Getting Started with R and RStudio

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

2019/09/24 & 2020/02/19

🔗 slides: bit.ly/berd_intro_r
pdf: bit.ly/berd_intro_r_pdf

1. Open slides: bit.ly/berd_intro_r

2. Install R

- Windows:
  - Download from https://cran.rstudio.com/bin/windows/base/
- Mac OS X:
  - Download the latest .pkg file (currently R-3.6.2.pkg) from https://cran.rstudio.com/bin/macosx/

3. Install RStudio Desktop Open Source License

- Select download file corresponding to your operating system from https://www.rstudio.com/products/rstudio/download/#download

4. Download folder of data (unzip completely)

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

Questions

- Who has used R?
- What other statistical software have you used?
- Has anyone used other programming languages (C, java, python, etc)?
- Why do you want to learn R?

Learning Objectives

- Basic operations in R/RStudio
- Understand data structures
- Be able to load in data
- Basic operations on data
- Be able to make a plot
- Know how to get help

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

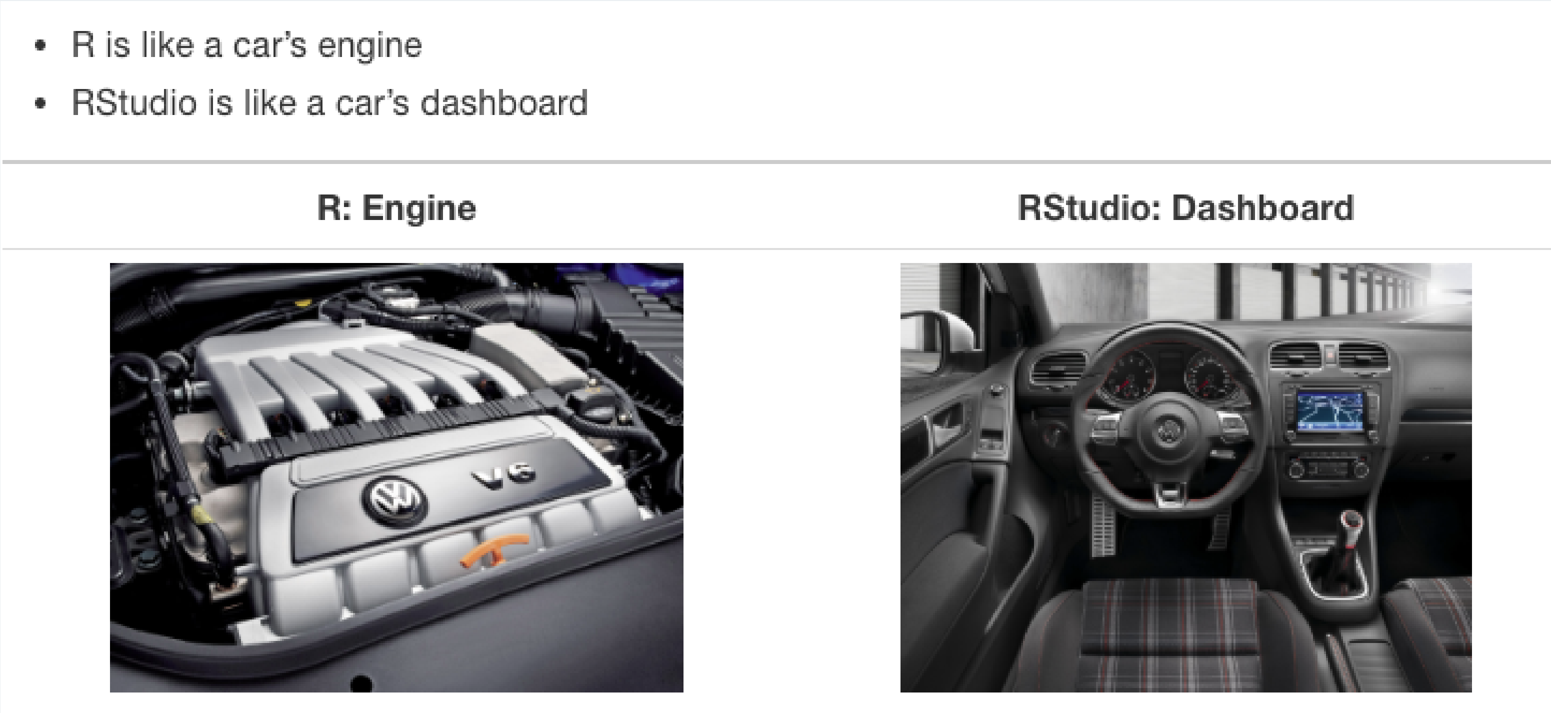
For the history and details: Wikipedia

- 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)

What is RStudio?

R is a programming language

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



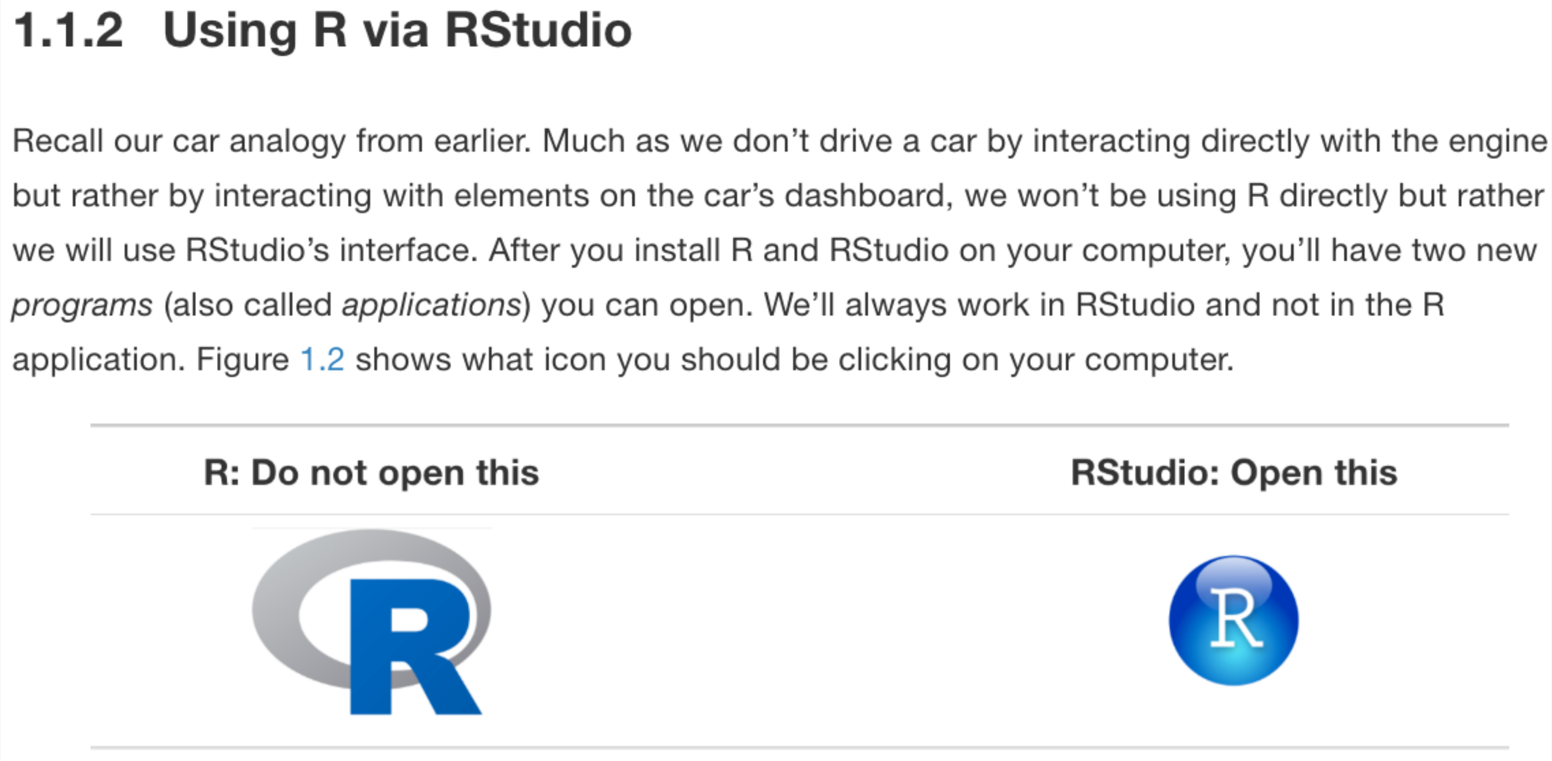
- R is like a car’s engine
- RStudio is like a car’s dashboard

R: Engine	RStudio: Dashboard

Modern Dive

Start RStudio

Double click on the `berd_intro_project.Rproj` file.



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

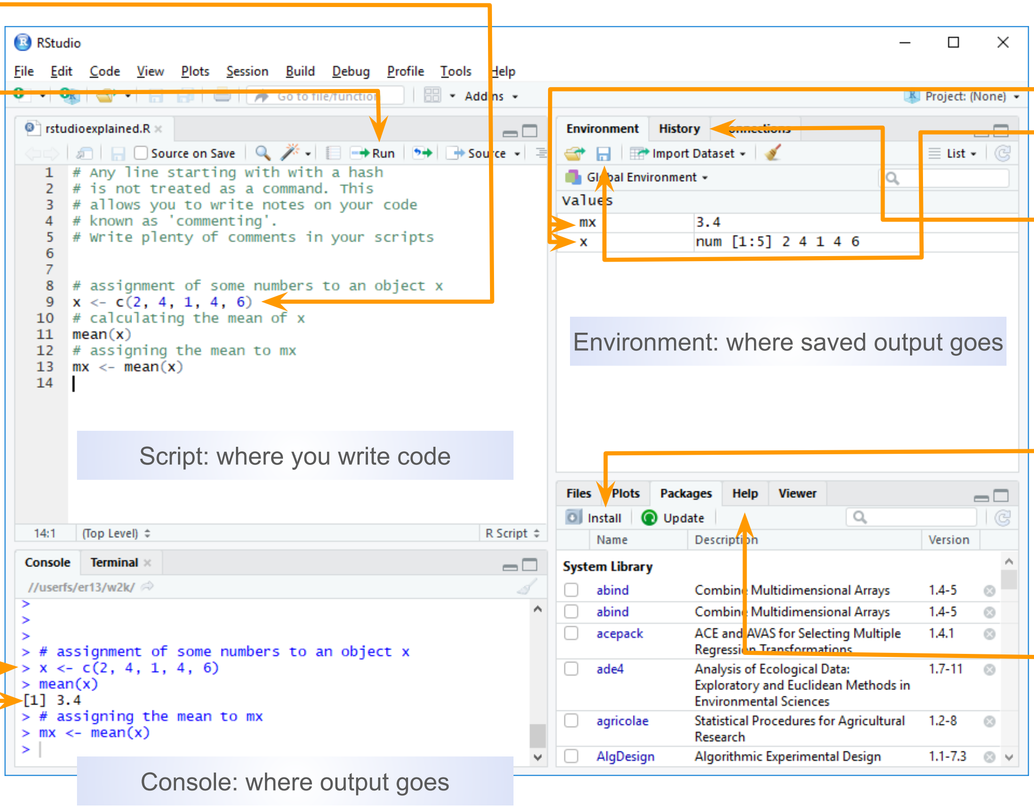
⇒ appears when R is ready for next command

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



```
# Any line starting with with a hash
# is not treated as a command. This
# allows you to write notes on your code
# known as 'commenting'.
# Write plenty of comments in your scripts


# assignment of some numbers to an object x
x <- c(2, 4, 1, 4, 6)
# calculating the mean of x
mean(x)
# assigning the mean to mx
mx <- mean(x)

```

Script: where you write code

```
>
>
>
> # assignment of some numbers to an object x
> x <- c(2, 4, 1, 4, 6)
> mean(x)
[1] 3.4
> # assigning the mean to mx
> mx <- mean(x)
>
```

Console: where output goes

```
Values
mx          3.4
x           num [1:5] 2 4 1 4 6
```

Environment: where saved output goes

	Name	Description	Version	
	System Library			
□	abind	Combine Multidimensional Arrays	1.4-5	⊗
□	abind	Combine Multidimensional Arrays	1.4-5	⊗
□	acepack	ACE and AVAS for Selecting Multiple Regression Transformations	1.4.1	⊗
□	ade4	Analysis of Ecological Data: Exploratory and Euclidean Methods in Environmental Sciences	1.7-11	⊗
□	agricolae	Statistical Procedures for Agricultural Research	1.2-8	⊗
□	AlgDesign	Algorithmic Experimental Design	1.1-7.3	⊗

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

Rstudio demo

R Projects (why .Rproj file?) & Good Practices

Use projects to keep everything together (read this)

- 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 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

Let's code!

Coding in the console

Typing and execting code in the console

- Type code in the console
- Press **return** to execute the code
- Output shown below

Coding in the console is not advisable for most situations!

- We only recommend this for short pieces of code that you don't need to save
- We will be using scripts (`.R` files) to run and save code (in a few slides)

```
> 7
```

```
[1] 7
```

```
> 3 + 5
```

```
[1] 8
```

```
> "hello"
```

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

```
> # this is a comment, nothing happens
> # 5 - 8
>
> # separate multiple commands with ;
> 3 + 5; 4 + 8
```

```
[1] 8
```

```
[1] 12
```

We can do math

```
> 10^2
```

```
[1] 100
```

```
> 3 ^ 7
```

```
[1] 2187
```

```
> 6/9
```

```
[1] 0.6666667
```

```
> 9-43
```

```
[1] -34
```

R follows the rules for order of operations and ignores spaces between numbers (or objects)

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

```
[1] 54.5
```

The equation above is computed as

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

Logarithms and exponentials

Logarithms: `log()` is base e

```
> log(10)
```

```
[1] 2.302585
```

```
> log10(10)
```

```
[1] 1
```

Exponentials

```
> exp(1)
```

```
[1] 2.718282
```

```
> exp(0)
```

```
[1] 1
```

Check that `log()` is base e

```
> log(exp(1))
```

```
[1] 1
```

Using functions

- `log()` is an example of a function
- functions have "arguments"
- `?log` in console will show help for `log()`

Arguments read in order:

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

```
[1] 2.5
```

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

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

Arguments read by name:

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

```
[1] 2.5
```

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

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

Variables

Data, information, everything is stored as a variable

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

Assigning just one value:

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

```
[1] 5
```

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

```
[1] 5
```

Assigning a **vector** of values

- Consecutive integers

```
> 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
```

Variable can include text (characters)

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

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

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

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

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 
```

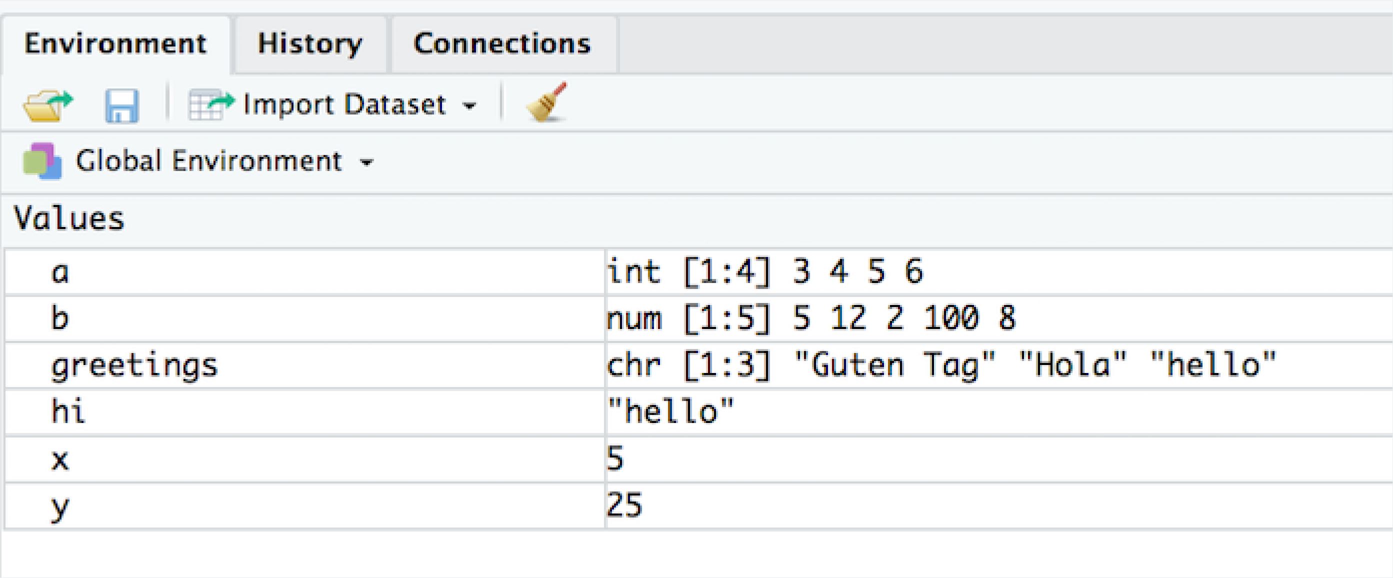
Viewing list of defined variables

- `ls()` is the R command to see what objects have been defined.
- This list includes all defined objects (including dataframes, functions, etc.)

```
> ls()
```

```
[1] "a"         "b"         "greetings" "hi"        "x"         "y"
```

- You can also look at the list in the Environment window:



Values

a	int [1:4] 3 4 5 6
b	num [1:5] 5 12 2 100 8
greetings	chr [1:3] "Guten Tag" "Hola" "hello"
hi	"hello"
x	5
y	25

Removing defined variables

- The R command to delete an object is `rm()`.

```
> ls()
```

```
[1] "a"         "b"         "greetings" "hi"        "x"         "y"
```

```
> rm("greetings", hi)  # Can run with or without quotes
> ls()
```

```
[1] "a" "b" "x" "y"
```

- Remove EVERYTHING - *Be careful!!*

```
> rm(list=ls())
> ls()
```

```
character(0)
```

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
```

R scripts (save your work!)

Coding in a script (1/3)

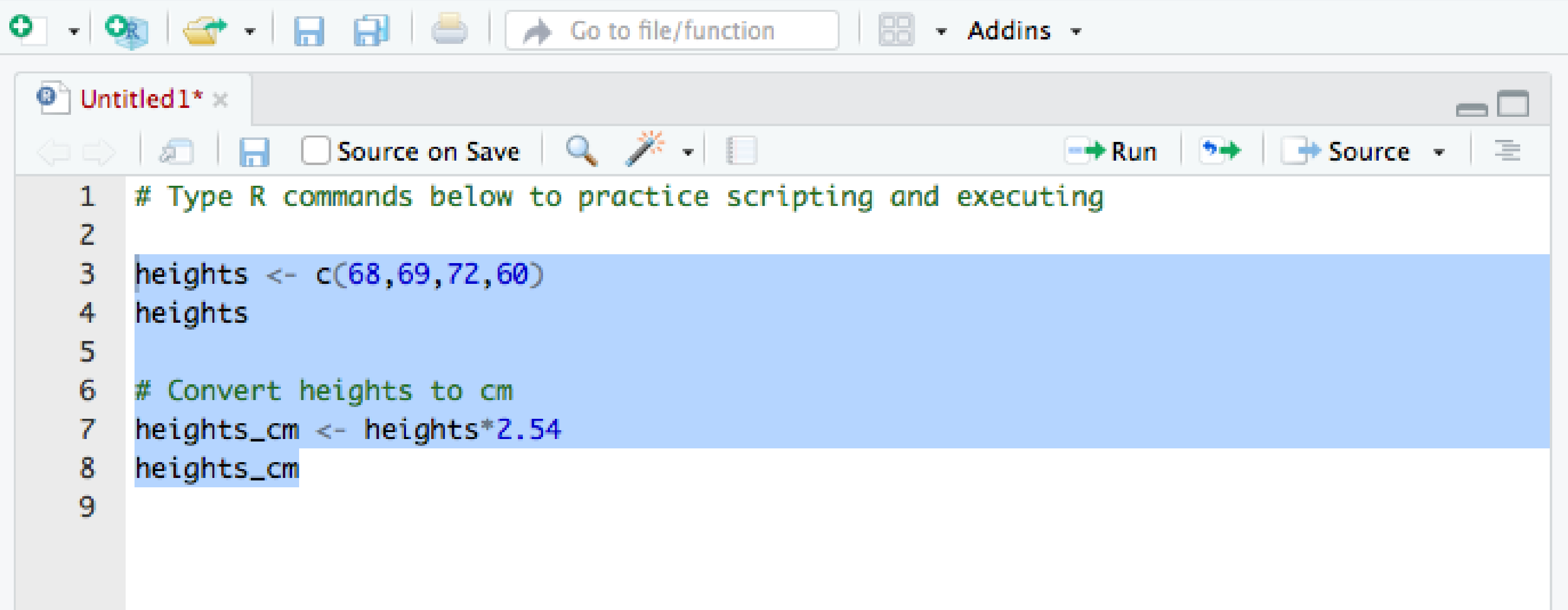
- **Create a new script** by
  - selecting `File -> New File -> R Script`,
  - *or* clicking on ⊕ ▾ (the left most button at the top of the scripting window), and then selecting the first option `R Script`

- **Type code** in the script
  - Type each R command on its own line
  - Use `#` to convert text to comments so that text doesn't accidentally get executed as an R command

Coding in a script (2/3)

- **Select code** you want to execute, by
  - placing the cursor in the line of code you want to execute,
  - *or* highlighting the code you want to execute

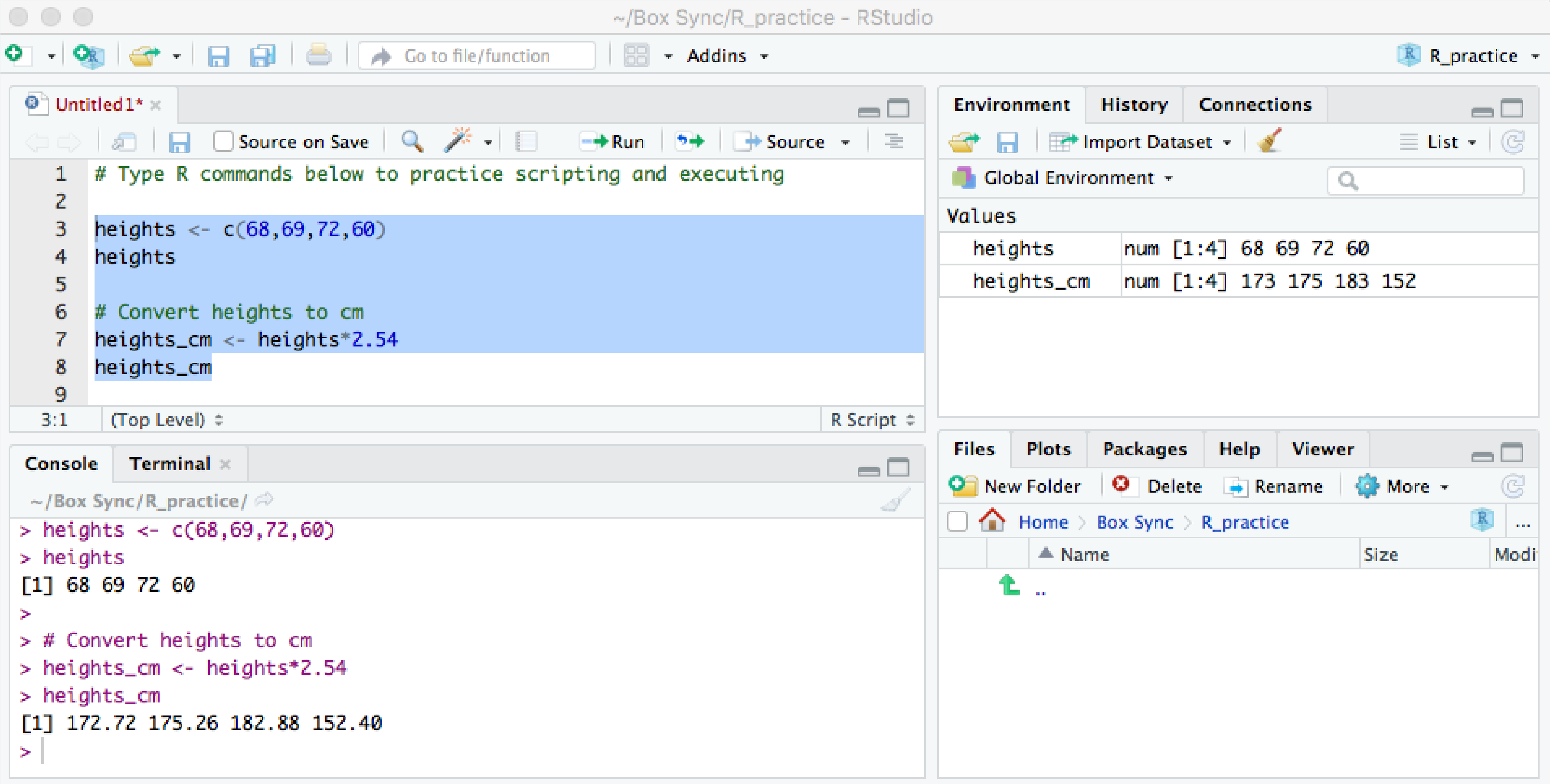
- **Execute code** in the script, by
  - clicking on the Run button in the top right corner of the scripting window,
  - or typing one of the following key combinations to execute the code
    - **Windows**: **ctrl + return**
    - **Mac**: **command + return**



```
# Type R commands below to practice scripting and executing

heights <- c(68,69,72,60)
heights

# Convert heights to cm
heights_cm <- heights*2.54
heights_cm

```

Coding in a script (3/3)

- The screenshot below shows code in the scripting window (top left window)
- The executed highlighted code and its output appear in the console window (bottom left window)



```
# Type R commands below to practice scripting and executing

heights <- c(68,69,72,60)
heights

# Convert heights to cm
heights_cm <- heights*2.54
heights_cm

```

```
> heights <- c(68,69,72,60)
> heights
[1] 68 69 72 60
>
> # Convert heights to cm
> heights_cm <- heights*2.54
> heights_cm
[1] 172.72 175.26 182.88 152.40
>
```

```
Values
  heights       num [1:4] 68 69 72 60
  heights_cm    num [1:4] 173 175 183 152
```

Useful keyboard shortcuts

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

Try typing (with shortcut) in a script and running

```
y <- 5
y
```

Now, in the *console*, press the up arrow.

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

Saving a script

- **Save a script** by
  - selecting `File -> Save`,
  - or clicking on 💾 (towards the left above the scripting window)
- You will need to specify
  - a **filename** to save the script as
    - ALWAYS use **.R** as the filename extension for R scripts
  - the **folder** to save the script in

Practice time!

Practice 1

1. Open a new R script and type code/answers for next tasks in it. Save as `Practice1.R`
2. Create a vector of all integers from 4 to 10, and save it as `a1`.
3. Create a vector of *even* integers from 4 to 10, and save it as `a2`.
4. What is the sum of `a1` and `a2`?
5. What does the command `sum(a1)` do?
6. What does the command `length(a1)` do?
7. Use the `sum` and `length` commands to calculate the average of the values in `a1`.
8. Compute the sum of all integers from 1 to 100. Then compare your answer to the one you get using the formula for sum of the first n integers: $n(n+1)/2$.
9. Compute the sum of the squares of all integers from 1 to 100.
10. Take a break!

Object types

Data frames

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

```
df <- data.frame(
  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
```

```
##   IDs gender  age     trt Veteran
## 1   1   male 28.0 control   FALSE
## 2   2 female 35.5       1    TRUE
## 3   3   Male 31.0       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, and tibbles.
  - 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)
```

```
## 'data.frame':    3 obs. of  5 variables:
##  $ IDs    : int  1 2 3
##  $ gender : Factor w/ 3 levels "female","male",..: 2 1 3
##  $ age    : num  28 35.5 31
##  $ trt    : Factor w/ 2 levels "1","control": 2 1 1
##  $ Veteran: logi  FALSE TRUE TRUE
```

Data frame cells, rows, or columns

Show whole data frame

```
df
```

```
##   IDs gender  age     trt Veteran
## 1   1   male 28.0 control   FALSE
## 2   2 female 35.5       1    TRUE
## 3   3   Male 31.0       1    TRUE
```

Specific cell value:

`DatSetName[row#, column#]`

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

```
## [1] 35.5
```

Entire column:

`DatSetName[, column#]`

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

```
## [1] 28.0 35.5 31.0
```

Entire row: `DatSetName[row#, ]`

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

```
##   IDs gender  age trt Veteran
## 2   2 female 35.5   1    TRUE
```

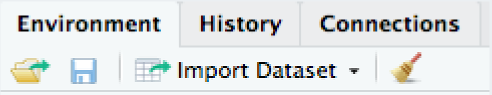
Getting the data into Rstudio

Load a data set

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

```
mydata <- read.csv("data/yrbss_demo.csv")
```

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



.

  - If you use this option, then copy and paste the importing code to your script so that you have a record of from where and how you loaded the data set.

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

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   14 years old         :1   Female:12   10th:8
##  1st Qu.: 925193   15 years old         :4   Male  : 8   11th:4
##  Median :1207132   16 years old         :7               12th:4
##  Mean   :1093150   17 years old         :7               9th :4
##  3rd Qu.:1313188   18 years old or older:1
##  Max.   :1316123
##                         race4        bmi          weight_kg
##  All other races          :5   Min.   :17.48   Min.   :43.09
##  Black or African American:3   1st Qu.:20.36   1st Qu.:57.27
##  Hispanic/Latino          :6   Median :22.23   Median :64.86
##  White                    :4   Mean   :23.01   Mean   :64.09
##  NA's                     :2   3rd Qu.:26.58   3rd Qu.:70.31
##                                Max.   :29.35   Max.   :84.82
##                        text_while_driving_30d smoked_ever bullied_past_12mo
##  0 days                          : 5          No   :10    Mode :logical
##  1 or 2 days                     : 2          Yes : 6     FALSE:11
##  3 to 5 days                     : 1          NA's: 4     TRUE :7
##  All 30 days                     : 1                      NA's :2
##  I did not drive the past 30 days: 1
##  NA's                            :10
```

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
```

```
## 'data.frame':    20 obs. of  10 variables:
##  $ id                    : int  335340 638618 922382 923122 923963 925603 93372
##  $ age                   : Factor w/ 5 levels "14 years old",..: 4 3 1 2 2 3 3
##  $ sex                   : Factor w/ 2 levels "Female","Male": 1 1 2 2 2 2 1 1
##  $ grade                 : Factor w/ 4 levels "10th","11th",..: 1 4 4 4 1 1 1 3
##  $ race4                 : Factor w/ 4 levels "All other races",..: 4 NA 4 4 2
##  $ bmi                   : num  27.6 29.3 18.2 21.4 19.6 ...
##  $ weight_kg             : num  66.2 84.8 57.6 60.3 63.5 ...
##  $ text_while_driving_30d: Factor w/ 5 levels "0 days","1 or 2 days",..: NA NA
##  $ smoked_ever           : Factor w/ 2 levels "No","Yes": NA 2 2 2 1 1 2 1 NA 1
##  $ bullied_past_12mo     : logi  NA NA FALSE FALSE TRUE TRUE ...
```

View the beginning of a data set

```
head(mydata)
```

```
##       id          age    sex grade                     race4     bmi weight_kg
## 1 335340 17 years old Female  10th                     White 27.5671     66.23
## 2 638618 16 years old Female   9th                      <NA> 29.3495     84.82
## 3 922382 14 years old   Male   9th                     White 18.1827     57.61
## 4 923122 15 years old   Male   9th                     White 21.3754     60.33
## 5 923963 15 years old   Male  10th Black or African American 19.5988     63.50
## 6 925603 16 years old   Male  10th           All other races 22.1910     70.31
##   text_while_driving_30d smoked_ever bullied_past_12mo
## 1                   <NA>        <NA>                NA
## 2                   <NA>         Yes                NA
## 3                   <NA>         Yes             FALSE
## 4                   <NA>         Yes             FALSE
## 5                   <NA>          No              TRUE
## 6                   <NA>          No              TRUE
```

View the end of a data set

```
tail(mydata)
```

```
##          id                   age    sex grade                     race4     bmi
## 15 1313153          16 years old Female  11th           Hispanic/Latino 26.5781
## 16 1313291          16 years old Female  11th                     White 24.8047
## 17 1313477          16 years old Female  10th           All other races 25.0318
## 18 1315121          17 years old Female  11th                      <NA> 22.2687
## 19 1315850          17 years old Female  12th           Hispanic/Latino 19.4922
## 20 1316123 18 years old or older Female  12th Black or African American 27.4894
##    weight_kg           text_while_driving_30d smoked_ever bullied_past_12mo
## 15     68.04                           0 days          No              TRUE
## 16     63.50                      3 to 5 days          No             FALSE
## 17     76.66                           0 days          No              TRUE
## 18     54.89 I did not drive the past 30 days         Yes             FALSE
## 19     49.90                           0 days        <NA>             FALSE
## 20     74.84                      All 30 days         Yes             FALSE
```

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

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

```
##       id          age    sex grade race4     bmi weight_kg
## 1 335340 17 years old Female  10th White 27.5671     66.23
## 2 638618 16 years old Female   9th  <NA> 29.3495     84.82
## 3 922382 14 years old   Male   9th White 18.1827     57.61
##   text_while_driving_30d smoked_ever bullied_past_12mo
## 1                   <NA>        <NA>                NA
## 2                   <NA>         Yes                NA
## 3                   <NA>         Yes             FALSE
```

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

```
##         id                   age    sex grade                     race4     bmi
## 20 1316123 18 years old or older Female  12th Black or African American 27.4894
##    weight_kg text_while_driving_30d smoked_ever bullied_past_12mo
## 20     74.84            All 30 days         Yes             FALSE
```

Working with the data

The $

Suppose we want to single out the column of BMI values.

- How did we previously learn to do this?

```
mydata[, 6]
```

```
##  [1] 27.5671 29.3495 18.1827 21.3754 19.5988 22.1910 20.9913 17.4814 22.4593
## [10] 26.5781 21.1874 19.4637 20.6121 27.4648 26.5781 24.8047 25.0318 22.2687
## [19] 19.4922 27.4894
```

The problem with this method, is that we need to know the column number which can change as we make changes to the data set.

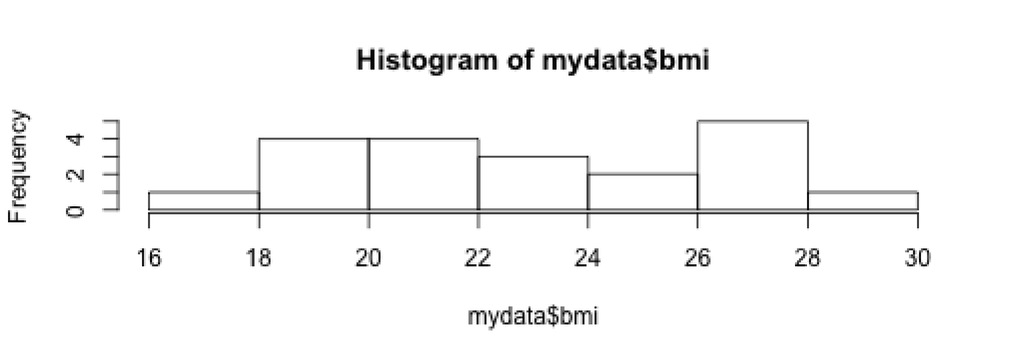
- Use the $ instead: `DatSetName$VariableName`

```
mydata$bmi
```

```
##  [1] 27.5671 29.3495 18.1827 21.3754 19.5988 22.1910 20.9913 17.4814 22.4593
## [10] 26.5781 21.1874 19.4637 20.6121 27.4648 26.5781 24.8047 25.0318 22.2687
## [19] 19.4922 27.4894
```

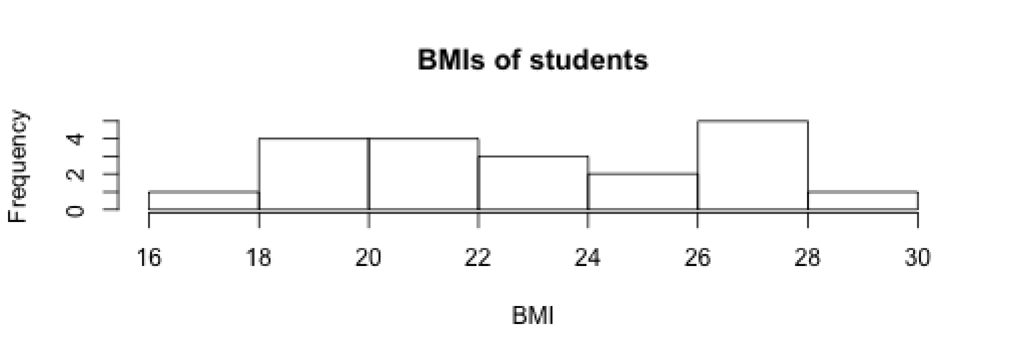
Basic plots of numeric data: Histogram

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



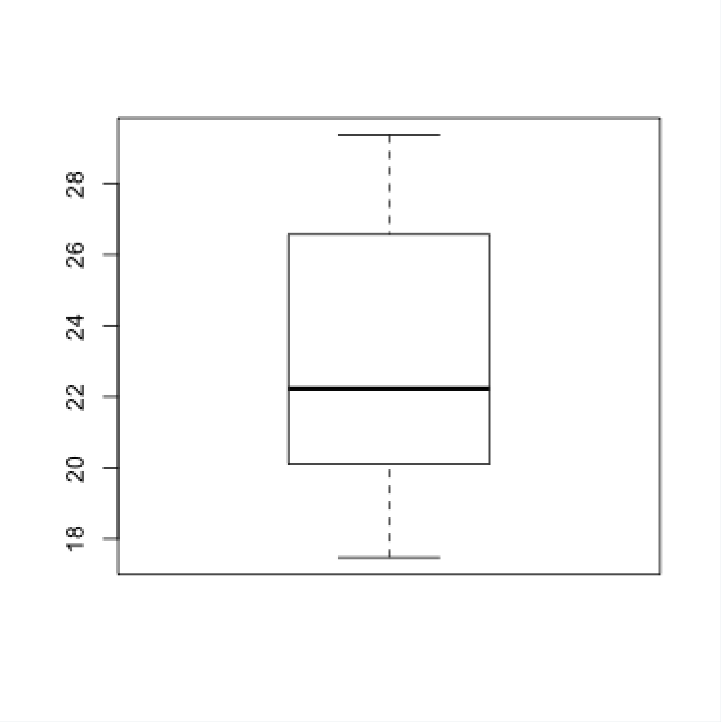
With extra features:

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

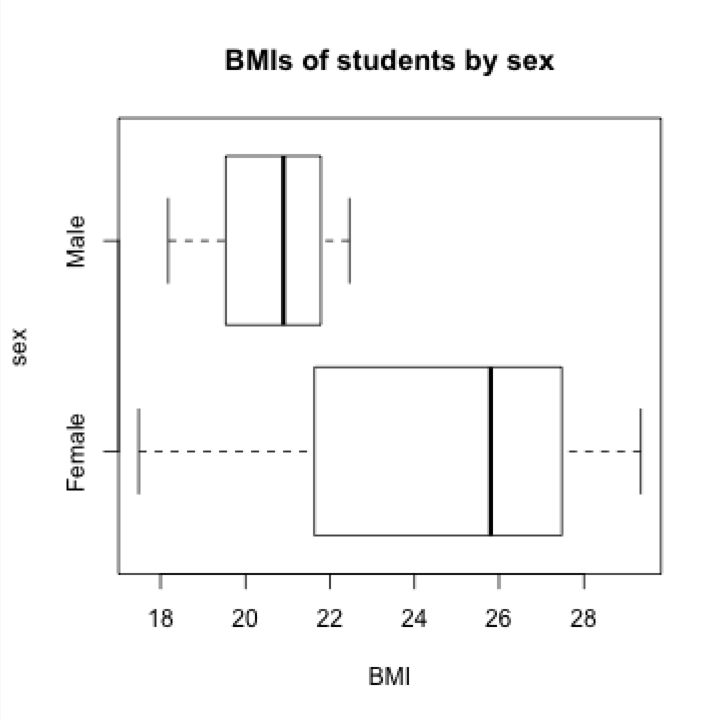


Basic plots of numeric data: Boxplot

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

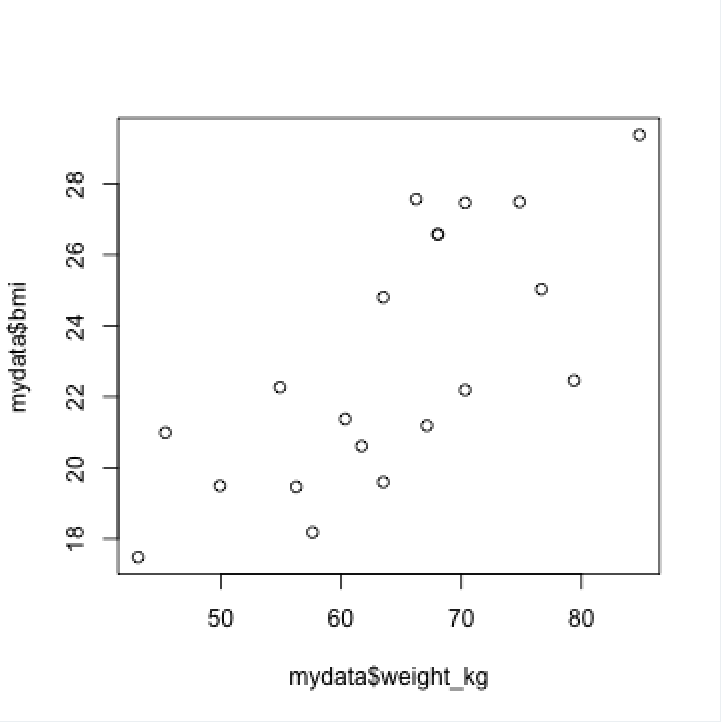


```
boxplot(mydata$bmi ~ mydata$sex,
  horizontal = TRUE,
  xlab = "BMI", ylab = "sex",
  main = "BMIs of students by sex")
```

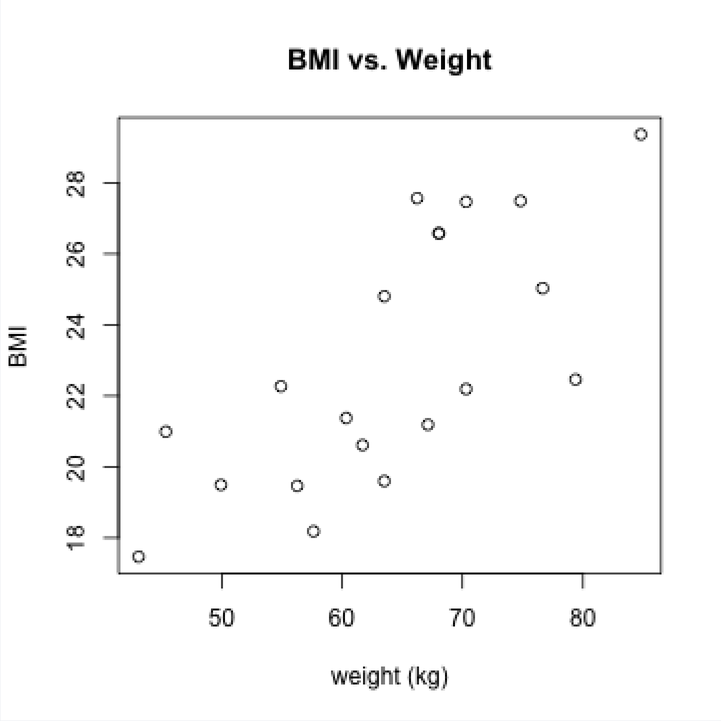


Basic plots of numeric data: Scatterplot

```
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
```

Add height column to data frame

Since $\mathrm{BMI} = \frac{kg}{m^2}$, we have $\textbf{height}(m) = \sqrt{\frac{\textbf{weight}(kg)}{\textbf{BMI}}}$

```
mydata$height_m <- sqrt( mydata$weight_kg / mydata$bmi )
mydata$height_m
```

```
##  [1] 1.550000 1.699999 1.779999 1.680001 1.799998 1.780000 1.469998 1.570002
##  [9] 1.879998 1.600001 1.779998 1.699999 1.730001 1.600001 1.600001 1.600000
## [17] 1.750001 1.569998 1.599999 1.650001
```

```
dim(mydata)
```

```
## [1] 20 11
```

```
names(mydata)
```

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

Access specific columns in data set

Previously we used `DatSetName[, column#]`

```
mydata[, c(2, 6)] # 2nd & 6th columns
```

```
##                        age     bmi
## 1             17 years old 27.5671
## 2             16 years old 29.3495
## 3             14 years old 18.1827
## 4             15 years old 21.3754
## 5             15 years old 19.5988
## 6             16 years old 22.1910
## 7             16 years old 20.9913
## 8             17 years old 17.4814
## 9             15 years old 22.4593
## 10            17 years old 26.5781
## 11            16 years old 21.1874
## 12            17 years old 19.4637
## 13            17 years old 20.6121
## 14            15 years old 27.4648
## 15            16 years old 26.5781
```

The code below uses *column names* instead of numbers.

```
mydata[, c("age", "bmi")]
```

```
##                        age     bmi
## 1             17 years old 27.5671
## 2             16 years old 29.3495
## 3             14 years old 18.1827
## 4             15 years old 21.3754
## 5             15 years old 19.5988
## 6             16 years old 22.1910
## 7             16 years old 20.9913
## 8             17 years old 17.4814
## 9             15 years old 22.4593
## 10            17 years old 26.5781
## 11            16 years old 21.1874
## 12            17 years old 19.4637
## 13            17 years old 20.6121
## 14            15 years old 27.4648
## 15            16 years old 26.5781
```

Access specific rows in data set

- Rows for 14 year olds only

```
mydata[mydata$age == "14 years old",] # 1 row since there is only one 14 year old
```

```
##        id          age  sex grade race4     bmi weight_kg text_while_driving_30d
## 3 922382 14 years old Male   9th White 18.1827     57.61                   <NA>
##   smoked_ever bullied_past_12mo height_m
## 3         Yes             FALSE 1.779999
```

- Rows for teens with BMI less than 19

```
mydata[mydata$bmi < 19,]
```

```
##        id          age    sex grade           race4     bmi weight_kg
## 3 922382 14 years old   Male   9th           White 18.1827     57.61
## 8 935435 17 years old Female  12th All other races 17.4814     43.09
##   text_while_driving_30d smoked_ever bullied_past_12mo height_m
## 3                   <NA>         Yes             FALSE 1.779999
## 8                   <NA>          No             FALSE 1.570002
```

Access specific values in data set

- Grade and race for 15 year olds only

```
mydata[mydata$age == "15 years old", c("age", "grade", "race4")]
```

```
##               age grade                     race4
## 4  15 years old   9th                     White
## 5  15 years old  10th Black or African American
## 9  15 years old  10th           All other races
## 14 15 years old  10th           Hispanic/Latino
```

- Age, sex, and BMI for students with BMI less than 19

```
mydata[mydata$bmi < 19, c("age", "sex", "bmi")]
```

```
##              age    sex     bmi
## 3 14 years old   Male 18.1827
## 8 17 years old Female 17.4814
```

Practice 2

1. Create a new script and save it as `Practice2.R`

2. Create data frames for males and females separately.

3. Do males and females have similar BMIs? Weights? Compares means, standard deviations, range, and boxplots.

4. Plot BMI vs. weight for each gender separately. Do they have similar relationships?

5. Are males or females more likely to be bullied in the past 12 months? Calculate the percentage bullied for each gender.

Save data frame

- Save **.RData** file: the standard R format, which is recommended if saving data for future use in R

```
save(mydata, file = "data/mydata.RData")  # saving mydata within the data folder
```

You can load .RData files using the load() command:

```
load("data/mydata.RData")
```

- Save **csv** file: comma-separated values

```
write.csv(mydata, file = "data/mydata.csv", col.names = TRUE, row.names = FALSE)
```

The more you know

Installing and using packages

- Packages are to R like apps are to your phone/OS
- Packages contain additional functions and data
- Install packages with `install.packages()`
  - Also can use the "Packages" tab in Files/Plots/Packages/Help/Viewer window
  - *Only install once (unless you want to update)*
  - Installs from Comprehensive R Archive Network (CRAN) = package mothership

```
install.packages("dplyr")    # only do this ONCE, use quotes
```

- Load packages: At the top of your script include `library()` commands to load each required package *every* time you open Rstudio.

```
library(dplyr)     # run this every time you open Rstudio
```

- Use a function without loading the package with `::`

```
dplyr::arrange(mydata, bmi)
```

Installing packages from other places (i.e. github, URLs)

- Need to have remotes package installed first:

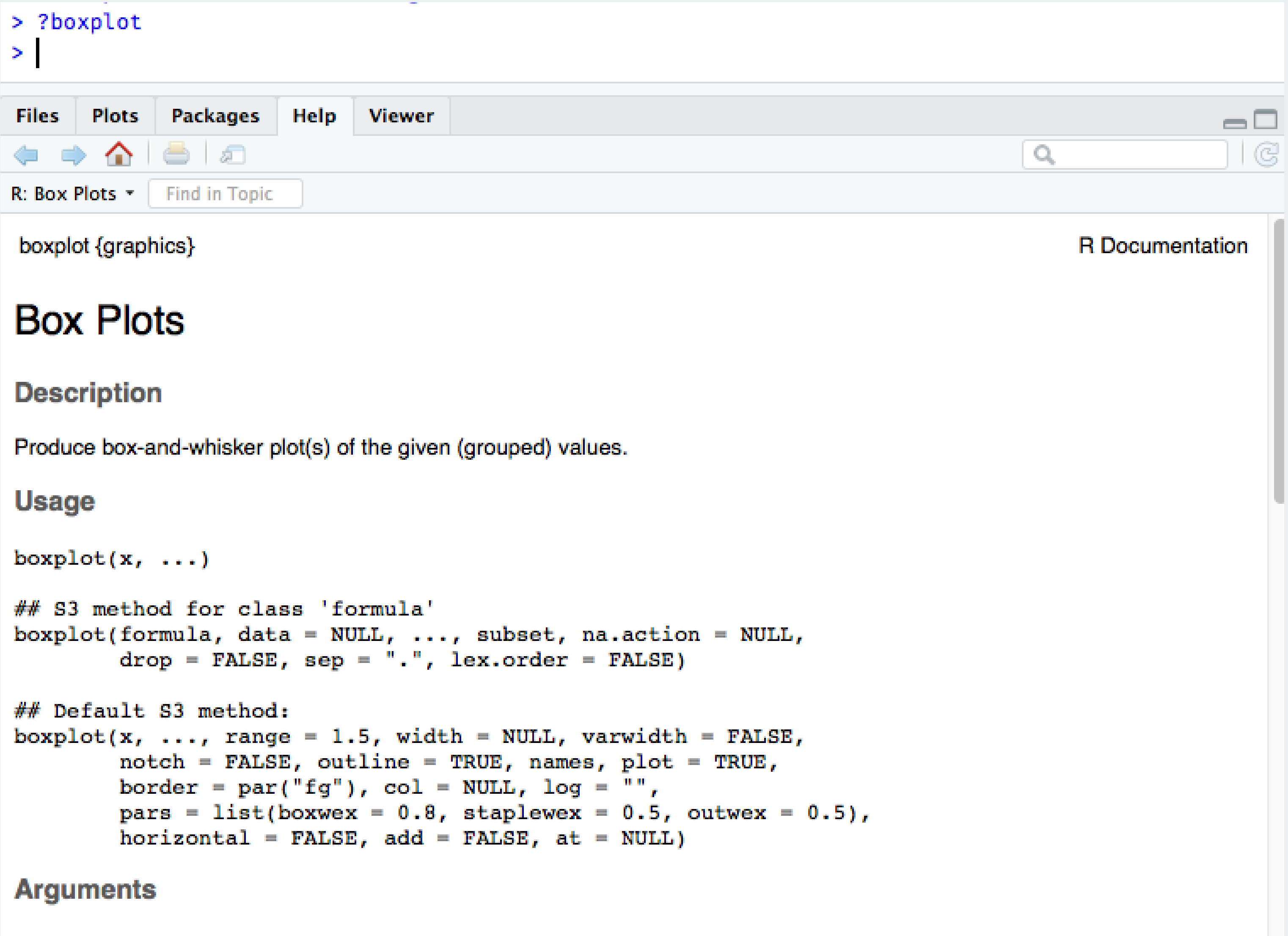
```
install.packages("remotes")
```

- To install a package from github (often in development) use `install_github()` from the remotes package

```
# https://github.com/hadley/yrbss
remotes::install_github("hadley/yrbss")

# Load it the same way
library(yrbss)
```

How to get help (1/2)

Use **?** in front of function name in console. Try this:



```
> ?boxplot
> 
```

boxplot {graphics} R Documentation

Box Plots

Description

Produce box-and-whisker plot(s) of the given (grouped) values.

Usage

```
boxplot(x, ...)

## S3 method for class 'formula'
boxplot(formula, data = NULL, ..., subset, na.action = NULL,
        drop = FALSE, sep = ".", lex.order = FALSE)

## Default S3 method:
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
        notch = FALSE, outline = TRUE, names, plot = TRUE,
        border = par("fg"), col = NULL, log = "",
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
        horizontal = FALSE, add = FALSE, at = NULL)
```

Arguments

How to get help (2/2)

- Use `??` (i.e `??dplyr` or `??read_csv`) for searching all documentation in installed packages (including unloaded packages)
- search Stack Overflow #r tag
- googlequestion + rcran or + r (i.e. "make a boxplot rcran" "make a boxplot r")
- google error in quotes (i.e. `"Evaluation error: invalid type (closure) for variable '***'"`)
- search github for your function name (to see examples) or error
- Rstudio community
- twitter #rstats

Resources

- Click on this List of resources for learning R
- Watch recordings of our other workshops
- **Highly recommend** ***Data Wrangling in R with Tidyverse***

Getting started:

- RStudio IDE Cheatsheet
- Install R/RStudio help video
- Basic Basics

Some of this is drawn from materials in online books/lessons:

- Intro to R/RStudio by Emma Rand
- Modern Dive - An Introduction to Statistical and Data Sciences via R by Chester Ismay & Albert Kim
- Cookbook for R by Winston Chang

Local resources

- OHSU's BioData club + active slack channel
- Portland's R user meetup group + active slack channel
- R-ladies PDX meetup group
- Cascadia R Conf - May 31, 2020 in Eugene with workshops

Allison Horst

Contact info:

Jessica Minnier: *minnier@ohsu.edu*

Meike Niederhausen: *niederha@ohsu.edu*

This workshop info:

- Code for these slides on github: jminnier/berd_r_courses
- all the R code in an R script
- answers to practice problems can be found here: html, pdf
- The project folder of examples can be downloaded at github.com/jminnier/berd_intro_project & the solutions are in the `solns/` folder.