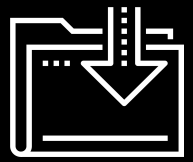
Introduction to R and Project 3

Data Boot Camp

Lesson 16.1



Class Objectives

In this lesson, you will:

- Apply the basics of R syntax
- Identify the fundamental R data types.
- Navigate RStudio.
- Create tibbles.
- Manipulate data in tibbles.
- Compare and contrast the features of Python and R.

Instructor Demonstration

Introduction to R

is a language used for data analysis, statistics, and machine learning, more popular in academia than Python.

Introduction to R

Which option is the superior one is up for debate; however, R offers compelling features, notably piping and its ease of plotting.

Introduction to R

Other R advantages include:

Speed

Specialized statistical packages

Great visualization libraries

Activity: Install RStudio

In this activity, you'll install R, RStudio and several packages.

Suggested Time:

10 Minutes

Activity: Install RStudio

Instructions	Download the page for your respective operating system: • R for Mac OSX • R for Windows
	Download the RStudio installer for your respective operating system.
	Once the file is open in RStudio, install the listed packages.
Hint	You may already have RStudio installed via Anaconda. But you may not have R itself installed and will need to install it via one of the links above.

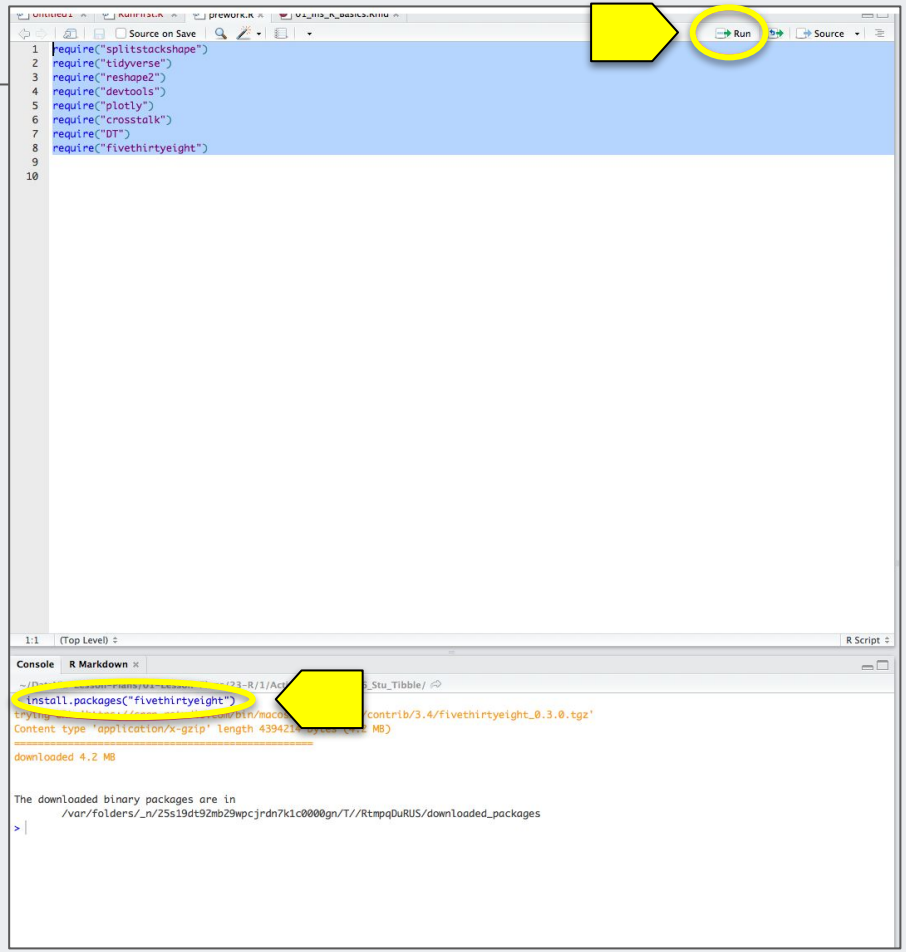
Activity: Install RStudio

Instructions	Install the packages for this week's activities. • prework.R file
	Open the file in RStudio with **File** then **Open File**.
	Once the file is open in RStudio, install the listed packages.
Hint	An alternate way to open R files: • **Mac:** pressControl while clicking • **Windows:** right-click You can also use the built-in file browser in RStudio to find and open files.

Activity: Install RStudio

There are several ways to run the installation commands in a batch.

- One way is to highlight all the lines in the text editor pane and click the **Run** button, as in the following image.
- Another way is to press
  Windows: Control + Shift + Enter
  Mac: Cmd + Shift + Enter



Activity: Install RStudio

If you encounter error messages with any of the packages, you can type `install.packages(<"package-name">)` in the console.

For example:

If an error message is returned with the `tidyverse` package,

we would type `install.packages("tidyverse")` in the console.

Try these R keyboard shortcuts.

Questions?

Instructor Demonstration

Basics of R Syntax

Basics of R Syntax

Much of the basic syntax is similar to Python and JavaScript.

Like Python

We can assign values to variables without specifying the data type.

Unlike Python

We use the left-pointing arrow `<-` to accomplish this task, with the value on the right assigned to the variable on the left.

Basics of R Syntax

Semantically `<-` is probably more accurate than the equality sign.

The equality sign can and will be used, as we will see.

However, for simple assignment operations, however, `<-` is preferred.

```
a <- 3
b < 3.1415
c <- "This is a string"
d <- "Yet another string"
e <- TRUE
f <- FALSE
g <- T
h <- F
```

The keyboard shortcut for the assignment operator `<-`

Mac: Option + - (hyphen)

Windows: Alt + - (hyphen)

Basics of R Syntax

Like Python lists, an R **vector** can hold multiple items.

Unlike Python lists, however, a vector must hold items of the same type:

```
disney_characters <- c("mickey", "minnie", "donald", "goofy")
presidents <- c("washington", "adams", "jefferson")
numbers_vector <- c(1, 3, 5, 7, 9, 11)
```

This next point is extremely important!

Basics of R Syntax

R data structures	Python and JavaScript data structures
Indexed at one	Indexed at zero
`presidents[1]` returns the first item from the vector, `"washington"`	`presidents[1]` returns the second item from the vector, `"adams"`

```
disney_characters <- c("mickey", "minnie", "donald", "goofy")

presidents <- c("washington", "adams", "jefferson")

numbers_vector <- c(1, 3, 5, 7, 9, 11)
```

Basics of R Syntax

Vectors are created using the `c()`, or concatenate, function.

We can combine two vectors into a single vector with the same operation:

```
combined_vector <- c(disney_characters, presidents)
```

Basics of R Syntax

A `for` loop in R, captured in the following code, is similar to a `for` loop in Python and JavaScript:

```
for (x in combined_vector){
    print(x)
}
```

Basics of R Syntax

Similarly, we can create a vector of integers by using the colon operator `(:)` and the `length` function.

We can even perform operations on them as a group, as in the following code:

```
numeric_vector <- 1:length(combined_vector)

squared_vector <- numeric_vector**2
```

Basics of R Syntax

An `if` statement in R, captured in the following code, works in much the same way as it does in Python:

```
for (prez in presidents){
    if (nchar(prez) > 5){
        next
    }
    else {
      print(prez)
    }
}
```

`nchar()` returns the number of characters in a string, while `next` stops the current loop iteration and starts a new iteration from the beginning.

Basics of R Syntax

R vectors can contain only a **single** data type.

A list in R can contain **multiple** data types.

```
random_list <- list("movies"=c("Star Wars", "Titanic", "Avatar"),
                    "states"=c("California", "Oklahoma", "Texas", "Virginia"),
                    "coins"=c("penny", "dime", "nickel", "quarter"),
                    "first_presidents"=presidents,
                    "nums"=c(1,2,3,4,5),
                    "bools"=c(T,F,T,T,T,F)
                    )
```

Basics of R Syntax

We can use **bracket notation** to access an item in a list:

```
random_list["states"]
```

We can also use a **dollar sign** to access an item in a list:

```
random_list$coins
```

We can verify that `random_list` is indeed a list with `typeof()`:

```
typeof(random_list)
```

Questions?

Activity: Student Homeroom

In this activity, you will:

- Practice the basics of R syntax.
- Create vectors, use `for` loops, use `if-else` statements, identify substrings of strings, and create functions.

Suggested Time:

15 Minutes

Time’s Up! Let’s Review.

Review: We R in Junior High Again

Part 1	You were required to research how to access the current date.	This can be done with `Sys.Date()`
Part 2	You were required to research how to generate a pseudo-random sample of three numbers from 1 through 33.	This can be done with `sample(33, 3)`
Part 3	You were required to research how to access a substring of a string.	This can be done with `substr(student, 2, 2)`

Questions?

Time to Code

Vectors and Pipes

Suggested Time:

10 Minutes

Vectors and Pipes

`names()`

Just as an array can be used to index a Series in Pandas, a vector in R can be paired up as names for another vector using the `names()` function.

`summary()`

We can store the results of `summary()` in a vector and then access features of the summary.

`summary()` provides a statistical summary of a data set.

Vectors and Pipes

We can use the familiar square brackets to index elements in a vector.

For example, to access the features of a summary, we use **single square brackets**:

```
precipitation_summary["Min."]
precipitation_summary["Mean"]
```

Note: the minimum value is stored with the key `Min.`, including the period, which must be included in the syntax.

Alternatively, if we want to access only the value, we use double square brackets, like `precipitation_summary[["Max."]]`.

The pipe operator `(%>%)` is a helpful feature that can improve our R workflow.

Vectors and Pipes

To obtain a summary of the `precipitation` vector, we use:

```
summary(precipitation)
```

The same result can be obtained by using the pipe operator:

```
precipitation %>% summary()
```

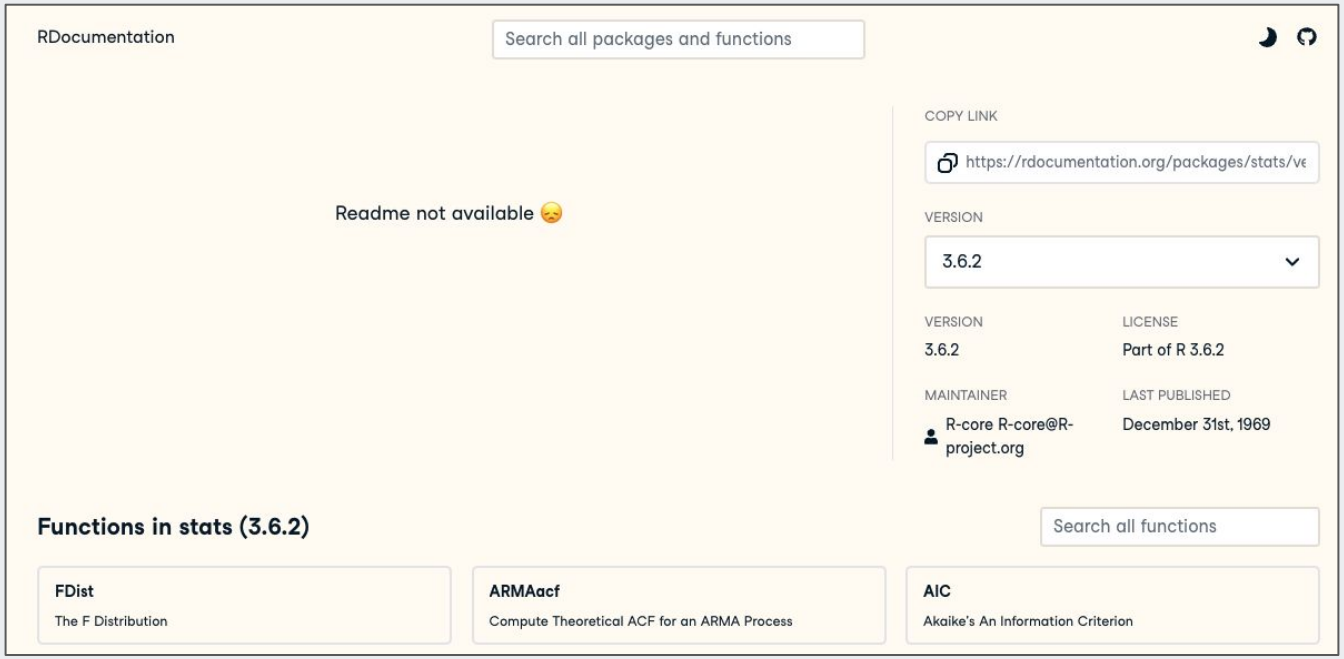
The pipe operator takes what's **on the left** (the `precipitation` vector) and performs the operation **on the right** (the `summary()` function).

The keyboard shortcut for the pipe operator `%>%` is:
Mac: Cmd + Shift + M
Windows: Ctrl + Shift + M

R Stats Package

R ships with a [stats package](#).

Here, we are using `sd()` to calculate the standard deviation of `precipitation`.



Questions?

File Structure Navigation

In order to work with external data files, such as **CSV files**, you will need to familiarize yourself with file structure navigation in R.

File Structure Navigation

File structure navigation in R shares some similarities to that in Unix-based environments.

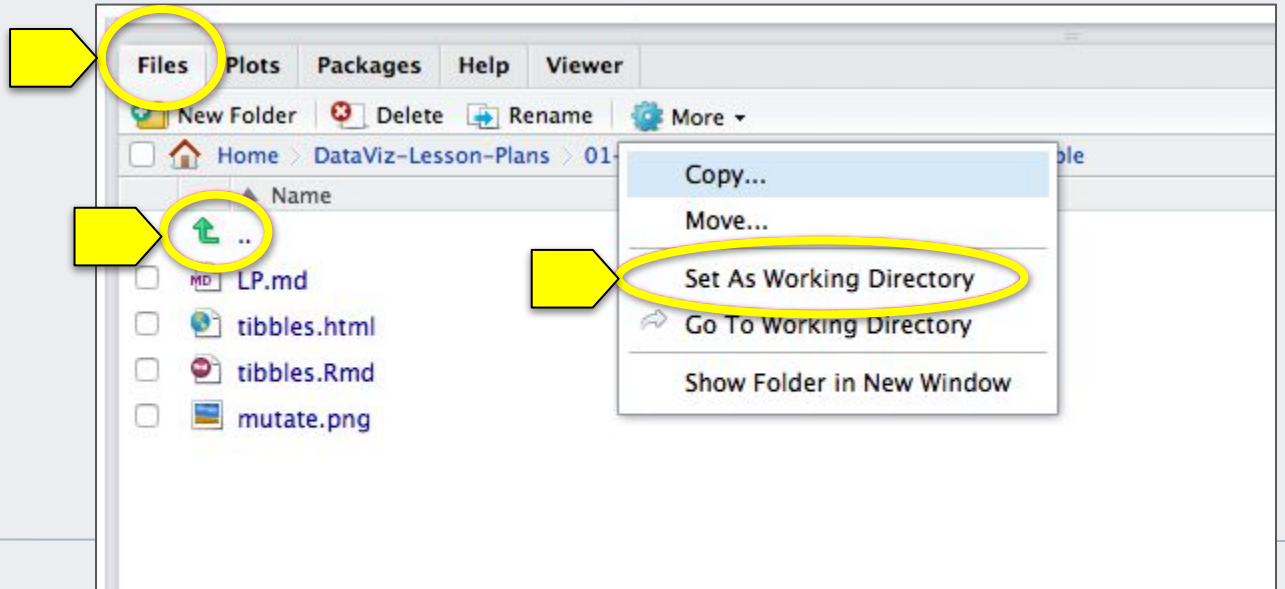
To display	R command	Terminal equivalent
Current directory	`getwd()`	`pwd`
Contents of the current directory	`dir()`	`ls`
Change the director	`setwd()`	`cd`

File Structure Navigation

A simpler way to set the working directory in RStudio is to select the **Files** panel, then use either the up arrow icons  to move up a directory or click on a directory name to navigate into it.

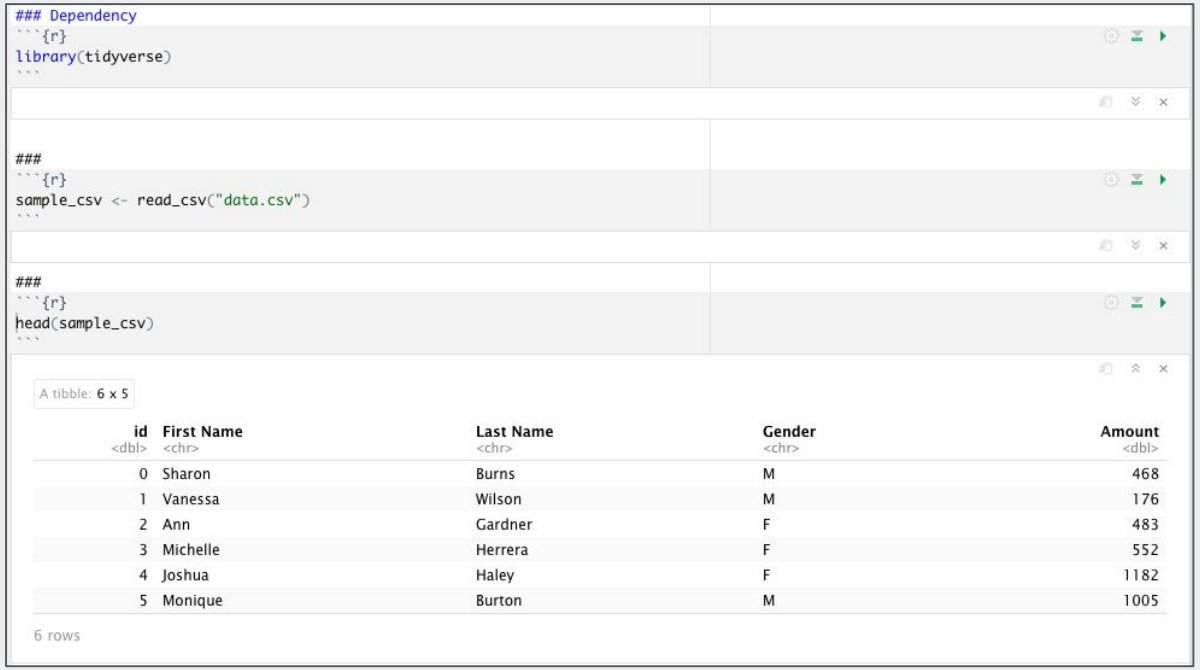
Under the 

menu, select **Set As Working Directory**:



File Structure Navigation

Navigate to the directory for this activity, and load the `data.csv` file.



Dependency

```{r}
library(tidyverse)
```

###

```{r}
sample_csv <- read_csv("data.csv")
```

###

```{r}
head(sample_csv)
```

A tibble: 6 x 5

id <dbl>	First Name <chr>	Last Name <chr>	Gender <chr>	Amount <dbl>
0	Sharon	Burns	M	468
1	Vanessa	Wilson	M	176
2	Ann	Gardner	F	483
3	Michelle	Herrera	F	552
4	Joshua	Haley	F	1182
5	Monique	Burton	M	1005

6 rows

To run a cell of code in an RMD file, you can click the green play button or press

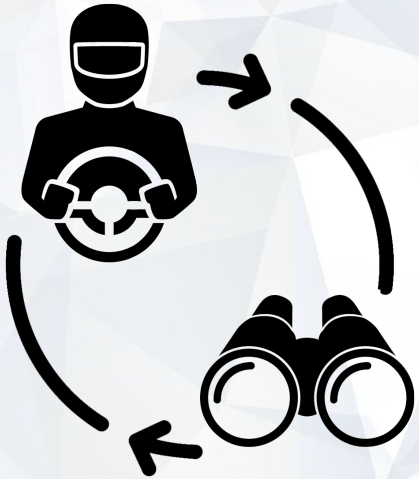
Mac: Command + Shift + Enter

Windows: Ctrl + Shift + Enter

File Structure Navigation

Here are some additional command to reference, but you are not required to learn them:

To create a directory called "data_science":	`dir.create("data_science")`
To create a file:	`file.create("my_first.R")`
To determine whether a file exists:	`file.exists()`
Obtain additional info on a file:	`file.info()`
Rename a file:	`file.rename(file1, file2)`
To copy a file:	`file.copy()`



Pair Programming Activity:

Partners Do: Discussion

In this activity, you will pair up with a partner to:

- Ensure that your partner is able to load `data.csv` in RStudio.
- Discuss the syntax of R, comparing and contrasting it with those of Python and JavaScript.

Suggested Time:

10 Minutes

Partners Do: Discussion

Instructions	Discuss topics including:
	• Assignment operator used to assign a variable
	• Declaring a function
	• Basic data structures
	• Concatenating a string
	• Logical operators (they will have to research on logical operators in R)
	• `If-else` statements

Questions?

DataFrames in R

A **tibble**, (a word play on “table”), is a DataFrame.

DataFrames in R

Tibbles in R are similar to DataFrames in Pandas:

Data are organized by rows and columns.

Data allow operations for computation and data-wrangling.

There are some compelling new aspects of R, such as piping, which we will discuss later.

`tibbles.html` provides a complete walk-through of the code, but here are some additional details...

DataFrames in R

`library(tidyverse)`:

`tidyverse` is a collection of data science-oriented packages.

- Tibbles are not available in standard R, they are enabled by `tidyverse` and are generally superior to R's standard data frame.
- The `library()` function loads this package.

```
### Load dependency and sample data set
library(tidyverse)
data(diamonds, package='ggplot2')
```

DataFrames in R

`data(diamonds, package='ggplot2')`:

- the `data()` unction loads data sources.
- `diamonds` is a sample data set that comes with `ggplot2`, a plotting package for R.

```
### Load dependency and sample data set
library(tidyverse)
data(diamonds, package='ggplot2')
```

DataFrames in R

Unlike other languages that we have encountered so far, R allows the use of periods/dots in regular variable names:

```
total.volume2 <- mutate(diamonds, total.volume=(x*y*z))
```

- `total.volume2` does not refer to a `volume2` property of `total` object.
- It is simply a variable name. It is equivalent to `total_volume2`, for example.

DataFrames in R

The `mutate()` function adds a new columnar variable to the tibble.

```
### Operate over a column, temporarily add a new column
```{r}
total_volume <- mutate(diamonds, x * y * z)
total_volume
```
```

cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>	x * y * z <dbl>
Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	38.20203
Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	34.50586
Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31	38.07688
Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	46.72458
Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	51.91725
Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48	38.69395
Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47	38.83087
Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53	42.32108
Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49	36.42521
Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39	38.71800

1-10 of 53,940 rows | 2-11 of 11 columns Previous 1 2 3 4 5 6 ... 100 Next

```
### Name the new column, save the results to another tibble
```{r}
# In R, variables can contain periods
total.volume2 <- mutate(diamonds, total.volume=(x*y*z))
total.volume2
```
```

cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>	total.volume <dbl>
Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	38.20203
Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	34.50586
Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31	38.07688
Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	46.72458
Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	51.91725

Note:

In the `Rmd` file, it may be necessary to click on the arrow to reveal more columns of a tibble.

Activity: Back to School

In this activity, you will perform some of the same data operations that you ran in the PySchool homework assignment from the fourth week: this time in R!

Suggested Time:

20 Minutes

Activity: Install RStudio

Instructions

You will perform some of the same data operations you ran in the homework assignment, including the following:

- Create a list of all schools.
- Calculate the total count of schools.
- Calculate the total number of students.
- Calculate the average reading and math scores.
- Calculate the percentage of students with passing reading scores (scores of 70% or higher).
- Calculate the percentage of students with passing math scores (scores of 70% or higher).
- Calculate the overall passing rate (the average of passing math and reading percentages).

(Data Source)

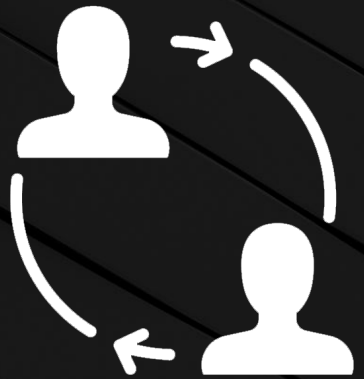
Time’s Up! Let’s Review.

Review: Back to School

- First, we load `tidyverse` with `library(tidyverse)`, then `use read_csv()` to load an external CSV file.
- `head()` and `tail()` can be used to preview the tibble.
- `unique()` is used to display the unique entries across a column.
- `summarize()` provides a basic statistical summary.
- `filter()` can be used to filter rows.
- `group_by()` can be used to perform aggregate calculations, such as mean and sum.

Questions?


Break



Project 3 Group Work

Suggested Time:

55 Minutes

Project 3 Group Work

You have two weeks to work on this project.

- The goal of this project is to tell a story using data visualizations.
- Interactivity should be built-in, allowing users to explore data on their own.
- Teams will also create a 10-minute presentation that lays out your theme, coding approach, data wrangling techniques, and final visualization.

Project 3 Group Work

This project's rubric has five categories:

01 Data and data delivery

02 Back end (ETL)

03 Visualizations

04 Group presentations

05 Slide deck

Specific Requirements

01 Your visualization must include a Python Flask–powered API, HTML/CSS, JavaScript, and at least one database (SQL, MongoDB, SQLite, etc.).

02 Your project should fall into one of these tracks:

- A combination of web scraping and Leaflet or Plotly
- A dashboard page with multiple charts that update from the same data

03 Your project should include at least one JS library that we did not cover.

04 Your project must be powered by a dataset with at least 100 records.

05 Your project must include some level of user-driven interaction (e.g., menus, dropdowns, textboxes).

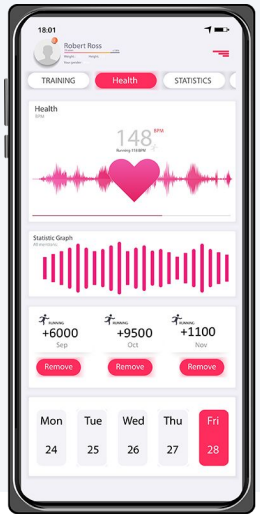
06 Your final visualization should ideally include at least three views.

Project 3 Group Work

You can focus on:

Finance

Healthcare



Custom

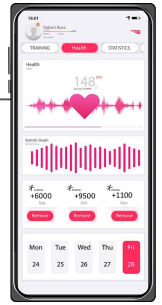
Dashboard Example: Finance

Tracking market data is crucial for equity traders. Not all traders code and are able to create custom-tailored visualizations. What's the best way for them to get what they need for success?

One option is offered by the Wall Street Journal.

- Their website offers a dashboarding tool providing a high-level view of market performance.
- This highly interactive tool allows users to easily explore stocks, bonds, currencies, and commodities.
- Users of all skill levels can utilize these data.
- Visualizations help make the data easier to understand.
- Multiple views are available for customized content.

Dashboard Example: Healthcare



Imagine: Vacation time is coming up—and so is flu season. Trying to plan a road trip across the United States while keeping everyone's health in mind can be tricky. Using the FluView dashboard provided by the CDC, users can easily confirm which areas to avoid. Different interactive features include:

- An overall view of the United States, or customizable view (state by state)
- Historic and current cases
- A chart showing the count of cases, broken down by strain
- With this, data is delivered quickly and navigated through with ease.

Dashboard Example: Weather Tracking

While on the way to work one morning, you notice dark clouds on the horizon. You don't remember hearing about a storm front coming in, but this looks ominous.

- A quick visit to Weather Underground's Dashboard helps illuminate the situation.
- Updated with live data, you can view a live map as well as specific conditions such as temperature, pressure, and even feed from a live webcam.
- The data delivery is up-to-date and seamless, making it easy to understand current conditions without digging too deeply.

Time to divide into teams!

Questions?


The End