Assignment 7 Data Wrangling

Yaniv Bronshtein

3/17/2021

Import the necessary libraries

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rvest)
```

```
## Loading required package: xml2

##
## Attaching package: 'rvest'

## The following object is masked from 'package:purrr':
##
##     pluck

## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
library(broom)
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 4.0.4

## Loading required package: lattice

## Loading required package: survival
## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following object is masked from 'package:rvest':
##
##     html

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.0.4
```

```
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'

## The following object is masked from 'package:purrr':
##
##     flatten
```

PROBLEM 1:

1.From the worldometer webpage https://www.worldometers.info/coronavirus/usa/new-jersey/ extract the county-wise COVID data (total cases, new cases, total deaths and new deaths). 2.Show a nice graphical representation of the county-wise total cases and total deaths in a single plot. Use your imagination and Chapter 3 of R for Data Science to come up with an appropriate visual representation. 3.Identify the top two counties reporting most new cases.

Part 1

```
url <- "https://www.worldometers.info/coronavirus/usa/new-york/"

worldometer_t <- url %>%
  read_html() %>%
  html_nodes("table") %>%
  html_table(fill = TRUE)
```

Use yesterday's coronavirus data for New York State since it is fully updated Perform the following steps:: a). Convert worldometer_t[[2]] which is yesterdays data to a tibble b). From this tibble extract only the County, TotalCases, NewCases, TotalDeaths, and NewDeaths c). Remove the rows that contain the string Total signifying state totals since we only want county-wide data

```r
county_covid_t <- worldometer_t[[2]] %>%
  as_tibble() %>%
  select(County, TotalCases, NewCases, TotalDeaths, NewDeaths) %>%
  filter(!str_detect(County ,"Total"))
```

Create a function clean_column that takes in the tibble data and the column to clean and returns a numeric vector of the cleaned up column

```r
clean_column <- function(data, col_name) {
  #Cast data as a tibble and extract the col_name as a vector
  col_vec <- as_tibble(data) %>%
    pull(col_name)

  #Check if the vector is numeric. If not, the data requires processing
  #All commas and + need to be removed, the vector needs to be converted to numeric,
  #and the NA's need to be replaced with 0's
  if (!all.is.numeric(col_vec)) {
    return(
      col_vec %>%
        str_replace_all("[,+]", "") %>%
        as.numeric() %>%
        replace_na(0)
    )
  }
  #If the data is already numeric, only the NA's need to be replaced with 0's
  else {
    return(
      col_vec %>%
        replace_na(0)
    )
  }
}
```

Call clean_column() on each column of county_covid_t

```r
total_cases <- clean_column(data = county_covid_t, col_name = "TotalCases")
new_cases <- clean_column(data = county_covid_t, col_name = "NewCases")
total_deaths <- clean_column(data = county_covid_t, col_name = "TotalDeaths")
new_deaths <- clean_column(data = county_covid_t, col_name = "NewDeaths")
```

Create a tibble from the TotalCases and TotalDeaths called cleaned_total_cases_deaths Add a calculated column PercentDeaths that shows the death rate

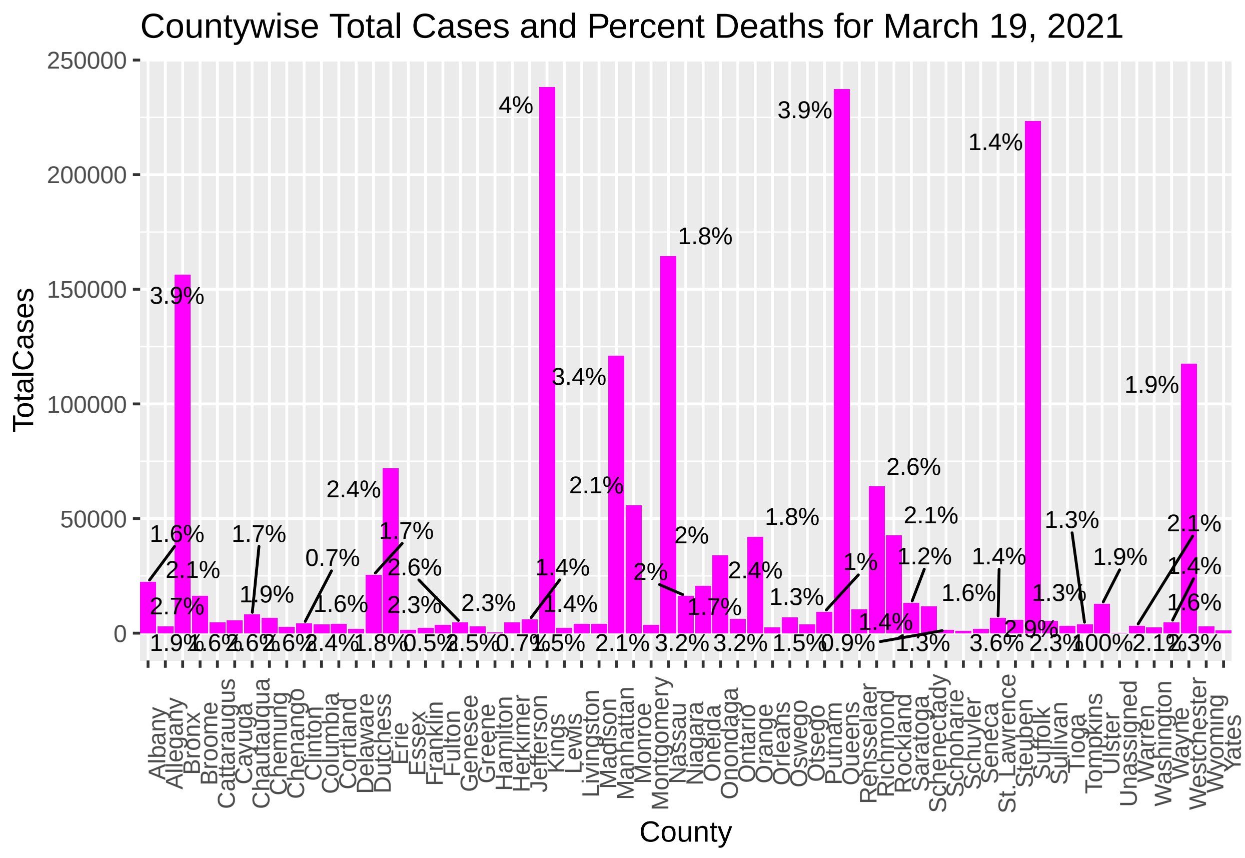
```r
cleaned_total_cases_deaths <-
  tibble(TotalCases = total_cases,
         TotalDeaths = total_deaths) %>%
  mutate(PercentDeaths =
           paste0(round((total_deaths / total_cases) * 100, 1), "%"))
```

Add county data to the cleaned__total__cases__deaths tibble and sort by county

```
cleaned_total_cases_deaths <-
  bind_cols(county_covid_t %>% select(County), cleaned_total_cases_deaths) %>%
  arrange(County)
```

Create a ggplot where the bars represent the total cases and the labels represent the death rate for each county Use geom__text__repel for the death rate adn title the x labels 90 degrees to prevent overlap

```
ggplot(data = cleaned_total_cases_deaths,
       mapping = aes(x = County, y = TotalCases)) +
  geom_bar(stat = "identity", fill = "magenta") +
  geom_text_repel(aes(label = PercentDeaths), max.overlaps = 25, size = 3.0) +
  theme(axis.text.x = element_text(angle = 90, hjust = 0.5)) +
  scale_x_discrete(expand = expansion(mult = 0.0)) +
  labs(title = "Countywise Total Cases and Percent Deaths for March 19, 2021")
```



3.Identify the top two counties reporting most new cases.

```
top_2_newcases_counties <-
  tibble(county_covid_t %>% select(County), new_cases) %>%
  arrange(desc(new_cases)) %>%
  head(2)

top_2_newcases_counties
```

```
## # A tibble: 2 x 2
##   County new_cases
##   <chr>      <dbl>
## 1 Kings       1425
## 2 Queens      1276
```

PROBLEM 2:

Obtain your free API for https://spoonacular.com/food-api Use it to obtain a list of 10 recipes that have carbohydrates not exceeding 30 grams. Present your output as a 10x3 tibble, where the column names are "Recipe" (the title of the recipe), "Carbs" (the carb content) and "ID" (the ID of the recipe) Find 10 types of Riesling wines whose prices do not exceed $50 and present your results as a 10x3 tibble, where the columns represent the title of the wine, its ID and its price.

Define the API key to be used for both parts

```
api_key <-"87c3b12b27b04583b7fe359c409b8fd1"
```

Construct the url to be used for the recipes

```
base_url_recipe <- "https://api.spoonacular.com/recipes/findByNutrients?"
query_recipe <- "maxCarbs=30&number=10&apiKey="
url_recipe <- paste0(base_url_recipe, query_recipe, api_key)
```

Perform the API call, save the results into a tibble,extracting only the necessary columns. Rename the columns to match the names in the problem statement

```
json_result_recipe <- url_recipe %>%
  fromJSON()

recipes_t <- json_result_recipe %>%
  as_tibble() %>%
  select(title, carbs, id) %>%
  rename(Recipe = title, Carbs = carbs, ID = id)

recipes_t
```

```
## # A tibble: 10 x 3
##    Recipe                                                      Carbs     ID
##    <chr>                                                       <chr>  <int>
##  1 Anticuchos Of White Seabass With Aji Chile Honey Marinade & Sem~ 0g    632426
##  2 Banana Oatmeal Breakfast Muffins                            27g   634141
##  3 Gluten Free Dairy Free Sugar Free Chinese Chicken Salad     27g   644826
##  4 Lychee Granita                                              21g   650499
##  5 Nori Seaweed Muffins                                        27g   653270
##  6 Pumpkin Pie Smoothie                                        22g   657359
##  7 Simple Sage Pesto                                           2g    660130
##  8 Slow Cooked Applesauce                                      16g   660261
##  9 Trinidadian Chicken Potato Curry                            20g   663824
## 10 Spicy Indian-Style Hummus                                   15g   716195
```

Construct the url to be used for the recipes

```
base_url_wine <-"https://api.spoonacular.com/food/wine/recommendation?"
query_wine <- "wine=Riesling&maxPrice=50&number=10&apiKey="
url_wine <- paste0(base_url_wine, query_wine,api_key)
```

Perform the API call

```
json_result_wine <- url_wine %>%
  fromJSON()
```

Given the more complex output, extract the respective fields directly from the the result of fromJSON(). Use successive $ to go deeper Create a tibble from the fields requested in the problem statement

```
cleaned_wines_t <- tibble(Wine = json_result_wine$recommendedWines$title,
                          ID = json_result_wine$recommendedWines$id,
                          Price = json_result_wine$recommendedWines$price)



cleaned_wines_t
```

```
## # A tibble: 10 x 3
##    Wine                                                           ID Price
##    <chr>                                                       <int> <chr>
##  1 Schloss Johannisberg Rotlack Riesling Kabinett Feinherb    492948 $39.99
##  2 Bollig-Lehnert Piesporter Goldtroopfchen Riesling Spatlese 474522 $23.99
##  3 J.J. Prum Graacher Himmelreich Kabinett Riesling           437965 $21.99
##  4 Alsace Willm Cuvee Emile Willm Riesling                    496530 $22.99
##  5 J.J. Christoffel Erdener Treppchen Riesling Spatlese       461360 $26.99
##  6 Ostertag Les Jardins Riesling                              482286 $26.99
##  7 Eroica Riesling                                            434655 $23.99
##  8 Fritz Haag Brauneberger Juffer Spatlese Riesling           446064 $29.99
##  9 J.J. Prum Wehlener Sonnenuhr Auslese Riesling              438482 $39.99
## 10 Montinore Estate Almost Dry Riesling                       495768 $15.99
```