Map visualization with `ggplot2` of Tidyverse

Monika Baloda

In this R-markdown, I play with maps using tidyverse/ggplot basic regressions. also show some visualizations and data cleaning steps before jumping to regressions. Main steps done in this exercise can be put in the following points:

1. Plotting USA maps and state maps
2. Plotting by manipulations
3. Application of maps with ggplot using NYC flight destination map

Load necessary packages

```
#install.packages("maps")
#install.packages("mapproj")

library(tidyverse) # for `ggplot2`, `dplyr`, and more
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error

library(maps) # for map visualization
##
## Attaching package: 'maps'
##
## The following object is masked from 'package:purrr':
##
##     map
#library(mapdata)
library(mapproj)

library(datasets) # for `state` data set
library(nycflights13) # for the 2013 NYC flights data set
```

Map visualization with ggplot2

In this work, we will draw maps with `ggplot2`.

The `maps` package comes with a plotting function, but, we will opt to use `ggplot2` functions (`geom_polygon` and `geom_map`) to plot the maps in the `maps` package.

Recall that `ggplot2` operates on data frames. We will use the `map_data()` function (provided by `ggplot2`), which turns a series of points along an outline into a data frame of those points.

1) Plot the USA map

First we load the USA map from `maps`.

```
usa_map <- map_data("usa")
dim(usa_map)
## [1] 7243    6
glimpse(usa_map)
## Rows: 7,243
## Columns: 6
## $ long      <dbl> -101.4078, -101.3906, -101.3620, -101.3505, -101.3219, -101.~
## $ lat       <dbl> 29.74224, 29.74224, 29.65056, 29.63911, 29.63338, 29.64484, ~
## $ group     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ order     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1~
## $ region    <chr> "main", "main", "main", "main", "main", "main", "main", "mai~
## $ subregion <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

We use `geom_polygon()` to make a simple black map (no line color, but with a black fill).

```
ggplot(data = usa_map) +
  geom_polygon(aes(x = long, y = lat, group = group)) +
  coord_quickmap()
```

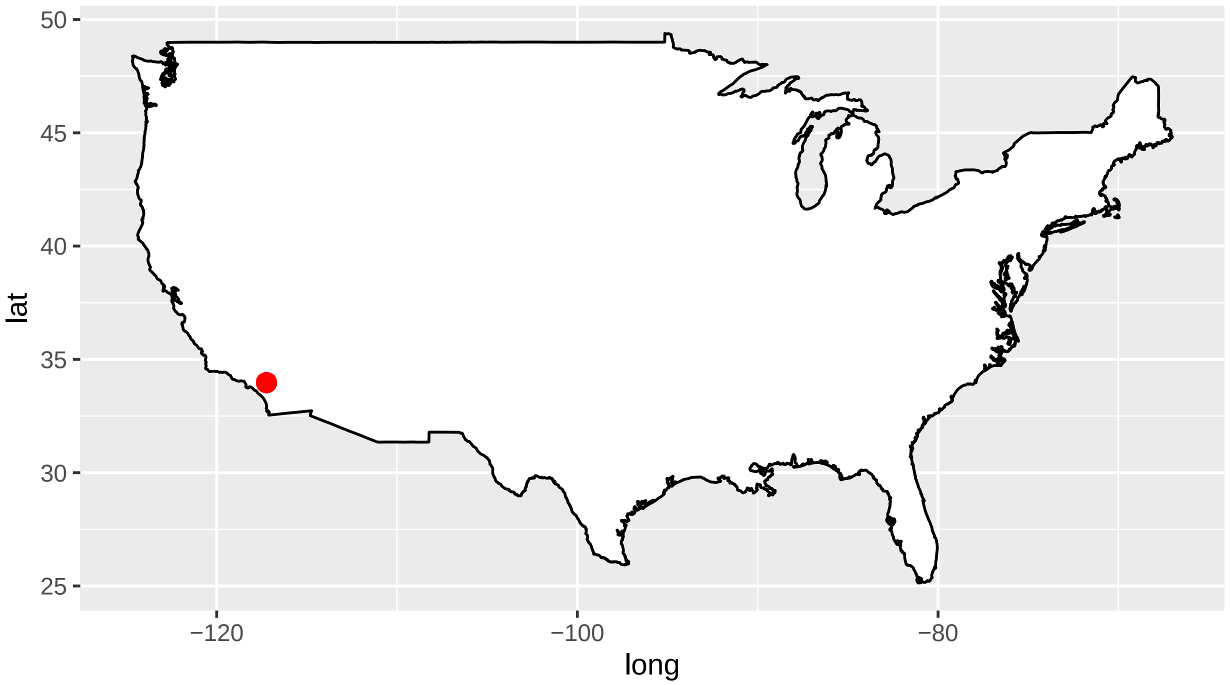
50
45
40
35
30
25
lat
−120
−100
−80
long

We google the coordinates of University of California Riverside, and then use `geom_point()` to mark the location of UCR on the USA map. In addition, change the outline/border color as well as the fill-in color of your map.

plot the USA map with white fill and black outline below

```
usa_map= map_data("usa")

ggplot(data = usa_map) +
  geom_polygon(aes(x = long, y = lat, group = group), fill= "white", color = "black") +
  coord_quickmap()+
  ##point to mark the location of UCR, with coordinates of UCR
  geom_point(x = -117.2387, y = 33.9738, size= 3, colour = "red")
```



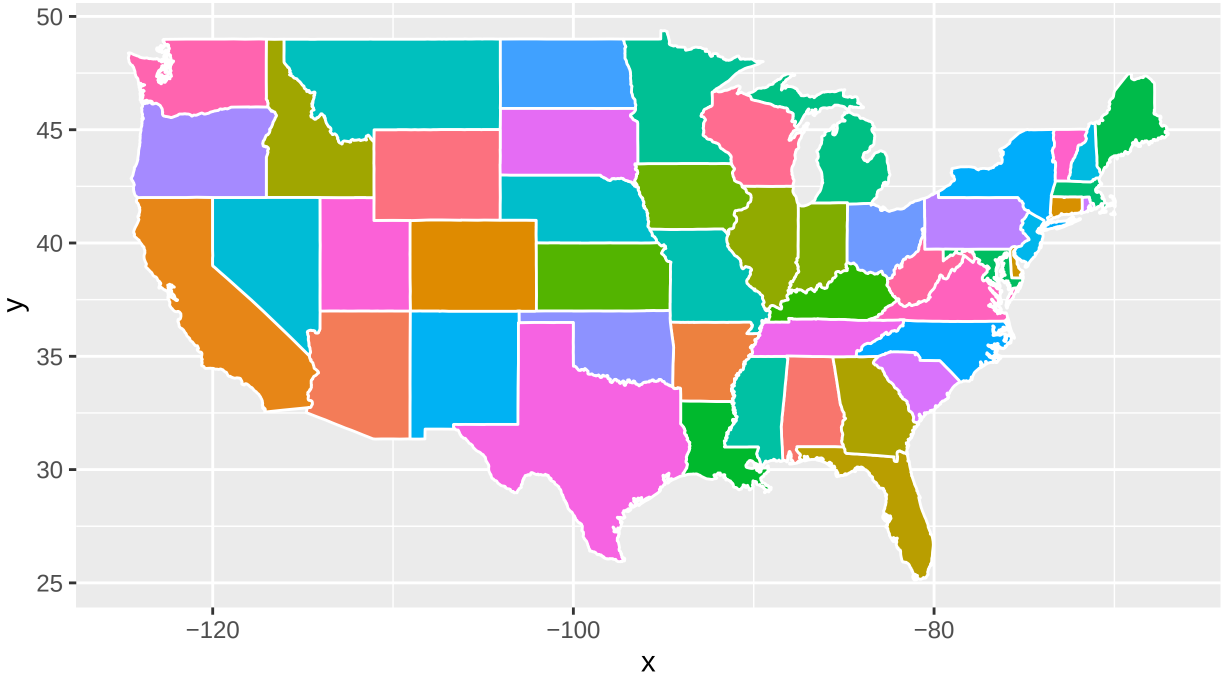
2) Plot the states map

In addition to `geom_polygon()`, we can use `geom_map()` to draw maps too. Basically, `geom_map()` acts as a wrapper of `geom_polygon()`. See more details in the `geom_map()` documentation @ http://ggplot2.tidyverse.org/reference/geom_map.html

Here is the example code of a states map. We can plot all the states, map the `fill` aesthetic to `region` and set the the lines of state borders to white color.

```
states_map =map_data("state")
dim(states_map)
## [1] 15537     6
glimpse(states_map)
## Rows: 15,537
## Columns: 6
## $ long      <dbl> -87.46201, -87.48493, -87.52503, -87.53076, -87.57087, -87.5~
## $ lat       <dbl> 30.38968, 30.37249, 30.37249, 30.33239, 30.32665, 30.32665, ~
## $ group     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ order     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1~
## $ region    <chr> "alabama", "alabama", "alabama", "alabama", "alabama", "alab~
## $ subregion <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
ggplot() +
  geom_map(data = states_map, map = states_map,
           aes(map_id = region, fill = region), color="white") +
  # geom_map() doesn't work in such a way that ggplot2 knows the extend of the map values, so you alway
  expand_limits(x = states_map$long, y = states_map$lat) +
  coord_quickmap() +
  guides(fill = FALSE) # do this to leave off the color legend
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



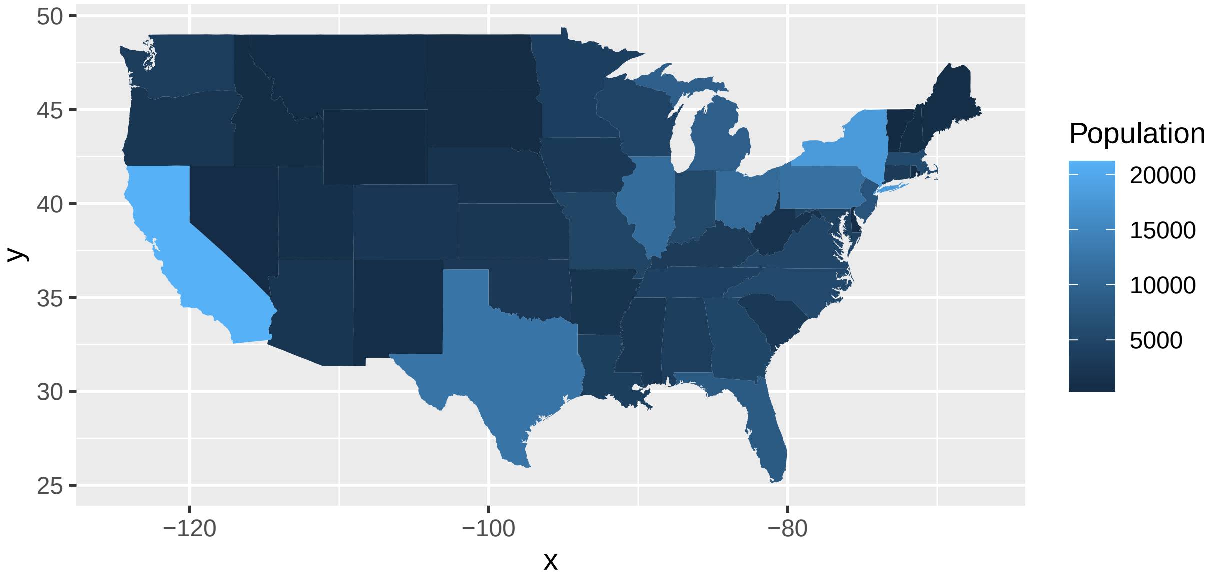
Next, we will use the built-in `state` data sets in R to annotate our states map. In particular, `state.x77` is a two-dimensional array containing 8 variables and data from all 50 states.

```
#?state.x77
head(state.x77)
##            Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
## Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
## Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
## Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
## California      21198   5114        1.1    71.71   10.3    62.6    20 156361
## Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766

state_data <- as.data.frame(state.x77)
state_data$State <- tolower(rownames(state_data))
state_data %>% glimpse()
## Rows: 50
## Columns: 9
## $ Population <dbl> 3615, 365, 2212, 2110, 21198, 2541, 3100, 579, 8277, 4931, ~
## $ Income     <dbl> 3624, 6315, 4530, 3378, 5114, 4884, 5348, 4809, 4815, 4091,~
## $ Illiteracy <dbl> 2.1, 1.5, 1.8, 1.9, 1.1, 0.7, 1.1, 0.9, 1.3, 2.0, 1.9, 0.6,~
## $ `Life Exp` <dbl> 69.05, 69.31, 70.55, 70.66, 71.71, 72.06, 72.48, 70.06, 70.~
## $ Murder     <dbl> 15.1, 11.3, 7.8, 10.1, 10.3, 6.8, 3.1, 6.2, 10.7, 13.9, 6.2~
## $ `HS Grad`  <dbl> 41.3, 66.7, 58.1, 39.9, 62.6, 63.9, 56.0, 54.6, 52.6, 40.6,~
## $ Frost      <dbl> 20, 152, 15, 65, 20, 166, 139, 103, 11, 60, 0, 126, 127, 12~
## $ Area       <dbl> 50708, 566432, 113417, 51945, 156361, 103766, 4862, 1982, 5~
## $ State      <chr> "alabama", "alaska", "arizona", "arkansas", "california", "~
```

Below is the example code from the lecture for a state population map. We first create an aesthetic mapping for `map_id` to the column `State` (state names in lower case) in the `state_data` data frame. We then call `geom_map` again and map the `fill` aesthetic to the `Population` variable in `state_data`.

```
ggplot(data = state_data, aes(map_id = State))+
  geom_map(map = states_map,
           aes(fill = Population)) +
  expand_limits(x = states_map$long, y = states_map$lat) +
  coord_quickmap()
```



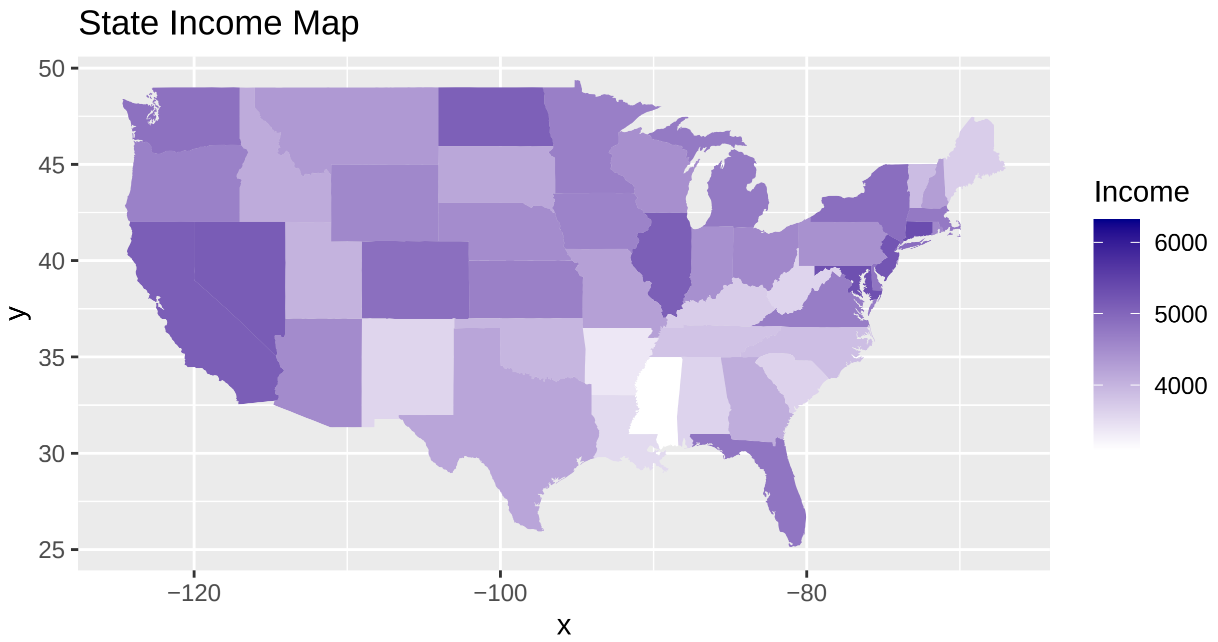
a) Drawing sample states map and map the `fill` aesthetic to `Income` in the `state.x77` data set

```
states_map= map_data("state")

#converts the built-in dataset "state.x77" into a data frame
```

```
state_data =as.data.frame(state.x77)
state_data$State=tolower(rownames(state_data))

ggplot(data = state_data, aes(map_id = State))+
  geom_map(map = states_map,
           aes(fill = Income)) +
  expand_limits(x = states_map$long, y = states_map$lat) +
  coord_quickmap() +
  labs(title = "State Income Map") +
  scale_fill_gradient(low = "white", high = "darkblue") +
  guides(fill = guide_colorbar(title = "Income"))
```



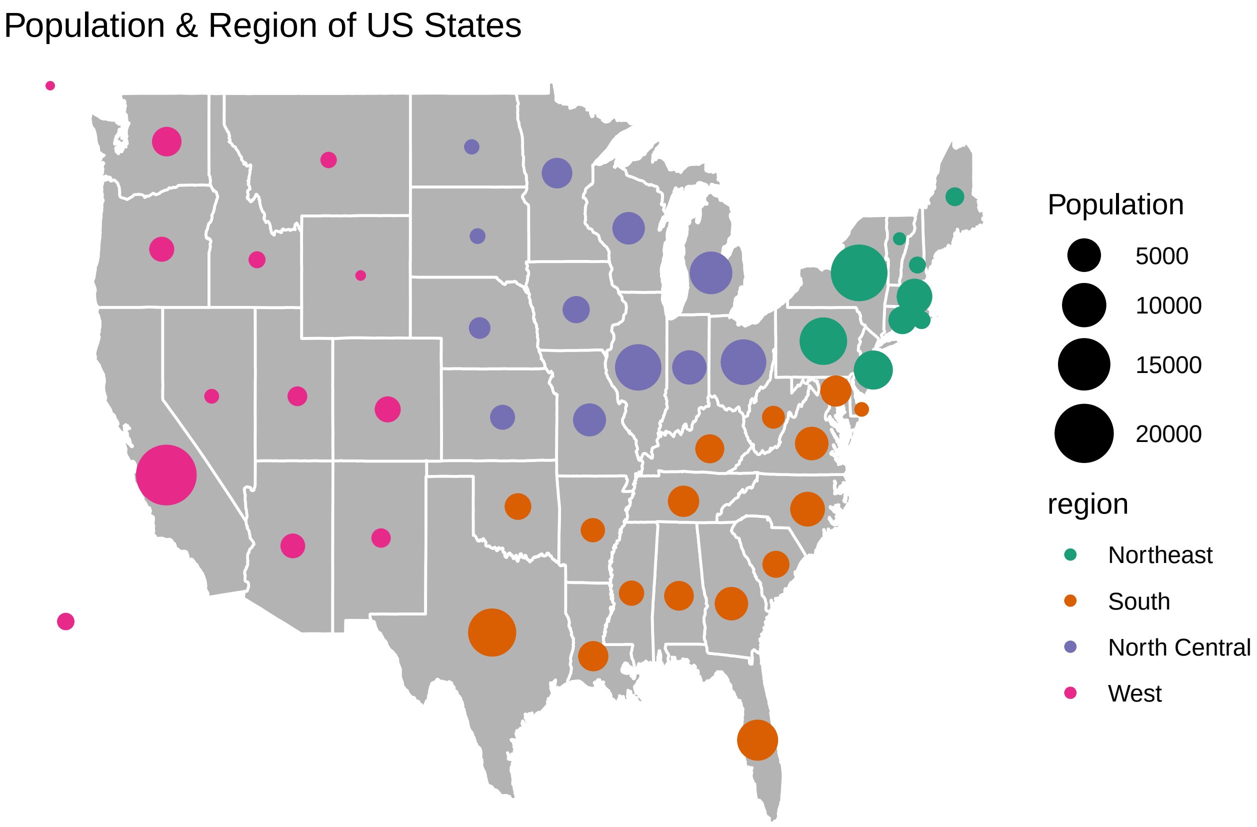
b) Adding 50 colorful points to your map We use one point to mark one state (state coordinates can be found in `state.center`). Map the color of the points to `state.region`. Map the `size` aesthetic of the points to `Population`.

```
state_coords= data.frame(state.center)

#adds a new column "region" to the data frame "state_coords".
state_coords$region=state.region
state_coords$Population= state.x77[,"Population"]

states_map= map_data("state")

ggplot() +
  geom_polygon(data = states_map, aes(x = long, y= lat, group = group),color = "white", fill = "gray70"
  geom_point(data = state_coords, aes(x = x, y = y, color = region, size = Population)) +
  scale_color_brewer(palette = "Dark2") +
  scale_size(range = c(1, 10)) +
  labs(title = "Population & Region of US States",
       x = "", y = "") +
 theme_void()
```



***

3. NYC flight destination map : application of maps using ggplot

`nycflights13::flights` data set contains all 336,776 flights that departed from New York City in 2013.

```
#?flights # full documentation
#glimpse(flights)
```

a) Counting the number of flights per destination

How many unique destination airports did these NYC flights connected to? How many **non-canceled** flights per destination? Which destination had the largest number of arrival flights from NYC? Which destination had the smallest number of arrival flights from NYC?

ANSWER

```
num_destinations= length(unique(nycflights13::flights$dest))
cat("Number of unique destination airports: ", num_destinations, "\n")
## Number of unique destination airports:  105
```

```
non_cancelled_flights= subset(nycflights13::flights, !is.na(arr_delay))
#aggregates the "arr_delay" variable
non_cancelled_by_dest= aggregate(non_cancelled_flights$arr_delay, by = list(non_cancelled_flights$dest)
names(non_cancelled_by_dest)= c("destination", "num_flights")
cat("Number of non-cancelled flights per destination:\n")
## Number of non-cancelled flights per destination:
print(non_cancelled_by_dest)
##     destination num_flights
## 1           ABQ         254
## 2           ACK         264
## 3           ALB         418
## 4           ANC           8
## 5           ATL       16837
## 6           AUS        2411
## 7           AVL         261
## 8           BDL         412
## 9           BGR         358
## 10          BHM         269
## 11          BNA        6084
## 12          BOS       15022
## 13          BQN         888
## 14          BTV        2510
## 15          BUF        4570
## 16          BUR         370
## 17          BWI        1687
## 18          BZN          35
## 19          CAE         106
## 20          CAK         842
## 21          CHO          46
## 22          CHS        2759
## 23          CLE        4394
## 24          CLT       13674
## 25          CMH        3326
## 26          CRW         134
## 27          CVG        3725
## 28          DAY        1399
## 29          DCA        9111
## 30          DEN        7169
## 31          DFW        8388
## 32          DSM         523
## 33          DTW        9031
## 34          EGE         207
## 35          EYW          17
## 36          FLL       11897
## 37          GRR         728
## 38          GSO        1492
## 39          GSP         790
## 40          HDN          14
## 41          HNL         701
## 42          HOU        2083
## 43          IAD        5383
## 44          IAH        7085
## 45          ILM         107
## 46          IND        1981
## 47          JAC          21
## 48          JAX        2623
## 49          LAS        5952
## 50          LAX       16026
## 51          LEX           1
## 52          LGB         661
## 53          MCI        1885
## 54          MCO       13967
## 55          MDW        4025
## 56          MEM        1686
## 57          MHT         932
## 58          MIA       11593
## 59          MKE        2709
## 60          MSN         556
## 61          MSP        6929
## 62          MSY        3715
## 63          MTJ          14
## 64          MVY         210
## 65          MYR          58
## 66          OAK         309
## 67          OKC         315
## 68          OMA         817
## 69          ORD       16566
## 70          ORF        1434
## 71          PBI        6487
## 72          PDX        1342
## 73          PHL        1541
## 74          PHX        4606
## 75          PIT        2746
## 76          PSE         358
## 77          PSP          18
## 78          PVD         358
## 79          PWM        2288
## 80          RDU        7770
## 81          RIC        2346
## 82          ROC        2358
## 83          RSW        3502
## 84          SAN        2709
## 85          SAT         659
## 86          SAV         749
## 87          SBN          10
## 88          SDF        1104
## 89          SEA        3885
## 90          SFO       13173
## 91          SJC         328
## 92          SJU        5773
## 93          SLC        2451
## 94          SMF         282
## 95          SNA         812
## 96          SRQ        1201
## 97          STL        4142
## 98          STT         518
## 99          SYR        1707
## 100         TPA        7390
## 101         TUL         294
## 102         TVC          95
## 103         TYS         578
## 104         XNA         992

max_dest= non_cancelled_by_dest[which.max(non_cancelled_by_dest$num_flights),"destination"]
#cat function to print a message to the console
cat("Destination with the largest number of arrival flights: ", max_dest, "\n")
## Destination with the largest number of arrival flights:  ATL

min_dest=non_cancelled_by_dest[which.min(non_cancelled_by_dest$num_flights), "destination"]
cat("Destination with the smallest number of arrival flights: ", min_dest, "\n")
## Destination with the smallest number of arrival flights:  LEX
```

b) Marking all destination airports on a states map Find out the coordinates of the destination airports from `nycflights13::airports`. Draw each destination airport as a point on a states map, and map a point aesthetic to the number of **non-canceled** flights flew to that destination from NYC in 2013.

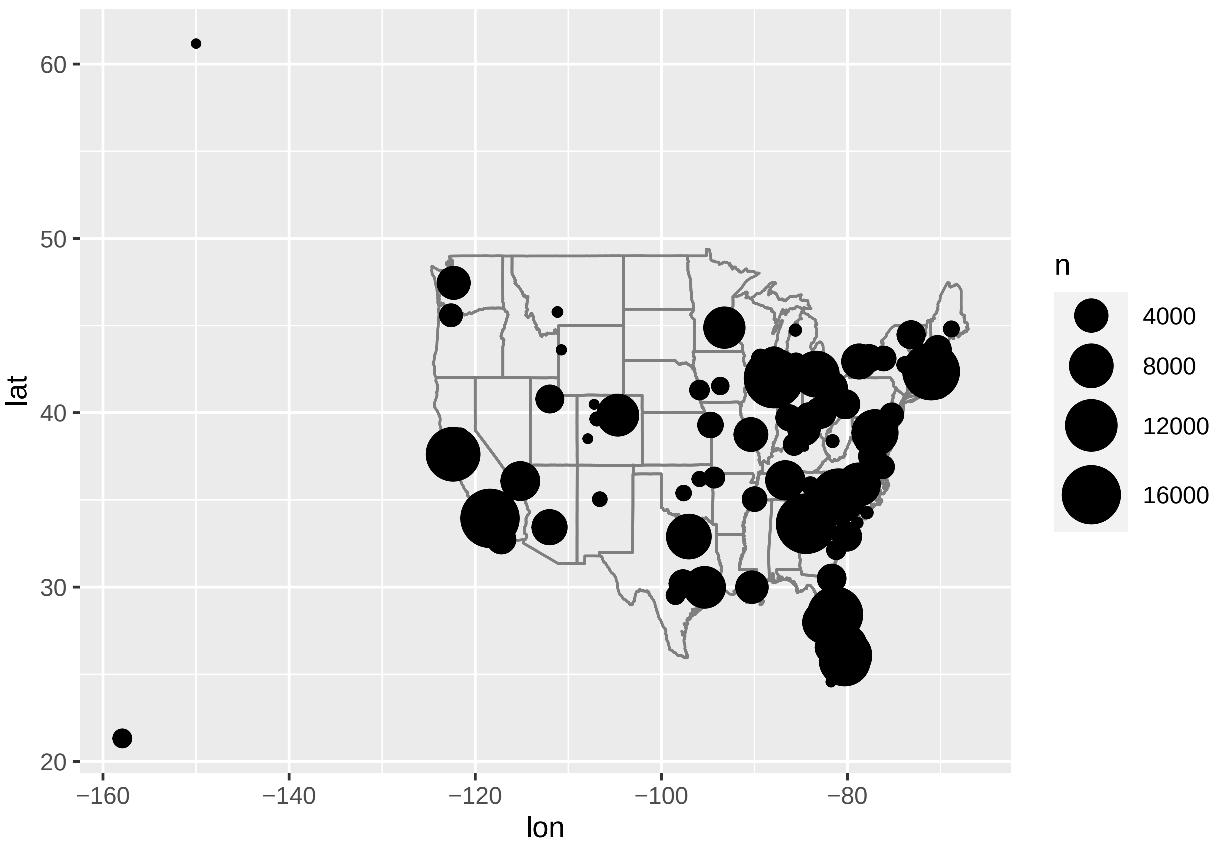
```
#?airports
airports %>% glimpse()
left_join(per_dest, airports, by=c("dest"="faa"))
```

```
library(ggplot2)
library(nycflights13)
library(magrittr)
##
## Attaching package: 'magrittr'
## The following object is masked from 'package:purrr':
##
##     set_names
## The following object is masked from 'package:tidyr':
##
##     extract
library(dplyr)

per_dest= flights %>%
  filter(!is.na(dep_delay)) %>% # remove cancelled flights
  group_by(dest) %>% #groups the filtered data by the dest column
  summarize(n = n()) # count the non-cancelled flights/destination

 #performs a left join between the per_dest & airports dataframe
df= left_join(per_dest, airports, by = c("dest" = "faa"))

ggplot(df, aes(x = lon, y =lat, size = n)) +
  borders("state", colour = "grey50", fill = NA) +
  geom_point() +
  scale_size_continuous(range = c(1, 10))
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```



Question Which destination airports have missing values?

ANSWER

```r
#here dep_delay column is not NA (i.e., missing values).
per_dest = flights %>%
  filter(!is.na(dep_delay)) %>%
  group_by(dest) %>%
  summarize(n = n()) %>%
  arrange(desc(n))

# to find the airports with missing
missing_airports = flights %>%
  filter(is.na(dep_delay)) %>%
  distinct(dest)

# Print the list of airports with missing values
#missing_airports that shows the unique destinations
missing_airports
## # A tibble: 99 x 1
##    dest
##    <chr>
##  1 RDU
##  2 DFW
##  3 MIA
##  4 FLL
##  5 CVG
##  6 PIT
##  7 MHT
##  8 ATL
##  9 IND
## 10 LAX
## # i 89 more rows
```