HW 7

April 1, 2021

1 IST 387 HW 7

[1]:
```
# Enter your name here: Connor Hanan
```

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

[2]:
```
# 1. I did this homework by myself, with help from the book and the professor.
```

Chapter 13 of Introduction to Data Science

This module: Last module we explored **data visualization** in R using the **ggplot2** package. This module continues to use ggplot, together with a companion package called **ggmap**. This package enhances the capabilities of ggplot by adding the capability to **draw geographic outlines (polygons), shading, labeling, and other map markings**. In addition, we will merge datasets using the built-in **merge( )** function, which provides a similar capability to a **JOIN in SQL**. Many analytical strategies require joining data from different sources based on a **"key"** – a field that two datasets have in common.

1.1 Step 1: Load the population data

A. The following lines of code will help you read a json file into an R dataframe. Examine the resulting pop dataframe with View() and add comments explaining what each column contains.

[2]:
```
library(jsonlite)
url="https://ist387.s3.us-east-2.amazonaws.com/data/cities.json"
pop <- jsonlite::fromJSON(url)

pop

#City name, percent growth, lat, lon, pop, ranked pop, state name
```

A data.frame: 1000 × 7

	city	growth_from_2000_to_2013	latitude	longitud
	<chr>	<chr>	<dbl>	<dbl>
1	New York	4.8%	40.71278	-74.0059
2	Los Angeles	4.8%	34.05223	-118.243
3	Chicago	-6.1%	41.87811	-87.6298
4	Houston	11.0%	29.76043	-95.3698
5	Philadelphia	2.6%	39.95258	-75.1652
6	Phoenix	14.0%	33.44838	-112.074
7	San Antonio	21.0%	29.42412	-98.4936
8	San Diego	10.5%	32.71574	-117.161
9	Dallas	5.6%	32.77666	-96.7969
10	San Jose	10.5%	37.33821	-121.886
11	Austin	31.7%	30.26715	-97.7430
12	Indianapolis	7.8%	39.76840	-86.1580
13	Jacksonville	14.3%	30.33218	-81.6556
14	San Francisco	7.7%	37.77493	-122.419
15	Columbus	14.8%	39.96118	-82.9987
16	Charlotte	39.1%	35.22709	-80.8431
17	Fort Worth	45.1%	32.75549	-97.3307
18	Detroit	-27.1%	42.33143	-83.0457
19	El Paso	19.4%	31.77758	-106.442
20	Memphis	-5.3%	35.14953	-90.0489
21	Seattle	15.6%	47.60621	-122.332
22	Denver	16.7%	39.73924	-104.990
23	Washington	13.0%	38.90719	-77.0368
24	Boston	9.4%	42.36008	-71.0588
25	Nashville-Davidson	16.2%	36.16266	-86.7816
26	Baltimore	-4.0%	39.29038	-76.6121
27	Oklahoma City	20.2%	35.46756	-97.5164
28	Louisville/Jefferson County	10.0%	38.25266	-85.7584
29	Portland	15.0%	45.52306	-122.676
30	Las Vegas	24.5%	36.16994	-115.139
971	Brookfield	-1.9%	43.06057	-88.1064
972	Park Ridge	0.1%	42.01114	-87.8406
973	Florence	19.8%	34.19543	-79.7625
974	Roy	13.3%	41.16161	-112.026
975	Winter Garden	142.5%	28.56528	-81.5861
976	Chelsea	7.3%	42.39176	-71.0328
977	Valley Stream	3.6%	40.66427	-73.7084
978	Spartanburg	-6.2%	34.94957	-81.9320
979	Lake Oswego	5.3%	45.42067	-122.670
980	Friendswood	28.6%	29.52940	-95.2010
981	Westerville	5.7%	40.12617	-82.9290
982	Northglenn	15.5%	39.89618	-104.981
983	Phenix City	31.9%	32.47098	-85.0007
984	Grove City	35.6%	39.88145	-83.0929
985	Texarkana	7.4%	33.42513	-94.0476
986	Addison	2.6%	41.93170	-87.9889
987	Dover	16.0%	39.15817	-75.5243
988	Lincoln Park	-6.7%	42.25059	-83.1785
989	Calumet City	-4.5%	41.61559	-87.5294
990	Muskegon	-7.1%	43.23418	-86.2483

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.

[3]:
```
mean(as.numeric(pop$population))
```

131132.443

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

[4]:
```
pop[which.min(as.numeric(pop$population)),]
```

A data.frame: 1 × 7

	city	growth_from_2000_to_2013	latitude	longitude	population	r
	<chr>	<chr>	<dbl>	<dbl>	<chr>	<
1000	Panama City	0.1%	30.15881	-85.66021	36877	1

1.2 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://ist387.s3.us-east-2.amazonaws.com/data/states.csv

[5]:
```
library(tidyverse)
```

```
  Attaching packages                              tidyverse
1.3.0

  ggplot2 3.3.2       purrr   0.3.4
  tibble  3.0.4       dplyr   1.0.2
  tidyr   1.1.2       stringr 1.4.0
  readr   1.4.0       forcats 0.5.0

  Conflicts
tidyverse_conflicts()
  dplyr::filter()  masks stats::filter()
  purrr::flatten() masks
jsonlite::flatten()
  dplyr::lag()     masks stats::lag()
```

[6]:
```
abbr <- read_csv("https://ist387.s3.us-east-2.amazonaws.com/data/states.csv")
```

```
  Column specification

cols(
  State = col_character(),
  Abbreviation = col_character()
)
```

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**.

[10]:
```
#did it by naming columns below
#otherwise I would coerce the colnames to lower
```

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

[11]:
```
pop %>%
inner_join(.,abbr, by = c('state' = 'State')) -> dfNew
```

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

[13]:
```
str(dfNew)

#same as pop, only columns that are in numeric form are the lat lon coordinates
#abbreviation was added as a column in merge
```

```
'data.frame':   1000 obs. of  8 variables:
 $ city                      : chr  "New York" "Los Angeles" "Chicago" "Houston"
…
 $ growth_from_2000_to_2013: chr  "4.8%" "4.8%" "-6.1%" "11.0%" …
 $ latitude                  : num  40.7 34.1 41.9 29.8 40 …
 $ longitude                 : num  -74 -118.2 -87.6 -95.4 -75.2 …
 $ population                : chr  "8405837" "3884307" "2718782" "2195914" …
 $ rank                      : chr  "1" "2" "3" "4" …
 $ state                     : chr  "New York" "California" "Illinois" "Texas" …
 $ Abbreviation              : chr  "NY" "CA" "IL" "TX" …
```

1.3 Step 3: Visualize the data

H) Plot points (on top of a map of the US) for **each city** (don't forget to library **ggplot2** and **ggmap**). Have the **color** represent the **population**.

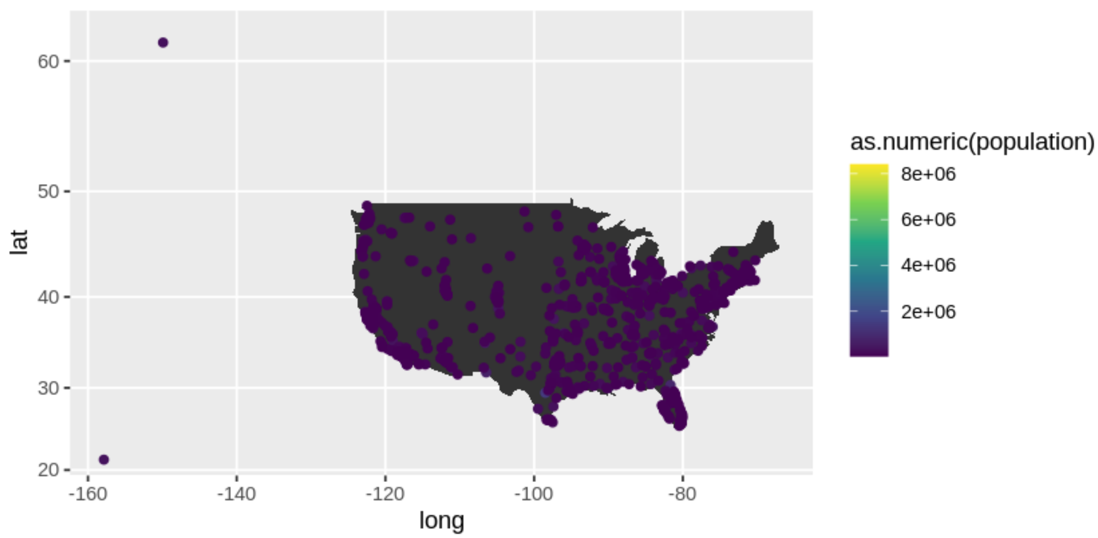
[28]:
```
#install.packages('ggmap')
#install.packages('maps')
#install.packages('mapproj')
library(ggmap)
library(maps)
library(mapproj)
```

[49]:
```
us <- map_data('state')
us$statename <- tolower(us$region)
us
```

A data.frame: 15537 × 7

	long	lat	group	order	region	subregion	statename
	<dbl>	<dbl>	<dbl>	<int>	<chr>	<chr>	<chr>
1	-87.46201	30.38968	1	1	alabama	NA	alabama
2	-87.48493	30.37249	1	2	alabama	NA	alabama
3	-87.52503	30.37249	1	3	alabama	NA	alabama
4	-87.53076	30.33239	1	4	alabama	NA	alabama
5	-87.57087	30.32665	1	5	alabama	NA	alabama
6	-87.58806	30.32665	1	6	alabama	NA	alabama
7	-87.59379	30.30947	1	7	alabama	NA	alabama
8	-87.59379	30.28655	1	8	alabama	NA	alabama
9	-87.67400	30.27509	1	9	alabama	NA	alabama
10	-87.81152	30.25790	1	10	alabama	NA	alabama
11	-87.88026	30.24644	1	11	alabama	NA	alabama
12	-87.92037	30.24644	1	12	alabama	NA	alabama
13	-87.95475	30.24644	1	13	alabama	NA	alabama
14	-88.00632	30.24071	1	14	alabama	NA	alabama
15	-88.01778	30.25217	1	15	alabama	NA	alabama
16	-88.01205	30.26936	1	16	alabama	NA	alabama
17	-87.99486	30.27509	1	17	alabama	NA	alabama
18	-87.95475	30.27509	1	18	alabama	NA	alabama
19	-87.90318	30.28082	1	19	alabama	NA	alabama
20	-87.82870	30.28655	1	20	alabama	NA	alabama
21	-87.80006	30.28655	1	21	alabama	NA	alabama
22	-87.80006	30.32665	1	22	alabama	NA	alabama
23	-87.81724	30.34385	1	23	alabama	NA	alabama
24	-87.84016	30.38395	1	24	alabama	NA	alabama
25	-87.85162	30.40114	1	25	alabama	NA	alabama
26	-87.87453	30.41260	1	26	alabama	NA	alabama
27	-87.90318	30.42406	1	27	alabama	NA	alabama
28	-87.92610	30.44698	1	28	alabama	NA	alabama
29	-87.93183	30.49281	1	29	alabama	NA	alabama
30	-87.94329	30.52719	1	30	alabama	NA	alabama
15570	-105.0289	45.00583	63	15570	wyoming	NA	wyoming
15571	-104.9258	45.00583	63	15571	wyoming	NA	wyoming
15572	-104.7825	45.00583	63	15572	wyoming	NA	wyoming
15573	-104.5820	45.00583	63	15573	wyoming	NA	wyoming
15574	-104.3413	45.00583	63	15574	wyoming	NA	wyoming
15575	-104.1580	45.00583	63	15575	wyoming	NA	wyoming
15576	-104.0549	45.00583	63	15576	wyoming	NA	wyoming
15577	-104.0549	44.58185	63	15577	wyoming	NA	wyoming
15578	-104.0549	44.18077	63	15578	wyoming	NA	wyoming
15579	-104.0606	44.13494	63	15579	wyoming	NA	wyoming
15580	-104.0549	43.84846	63	15580	wyoming	NA	wyoming
15581	-104.0606	43.49895	63	15581	wyoming	NA	wyoming
15582	-104.0663	43.47604	63	15582	wyoming	NA	wyoming
15583	-104.0606	43.00621	63	15583	wyoming	NA	wyoming
15584	-104.0606	42.61087	63	15584	wyoming	NA	wyoming
15585	-104.0549	41.99781	63	15585	wyoming	NA	wyoming
15586	-104.0606	41.69987	63	15586	wyoming	NA	wyoming
15587	-104.0606	41.56236	63	15587	wyoming	NA	wyoming
15588	-104.0606	41.39620	63	15588	wyoming	NA	wyoming
15589	-104.0606	41.00659	63	15589	wyoming	NA	wyoming

[55]:
```
ggplot()+
    geom_polygon(data = us, aes(x = long, y = lat, group = group))+
    expand_limits(x=us$long, y=us$lat)+
    coord_map(projection = 'mercator')+
    geom_point(data=pop,aes(x=longitude,y=latitude,color=as.
 ↪numeric(population)))+
    scale_color_viridis_c()
```



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

[72]:
```
#it's much too far zoomed out because it is including alaska and hawaii
#so many points you cannot tell what is what
```

1.4 Step 4: Use aggregate() to make a dataframe of state-by-state population

Run the following lines of code to create a new data frame:

[63]:
```
dfNew$population <- as.numeric(dfNew$population) #fully converts all values to
↪numeric and saves
dfSimple = aggregate(dfNew$population,
                     by = list(dfNew$state),
                     FUN = sum) #sums all populations of cities within each
↪state, iterating over itself
dfSimple$name <- dfSimple$Group.1 #renaming column
dfSimple$Group.1 <- NULL #deleting column
dfSimple$statePop <- dfSimple$x #renaming column
dfSimple$x <- NULL #delete column
```

J) Add a comment describing what each line of code does. Make sure to describe how many rows there are in **dfSimple** (and why there are that many rows).

[62]:
```
#see above
```

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

[73]:
```
dfSimple[which.max(dfSimple$statePop),'name']

dfSimple[which.min(dfSimple$statePop),'name']
```

'California'

'Vermont'

A data.frame: 51 × 2

name	statePop
<chr>	<dbl>
Alabama	1279813
Alaska	300950
Arizona	4691466
Arkansas	787011
California	27910620
Colorado	3012284
Connecticut	1239817
Delaware	108891
District of Columbia	646449
Florida	7410114
Georgia	1995615
Hawaii	347884
Idaho	638333
Illinois	6055539
Indiana	2393472
Iowa	1037690
Kansas	1327215
Kentucky	1079181
Louisiana	1238263
Maine	66318
Maryland	954852
Massachusetts	3007084
Michigan	2979267
Minnesota	2055749
Mississippi	427944
Missouri	1843953
Montana	277392
Nebraska	807304
Nevada	1481832
New Hampshire	239934
New Jersey	1859793
New Mexico	953296
New York	9933332
North Carolina	3358746
North Dakota	281945
Ohio	3480839
Oklahoma	1666530
Oregon	1680656
Pennsylvania	2598080
Rhode Island	499878
South Carolina	812734
South Dakota	235488
Tennessee	2483464
Texas	14836230
Utah	1440569
Vermont	42284
Virginia	2236964
Washington	2956938
West Virginia	99998
Wisconsin	1910367
Wyoming	122076

1.5 Step 5: Use ggplot and ggmap to shade a map of the U.S. with state population

L) Copy the ggplot code from Step 3. In the initial ggplot statement, you will need to use your new dataframe, so substitute **dfSimple** in place of **dfNew**. Additionally, instead of using **geom_point** to plot points, use this aesthetic to fill the polygons with a **color** for each state. Make sure to expand the limits correctly and that you have used **coord_map** appropriately.

[83]:
```
dfSimple$name <- tolower(dfSimple$name)
```

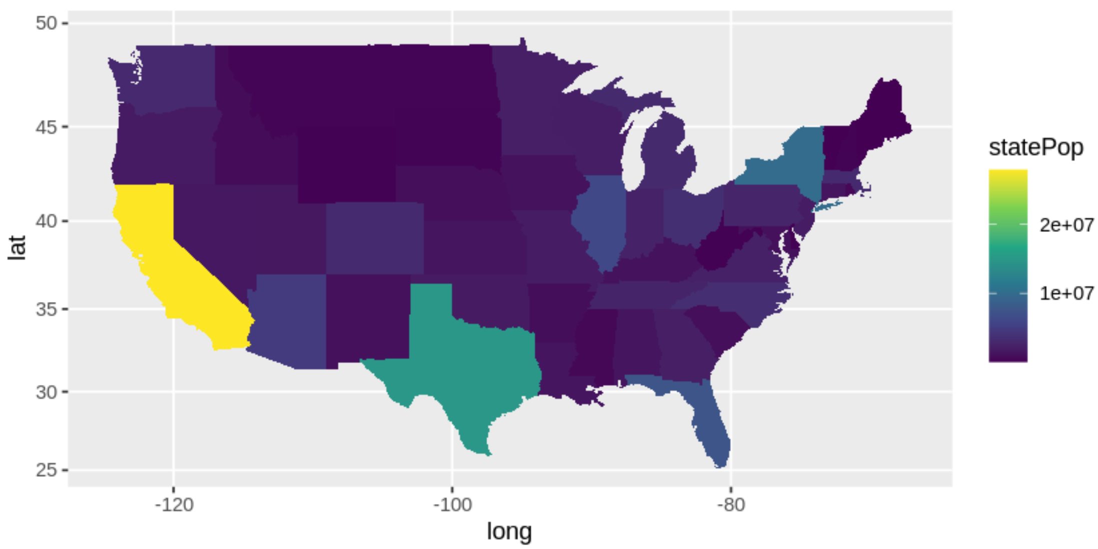
[84]:
```
us %>%
left_join(.,dfSimple, by = c('statename' = 'name')) -> joined_state
```

[85]:
```
joined_state
```

A data.frame: 15537 × 8

long	lat	group	order	region	subregion	statename	statePop
<dbl>	<dbl>	<dbl>	<int>	<chr>	<chr>	<chr>	<dbl>
-87.46201	30.38968	1	1	alabama	NA	alabama	1279813
-87.48493	30.37249	1	2	alabama	NA	alabama	1279813
-87.52503	30.37249	1	3	alabama	NA	alabama	1279813
-87.53076	30.33239	1	4	alabama	NA	alabama	1279813
-87.57087	30.32665	1	5	alabama	NA	alabama	1279813
-87.58806	30.32665	1	6	alabama	NA	alabama	1279813
-87.59379	30.30947	1	7	alabama	NA	alabama	1279813
-87.59379	30.28655	1	8	alabama	NA	alabama	1279813
-87.67400	30.27509	1	9	alabama	NA	alabama	1279813
-87.81152	30.25790	1	10	alabama	NA	alabama	1279813
-87.88026	30.24644	1	11	alabama	NA	alabama	1279813
-87.92037	30.24644	1	12	alabama	NA	alabama	1279813
-87.95475	30.24644	1	13	alabama	NA	alabama	1279813
-88.00632	30.24071	1	14	alabama	NA	alabama	1279813
-88.01778	30.25217	1	15	alabama	NA	alabama	1279813
-88.01205	30.26936	1	16	alabama	NA	alabama	1279813
-87.99486	30.27509	1	17	alabama	NA	alabama	1279813
-87.95475	30.27509	1	18	alabama	NA	alabama	1279813
-87.90318	30.28082	1	19	alabama	NA	alabama	1279813
-87.82870	30.28655	1	20	alabama	NA	alabama	1279813
-87.80006	30.28655	1	21	alabama	NA	alabama	1279813
-87.80006	30.32665	1	22	alabama	NA	alabama	1279813
-87.81724	30.34385	1	23	alabama	NA	alabama	1279813
-87.84016	30.38395	1	24	alabama	NA	alabama	1279813
-87.85162	30.40114	1	25	alabama	NA	alabama	1279813
-87.87453	30.41260	1	26	alabama	NA	alabama	1279813
-87.90318	30.42406	1	27	alabama	NA	alabama	1279813
-87.92610	30.44698	1	28	alabama	NA	alabama	1279813
-87.93183	30.49281	1	29	alabama	NA	alabama	1279813
-87.94329	30.52719	1	30	alabama	NA	alabama	1279813
-105.0289	45.00583	63	15570	wyoming	NA	wyoming	122076
-104.9258	45.00583	63	15571	wyoming	NA	wyoming	122076
-104.7825	45.00583	63	15572	wyoming	NA	wyoming	122076
-104.5820	45.00583	63	15573	wyoming	NA	wyoming	122076
-104.3413	45.00583	63	15574	wyoming	NA	wyoming	122076
-104.1580	45.00583	63	15575	wyoming	NA	wyoming	122076
-104.0549	45.00583	63	15576	wyoming	NA	wyoming	122076
-104.0549	44.58185	63	15577	wyoming	NA	wyoming	122076
-104.0549	44.18077	63	15578	wyoming	NA	wyoming	122076
-104.0606	44.13494	63	15579	wyoming	NA	wyoming	122076
-104.0549	43.84846	63	15580	wyoming	NA	wyoming	122076
-104.0606	43.49895	63	15581	wyoming	NA	wyoming	122076
-104.0663	43.47604	63	15582	wyoming	NA	wyoming	122076
-104.0606	43.00621	63	15583	wyoming	NA	wyoming	122076
-104.0606	42.61087	63	15584	wyoming	NA	wyoming	122076
-104.0549	41.99781	63	15585	wyoming	NA	wyoming	122076
-104.0606	41.69987	63	15586	wyoming	NA	wyoming	122076
-104.0606	41.56236	63	15587	wyoming	NA	wyoming	122076
-104.0606	41.39620	63	15588	wyoming	NA	wyoming	122076
-104.0606	41.00659	63	15589	wyoming	NA	wyoming	122076

[90]:
```
ggplot()+
    geom_polygon(data = joined_state, aes(x = long, y = lat, group = group,␣
↪fill = statePop))+
    expand_limits(x=us$long, y=us$lat)+
    coord_map(projection = 'mercator')+
    scale_fill_viridis_c()
```



[ ]:
```
```