Data Science: Wrangling - Assessments

Hector Antigua

Last compiled on January, 2022

## Assessment Part 1: Data Import

In this part of the assessment, you will answer several multiple choice questions that review the concepts of data import. You can answer these questions without using R, although you may find it helpful to experiment with commands in your console.

In the second part of the assessment on the next page, you will import real datasets and learn more about useful arguments to readr functions. The second part of the assessment will require you to program in R.

### **Question 1**

Which of the following is NOT part of the data wrangling process?

**Answer:** *Checking correlations between your variables*

### **Question 2**

Which files could be opened in a basic text editor?

**Answer:** *data.txt* *data.csv* *data.tsv*

### **Question 3**

You want to analyze a file containing race finish times for a recent marathon. You open the file in a basic text editor and see lines that look like the following:

initials,state,age,time  
vib,MA,61,6:01  
adc,TX,45,5:45  
kme,CT,50,4:19

**Answer:** *A comma-delimited file with a header*

### **Question 4**

Assume the following is the full path to the directory that a student wants to use as their working directory in R: “/Users/student/Documents/projects/”

Which of the following lines of code CANNOT set the working directory to the desired “projects” directory?

**Answer:** *setwd(/Users/student/Documents/projects/)*

### **Question 5**

We want to copy the “murders.csv” file from the dslabs package into an existing folder “data”, which is located in our HarvardX-Wrangling projects folder. We first enter the code below into our RStudio console.

> getwd()  
[1] "C:/Users/UNIVERSITY/Documents/Analyses/HarvardX-Wrangling"  
> filename <- "murders.csv"  
> path <- system.file("extdata", package = "dslabs")

Which of the following commands would NOT successfully copy “murders.csv” into the folder “data”?

**Answer:** *file.copy(file.path(path, “murders.csv”), getwd())*

### **Question 6**

You are not sure whether the murders.csv file has a header row. How could you check this?

**Answer:** *Open the file in a basic text editor.* *In the RStudio “Files” pane, click on your file, then select “View File”.* *Use the command read\_lines (remembering to specify the number of rows with the n\_max argument).*

### **Question 7**

What is one difference between read\_excel() and read\_xlsx()?

**Answer:** *read\_excel() reads both .xls and .xlsx files by detecting the file format from its extension, while read\_xlsx() only reads .xlsx files.*

### **Question 8**

You have a file called “times.txt” that contains race finish times for a marathon. The first four lines of the file look like this:

initials,state,age,time  
vib,MA,61,6:01  
adc,TX,45,5:45  
kme,CT,50,4:19

Which line of code will NOT produce a tibble with column names “initials”, “state”, “age”, and “time”?

**Answer:** *race\_times <- read.csv(“times.txt”)*

### **Question 9**

You also have access to marathon finish times in the form of an Excel document named “times.xlsx”. In the Excel document, different sheets contain race information for different years. The first sheet is named “2015”, the second is named “2016”, and the third is named “2017”.

Which line of code will NOT import the data contained in the “2016” tab of this Excel sheet?

**Answer:** *times\_2016 <- read\_xlsx(“times.xlsx”, sheet = “2”)*

### **Question 10**

You have a comma-separated values file that contains the initials, home states, ages, and race finish times for marathon runners. The runners’ initials contain three characters for the runners’ first, middle, and last names (for example, “KME”).

You read in the file using the following code.

race\_times <- read.csv(“times.csv”)

What is the data type of the initials in the object race\_times?

**Answer:** *factors* Nota:*In previous versions of R, this was true, but is not any more. read.csv() no longer automatically converts characters to factors. If you want to read in character columns as factors, you can supply the argument stringsAsFactors = T.*

### **Question 11**

Which of the following is NOT a real difference between the readr import functions and the base R import functions?

**Answer:** *The base R import functions can read .csv files, but cannot read files with other delimiters, such as .tsv files, or fixed-width files.*

### **Question 12**

You read in a file containing runner information and marathon finish times using the following code.

race\_times <- read.csv(“times.csv”, stringsAsFactors = F)

What is the class of the object race\_times?

**Answer:** *data frame*

### **Question 13**

Select the answer choice that summarizes all of the actions that the following lines of code can perform. Please note that the url below is an example and does not lead to data.

url <- "https://raw.githubusercontent.com/MyUserName/MyProject/master/MyData.csv "  
dat <- read\_csv(url)  
download.file(url, "MyData.csv")

**Answer:** *Create a tibble in R called dat that contains the information contained in the csv file stored on Github. Download the csv file to the working directory and name the downloaded file “MyData.csv”.*

# Assessment Part 2: Data Import

In this part of the assessment, you will import real datasets and learn more about useful arguments to readr functions. You will encounter common issues that arise when importing raw data. This part of the assessment will require you to program in R.

Use the readr package in the tidyverse library:

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.4 v purrr 0.3.4  
## v tibble 3.1.2 v dplyr 1.0.6  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

### **Question 14**

Inspect the file at the following URL:

<https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data>

Which **readr** function should be used to import this file?

**Answer:** read\_csv()

### **Question 15**

Check the documentation for the readr function you chose in the previous question to learn about its arguments. Determine which arguments you need to the file from the previous question:

url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"

Does this file have a header row? Does the readr function you chose need any additional arguments to import the data correctly?

**Answer:** *No, there is no header. The col\_names=FALSE argument is necessary.*

### **Question 16**

Inspect the imported data from the previous question.

d\_q16 <- read\_csv(url, col\_names = FALSE)

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_double(),  
## X2 = col\_character()  
## )  
## i Use `spec()` for the full column specifications.

How many rows are in the dataset? *569*

How many columns are in the dataset? *32*

# Assessment Part 1: Reshaping Data

Part 1 consists of 8 questions are conceptual questions about tidy data and reshaping data. They do not necessarily require R, but you may benefit from checking your work on the console.

Part 2 consists of 7 questions which require you to write code in R to apply the new concepts about tidy data and reshaping data.

### **Question 1**

A collaborator sends you a file containing data for three years of average race finish times.

age\_group,2015,2016,2017  
20,3:46,3:22,3:50  
30,3:50,3:43,4:43  
40,4:39,3:49,4:51  
50,4:48,4:59,5:01

Are these data considered “tidy” in R? Why or why not?

**Answer:** *No. These data are not considered “tidy” because the variable “year” is stored in the header.*

### **Question 2**

Below are four versions of the same dataset. Which one is in a tidy format?

**Answer:**

state abb region population total  
Alabama AL South 4779736 135  
Alaska AK West 710231 19  
Arizona AZ West 6392017 232  
Arkansas AR South 2915918 93  
California CA West 37253956 1257  
Colorado CO West 5029196 65

### **Question 3**

Your file called “times.csv” has age groups and average race finish times for three years of marathons.

age\_group,2015,2016,2017  
20,3:46,3:22,3:50  
30,3:50,3:43,4:43  
40,4:39,3:49,4:51  
50,4:48,4:59,5:01

You read in the data file using the following command.

d <- read\_csv("times.csv")

**Answer:**

tidy\_data <- d %>%  
 gather(year, time, `2015`:`2017`)

### **Question 4**

You have a dataset on U.S. contagious diseases, but it is in the following wide format:

> head(dat\_wide)  
state year population HepatitisA Mumps Polio Rubella  
Alabama 1990 4040587 86 19 76 1  
Alabama 1991 4066003 39 14 65 0  
Alabama 1992 4097169 35 12 24 0  
Alabama 1993 4133242 40 22 67 0  
Alabama 1994 4173361 72 12 39 0  
Alabama 1995 4216645 75 2 38 0

You want to transform this into a tidy dataset, with each row representing an observation of the incidence of each specific disease (as shown below):

> head(dat\_tidy)  
state year population disease count  
Alabama 1990 4040587 HepatitisA 86  
Alabama 1991 4066003 HepatitisA 39  
Alabama 1992 4097169 HepatitisA 35  
Alabama 1993 4133242 HepatitisA 40  
Alabama 1994 4173361 HepatitisA 72  
Alabama 1995 4216645 HepatitisA 75

Which of the following commands would achieve this transformation to tidy the data?

**Answer:**

dat\_tidy <- dat\_wide %>%  
 gather(key = disease, value = count, HepatitisA:Rubella)

### **Question 5**

You have successfully formatted marathon finish times into a tidy object called tidy\_data. The first few lines are shown below.

age\_group year time  
20 2015 03:46  
30 2015 03:50  
40 2015 04:39  
50 2015 04:48  
20 2016 03:22

Select the code that converts these data back to the wide format, where each year has a separate column.

**Answer:**  
*tidy\_data %>% spread(year, time)*

### **Question 6**

You have the following dataset:

> head(dat)  
state abb region var people  
Alabama AL South population 4779736  
Alabama AL South total 135  
Alaska AK West population 710231  
Alaska AK West total 19  
Arizona AZ West population 6392017  
Arizona AZ West total 232

You would like to transform it into a dataset where population and total are each their own column (shown below):

state abb region population total  
Alabama AL South 4779736 135  
Alaska AK West 710231 19  
Arizona AZ West 6392017 232  
Arkansas AR South 2915918 93  
California CA West 37253956 1257  
Colorado CO West 5029196 65

Which code would best accomplish this?

**Answer:**  
*dat\_tidy <- dat %>% spread(key = var, value = people)*

### **Question 7**

A collaborator sends you a file containing data for two years of average race finish times, “times.csv”:

age\_group,2015\_time,2015\_participants,2016\_time,2016\_participants  
20,3:46,54,3:22,62  
30,3:50,60,3:43,58  
40,4:39,29,3:49,33  
50,4:48,10,4:59,14

You read in the data file:

d <- read\_csv("times.csv")

Which of the answers below best makes the data tidy?

**Answer:**

tidy\_data <- d %>%  
 gather(key = “key”, value = “value”, -age\_group) %>%  
 separate(col = key, into = c(“year”, “variable\_name”), sep = “\_”) %>%   
 spread(key = variable\_name, value = value)

### **Question 8**

You are in the process of tidying some data on heights, hand length, and wingspan for basketball players in the draft. Currently, you have the following:

> head(stats)  
key value  
allen\_height 75  
allen\_hand\_length 8.25  
allen\_wingspan 79.25  
bamba\_height 83.25  
bamba\_hand\_length 9.75  
bamba\_wingspan 94

Select all of the correct commands below that would turn this data into a “tidy” format with columns “height”, “hand\_length” and “wingspan”.

**Answer:**

tidy\_data <- stats %>%  
 separate(col = key, into = c("player", "variable\_name"), sep = "\_", extra = "merge") %>%   
 spread(key = variable\_name, value = value)

# Assessment Part 2: Reshaping Data

Use the following libraries for these questions:

library(tidyverse)  
library(dslabs)

### **Question 9**

Examine the built-in dataset co2. This dataset comes with base R, not **dslabs** - just type co2 to access the dataset.

Is co2 tidy? Why or why not?

head(co2)

## [1] 315.42 316.31 316.50 317.56 318.13 318.00

**Answer:** *co2 is not tidy: to be tidy we would have to wrangle it to have three columns (year, month and value), and then each co2 observation would have a row.*

### **Question 10**

Run the following command to define the co2\_wide object:

co2\_wide <- data.frame(matrix(co2, ncol = 12, byrow = TRUE)) %>%   
 setNames(1:12) %>%   
 mutate(year = as.character(1959:1997))

Use the gather() function to make this dataset tidy. Call the column with the CO2 measurements co2 and call the month column month. Name the resulting object co2\_tidy.

Which code would return the correct tidy format?

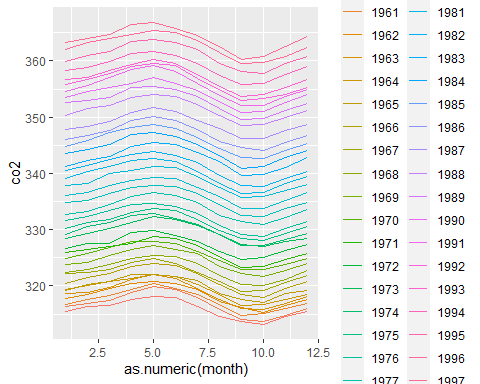
**Answer:**

co2\_tidy <- gather(co2\_wide,month,co2,-year)

### **Question 11**

Use co2\_tidy to plot CO2 versus month with a different curve for each year:

co2\_tidy %>% ggplot(aes(as.numeric(month), co2, color = year)) + geom\_line()



What can be concluded from this plot?

**Answer:** *CO2 concentrations are highest around May and the yearly average increased from 1959 to 1997.*

### **Question 12**

Load the admissions dataset from **dslabs**, which contains college admission information for men and women across six majors, and remove the applicants percentage column:

library(dslabs)  
data(admissions)  
dat <- admissions %>% select(-applicants)

Your goal is to get the data in the shape that has one row for each major, like this:

major men women  
A 62 82   
B 63 68   
C 37 34   
D 33 35   
E 28 24   
F 6 7

Which command could help you to wrangle the data into the desired format?

**Answer:**

dat\_tidy <- spread(dat, gender, admitted)

### **Question 13**

Now use the admissions dataset to create the object tmp, which has columns major, gender, key and value:

tmp <- gather(admissions, key, value, admitted:applicants)  
tmp

## major gender key value  
## 1 A men admitted 62  
## 2 B men admitted 63  
## 3 C men admitted 37  
## 4 D men admitted 33  
## 5 E men admitted 28  
## 6 F men admitted 6  
## 7 A women admitted 82  
## 8 B women admitted 68  
## 9 C women admitted 34  
## 10 D women admitted 35  
## 11 E women admitted 24  
## 12 F women admitted 7  
## 13 A men applicants 825  
## 14 B men applicants 560  
## 15 C men applicants 325  
## 16 D men applicants 417  
## 17 E men applicants 191  
## 18 F men applicants 373  
## 19 A women applicants 108  
## 20 B women applicants 25  
## 21 C women applicants 593  
## 22 D women applicants 375  
## 23 E women applicants 393  
## 24 F women applicants 341

Combine the key and gender and create a new column called column\_name to get a variable with the following values: admitted\_men, admitted\_women, applicants\_men and applicants\_women. Save the new data as tmp2.

Which command could help you to wrangle the data into the desired format?

**Answer:**

tmp2 <- unite(tmp, column\_name, c(key, gender))

### **Question 14**

Which function can reshape tmp2 to a table with six rows and five columns named major, admitted\_men, admitted\_women, applicants\_men and applicants\_women?

**Answer:**

spread(tmp2,column\_name,value)

## major admitted\_men admitted\_women applicants\_men applicants\_women  
## 1 A 62 82 825 108  
## 2 B 63 68 560 25  
## 3 C 37 34 325 593  
## 4 D 33 35 417 375  
## 5 E 28 24 191 393  
## 6 F 6 7 373 341

# Assessment: Combining Tables

### **Question 1**

You have created data frames tab1 and tab2 of state population and election data, similar to our module videos:

> tab1  
state population  
Alabama 4779736  
Alaska 710231  
Arizona 6392017  
Delaware 897934  
District of Columbia 601723  
  
> tab2  
state electoral\_votes  
Alabama 9  
Alaska 3  
Arizona 11  
California 55  
Colorado 9  
Connecticut 7  
  
> dim(tab1)  
[1] 5 2  
  
> dim(tab2)  
[1] 6 2

What are the dimensions of the table dat, created by the following command?

dat <- left\_join(tab1, tab2, by = “state”)

**Answer:** *5 rows by 3 columns*

### **Question 2**

We are still using the tab1 and tab2 tables shown in question 1. What join command would create a new table “dat” with three rows and two columns?

**Answer:** *dat <- semi\_join(tab1, tab2, by = “state”)*

### **Question 3**

Which of the following are real differences between the join and bind functions?

**Answer:** *Binding functions combine by position, while join functions match by variables.* *Joining functions can join datasets of different dimensions, but the bind functions must match on the appropriate dimension (either same row or column numbers).* *Bind functions can combine both vectors and dataframes, while join functions work for only for dataframes.*

### **Question 4**

We have two simple tables, shown below, with columns x and y:

> df1  
 x y   
 a a   
 b a   
  
> df2  
 x y   
 a a   
 a b

Which command would result in the following table?

> final  
 x y   
 b a

**Answer:**

final <- setdiff(df1, df2)

**Introduction to Questions 5-7**

Install and load the Lahman library. This library contains a variety of datasets related to US professional baseball. We will use this library for the next few questions and will discuss it more extensively in the Regression course. For now, focus on wrangling the data rather than understanding the statistics.

The Batting data frame contains the offensive statistics for all baseball players over several seasons. Filter this data frame to define top as the top 10 home run (HR) hitters in 2016:

install.packages("Lahman")

## Installing package into 'C:/Users/guy\_l/OneDrive/Documentos/R/win-library/4.1'  
## (as 'lib' is unspecified)

## package 'Lahman' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\guy\_l\AppData\Local\Temp\RtmpQHrmB5\downloaded\_packages

library(Lahman)

## Warning: package 'Lahman' was built under R version 4.1.2

top <- Batting %>%   
 filter(yearID == 2016) %>%  
 arrange(desc(HR)) %>% # arrange by descending HR count  
 slice(1:10) # take entries 1-10  
top %>% as\_tibble()

## # A tibble: 10 x 22  
## playerID yearID stint teamID lgID G AB R H X2B X3B HR  
## <chr> <int> <int> <fct> <fct> <int> <int> <int> <int> <int> <int> <int>  
## 1 trumbma01 2016 1 BAL AL 159 613 94 157 27 1 47  
## 2 cruzne02 2016 1 SEA AL 155 589 96 169 27 1 43  
## 3 daviskh01 2016 1 OAK AL 150 555 85 137 24 2 42  
## 4 doziebr01 2016 1 MIN AL 155 615 104 165 35 5 42  
## 5 encared01 2016 1 TOR AL 160 601 99 158 34 0 42  
## 6 arenano01 2016 1 COL NL 160 618 116 182 35 6 41  
## 7 cartech02 2016 1 MIL NL 160 549 84 122 27 1 41  
## 8 frazito01 2016 1 CHA AL 158 590 89 133 21 0 40  
## 9 bryankr01 2016 1 CHN NL 155 603 121 176 35 3 39  
## 10 canoro01 2016 1 SEA AL 161 655 107 195 33 2 39  
## # ... with 10 more variables: RBI <int>, SB <int>, CS <int>, BB <int>,  
## # SO <int>, IBB <int>, HBP <int>, SH <int>, SF <int>, GIDP <int>

Also Inspect the Master data frame, which has demographic information for all players:

Master %>% as\_tibble()

## # A tibble: 20,093 x 26  
## playerID birthYear birthMonth birthDay birthCountry birthState birthCity   
## <chr> <int> <int> <int> <chr> <chr> <chr>   
## 1 aardsda01 1981 12 27 USA CO Denver   
## 2 aaronha01 1934 2 5 USA AL Mobile   
## 3 aaronto01 1939 8 5 USA AL Mobile   
## 4 aasedo01 1954 9 8 USA CA Orange   
## 5 abadan01 1972 8 25 USA FL Palm Beach   
## 6 abadfe01 1985 12 17 D.R. La Romana La Romana   
## 7 abadijo01 1850 11 4 USA PA Philadelphia  
## 8 abbated01 1877 4 15 USA PA Latrobe   
## 9 abbeybe01 1869 11 11 USA VT Essex   
## 10 abbeych01 1866 10 14 USA NE Falls City   
## # ... with 20,083 more rows, and 19 more variables: deathYear <int>,  
## # deathMonth <int>, deathDay <int>, deathCountry <chr>, deathState <chr>,  
## # deathCity <chr>, nameFirst <chr>, nameLast <chr>, nameGiven <chr>,  
## # weight <int>, height <int>, bats <fct>, throws <fct>, debut <chr>,  
## # finalGame <chr>, retroID <chr>, bbrefID <chr>, deathDate <date>,  
## # birthDate <date>

### **Question 5**

Use the correct join or bind function to create a combined table of the names and statistics of the top 10 home run (HR) hitters for 2016. This table should have the player ID, first name, last name, and number of HR for the top 10 players. Name this data frame top\_names.

Identify the join or bind that fills the blank in this code to create the correct table:

top\_names <- top %>% left\_join(Master, by="playerID") %>%  
 select(playerID, nameFirst, nameLast, HR)  
top\_names

## playerID nameFirst nameLast HR  
## 1 trumbma01 Mark Trumbo 47  
## 2 cruzne02 Nelson Cruz 43  
## 3 daviskh01 Khris Davis 42  
## 4 doziebr01 Brian Dozier 42  
## 5 encared01 Edwin Encarnacion 42  
## 6 arenano01 Nolan Arenado 41  
## 7 cartech02 Chris Carter 41  
## 8 frazito01 Todd Frazier 40  
## 9 bryankr01 Kris Bryant 39  
## 10 canoro01 Robinson Cano 39

Which bind or join function fills the blank to generate the correct table?

**Answer:**

left\_join(Master)

### **Question 6**

Inspect the Salaries data frame. Filter this data frame to the 2016 salaries, then use the correct bind join function to add a salary column to the top\_names data frame from the previous question. Name the new data frame top\_salary. Use this code framework:

top\_salary <- Salaries %>% filter(yearID == 2016) %>%  
 right\_join(top\_names) %>%  
 select(nameFirst, nameLast, teamID, HR, salary)

## Joining, by = "playerID"

Which bind or join function fills the blank to generate the correct table?

**Answer:**

right\_join(Master)

### **Question 7**

Inspect the AwardsPlayers table. Filter awards to include only the year 2016.

How many players from the top 10 home run hitters won at least one award in 2016?

AwardsPlayers %>% filter(yearID == 2016) %>% semi\_join(top\_names, by="playerID") %>% distinct(playerID) %>% count()

## n  
## 1 3

How many players won an award in 2016 but were not one of the top 10 home run hitters in 2016?

AwardsPlayers %>% filter(yearID == 2016) %>% anti\_join(top\_names, by="playerID") %>% distinct(playerID) %>% count()

## n  
## 1 44

# Assessment: Web Scraping

### **Introduction: Questions 1-3**

Load the following web page, which contains information about Major League Baseball payrolls, into R: <https://web.archive.org/web/20181024132313/http://www.stevetheump.com/Payrolls.htm>

library(rvest)

## Warning: package 'rvest' was built under R version 4.1.2

##   
## Attaching package: 'rvest'

## The following object is masked from 'package:readr':  
##   
## guess\_encoding

url <- "https://web.archive.org/web/20181024132313/http://www.stevetheump.com/Payrolls.htm"  
h <- read\_html(url)

We learned that tables in html are associated with the table node. Use the html\_nodes() function and the table node type to extract the first table. Store it in an object nodes:

nodes <- html\_nodes(h, "table")

The html\_nodes() function returns a list of objects of class xml\_node. We can see the content of each one using, for example, the html\_text() function. You can see the content for an arbitrarily picked component like this:

html\_text(nodes[[8]])

## [1] "Team\nPayroll\nAverge\nMedianNew York Yankees\n$ 197,962,289\n$ 6,186,321\n$ 1,937,500Philadelphia Phillies\n$ 174,538,938\n$ 5,817,964\n$ 1,875,000Boston Red Sox\n$ 173,186,617\n$ 5,093,724\n$ 1,556,250Los Angeles Angels\n$ 154,485,166\n$ 5,327,074\n$ 3,150,000Detroit Tigers\n$ 132,300,000\n$ 4,562,068\n$ 1,100,000Texas Rangers\n$ 120,510,974\n$ 4,635,037\n$ 3,437,500Miami Marlins\n$ 118,078,000\n$ 4,373,259\n$ 1,500,000San Francisco Giants\n$ 117,620,683\n$ 3,920,689\n$ 1,275,000St. Louis Cardinals\n$ 110,300,862\n$ 3,939,316\n$ 800,000Milwaukee Brewers\n$ 97,653,944\n$ 3,755,920\n$ 1,981,250Chicago White Sox\n$ 96,919,500\n$ 3,876,780\n$ 530,000Los Angeles Dodgers\n$ 95,143,575\n$ 3,171,452\n$ 875,000Minnesota Twins\n$ 94,085,000\n$ 3,484,629\n$ 750,000New York Mets\n$ 93,353,983\n$ 3,457,554\n$ 875,000Chicago Cubs\n$ 88,197,033\n$ 3,392,193\n$ 1,262,500Atlanta Braves\n$ 83,309,942\n$ 2,776,998\n$ 577,500Cincinnati Reds\n$ 82,203,616\n$ 2,935,843\n$ 1,150,000Seattle Mariners\n$ 81,978,100\n$ 2,927,789\n$ 495,150Baltimore Orioles\n$ 81,428,999\n$ 2,807,896\n$ 1,300,000Washington Nationals\n$ 81,336,143\n$ 2,623,746\n$ 800,000Cleveland Indians\n$ 78,430,300\n$ 2,704,493\n$ 800,000Colorado Rockies\n$ 78,069,571\n$ 2,692,054\n$ 482,000Toronto Blue Jays\n$ 75,489,200\n$ 2,696,042\n$ 1,768,750Arizona Diamondbacks\n$ 74,284,833\n$ 2,653,029\n$ 1,625,000Tampa Bay Rays\n$ 64,173,500\n$ 2,291,910\n$ 1,425,000Pittsburgh Pirates\n$ 63,431,999\n$ 2,187,310\n$ 916,666Kansas City Royals\n$ 60,916,225\n$ 2,030,540\n$ 870,000Houston Astros\n$ 60,651,000\n$ 2,332,730\n$ 491,250Oakland Athletics\n$ 55,372,500\n$ 1,845,750\n$ 487,500San Diego Padres\n$ 55,244,700\n$ 1,973,025\n$ 1,207,500"

If the content of this object is an html table, we can use the html\_table() function to convert it to a data frame:

html\_table(nodes[[8]])

## # A tibble: 30 x 4  
## Team Payroll Averge Median   
## <chr> <chr> <chr> <chr>   
## 1 New York Yankees $ 197,962,289 $ 6,186,321 $ 1,937,500  
## 2 Philadelphia Phillies $ 174,538,938 $ 5,817,964 $ 1,875,000  
## 3 Boston Red Sox $ 173,186,617 $ 5,093,724 $ 1,556,250  
## 4 Los Angeles Angels $ 154,485,166 $ 5,327,074 $ 3,150,000  
## 5 Detroit Tigers $ 132,300,000 $ 4,562,068 $ 1,100,000  
## 6 Texas Rangers $ 120,510,974 $ 4,635,037 $ 3,437,500  
## 7 Miami Marlins $ 118,078,000 $ 4,373,259 $ 1,500,000  
## 8 San Francisco Giants $ 117,620,683 $ 3,920,689 $ 1,275,000  
## 9 St. Louis Cardinals $ 110,300,862 $ 3,939,316 $ 800,000   
## 10 Milwaukee Brewers $ 97,653,944 $ 3,755,920 $ 1,981,250  
## # ... with 20 more rows

### **Question 1**

Many tables on this page are team payroll tables, with columns for rank, team, and one or more money values.

Convert the first four tables in nodes to data frames and inspect them. (Note that “parsing errors” and/or “empty tables” still count towards the table index!)

Which of the first four nodes are tables of team payroll?

**Answer:**

*Table 2, Table 3 and Table 4*

html\_table(nodes[1:4])

## [[1]]  
## # A tibble: 1 x 2  
## X1 X2   
## <lgl> <chr>   
## 1 NA "Salary Stats 1967-2019\nTop ML Player Salaries / Baseball's Luxury Tax"  
##   
## [[2]]  
## # A tibble: 30 x 3  
## RANK TEAM Payroll   
## <int> <chr> <chr>   
## 1 1 Boston Red Sox $235.65M   
## 2 2 San Francisco Giants $208.51M   
## 3 3 Los Angeles Dodgers $186.14M   
## 4 4 Chicago Cubs $183.46M   
## 5 5 Washington Nationals $181.59M   
## 6 6 Los Angeles Angels $175.1M   
## 7 7 New York Yankees $168.54M   
## 8 8 Seattle Mariners $162.48M   
## 9 9 Toronto Blue Jays $162.316M  
## 10 10 St. Louis Cardinals $161.01M   
## # ... with 20 more rows  
##   
## [[3]]  
## # A tibble: 31 x 5  
## X1 X2 X3 X4 X5   
## <chr> <chr> <chr> <chr> <chr>   
## 1 Rank Team 25 Man Disabled List Total Payroll  
## 2 1 Los Angeles Dodgers $155,887,854 $37,354,166 $242,065,828   
## 3 2 New York Yankees $168,045,699 $5,644,000 $201,539,699   
## 4 3 Boston Red Sox $136,780,500 $38,239,250 $199,805,178   
## 5 4 Detroit Tigers $168,500,600 $11,750,000 $199,750,600   
## 6 5 Toronto Blue Jays $159,175,968 $2,169,400 $177,795,368   
## 7 6 Texas Rangers $115,162,703 $39,136,360 $175,909,063   
## 8 7 San Francisco Giants $169,504,611 $2,500,000 $172,354,611   
## 9 8 Chicago Cubs $170,189,880 $2,000,000 $172,189,880   
## 10 9 Washington Nationals $163,111,918 $535,000 $167,846,918   
## # ... with 21 more rows  
##   
## [[4]]  
## # A tibble: 30 x 5  
## Rank Team `Opening Day` `Avg Salary` Median   
## <int> <chr> <chr> <chr> <chr>   
## 1 1 Dodgers $ 223,352,402 $ 7,445,080 $ 5,166,666  
## 2 2 Yankees $ 213,472,857 $ 7,361,133 $ 3,300,000  
## 3 3 Red Sox $ 182,161,414 $ 6,072,047 $ 3,500,000  
## 4 4 Tigers $ 172,282,250 $ 6,891,290 $ 3,000,000  
## 5 5 Giants $ 166,495,942 $ 5,946,284 $ 4,000,000  
## 6 6 Nationals $ 166,010,977 $ 5,724,516 $ 2,500,000  
## 7 7 Angels $ 146,449,583 $ 5,049,986 $ 1,312,500  
## 8 8 Rangers $ 144,307,373 $ 4,509,605 $ 937,500   
## 9 9 Phillies $ 133,048,000 $ 4,434,933 $ 700,000   
## 10 10 Blue Jays $ 126,369,628 $ 4,357,573 $ 1,650,000  
## # ... with 20 more rows

### **Question 2**

For the last 3 components of nodes, which of the following are true? (Check all correct answers.)

**Answer:**

*All three entries are tables.*  *The last entry shows the average across all teams through time, not payroll per team.*

html\_table(tail(nodes,n=3))

## [[1]]  
## # A tibble: 31 x 3  
## X1 X2 X3   
## <chr> <chr> <chr>   
## 1 Team Payroll Average   
## 2 NY Yankees $109,791,893 $3,541,674  
## 3 Boston $109,558,908 $3,423,716  
## 4 Los Angeles $108,980,952 $3,757,964  
## 5 NY Mets $93,174,428 $3,327,658  
## 6 Cleveland $91,974,979 $3,065,833  
## 7 Atlanta $91,851,687 $2,962,958  
## 8 Texas $88,504,421 $2,854,981  
## 9 Arizona $81,206,513 $2,900,233  
## 10 St. Louis $77,270,855 $2,664,512  
## # ... with 21 more rows  
##   
## [[2]]  
## # A tibble: 31 x 3  
## X1 X2 X3   
## <chr> <chr> <chr>   
## 1 Team Payroll Average   
## 2 NY Yankees $92,538,260 $3,190,974  
## 3 Los Angeles $88,124,286 $3,263,862  
## 4 Atlanta $84,537,836 $2,817,928  
## 5 Baltimore $81,447,435 $2,808,532  
## 6 Arizona $81,027,833 $2,893,851  
## 7 NY Mets $79,509,776 $3,180,391  
## 8 Boston $77,940,333 $2,598,011  
## 9 Cleveland $75,880,871 $2,918,495  
## 10 Texas $70,795,921 $2,722,920  
## # ... with 21 more rows  
##   
## [[3]]  
## # A tibble: 54 x 4  
## X1 X2 X3 X4   
## <chr> <chr> <chr> <chr>   
## 1 Year Minimum "Average" "% Chg"  
## 2 2019 $555,000 "" "-"   
## 3 2018 $545,000 "$4,520,000" ""   
## 4 2017 $535,000 "$4,470,000" "5.4"   
## 5 2016 $507,500 "$4,400,000" "-"   
## 6 2015 $507,500 "$4,250,000" "-"   
## 7 2014 $507,500 "$3,820,000" "12.8"   
## 8 2013 $480,000 "$3,386,212" "5.4"   
## 9 2012 $480,000 "$3,440,000" "3.8"   
## 10 2011 $414,500 "$3,305,393" "0.2"   
## # ... with 44 more rows

### **Question 3**

Create a table called tab\_1 using entry 10 of nodes. Create a table called tab\_2 using entry 19 of nodes.

Note that the column names should be c("Team", "Payroll", "Average"). You can see that these column names are actually in the first data row of each table, and that tab\_1 has an extra first column No. that should be removed so that the column names for both tables match.

Remove the extra column in tab\_1, remove the first row of each dataset, and change the column names for each table to c("Team", "Payroll", "Average"). Use a full\_join() by the Team to combine these two tables.

How many rows are in the joined data table?

**Answer:** *58*

tab\_1 <- html\_table(nodes[[10]],header = TRUE)[,-1]   
tab\_2 <- html\_table(nodes[[19]],header = TRUE)  
tab\_1 %>% full\_join(tab\_2,"Team")

## # A tibble: 58 x 5  
## Team Payroll.x Average.x Payroll.y Average.y   
## <chr> <chr> <chr> <chr> <chr>   
## 1 New York Yankees $206,333,389 $8,253,336 <NA> <NA>   
## 2 Boston Red Sox $162,747,333 $5,611,977 <NA> <NA>   
## 3 Chicago Cubs $146,859,000 $5,439,222 $64,015,833 $2,462,147  
## 4 Philadelphia Phillies $141,927,381 $5,068,835 <NA> <NA>   
## 5 New York Mets $132,701,445 $5,103,902 <NA> <NA>   
## 6 Detroit Tigers $122,864,929 $4,550,553 <NA> <NA>   
## 7 Chicago White Sox $108,273,197 $4,164,354 $62,363,000 $2,309,741  
## 8 Los Angeles Angels $105,013,667 $3,621,161 <NA> <NA>   
## 9 Seattle Mariners $98,376,667 $3,513,452 <NA> <NA>   
## 10 San Francisco Giants $97,828,833 $3,493,887 <NA> <NA>   
## # ... with 48 more rows

### **Introduction: Questions 4 and 5**

The Wikipedia page on [opinion polling for the Brexit referendum](https://en.wikipedia.org/w/index.php?title=Opinion_polling_for_the_United_Kingdom_European_Union_membership_referendum&oldid=896735054) , in which the United Kingdom voted to leave the European Union in June 2016, contains several tables. One table contains the results of all polls regarding the referendum over 2016:

Use the **rvest** library to read the HTML from this Wikipedia page (make sure to copy both lines of the URL):

library(rvest)  
library(tidyverse)  
url <- "https://en.wikipedia.org/w/index.php?title=Opinion\_polling\_for\_the\_United\_Kingdom\_European\_Union\_membership\_referendum&oldid=896735054"

### **Question 4**

Assign tab to be the html nodes of the “table” class.

How many tables are in this Wikipedia page?

**Answer:** *41*

h <- read\_html(url)  
tab <- html\_nodes(h, "table")  
tab

## {xml\_nodeset (42)}  
## [1] <table class="sidebar sidebar-collapse nomobile vcard"><tbody>\n<tr><td ...  
## [2] <table style="width:100%;border-collapse:collapse;border-spacing:0px 0px ...  
## [3] <table style="width:100%;border-collapse:collapse;border-spacing:0px 0px ...  
## [4] <table class="wikitable sortable" style="text-align:center;font-size:95% ...  
## [5] <table class="plainlinks metadata ambox mbox-small-left ambox-notice" ro ...  
## [6] <table class="wikitable sortable" style="text-align:center;font-size:95% ...  
## [7] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [8] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [9] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [10] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [11] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [12] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [13] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [14] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [15] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [16] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [17] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## [18] <table class="box-More\_citations\_needed\_section plainlinks metadata ambo ...  
## [19] <table class="wikitable sortable collapsible" style="text-align:center; ...  
## [20] <table class="wikitable sortable collapsible" style="text-align:center;f ...  
## ...

### **Question 5**

Inspect the first several html tables using html\_table() with the argument fill=TRUE (you can read about this argument in the documentation). Find the first table that has 9 columns with the first column named “Date(s) conducted”.

What is the first table number to have 9 columns where the first column is named “Date(s) conducted”?

**Answer:** *6*

html\_table(tab[[6]], fill = TRUE)

## # A tibble: 134 x 9  
## `Date(s) conduct~ Remain Leave Undecided Lead Sample `Conducted by`   
## <chr> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 Date(s) conducted "" "" Undecided Lead Sample Conducted by   
## 2 23 June 2016 "48.1%" "51.~ N/A 3.8% 33,57~ Results of the United~  
## 3 23 June "52%" "48%" N/A 4% 4,772 YouGov   
## 4 22 June "55%" "45%" N/A 10% 4,700 Populus   
## 5 20–22 June "51%" "49%" N/A 2% 3,766 YouGov   
## 6 20–22 June "49%" "46%" 1% 3% 1,592 Ipsos MORI   
## 7 20–22 June "44%" "45%" 9% 1% 3,011 Opinium   
## 8 17–22 June "54%" "46%" N/A 8% 1,032 ComRes   
## 9 17–22 June "48%" "42%" 11% 6% 1,032 ComRes   
## 10 16–22 June "41%" "43%" 16% 2% 2,320 TNS   
## # ... with 124 more rows, and 2 more variables: Polling type <chr>, Notes <chr>

# Assessment: String Processing Part 1

### **Question 1**

Which of the following is NOT an application of string parsing?

**Answer:** *Formatting numbers and characters so they can easily be displayed in deliverables like papers and presentations.*

### **Question 2**

Which of the following commands would not give you an error in R?

**Answer:** *cat(" LeBron James is 6’8" ")*

### **Question 3**

Which of the following are advantages of the stringr package over string processing functions in base R? Select all that apply.

**Answer:** *Functions in stringr all start with “str\_”, which makes them easy to look up using autocomplete.* *Stringr functions work better with pipes.* *The order of arguments is more consistent in stringr functions than in base R.*

### **Question 4**

You have a data frame of monthly sales and profits in R:

> head(dat)  
# A tibble: 5 x 3  
Month Sales Profit   
<chr> <chr> <chr>   
January $128,568 $16,234  
February $109,523 $12,876  
March $115,468 $17,920  
April $122,274 $15,825  
May $117,921 $15,437

Which of the following commands could convert the sales and profits columns to numeric? Select all that apply.

**Answer:** *dat %>% mutate\_at(2:3, parse\_number)* *dat %>% mutate\_at(2:3, funs(str\_replace\_all(., c(“\$|,”), ""))) %>% mutate\_at(2:3, as.numeric)*

# Assessment: String Processing Part 2

### **Question 1**

In the video, we use the function not\_inches to identify heights that were incorrectly entered

not\_inches <- function(x, smallest = 50, tallest = 84) {  
 inches <- suppressWarnings(as.numeric(x))  
 ind <- is.na(inches) | inches < smallest | inches > tallest   
 ind  
}

In this function, what TWO types of values are identified as not being correctly formatted in inches?

**Answer:** *Values that result in NA’s when converted to numeric* *Values less than 50 inches or greater than 84 inches*

### **Question 2**

Which of the following arguments, when passed to the function not\_inches(), would return the vector c(FALSE)?

**Answer:** *c(70)*

not\_inches(c(70))

## [1] FALSE

### **Question 3**

Our function not\_inches() returns the object ind. Which answer correctly describes ind?

**Answer:**

*ind is a logical vector of TRUE and FALSE, equal in length to the vector x (in the arguments list). TRUE indicates that a height entry is incorrectly formatted.*

### **Question 4**

Given the following code

> s  
[1] "70" "5 ft" "4'11" "" "." "Six feet"

What pattern vector yields the following result?

str\_view\_all(s, pattern)  
70  
5 ft  
4’11  
.  
Six feet

**Answer:**

*pattern <- “\d|ft”*

### **Question 5**

You enter the following set of commands into your R console. What is your printed result?

**Answer:** *TRUE TRUE TRUE FALSE*

animals <- c("cat", "puppy", "Moose", "MONKEY")  
pattern <- "[a-z]"  
str\_detect(animals, pattern)

## [1] TRUE TRUE TRUE FALSE

### **Question 6**

You enter the following set of commands into your R console. What is your printed result?

**Answer:** *FALSE FALSE FALSE TRUE*

animals <- c("cat", "puppy", "Moose", "MONKEY")  
pattern <- "[A-Z]$"  
str\_detect(animals, pattern)

## [1] FALSE FALSE FALSE TRUE

### **Question 7**

You enter the following set of commands into your R console. What is your printed result?

**Answer:** *FALSE FALSE FALSE TRUE* 4 or 5 lowercase letters in a row

animals <- c("cat", "puppy", "Moose","MONKEY")  
pattern <- "[a-z]{4,5}"  
str\_detect(animals, pattern)

## [1] FALSE TRUE TRUE FALSE

### **Question 8**

Given the following code:

animals <- c("moose", "monkey", "meerkat", "mountain lion")

Which TWO “pattern” vectors would yield the following result?

> str\_detect(animals, pattern)  
[1] TRUE TRUE TRUE TRUE

**Answer:**

pattern <- "mo\*"  
pattern <- "mo?"  
str\_detect(animals, pattern)

## [1] TRUE TRUE TRUE TRUE

### **Question 9**

You are working on some data from different universities. You have the following vector:

schools  
[1] "U. Kentucky" "Univ New Hampshire" "Univ. of Massachusetts" "University Georgia"   
[5] "U California" "California State University"

You want to clean this data to match the full names of each university:

final  
[1] "University of Kentucky" "University of New Hampshire" "University of Massachusetts" "University of Georgia"   
[5] "University of California" "California State University"

What of the following commands could accomplish this?

**Answer:**

schools %>%   
 str\_replace("^Univ\\.?\\s|^U\\.?\\s", "University ") %>%   
 str\_replace("^University of |^University ", "University of ")

### **Question 10**

Rather than using the pattern\_with\_groups vector from the video, you accidentally write in the following code:

problems <- c("5.3", "5,5", "6 1", "5 .11", "5, 12")  
pattern\_with\_groups <- "^([4-7])[,\\.](\\d\*)$"  
str\_replace(problems, pattern\_with\_groups, "\\1'\\2")

## [1] "5'3" "5'5" "6 1" "5 .11" "5, 12"

What is your result?

**Answer:** *[1] “5’3” “5’5” “6 1” “5 .11” “5, 12”*

### **Question 11**

You notice your mistake and correct your pattern regex to the following

problems <- c("5.3", "5,5", "6 1", "5 .11", "5, 12")  
pattern\_with\_groups <- "^([4-7])[,\\.\\s](\\d\*)$"  
str\_replace(problems, pattern\_with\_groups, "\\1'\\2")

## [1] "5'3" "5'5" "6'1" "5 .11" "5, 12"

What is your result?

**Answer:** *[1] “5’3” “5’5” “6’1” “5 .11” “5, 12”*

### **Question 12**

In our example, we use the following code to detect height entries that do not match our pattern of x’y”:

converted <- problems %>%   
 str\_replace("feet|foot|ft", "'") %>%   
 str\_replace("inches|in|''|\"", "") %>%   
 str\_replace("^([4-7])\\s\*[,\\.\\s+]\\s\*(\\d\*)$", "\\1'\\2")  
  
pattern <- "^[4-7]\\s\*'\\s\*\\d{1,2}$"  
index <- str\_detect(converted, pattern)  
converted[!index]

## character(0)

Which answer best describes the differences between the regex string we use as an argument in str\_replace("^([4-7])\\s\*[,\\.\\s+]\\s\*(\\d\*)$", "\\1'\\2") and the regex string in pattern <- "^[4-7]\\s\*'\\s\*\\d{1,2}$"?

**Answer:** *The regex used in str\_replace() looks for either a comma, period or space between the feet and inches digits, while the pattern regex just looks for an apostrophe; the regex in str\_replace allows for none or more digits to be entered as inches, while the pattern regex only allows for one or two dig*

### **Question 13**

You notice a few entries that are not being properly converted using your str\_replace() and str\_detect() code:

yes <- c("5 feet 7inches", "5 7")  
no <- c("5ft 9 inches", "5 ft 9 inches")  
s <- c(yes, no)  
  
converted <- s %>%   
 str\_replace("feet|foot|ft", "'") %>%   
 str\_replace("inches|in|''|\"", "") %>%   
 str\_replace("^([4-7])\\s\*[,\\.\\s+]\\s\*(\\d\*)$", "\\1'\\2")  
  
pattern <- "^[4-7]\\s\*'\\s\*\\d{1,2}$"  
str\_detect(converted, pattern)

## [1] TRUE TRUE FALSE FALSE

It seems like the problem may be due to spaces around the words feet|foot|ft and inches|in. What is another way you could fix this problem?

**Answer:**

converted <- s %>%   
 str\_replace("\\s\*(feet|foot|ft)\\s\*", "'") %>%   
 str\_replace("\\s\*(inches|in|''|\")\\s\*", "") %>%   
 str\_replace("^([4-7])\\s\*[,\\.\\s+]\\s\*(\\d\*)$", "\\1'\\2")

# Assessment Part 1: String Processing Part 3

### **Question 1**

You have the following table, schedule:

day <- c("monday","tuesday")  
staff <- c("Mandy, Chris and Laura", "Steve, Ruth and Frank")  
  
schedule <- data.frame(day,staff)

>schedule  
day staff  
Monday Mandy, Chris and Laura  
Tuesday Steve, Ruth and Frank

You want to turn this into a more useful data frame.

Which two commands would properly split the text in the “staff” column into each individual name? Select ALL that apply.

**Answer:**

str\_split(schedule$staff, ",\\s|\\sand\\s")

## [[1]]  
## [1] "Mandy" "Chris" "Laura"  
##   
## [[2]]  
## [1] "Steve" "Ruth" "Frank"

str\_split(schedule$staff, ", | and ")

## [[1]]  
## [1] "Mandy" "Chris" "Laura"  
##   
## [[2]]  
## [1] "Steve" "Ruth" "Frank"

### **Question 2**

You have the following table, schedule:

> schedule  
day staff  
Monday Mandy, Chris and Laura  
Tuesday Steve, Ruth and Frank

What code would successfully turn your “Schedule” table into the following tidy table?

> tidy  
day staff  
<chr> <chr>  
Monday Mandy  
Monday Chris  
Monday Laura  
Tuesday Steve  
Tuesday Ruth   
Tuesday Frank

**Answer:**

tidy <- schedule %>%   
 mutate(staff = str\_split(staff, ", | and ")) %>%   
 unnest()

## Warning: `cols` is now required when using unnest().  
## Please use `cols = c(staff)`

### **Question 3**

Using the gapminder data, you want to recode countries longer than 12 letters in the region “Middle Africa” to their abbreviations in a new column, “country\_short”. Which code would accomplish this?

**Answer:**

library(dslabs)  
data("gapminder")  
 dat <- gapminder %>% filter(region == "Middle Africa") %>%   
 mutate(country\_short = recode(country,   
 "Central African Republic" = "CAR",   
 "Congo, Dem. Rep." = "DRC",  
 "Equatorial Guinea" = "Eq. Guinea"))

# Assessment Part 2: String Processing Part 3

Import raw Brexit referendum polling data from Wikipedia:

library(rvest)  
library(tidyverse)  
library(stringr)  
url <- "https://en.wikipedia.org/w/index.php?title=Opinion\_polling\_for\_the\_United\_Kingdom\_European\_Union\_membership\_referendum&oldid=896735054"  
tab <- read\_html(url) %>% html\_nodes("table")  
polls <- tab[[6]] %>% html\_table(fill = TRUE)

You will use a variety of string processing techniques learned in this section to reformat these data.

### **Question 4**

Some rows in this table do not contain polls. You can identify these by the lack of the percent sign (%) in the Remain column.

Update polls by changing the column names to c("dates", "remain", "leave", "undecided", "lead", "samplesize", "pollster", "poll\_type", "notes") and only keeping rows that have a percent sign (%) in the remain column.

How many rows remain in the polls data frame?

**Answer:**

polls <- polls[-1,] %>% setNames(c("dates", "remain", "leave", "undecided", "lead", "samplesize", "pollster", "poll\_type", "notes")) %>% filter(str\_detect(.$remain, "\\%"))  
  
polls %>% nrow()

## [1] 129

### **Question 5**

The remain and leave columns are both given in the format “48.1%”: percentages out of 100% with a percent symbol.

Which of these commands converts the remain vector to a proportion between 0 and 1?

**Answer:**

as.numeric(str\_replace(polls$remain, "%", ""))/100

## [1] 0.481 0.520 0.550 0.510 0.490 0.440 0.540 0.480 0.410 0.450 0.420 0.530  
## [13] 0.450 0.440 0.440 0.420 0.420 0.370 0.460 0.430 0.390 0.450 0.440 0.460  
## [25] 0.400 0.480 0.530 0.420 0.440 0.450 0.430 0.430 0.480 0.410 0.430 0.400  
## [37] 0.410 0.420 0.440 0.510 0.440 0.440 0.410 0.410 0.450 0.550 0.440 0.440  
## [49] 0.520 0.550 0.470 0.430 0.550 0.380 0.360 0.380 0.440 0.420 0.440 0.430  
## [61] 0.420 0.490 0.390 0.410 0.450 0.430 0.440 0.510 0.510 0.490 0.480 0.430  
## [73] 0.530 0.380 0.400 0.390 0.350 0.450 0.420 0.400 0.390 0.440 0.510 0.390  
## [85] 0.350 0.410 0.510 0.450 0.490 0.400 0.480 0.410 0.460 0.470 0.430 0.450  
## [97] 0.480 0.490 0.400 0.400 0.400 0.390 0.410 0.390 0.480 0.480 0.370 0.380  
## [109] 0.420 0.510 0.450 0.400 0.540 0.360 0.430 0.490 0.410 0.360 0.420 0.380  
## [121] 0.550 0.440 0.540 0.410 0.520 0.420 0.380 0.420 0.440

parse\_number(polls$remain)/100

## [1] 0.481 0.520 0.550 0.510 0.490 0.440 0.540 0.480 0.410 0.450 0.420 0.530  
## [13] 0.450 0.440 0.440 0.420 0.420 0.370 0.460 0.430 0.390 0.450 0.440 0.460  
## [25] 0.400 0.480 0.530 0.420 0.440 0.450 0.430 0.430 0.480 0.410 0.430 0.400  
## [37] 0.410 0.420 0.440 0.510 0.440 0.440 0.410 0.410 0.450 0.550 0.440 0.440  
## [49] 0.520 0.550 0.470 0.430 0.550 0.380 0.360 0.380 0.440 0.420 0.440 0.430  
## [61] 0.420 0.490 0.390 0.410 0.450 0.430 0.440 0.510 0.510 0.490 0.480 0.430  
## [73] 0.530 0.380 0.400 0.390 0.350 0.450 0.420 0.400 0.390 0.440 0.510 0.390  
## [85] 0.350 0.410 0.510 0.450 0.490 0.400 0.480 0.410 0.460 0.470 0.430 0.450  
## [97] 0.480 0.490 0.400 0.400 0.400 0.390 0.410 0.390 0.480 0.480 0.370 0.380  
## [109] 0.420 0.510 0.450 0.400 0.540 0.360 0.430 0.490 0.410 0.360 0.420 0.380  
## [121] 0.550 0.440 0.540 0.410 0.520 0.420 0.380 0.420 0.440

### **Question 6**

The undecided column has some “N/A” values. These “N/A”s are only present when the remain and leave columns total 100%, so they should actually be zeros.

Use a function from **stringr** to convert “N/A” in the undecided column to 0. The format of your command should be function\_name(polls$undecided, "arg1", "arg2").

What function replaces function\_name? What argument replaces arg1? What argument replaces arg2?

**Answer:**

str\_replace(polls$undecided, "N/A", "0")

## [1] "0" "0" "0" "0" "1%" "9%" "0" "11%" "16%" "11%" "13%" "2%"   
## [13] "13%" "9%" "12%" "9%" "13%" "16%" "11%" "3%" "15%" "5%" "7%" "9%"   
## [25] "13%" "3%" "0" "11%" "13%" "0" "11%" "9%" "5%" "11%" "16%" "16%"  
## [37] "13%" "15%" "9%" "3%" "12%" "18%" "13%" "16%" "10%" "3%" "14%" "12%"  
## [49] "7%" "5%" "14%" "10%" "5%" "21%" "22%" "16%" "11%" "13%" "11%" "11%"  
## [61] "14%" "0" "26%" "13%" "17%" "13%" "10%" "6%" "9%" "8%" "11%" "13%"  
## [73] "6%" "28%" "16%" "17%" "30%" "17%" "12%" "16%" "18%" "13%" "5%" "18%"  
## [85] "30%" "14%" "0" "12%" "10%" "19%" "11%" "17%" "19%" "4%" "16%" "16%"  
## [97] "7%" "15%" "19%" "18%" "19%" "19%" "18%" "18%" "15%" "0" "25%" "25%"  
## [109] "17%" "10%" "23%" "19%" "10%" "25%" "18%" "10%" "17%" "19%" "19%" "20%"  
## [121] "9%" "14%" "10%" "18%" "0" "17%" "22%" "12%" "18%"

### **Question 7**

The dates column contains the range of dates over which the poll was conducted. The format is “8-10 Jan” where the poll had a start date of 2016-01-08 and end date of 2016-01-10. Some polls go across month boundaries (16 May-12 June).

The end date of the poll will always be one or two digits, followed by a space, followed by the month as one or more letters (either capital or lowercase). In these data, all month abbreviations or names have 3, 4 or 5 letters.

Write a regular expression to extract the end day and month from dates. Insert it into the skeleton code below:

**Answer:**

pattern\_opt <- c("\\d+\\s[a-zA-Z]+","[0-9]+\\s[a-zA-Z]+","\\d{1,2}\\s[a-zA-Z]+","\\d+\\s[a-zA-Z]{3,5}")  
temp <- str\_extract\_all(polls$dates, "\\d+\\s[a-zA-Z]+")  
end\_date <- sapply(temp, function(x) x[length(x)]) # take last element (handles polls that cross month boundaries)