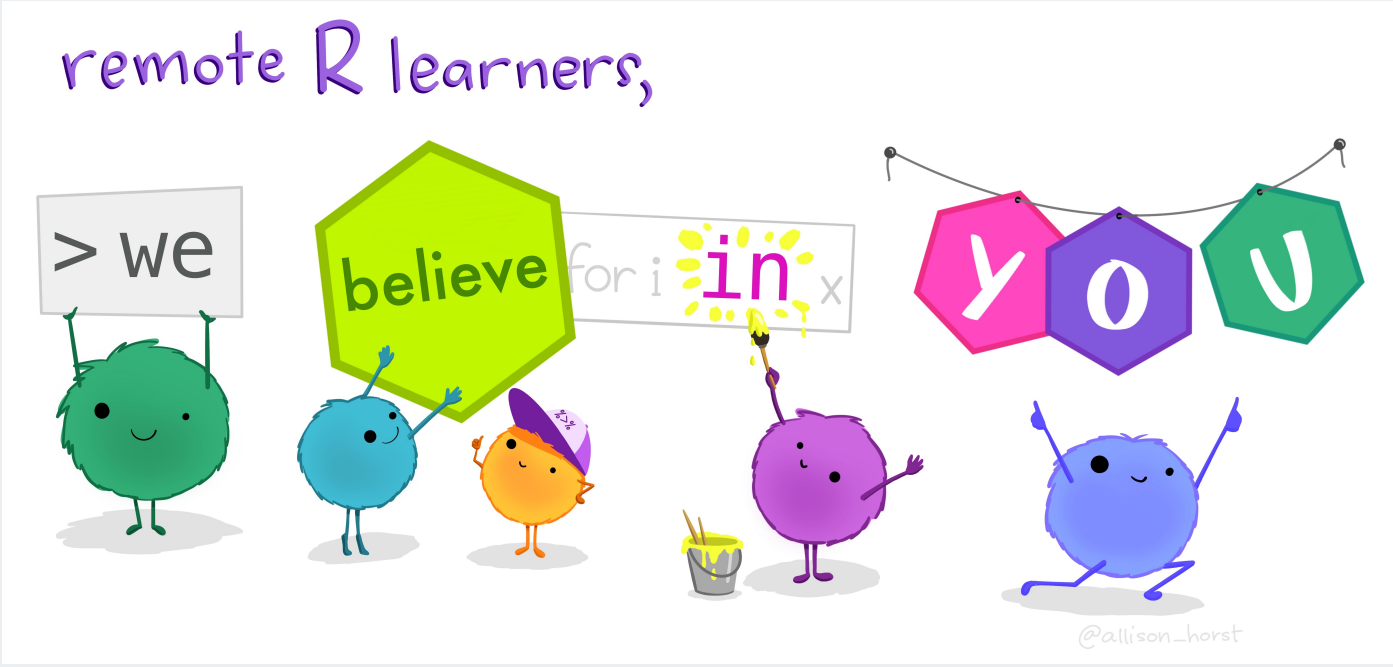
4. Download folder of data (unzip completely)

CHANGE TO DOWNLOAD CSV OF DATA?

- Go to bit.ly/intro_rproj and **unzip** folder
- Open (double click on) `berd_intro_project.Rproj` file.

Learning Objectives

- Basic operations in R/RStudio
- Understand data structures
- Be able to load in data
- Basic operations on data
- Some data wrangling
- Use Rstudio projects
- Be able to make a plot
- Basics of tidyverse and ggplot
- Know how to get help



Allison Horst

Introduction

Rrrrrr?

What is R?

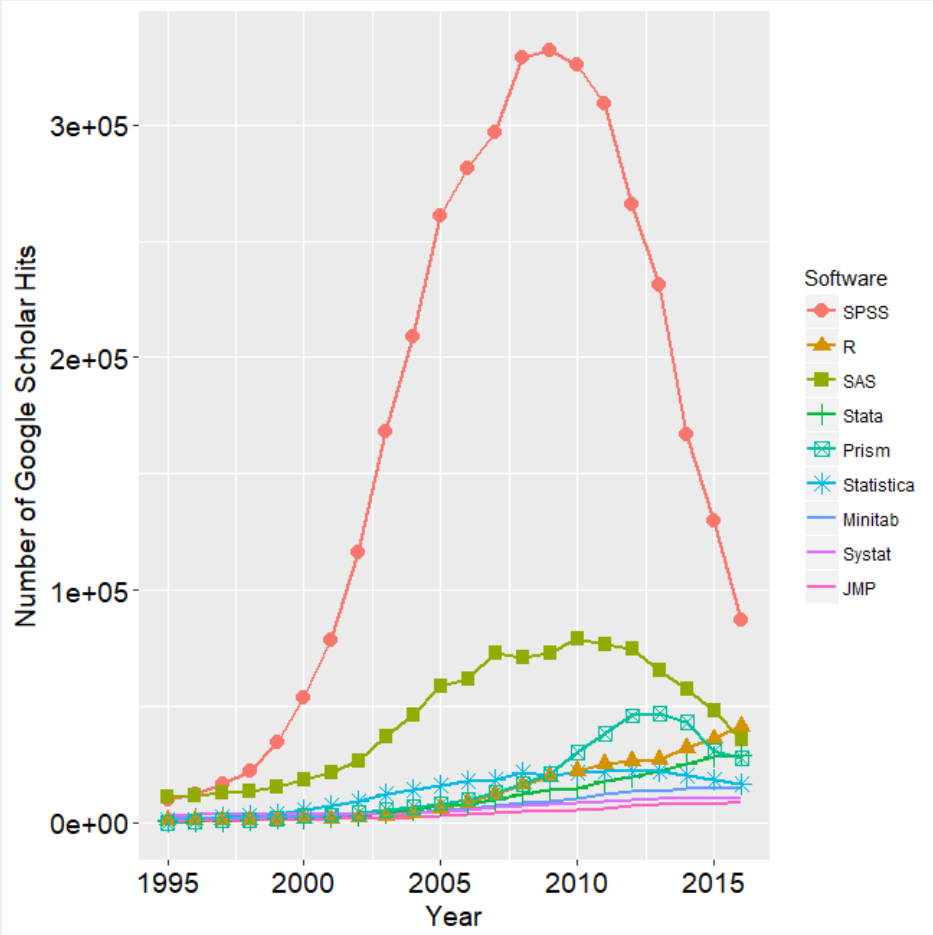
- A programming language
- Focus on statistical modeling and data analysis
  - import data, manipulate data, run statistics, make plots
- Useful for "Data Science"
- Great visualizations
- Also useful for most anything else you'd want to tell a computer to do
- Interfaces with other languages i.e. python, C++, bash

For the history and details: Wikipedia

- an interpreted language (run it through a command line)
- procedural programming with functions
- Why "R"?? Scheme (?) inspired S (invented at Bell Labs in 1976) which inspired R (**free and open source!** in 1992)

Why R?

- Free + Cross-platform (Mac/Windows)
- Flexible, fun, many more modern statistics methods, large community for learning and help
- One of the most popular data science tools for statistics in academia and industry
- SAS and STATA (and SPSS) are still used but becoming less popular (expensive, not as versatile/comprehensive)
- Constantly evolving and improving
- If you want a job doing stats and not be limited to specific research groups or some pharma companies, you absolutely *need to know R*



r4stats Robert A. Muenchen

What is RStudio?

R is a programming language

RStudio is an integrated development environment (IDE) = an interface to use R (with perks!)

- R is like a car's engine
- RStudio is like a car's dashboard

R: Engine	RStudio: Dashboard
	

Modern Dive

Start RStudio

1.1.2 Using R via RStudio

Recall our car analogy from earlier. Much as we don't drive a car by interacting directly with the engine but rather by interacting with elements on the car's dashboard, we won't be using R directly but rather we will use RStudio's interface. After you install R and RStudio on your computer, you'll have two new *programs* (also called *applications*) you can open. We'll always work in RStudio and not in the R application. Figure 1.2 shows what icon you should be clicking on your computer.

R: Do not open this	RStudio: Open this
	

Modern Dive

BuzzR

RStudio anatomy

https://buzzrbeeline.blog/

Emma Rand

Script file

Write code here

To run code put your cursor on the line and click the run button

Edit to correct errors

⇒ record of commands that worked

Save scripts with the **.R** extension

⇒ syntax will be highlighted

⇒ good practice

`<-` is the assignment operator

⇒ puts what is on the right in to the object on the left

⇒ Assign results if you want to use them again

Console

When you click run, code is sent to the console and executed

`>` is the prompt

⇒ do not type it

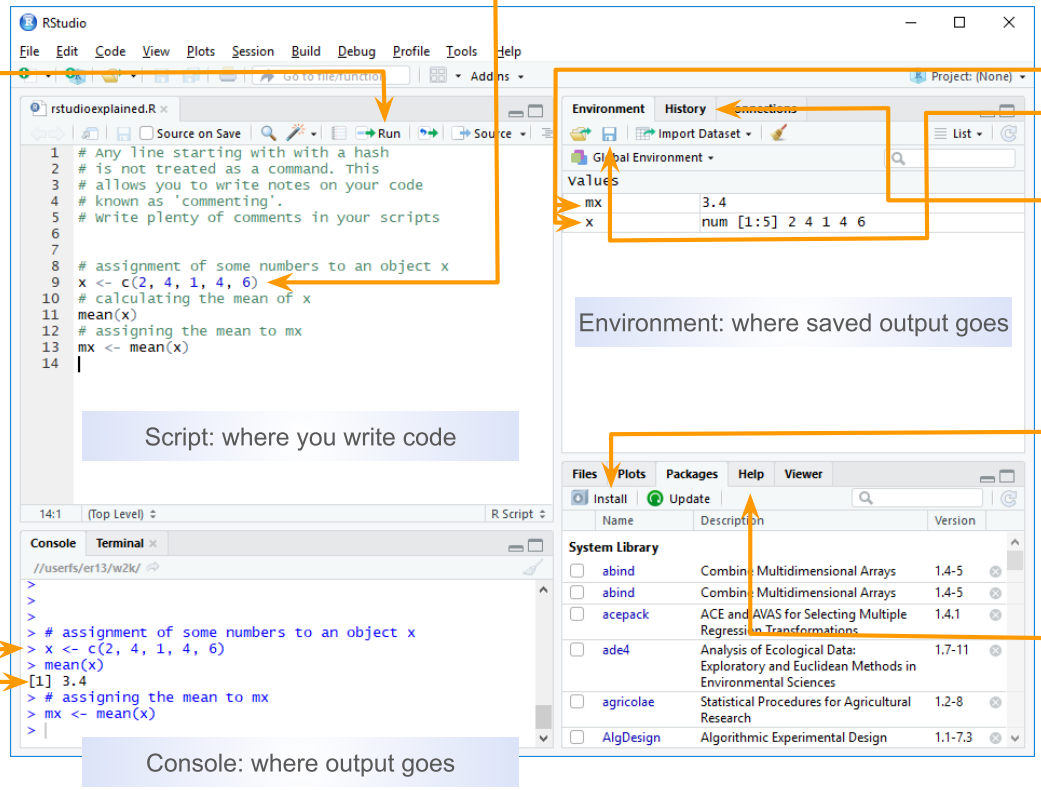
⇒ appears when R is ready for next command

Command output goes here by default

⇒ output is in a different colour

⇒ `[1]` indicates 3.4 is the first element of the output

⇒ many commands will not have output, the prompt just reappears



Environment

Name objects by assignment to use them again

All the objects you created in your session

Saving the environment saves all the objects, but not the code with a **.RData** extension

History

A history of every command you sent to the console, mistakes included.

File can be saved but usually you just need the script

Packages

Many functions come with R

A huge amount of extra functionality is available in packages

Packages can be installed by clicking the Install button

Help

Access to manual pages for all installed packages

Plots

Figure output appears here

Emma Rand

RStudio demo

- Start RStudio and explore

Bonus lessons

- gifs showing how to adjust panels, personalize how Rstudio looks, etc

Coding in the console

When you first open R, the console should be empty.

Typing and executing code in the console

- Type code in the console (blue text)
- Press **return** to execute the code
- Output shown below in black

We can do math

```
> 10^2
```

```
[1] 100
```

```
> 3 ^ 7
```

```
[1] 2187
```

```
> 6/9
```

```
[1] 0.6666667
```

```
> 9-43
```

```
[1] -34
```

- Rules for order of operations are followed
- Spaces between numbers and characters are ignored

```
> 4^3-2* 7+9 /2
```

```
[1] 54.5
```

The equation above is computed as

$$4^3 - (2 \cdot 7) + \frac{9}{2}$$

Variables

Variables are used to store data, figures, model output, etc.

- Can assign a variable using either = or <-
  - Using <- is preferable
  - type name of variable to print

Assign just one value:

```
> x = 5
> x
```

```
[1] 5
```

```
> x <- 5
> x
```

```
[1] 5
```

Assign a **vector** of values:

- Consecutive integers using :

```
> a <- 3:10
> a
```

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

- **Concatenate** a string of numbers

```
> b <- c(5, 12, 2, 100, 8)
> b
```

```
[1]   5  12   2 100   8
```

We can do math with variables

Math using variables with just one value

```
> x <- 5
> x
```

```
[1] 5
```

```
> x + 3
```

```
[1] 8
```

```
> y <- x^2
> y
```

```
[1] 25
```

Math on vectors of values: **element-wise** computation

```
> a <- 3:6
> a
```

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

```
> a+2; a*3
```

```
[1] 5 6 7 8
```

```
[1]  9 12 15 18
```

```
> a*a
```

```
[1]  9 16 25 36
```

Variables can include text (characters)

```
> hi <- "hello"
> hi
```

```
[1] "hello"
```

```
> greetings <- c("Guten Tag", "Hola", hi)
> greetings
```

```
[1] "Guten Tag" "Hola"      "hello"
```

Using functions

- `mean()` is an example of a function
- functions have "arguments" that are specified within the `()`
- `?mean` in console will show help for `mean()`

Arguments specified by name:

```
> mean(x = 1:4)
```

```
[1] 2.5
```

```
> seq(from = 1, to = 12, by = 3)
```

```
[1]  1  4  7 10
```

```
> seq(by = 3, to = 12, from = 1)
```

```
[1]  1  4  7 10
```

Arguments not specified, but listed in order:

```
> mean(1:4)
```

```
[1] 2.5
```

```
> seq(1,12,3)
```

```
[1]  1  4  7 10
```

Missing values

Missing values are denoted as `NA` and are handled differently depending on the operation. There are special functions for `NA` (i.e. `is.na()`, `na.omit()`).

```
> x <- c(1, 2, NA, 5)
> is.na(x)
```

```
[1] FALSE FALSE  TRUE FALSE
```

```
> mean(x)
```

```
[1] NA
```

```
> mean(x, na.rm=TRUE)
```

```
[1] 2.666667
```

```
> x <- c("a", "a", NA, "b")
> table(x)
```

```
x
a b
2 1
```

```
> table(x, useNA = "always")
```

```
x
   a    b <NA>
   2    1    1
```

Common console errors (1/2)

Incomplete commands

- When the console is waiting for a new command, the prompt line begins with >
  - If the console prompt is +, then a previous command is incomplete
  - You can finish typing the command in the console window

Example:

```
> 3 + (2*6
+ )
```

```
[1] 15
```

Common console errors (2/2)

Object is not found

- This happens when text is entered for a non-existent variable (object)

Example:

```
> hello
```

```
Error in eval(expr, envir, enclos): object 'hello' not found
```

- Can be due to missing quotes

```
> install.packages(dplyr) # need install.packages("dplyr")
```

```
Error in install.packages(dplyr): object 'dplyr' not found
```

Create an R Markdown file (`.Rmd`)

Save the Markdown file (`.Rmd`)

- **Save the file** by
  - selecting `File -> Save`,
  - or clicking on  (towards the left above the scripting window),
  - or keyboard shortcut
    - PC: *Ctrl + s*
    - Mac: *Command + s*

- You will need to specify
  - a **filename** to save the file as
    - ALWAYS use **.Rmd** as the filename extension for R markdown files
  - the **folder** to save the file in

Compare the .Rmd file with its html output

.Rmd file

default_html.Rmd

Knit | Insert | Run

```
---
title: "Untitled"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

html output

~/Google Drive/BERD R Classes/berd_rmarkdown_project/default/default_html.html

default_html.html | Open in Browser | Find | Publish

Untitled

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

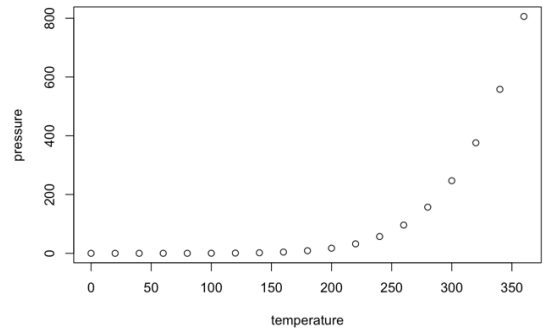
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Compare the .Rmd file with its html output

.Rmd file

default_html.Rmd

```
---
title: "Untitled"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

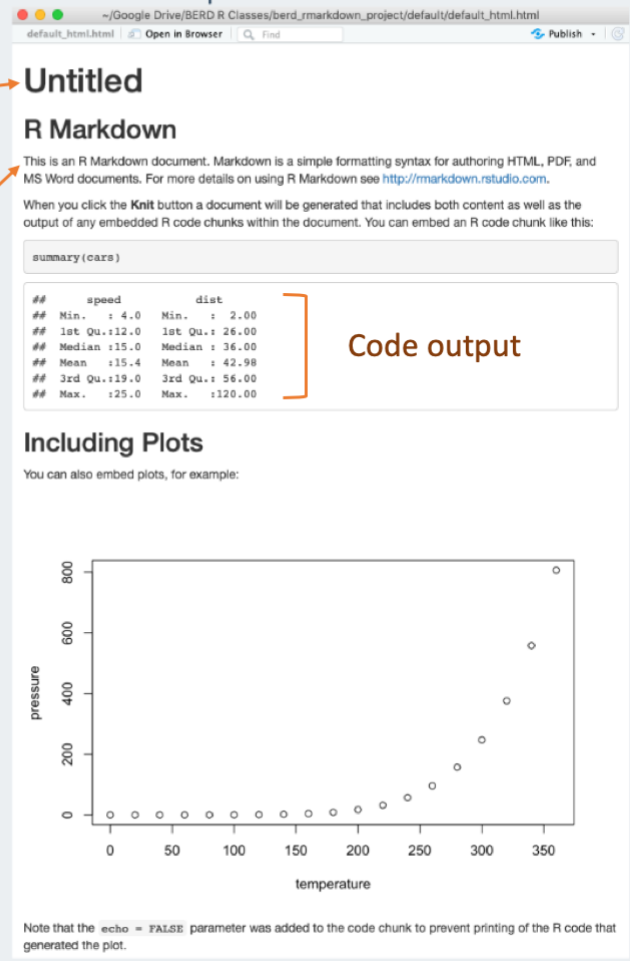
Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

YAML metadata

Text

Code chunk

html output



Code output

How to create the html file? *Knit* the .Rmd file!

To **knit** the .Rmd file, either

1. click on the knit icon  at the top of the editor window
2. or use keyboard shortcuts
    - Mac: *Command+Shift+K*
    - PC: *Ctrl+Shift+K*

- A new window will open with the html output.
- You will now see both .Rmd and .html files in the folder where you saved the .Rmd file.

Remarks:

- The template .Rmd file that RStudio creates will knit to an html file by default
- Watch the Reproducible Reports with R Markdown workshop for more customization and output formats (Word, pdf, slides).
    - Slides at https://jminnier-berd-r-courses.netlify.com/03-rmarkdown/03_rmarkdown_slides.html.

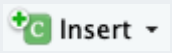
3 types of R Markdown content

1. *Code chunks*: type R code and execute it to see code output
2. Text: write about your analyses
3. YAML metadata: customize the report

We will first focus on using code chunks.

Create a code chunk

Code chunks can be created by either

1. Clicking on  →  at top right of the editor window, or
2. **Keyboard shortcut**
    - Mac: *Command + Option + I*
    - PC: *Ctrl + Alt + I*

- An empty code chunk looks like this:

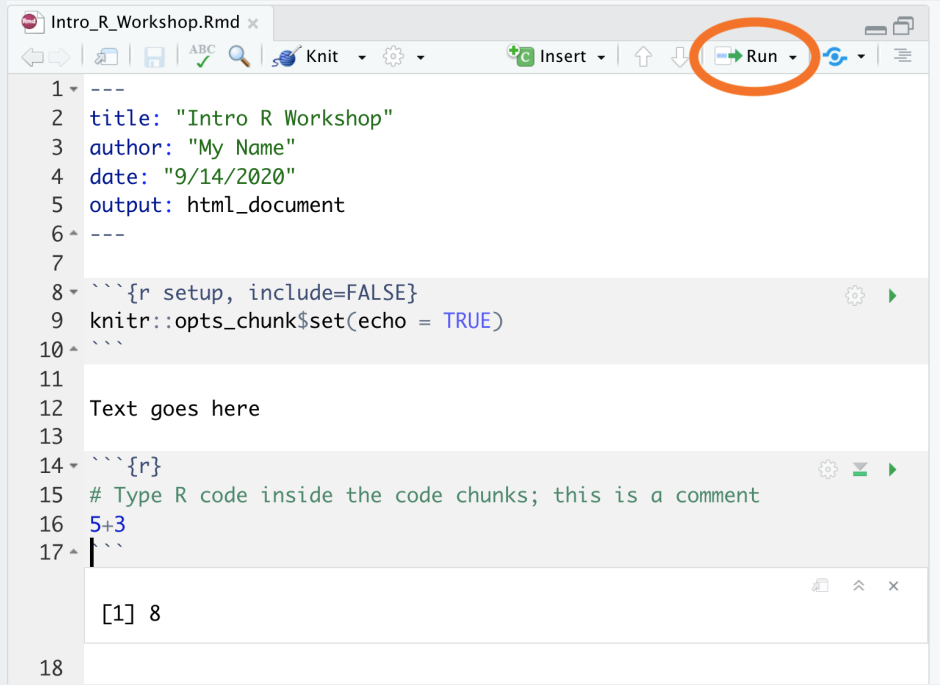
````
```{r}

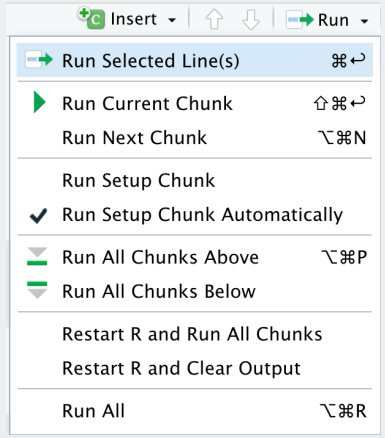

```
````

- Note that a code chunks start with ```` ```{r} ```` and ends with ```` ``` ````.

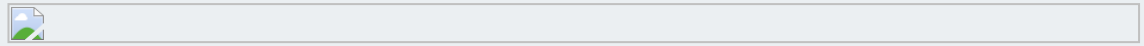
Enter and run code (1/n)

- **Type R code** inside code chunks
- **Select code** you want to run, by
  - placing the cursor in the line of code you want to run,
  - ***or*** highlighting the code you want to run
- **Run selected code** by
  - clicking on the Run button in the top right corner of the scripting window and choosing "Run Selected Line(s)",
  - or typing one of the following key combinations:
    - **Windows**: **ctrl + return**
    - **Mac**: **command + return**





Enter and run code (2/n)



- **Run all code** in a chunk by
  - by clicking the play button in the top right corner of the chunk
- The code output appears below the code chunk

Useful keyboard shortcuts

action	mac	windows/linux
Run code in Rmd or script	cmd + enter	ctrl + enter
<-	option + -	alt + -

Try typing in Rmd (with shortcut) and running

```
y <- 5
y
```

Others: (see full list)

action	mac	windows/linux
interrupt currently executing command	esc	esc
in console, go to previously run code	up/down	up/down
keyboard shortcut help	option + shift + k	alt + shift + k

Practice 1

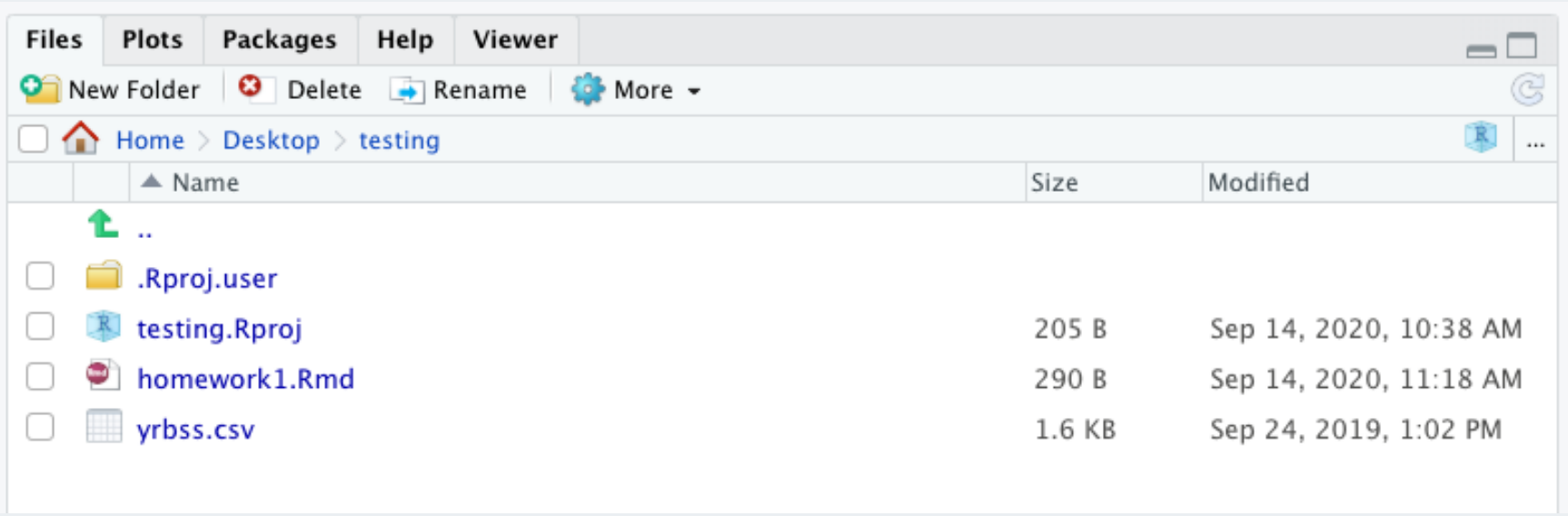
Intro to Data

How is data stored, how do we use it?

- Often, data is in an excel sheet, or a plain text file (.csv, .txt)
- .csv files open in Excel automatically, but actually are plain text
- Usually, columns are variables/measures and rows are observations (i.e. a person's measurements)

Open the data file `yrbss.csv` and look at it

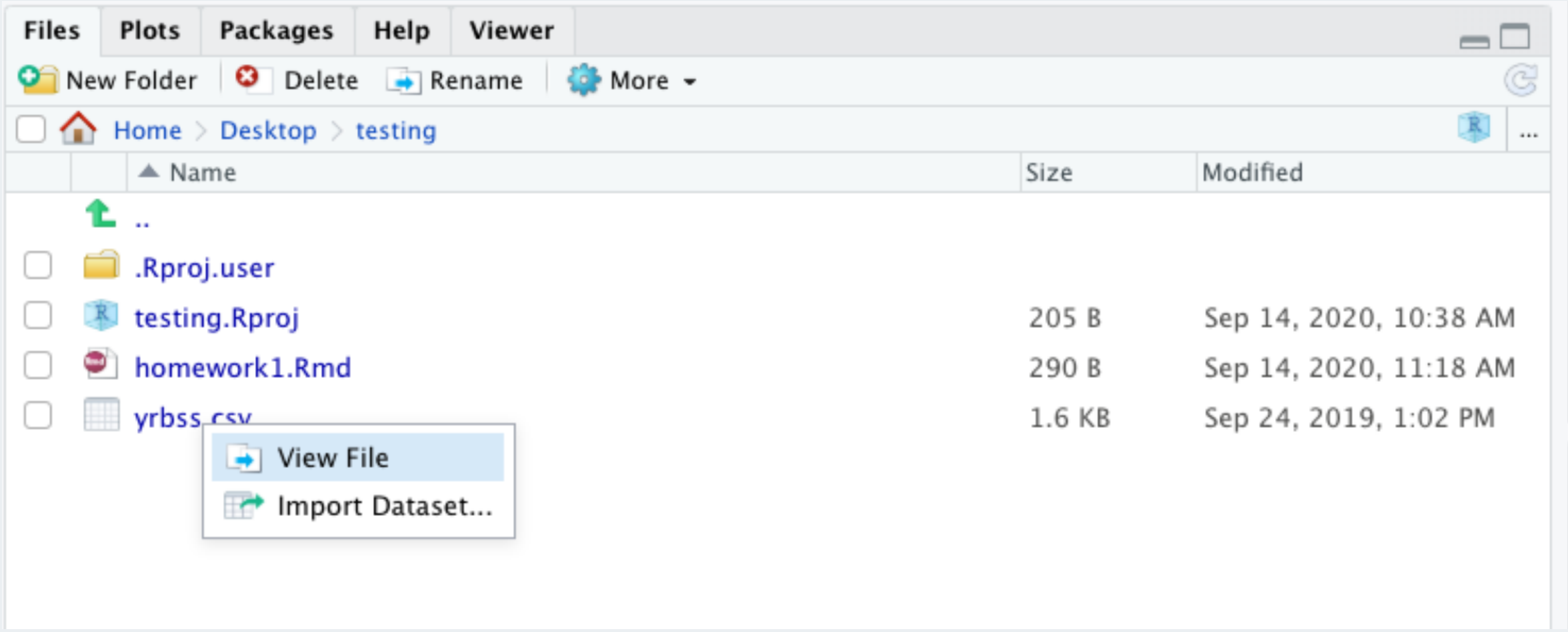
Move the data file `yrbss.csv` into the Rstudio project folder



Data in R/Rstudio

Open `yrbss.csv` in Rstudio and look at it

- Click on `yrbss.csv` in the Files pane, click *View File*



We will show you how to store and use this data in R as a data frame

First, load the packages we need

We will be using functions within the `tidyverse` package in R.

- In the console, run the installation code
  - if you get a message about restarting R, click Yes
  - if you get an error message IN RED (warnings in orange or black text are ok), ask a TA

```
install.packages("tidyverse")
install.packages("janitor")
```

- Then add this code to the setup chunk in the Rmd and run that chunk:

```
library(tidyverse)
library(janitor)
```

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(janitor)
```
````

- Now we can use functions in these packages, such as `read_csv()` and `%>%` and `mutate()` and `tabyl()`

Object types

Data frames (aka "tibbles" in tidyverse)

Vectors vs. **data frames**: a data frame is a collection (or array or table) of vectors

```
df <- tibble(
  IDs=1:3,
  gender=c("male", "female", "Male"),
  age=c(28, 35.5, 31),
  trt = c("control", "1", "1"),
  Veteran = c(FALSE, TRUE, TRUE)
  )
df
```

```
## # A tibble: 3 x 5
##     IDs gender   age trt     Veteran
##   <int> <chr>  <dbl> <chr>   <lgl>
## 1     1 male    28   control FALSE
## 2     2 female  35.5 1       TRUE
## 3     3 Male    31   1       TRUE
```

- Allows different columns to be of different data types (i.e. numeric vs. text)
- Both numeric and text can be stored within a column (stored together as *text*).
- Vectors and data frames are examples of ***objects*** in R.
  - There are other types of R objects to store data, such as matrices, lists.
  - These will be discussed in future R workshops.

Variable (column) types

type	description
integer	integer-valued numbers
numeric	numbers that are decimals
factor	categorical variables stored with levels (groups)
character	text, "strings"
logical	boolean (TRUE, FALSE)

- View the **structure** of our data frame to see what the variable types are:

```
str(df)
```

```
## tibble [3 × 5] (S3: tbl_df/tbl/data.frame)
##  $ IDs    : int [1:3] 1 2 3
##  $ gender : chr [1:3] "male" "female" "Male"
##  $ age    : num [1:3] 28 35.5 31
##  $ trt    : chr [1:3] "control" "1" "1"
##  $ Veteran: logi [1:3] FALSE TRUE TRUE
```

Data frame cells, rows, or columns

Show whole data frame

```
df
```

```
## # A tibble: 3 x 5
##     IDs gender   age trt     Vetera
##   <int> <chr>  <dbl> <chr>   <lgl>
## 1     1 male    28   control FALSE
## 2     2 female  35.5 1       TRUE
## 3     3 Male    31   1       TRUE
```

Specific cell: `DatSetName[row#, column#]`

```
# Second row, Third column
df[2, 3]
```

```
## # A tibble: 1 x 1
##     age
```

Entire col: `DatSetName[, column#]`

```
# Third column
df[, 3]
```

```
## # A tibble: 3 x 1
##     age
##   <dbl>
## 1  28
## 2  35.5
## 3  31
```

Entire row: `DatSetName[row#, ]`

```
# Second row
df[2,]
```

```
## # A tibble: 1 x 5
##     IDs gender   age trt   Veteran
##   <int> <chr>  <dbl> <chr> <lgl>
```

Getting the data into Rstudio

R Projects (.Rproj file) & Good Practices

Use projects to keep everything together (read this)

- A project keeps track of your coding environment and file structure.
- Create an RStudio project for each data analysis project, for each homework assignment, etc.
- A project is associated with a directory folder
  - Keep data files there
  - Keep code scripts there; edit them, run them in bits or as a whole
  - Save your outputs (plots and cleaned data) there
- Only use relative paths, never absolute paths
  - relative (good): `read.csv("data/mydata.csv")`
  - absolute (bad): `read.csv("/home/yourname/Documents/stuff/mydata.csv")`

Advantages of using projects

- standardizes file paths
- keep everything together
- a whole folder can be easily shared and run on another computer
- when you open the project everything is as you left it

Create a new R project

Let's go through it together. (Read this for more)

File -> New Project -> New Directory

Bonus lessons

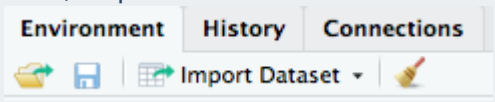
- Video on projects in R, most useful info in minutes 2:00-13:00

Load a data set

- Read in csv file from file path with code (filepath relative to Rproj directory)

```
mydata <- read_csv("yrbss.csv")
```

- Or, open saved file using Import Dataset button in Environment window:



+ From Text(readr).
    - If you use this option, **then copy and paste the importing code to your Rmd** so you have a record of from where and how you loaded the data set.

```
# Run in console:
View(mydata)
# Can also view the data by clicking on its name in the Environment tab
```

Load a data set: bonus lessons

- Importing Data, Rstudio support topic

About the data

Data from the CDC's Youth Risk Behavior Surveillance System (YRBSS)

- small subset (20 rows) of the full complex survey data
- national school-based survey conducted by CDC and state, territorial, tribal, and local surveys conducted by state, territorial, and local education and health agencies and tribal governments
- monitors health-related behaviors (including alcohol & drug use, unhealthy & dangerous behaviors, sexuality, physical activity); see Questionnaires
- original data in the R package `yrbss` which includes YRBSS from 1991-2013

Data set summary

```
summary(mydata)
```

```
##        id              age                sex                grade
##  Min.   : 335340   Length:20          Length:20          Length:20
##  1st Qu.: 925193   Class :character   Class :character   Class :character
##  Median :1207132   Mode  :character   Mode  :character   Mode  :character
##  Mean   :1093150
##  3rd Qu.:1313188
##  Max.   :1316123
##     race4                bmi           weight_kg     text_while_driving_30d
##  Length:20          Min.   :17.48   Min.   :43.09   Length:20
##  Class :character   1st Qu.:20.36   1st Qu.:57.27   Class :character
##  Mode  :character   Median :22.23   Median :64.86   Mode  :character
##                     Mean   :23.01   Mean   :64.09
##                     3rd Qu.:26.58   3rd Qu.:70.31
##                     Max.   :29.35   Max.   :84.82
##  smoked_ever        bullied_past_12mo
##  Length:20          Mode :logical
##  Class :character   FALSE:11
##  Mode  :character   TRUE :7
##                     NA's :2
##
##
```

Data set info

```
dim(mydata)
```

```
## [1] 20 10
```

```
nrow(mydata)
```

```
## [1] 20
```

```
ncol(mydata)
```

```
## [1] 10
```

```
names(mydata)
```

```
##  [1] "id"                                "age"
##  [4] "grade"                             "race4"
##  [7] "weight_kg"                         "text_while_dr
## [10] "bullied_past_12mo"
```

Data structure

- What are the different **variable types** in this data set?

```
str(mydata)   # structure of data
```

```
## tibble [20 × 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id                   : num [1:20] 335340 638618 922382 923122 923963 ...
##  $ age                  : chr [1:20] "17 years old" "16 years old" "14 years o
##  $ sex                  : chr [1:20] "Female" "Female" "Male" "Male" ...
##  $ grade                : chr [1:20] "10th" "9th" "9th" "9th" ...
##  $ race4                : chr [1:20] "White" NA "White" "White" ...
##  $ bmi                  : num [1:20] 27.6 29.3 18.2 21.4 19.6 ...
##  $ weight_kg            : num [1:20] 66.2 84.8 57.6 60.3 63.5 ...
##  $ text_while_driving_30d: chr [1:20] NA NA NA NA ...
##  $ smoked_ever          : chr [1:20] NA "Yes" "Yes" "Yes" ...
##  $ bullied_past_12mo    : logi [1:20] NA NA FALSE FALSE TRUE TRUE ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   age = col_character(),
##   ..   sex = col_character(),
##   ..   grade = col_character(),
##   ..   race4 = col_character(),
```

View the beginning of a data set

```
head(mydata)
```

```
## # A tibble: 6 x 10
##       id age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##    <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
## 1 335340 17 y… Fema… 10th  White  27.6      66.2 <NA>             <NA>
## 2 638618 16 y… Fema… 9th   <NA>   29.3      84.8 <NA>             Yes
## 3 922382 14 y… Male  9th   White  18.2      57.6 <NA>             Yes
## 4 923122 15 y… Male  9th   White  21.4      60.3 <NA>             Yes
## 5 923963 15 y… Male  10th  Blac…  19.6      63.5 <NA>             No
## 6 925603 16 y… Male  10th  All …  22.2      70.3 <NA>             No
## # … with 1 more variable: bullied_past_12mo <lgl>
```

View the end of a data set

```
tail(mydata)
```

```
## # A tibble: 6 x 10
##       id age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##    <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
## 1 1.31e6 16 y… Fema… 11th  Hisp…  26.6      68.0 0 days           No
## 2 1.31e6 16 y… Fema… 11th  White  24.8      63.5 3 to 5 days      No
## 3 1.31e6 16 y… Fema… 10th  All …  25.0      76.7 0 days           No
## 4 1.32e6 17 y… Fema… 11th  <NA>   22.3      54.9 I did not drive… Yes
## 5 1.32e6 17 y… Fema… 12th  Hisp…  19.5      49.9 0 days           <NA>
## 6 1.32e6 18 y… Fema… 12th  Blac…  27.5      74.8 All 30 days      Yes
## # … with 1 more variable: bullied_past_12mo <lgl>
```

Specify how many rows to view at beginning or end of a data set

```
head(mydata, 3)
```

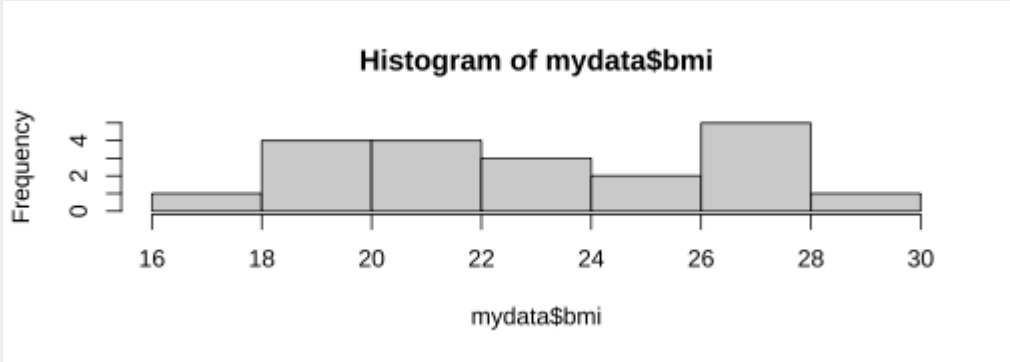
```
## # A tibble: 3 x 10
##       id age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##    <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
## 1 335340 17 y… Fema… 10th  White  27.6      66.2 <NA>             <NA>
## 2 638618 16 y… Fema… 9th   <NA>   29.3      84.8 <NA>             Yes
## 3 922382 14 y… Male  9th   White  18.2      57.6 <NA>             Yes
## # … with 1 more variable: bullied_past_12mo <lgl>
```

```
tail(mydata, 1)
```

```
## # A tibble: 1 x 10
##       id age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##    <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
## 1 1.32e6 18 y… Fema… 12th  Blac…  27.5      74.8 All 30 days      Yes
## # … with 1 more variable: bullied_past_12mo <lgl>
```

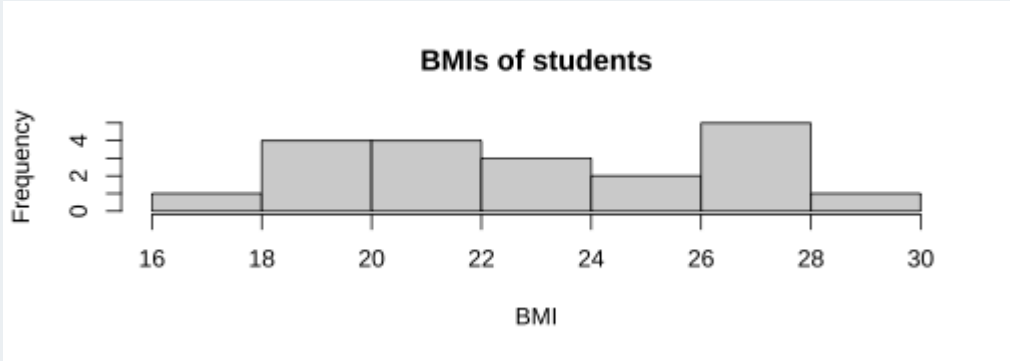
Basic plots of numeric data: Histogram

```
hist(mydata$bmi)
```



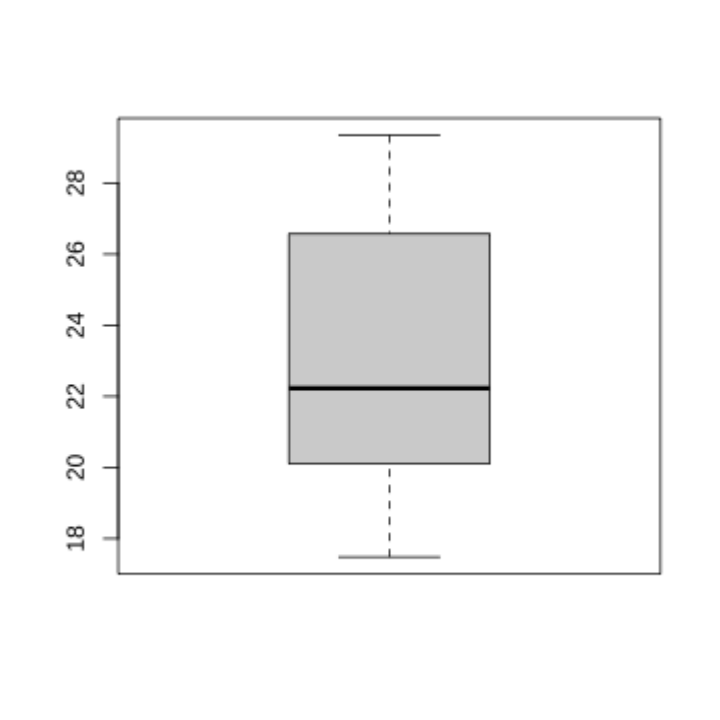
With extra features:

```
hist(mydata$bmi, xlab = "BMI", main="BMIs of students")
```

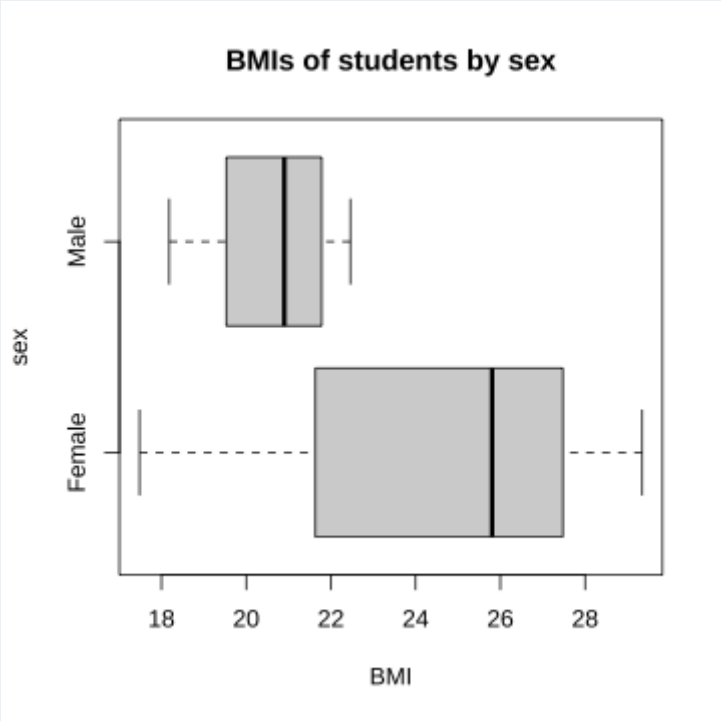


Basic plots of numeric data: Boxplot

```
boxplot(mydata$bmi)
```

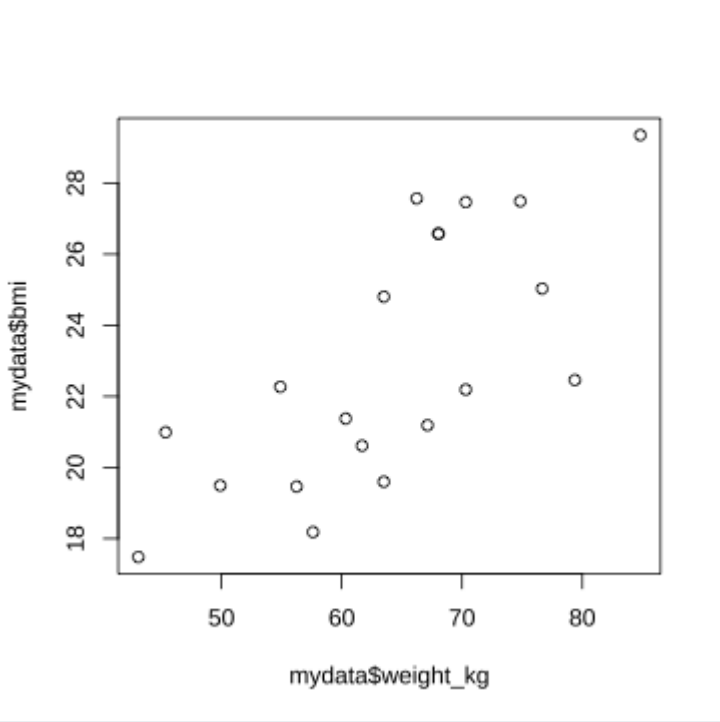



```
boxplot(mydata$bmi ~ mydata$sex,
  horizontal = TRUE,
  xlab = "BMI", ylab = "sex",
  main = "BMIs of students by sex")
```

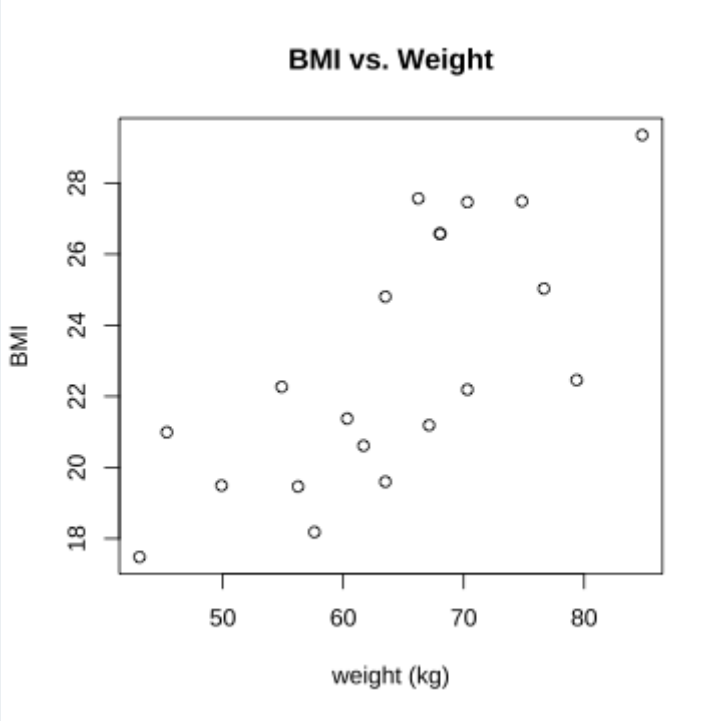


Basic plots of numeric data: Scatterplot

```
plot(mydata$weight_kg, mydata$bmi)
```




```
plot(mydata$weight_kg, mydata$bmi,
   xlab = "weight (kg)", ylab = "BMI",
   main = "BMI vs. Weight")
```



Summary stats of numeric data (1/2)

- Standard R `summary` command

```
summary(mydata$bmi)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17.48   20.36   22.23   23.01   26.58   29.35
```

- Mean and standard deviation

```
mean(mydata$bmi)
```

```
## [1] 23.00838
```

```
sd(mydata$bmi)
```

```
## [1] 3.56471
```

Summary stats of numeric data (2/2)

- Min, max, & median

```
min(mydata$bmi)
```

```
## [1] 17.4814
```

```
max(mydata$bmi)
```

```
## [1] 29.3495
```

```
median(mydata$bmi)
```

```
## [1] 22.22985
```

- Quantiles

```
quantile(mydata$bmi, prob=c(0, .25, .5, .75, 1))
```

```
##       0%      25%      50%      75%     100%
## 17.48140 20.35878 22.22985 26.57810 29.34950
```

Working with data, we will use the pipe %>%

The pipe operator %>% is part of the tidyverse, and strings together commands to be performed sequentially

```
mydata %>% head(n=3)       # prounounce %>% as "then"
```

```
## # A tibble: 3 x 10
##       id age   sex   grade race4   bmi weight_kg text_while_driv… smoked_ever
##    <dbl> <chr> <chr> <chr> <chr> <dbl>     <dbl> <chr>            <chr>
## 1 335340 17 y… Fema… 10th  White  27.6      66.2 <NA>             <NA>
## 2 638618 16 y… Fema… 9th   <NA>   29.3      84.8 <NA>             Yes
## 3 922382 14 y… Male  9th   White  18.2      57.6 <NA>             Yes
## # … with 1 more variable: bullied_past_12mo <lgl>
```

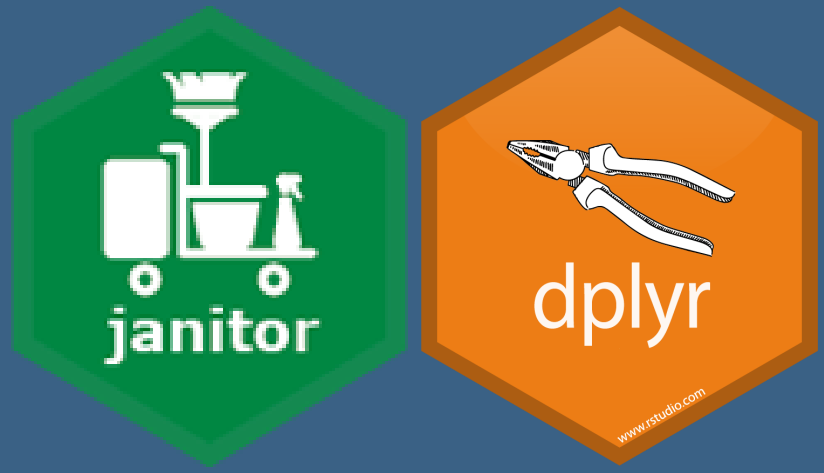
- Always *first list the tibble* that the commands are being applied to
- Can use **multiple pipes** to run multiple commands in sequence
  - What does the following code do?

```
mydata %>% head(n=2) %>% summary()
```

Quick tips on summarizing data

categorical data

numerical data



janitor, dplyr

Numerical data summaries: Using $ vs `summarize()`

We saw how to summarize a vector pulled with `$`, but there are easier ways to summarize multiple columns at once.

```
mean(mydata$bmi)
```

```
## [1] 23.00838
```

```
mean(mydata$weight_kg)
```

```
## [1] 64.094
```

```
mydata %>%
  summarize(mean(bmi), mean(weight_kg))
```

```
## # A tibble: 1 x 2
##   `mean(bmi)` `mean(weight_kg)`
##         <dbl>             <dbl>
## 1        23.0              64.1
```

Numerical data summaries: `summarize()`

- We can summarize data as a whole, or in groups with `group_by()`
- `group_by()` is very powerful, see data wrangling cheatsheet

```
# summary of all data as a whole
mydata %>%
  summarize(bmi_mean =mean(bmi),
            bmi_sd = sd(bmi),
            bmi_cv = sd(bmi)/mean(bmi))
```

```
## # A tibble: 1 x 3
##   bmi_mean bmi_sd bmi_cv
##      <dbl>  <dbl>  <dbl>
## 1     23.0   3.56  0.155
```

```
# summary by group variable
mydata %>%
  group_by(grade) %>%
  summarize(n_per_group = n(),
            bmi_mean =mean(bmi),
            bmi_sd = sd(bmi),
            bmi_cv = sd(bmi)/mean(bmi))
```

```
## # A tibble: 4 x 5
##   grade n_per_group bmi_mean bmi_sd
##   <chr>       <int>    <dbl>  <dbl>
## 1 10th            8     23.3   3.02
## 2 11th            4     23.6   2.65
## 3 12th            4     21.0   4.44
## 4 9th             4     23.9   5.03
```

Advanced `summarize(across())`

- Can also use `across()` to summarize multiple variables (more examples)

```
mydata %>%
  summarize(across(c(bmi, weight_kg),
```

```
## # A tibble: 1 x 2
##     bmi weight_kg
##   <dbl>     <dbl>
## 1  23.0      64.1
```

```
mydata %>%
  summarize(across(where(is.numeric),
```

```
## # A tibble: 1 x 3
##        id   bmi weight_kg
##     <dbl> <dbl>     <dbl>
## 1 1093150  23.0      64.1
```

```
mydata %>%
  summarize(across(where(is.character), n_distinct))
```

```
## # A tibble: 1 x 6
##     age   sex grade race4 text_while_driving_30d smoked_ever
##   <int> <int> <int> <int>                  <int>       <int>
## 1     5     2     4     5                      6           3
```

Frequency tables: simple `count()`

```
mydata %>% count(grade)
```

```
## # A tibble: 4 x 2
##   grade     n
##   <chr> <int>
## 1 10th      8
## 2 11th      4
## 3 12th      4
## 4 9th       4
```

```
mydata %>% count(grade, sex)
```

```
## # A tibble: 8 x 3
##   grade sex        n
##   <chr> <chr>  <int>
## 1 10th  Female     4
## 2 10th  Male       4
## 3 11th  Female     3
## 4 11th  Male       1
## 5 12th  Female     3
## 6 12th  Male       1
## 7 9th   Female     2
## 8 9th   Male       2
```

Fancier frequency tables: `janitor` package's `tabyl` function

```
# default table
mydata %>% tabyl(grade)
```

```
##  grade n percent
##   10th 8     0.4
##   11th 4     0.2
##   12th 4     0.2
##    9th 4     0.2
```

```
# output can be treated as tibble
mydata %>% tabyl(grade) %>% select(-n)
```

```
##  grade percent
##   10th     0.4
##   11th     0.2
##   12th     0.2
##    9th     0.2
```

adorn_ your table!

```
mydata %>%
  tabyl(grade) %>%
  adorn_totals("row") %>%
  adorn_pct_formatting(digits=2)
```

```
##  grade  n percent
##   10th  8  40.00%
##   11th  4  20.00%
##   12th  4  20.00%
##    9th  4  20.00%
##  Total 20 100.00%
```

2x2 tabyls

```
# default 2x2 table
mydata %>% tabyl(grade, sex)
```

```
##  grade Female Male
##   10th      4    4
##   11th      3    1
##   12th      3    1
##    9th      2    2
```

What adornments does the tabyl to right have?

```
mydata %>% tabyl(grade, sex) %>%
  adorn_percentages(denominator = "col") %>%
  adorn_totals("row") %>%
  adorn_pct_formatting(digits = 1) %>%
  adorn_ns()
```

```
##  grade       Female        Male
##   10th  33.3%  (4)  50.0% (4)
##   11th  25.0%  (3)  12.5% (1)
##   12th  25.0%  (3)  12.5% (1)
##    9th  16.7%  (2)  25.0% (2)
##  Total 100.0% (12) 100.0% (8)
```

- Notice `grade` is not sorted in a pleasing way. We will learn how to deal with this when we discuss `factors` as a data type in R.
- Base R has a `table` function, but it is clunkier and the output is not a data frame.
- See the tabyl vignette for more information, adorn options, & 3-way `tabyls`

3 way tabyls are possible

```
mydata %>% tabyl(grade, race4, sex)
```

```
## $Female
##  grade All other races Black or African American Hispanic/Latino White NA_
##   10th               2                        0               1     1   0
##   11th               0                        0               1     1   1
##   12th               1                        1               1     0   0
##    9th               0                        1               0     0   1
##
## $Male
##  grade All other races Black or African American Hispanic/Latino White
##   10th               2                        1               1     0
##   11th               0                        0               1     0
##   12th               0                        0               1     0
##    9th               0                        0               0     2
```

Practice 2

1. Create a new Rmd or continue in your current Rmd.