```
---
output:
  html_document: default
  pdf_document: default
---
# Intro to Data Science - HW 6
##### Copyright Jeffrey Stanton, Jeffrey Saltz, and Jasmina Tacheva


```{r}
# Enter your name here: Victoria Haley
```


Attribution statement: (choose only one and delete the rest)


```{r}
# 2. I did this homework with help from the book and the professor and these
Internet sources:https://www.rdocumentation.org/packages/ggplot2/versions/
3.2.1/topics/geom_map, www.statology.org
#Ben Heindl also helped me out with J
```

Last assignment we explored **data visualization** in R using the **ggplot2**
package. This homework continues to use ggplot, but this time, with maps. In
addition, we will merge datasets using the built-in **merge( )** function,
which provides a similar capability to a **JOIN in SQL** (don't worry if you
do not know SQL). Many analytical strategies require joining data from
different sources based on a **“key”** – a field that two datasets have in
common.

Step 1: Load the population data

A. Read the following JSON file, https://intro-datascience.s3.us-
east-2.amazonaws.com/cities.json and store it in a variable called **pop**.

Examine the resulting pop dataframe and add comments explaining what each
column contains.


```{r}
library(jsonlite)
library(tidyverse)
pop <- fromJSON("https://intro-datascience.s3.us-east-2.amazonaws.com/
cities.json")
```

B. Calculate the **average population** in the dataframe. Why is using mean()
directly not working? Find a way to correct the data type of this variable so
you can calculate the average (and then calculate the average)

Hint: use **str(pop)** or **glimpse(pop)** to help understand the dataframe
```

```{r}
str(pop)
pop$population <- as.numeric(pop$population)
mean(pop$population)
#mean() isn't directly working because the population wasn't numeric.
Updating the data type to numeric now lets us calculate the average
population, which is 131,132.4
```

C. What is the population of the smallest city in the dataframe? Which state is it in?

```{r}
pop[(which.min(pop$population)), ]
#The smallest city in the dataframe is Panama City, Florida and it has a
population of 36877
```

Step 2: Merge the population data with the state name data

D) Read in the state name .csv file from the URL below into a dataframe named **abbr** (for "abbreviation") – make sure to use the read_csv() function from the tidyverse package:
https://intro-datascience.s3.us-east-2.amazonaws.com/statesInfo.csv

```{r}
abbr <- read_csv("https://intro-datascience.s3.us-east-2.amazonaws.com/
statesInfo.csv")
```

E) To successfully merge the dataframe **pop** with the **abbr** dataframe, we need to identify a **column they have in common** which will serve as the **“key”** to merge on. One column both dataframes have is the **state column**. The only problem is the slight column name discrepancy – in **pop**, the column is called **“state”** and in **abbr** – **“State.”** These names need to be reconciled for the merge() function to work. Find a way to rename **abbr’s “State”** to **match** the **state column in pop**.

```{r}
cnames <- c("state", "abbreviation")
colnames(abbr) <- cnames
#pg 47 of the book shows how to rename column names by creating a character
vector (cnames) and then passing it through the colnames() function
```

F) Merge the two dataframes (using the **‘state’ column** from both dataframes), storing the resulting dataframe in **dfNew**.

```{r}
dfNew <- merge(pop,abbr, by = "state")
```

G) Review the structure of **dfNew** and explain the columns (aka attributes) in that dataframe.

```{r}
str(dfNew)
#The columns of dfNew show the population of different cities in the US, as well as how much they've grown from 2000-2013, where they rank compared to other cities in the list, as well as which state they are in, the abbreviation of that state, as well as the latitude and longitude of the city
```

Step 3: Visualize the data

H) Plot points (on top of a map of the US) for **each city**. Have the **color** represent the **population**.

```{r}
library(tidyverse)
library(ggplot2)
library(ggmap)
US <- data.frame(map_data("state"))
PopMap <- ggplot() + geom_polygon(data=US, aes(x=long, y=lat, group=group,), fill="white", color="black")
PopMap <- PopMap + geom_point(data=dfNew, aes(x=longitude, y=latitude, color=population))
PopMap <- PopMap + expand_limits(x=dfNew$longitude, y=dfNew$latitude) + coord_map()
PopMap
```

I) Add a block comment that criticizes the resulting map. It’s not very good.

```{r}
#This map is very hard to read. The data points from Alaska and Hawaii don't allow for the continental US data to be easily read. Further, the way the population is formatted makes all city points look basically the same.
```

Step 4: Group by State

J) Use group_by and summarise to make a dataframe of state-by-state population. Store the result in **dfSimple**.

```{r}
library(dplyr)
```

```
df <- dfNew %>%
  group_by(state)
dfSimple <- df %>%
  summarise(population = sum(population))
```

K) Name the most and least populous states in **dfSimple** and show the code you used to determine them.

```{r}
dfSimple[which.min(dfSimple$population), c("state", "population")]
#Vermont has the lowest population

dfSimple[which.max(dfSimple$population), c("state", "population")]
#California has the highest population
```

Step 5: Create a map of the U.S., with the color of the state representing the state population

L) Make sure to expand the limits correctly and that you have used **coord_map** appropriately.

```{r}
dfSimple$state <- tolower(dfSimple$state)
simpleUS <- merge(US,dfSimple, by.x = "region", by.y = "state")
Map <- ggplot() + geom_polygon(data=simpleUS, aes(x=long, y=lat, group=group,
fill=population))
Map <- Map + expand_limits(x=simpleUS$long, y=simpleUS$lat) + coord_map()
Map
```