STAT167 HW5 - Spring 2025

Ethan Choi

Contents

Homework #5 instructions

Review the ISLR textbook Chapter 3 "Linear Regression", as well as the lecture notes on map visualization and linear regression before answering the homework questions.

This homework contains 2 questions, each with multiple parts, 100 points in total.

Replace **INSERT_YOUR_ANSWER** with your own answers.

- First open this `rmd` file in RStudio and click `Knit -> Knit to PDF` to render it to PDF format. You need to have `LaTex` installed on the computer to render it to PDF format. If not, you can also render it to HTML format.
- It is best to read this `rmd` file and the rendered `pdf/html` file side-by-side, while you are working on this homework.
- If the question asks you to write some R code, remember to put your code into a **R code chunk**. Make sure both your R code chunk and its output are visible in the rendered `pdf/html` file.
- For this homework, use **ggplot2** to visualize your data.
- **Please comment your R code thoroughly, and follow the R coding style guideline (https://google.github.io/styleguide/Rguide.xml). Partial credit will be deducted for insufficient commenting or poor coding styles.**
- If you have any question about this homework assignment, we encourage you to post it on **Piazza**.

Homework submission guideline

- **This homework is DUE at *11:59 PM* on *Sunday May 11, 2025***.
- Late submission penalties.

  - Submissions up to 24 hours late will incur a 10% deduction.
  - Submissions up to 48 hours late will incur a 30% deduction.

- **If you are using one or both of your free late days, please state here: INSERT_YOUR_ANSWER**
- After you complete all questions, save your `rmd` file to `FirstnameLastname-SID-HW5.rmd` and save the rendered pdf file to `FirstnameLastname-SID-HW5.pdf`. If you can not knit it to pdf, knit it to html first and then print/save it to pdf format.
- Submit **BOTH your source `rmd` file and the knitted `pdf` file** to **GradeScope**. Do NOT create a zip file. For the `pdf` submission, please tag specific pages that correspond with each question in the assignment.
- You can submit multiple times, you last submission will be graded.

Acknowledgements

Please list all the help you have received for completing this homework.

Used some references from geeksforgeeks

Install necessary packages

Note that you only need to install each package once. Then you can comment out the following installation lines.

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

Load necessary packages

It is a recommended best practice to load all of your R packages and datasets at the beginning of your file in one chunk.

```
library(tidyverse) # for `ggplot2`, `dplyr`, and more
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- 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` dataset
library(nycflights13) # for the 2013 NYC flights dataset
```

Question 1 [45pt] Map visualization with ggplot2

Recall in HW1, we used the `maps` package to generate map visualizations. In this homework, 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.

(a) [5pt] Plot the USA map

First we load the USA map from `maps`.

```
usa_map <- map_data("usa")
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, ~
```

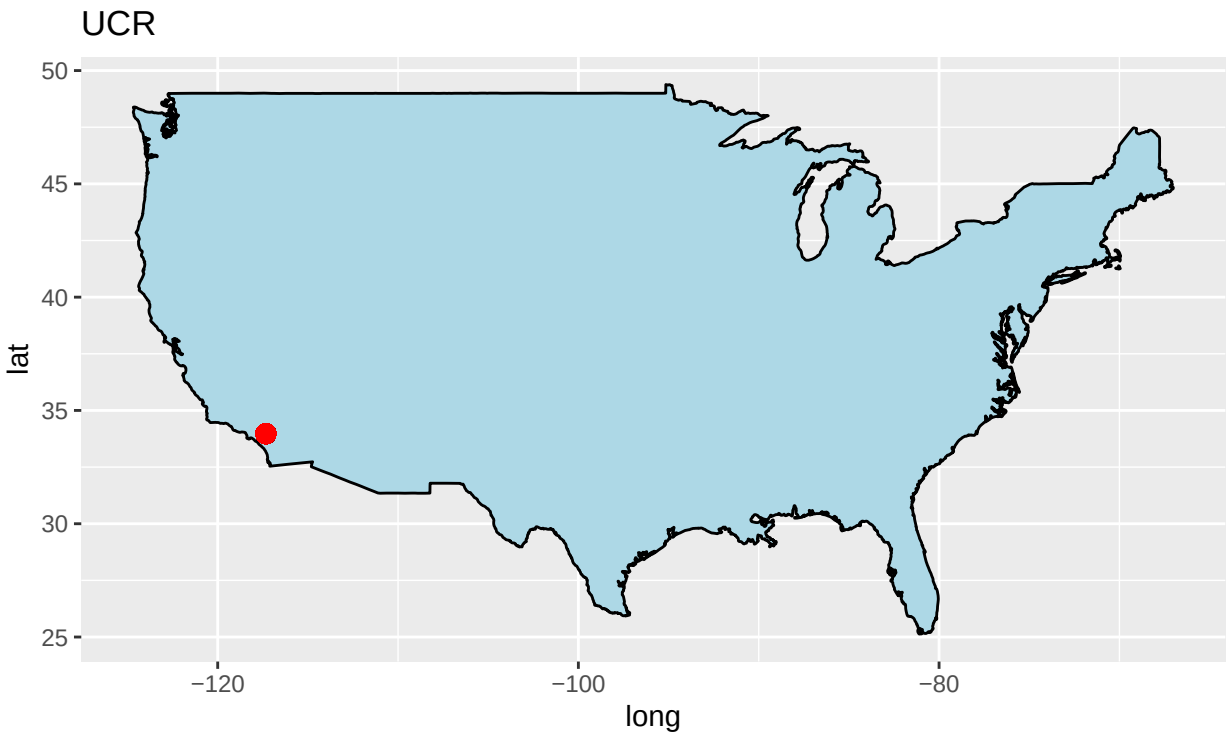
In the lecture, we have learned how to use `geom_polygon()` to make a simple black map (no line color, but with a black fill).

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

Google the coordinates of UCR, 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.

```
ggplot(data = usa_map) +
  geom_polygon(aes(x = long, y = lat, group = group), fill = "lightblue", color = "black") +
  geom_point(aes(x = -117.3281, y = 33.9737), color = "red", size = 3) +
  coord_quickmap() +
  labs(title = "UCR")
```

```
## Warning in geom_point(aes(x = -117.3281, y = 33.9737), color = "red", size = 3): All aesthetics have
## i Please consider using ‘annotate()‘ or provide this layer with data containing
##   a single row.
```



(b) [15pt] Plot the states map

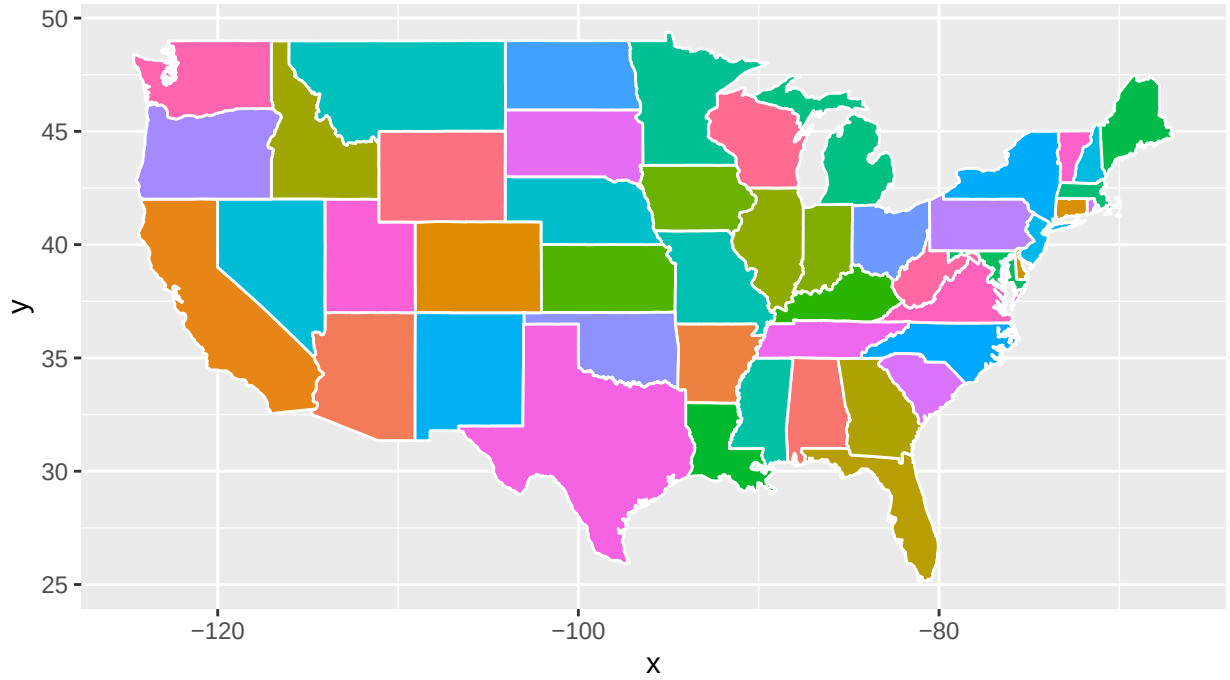
In addition to `geom_polygon()`, we can use `geom_map()` to draw maps too. Basically, `geom_map()` acts as a wrapper of `geom_polygon()`. See more details in the `geom_map()` documentation @ http://ggplot2.tidyverse.org/reference/geom_map.html

Here is the example code of a states map from the lecture. 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")
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 = "none") # do this to leave off the color legend
```



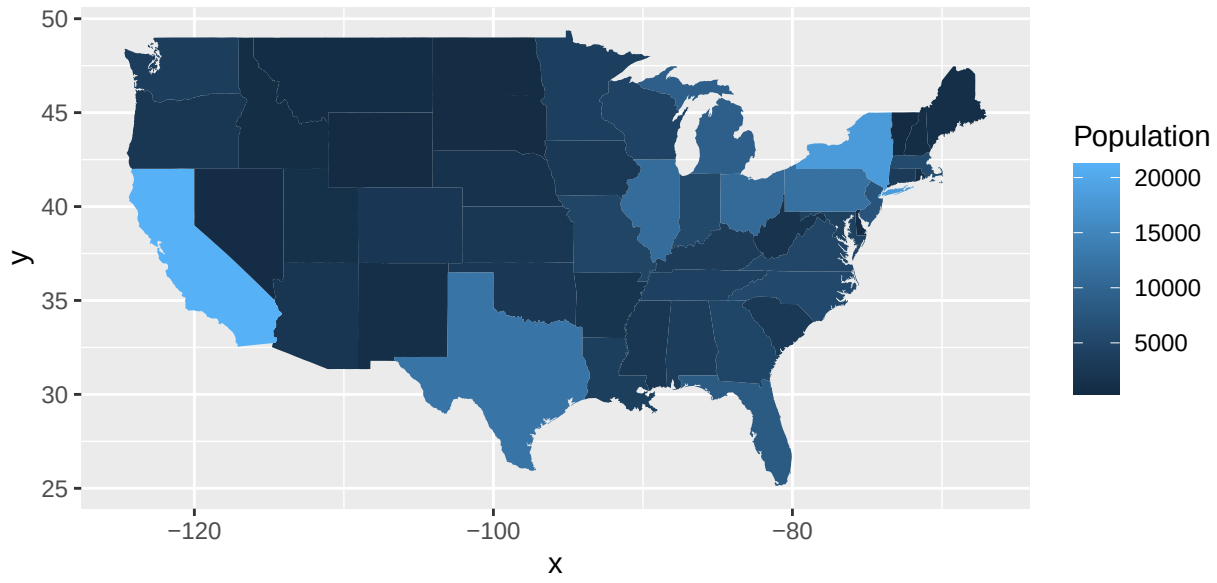
Next, we will use the built-in `state` datasets in R to annotate our states map. In particular, `state.x77` is a two-dimensional array containing 8 variables and data from all 50 states.

```
?state.x77
## starting httpd help server ... done
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", "~
```

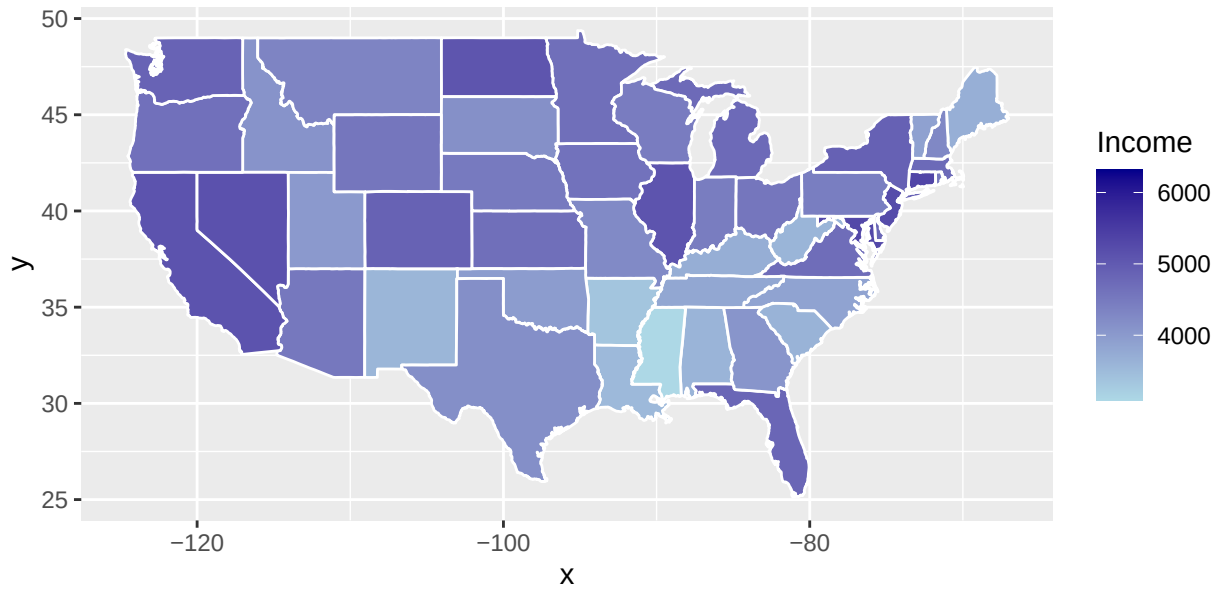
Below is the example code from the lecture for a state population map. We first create an aesthetic mapping for `map_id` to the column `State` (state names in lower case) in the `state_data` data frame. We then call `geom_map` again and map the `fill` aesthetic to the `Population` variable in `state_data`.

```
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()
```



(i) [5pt] Draw your own states map Map the `fill` aesthetic to `Income` in the `state.x77` dataset

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

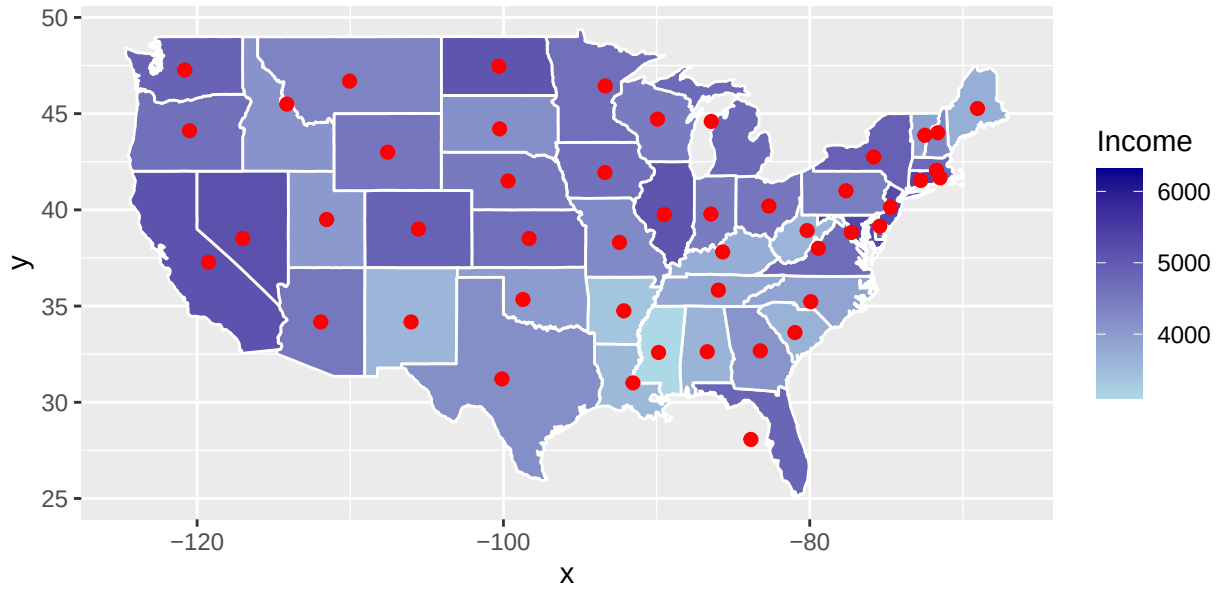


(ii) [10pt] Add 50 colorful points to your map Use one point to mark one state (state coordinates can be found in `state.center`). Map the color of the points to `state.region`. Map the `size` aesthetic of the points to `Population`.

```
head(state.center)
## $x
##  [1]  -86.7509 -127.2500 -111.6250  -92.2992 -119.7730 -105.5130  -72.3573
##  [8]  -74.9841  -81.6850  -83.3736 -126.2500 -113.9300  -89.3776  -86.0808
## [15]  -93.3714  -98.1156  -84.7674  -92.2724  -68.9801  -76.6459  -71.5800
## [22]  -84.6870  -94.6043  -89.8065  -92.5137 -109.3200  -99.5898 -116.8510
## [29]  -71.3924  -74.2336 -105.9420  -75.1449  -78.4686 -100.0990  -82.5963
## [36]  -97.1239 -120.0680  -77.4500  -71.1244  -80.5056  -99.7238  -86.4560
## [43]  -98.7857 -111.3300  -72.5450  -78.2005 -119.7460  -80.6665  -89.9941
## [50] -107.2560
##
## $y
##  [1] 32.5901 49.2500 34.2192 34.7336 36.5341 38.6777 41.5928 38.6777 27.8744
## [10] 32.3329 31.7500 43.5648 40.0495 40.0495 41.9358 38.4204 37.3915 30.6181
## [19] 45.6226 39.2778 42.3645 43.1361 46.3943 32.6758 38.3347 46.8230 41.3356
## [28] 39.1063 43.3934 39.9637 34.4764 43.1361 35.4195 47.2517 40.2210 35.5053
## [37] 43.9078 40.9069 41.5928 33.6190 44.3365 35.6767 31.3897 39.1063 44.2508
## [46] 37.5630 47.4231 38.4204 44.5937 43.0504
head(state.region)
## [1] South West  West  South West  West
## Levels: Northeast South North Central West
```

```r
# Was running into a lot of errors for columns and not sure if I did previous problem correct so here i

state_data <- as.data.frame(state.x77)
state_data$State <- tolower(rownames(state_data))

states_map <- map_data("state")

state_centers <- states_map %>%
  group_by(region) %>%
  summarize(Lon = mean(range(long)), Lat = mean(range(lat)))

state_data <- left_join(state_data, state_centers, by = c("State" = "region"))

ggplot(data = state_data, aes(map_id = State)) +
  geom_map(map = states_map, aes(fill = Income), color = "white") +
  expand_limits(x = states_map$long, y = states_map$lat) +
  coord_quickmap() +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(fill = "Income") +
  geom_point(aes(x = Lon, y = Lat), color = "red", size = 2)
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



(c) [25pt] NYC flight destination map

Recall `nycflights13::flights` dataset contains all 336,776 flights that departed from New York City in 2013. https://nycflights13.tidyverse.org

```
?flights # full documentation
glimpse(flights)
## Rows: 336,776
## Columns: 19
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, ~
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, ~
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1~
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,~
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,~
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1~
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "~
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4~
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394~
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",~
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",~
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1~
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, ~
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6~
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0~
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0~
```

(i) [10pt] Count 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?

Note: Recall in HW4 we have cleaned the `flights` data by removing flight record that has missing values in `dep_delay` or `arr_delay`, and save the result as the **non-canceled** flights.

```
non_canceled <- flights %>%
  filter(!is.na(dep_delay) & !is.na(arr_delay))

per_dest <- non_canceled %>%
  count(dest, name = "flight_count") %>%
  arrange(desc(flight_count))

nrow(per_dest)
```

```
## [1] 104
```

```
head(per_dest, 1)
```

```
## # A tibble: 1 x 2
##   dest  flight_count
##   <chr>        <int>
## 1 ATL          16837
```

```
tail(per_dest, 1)
```

```
## # A tibble: 1 x 2
##   dest  flight_count
##   <chr>        <int>
## 1 LEX              1
```

(ii) [15pt] Mark 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.

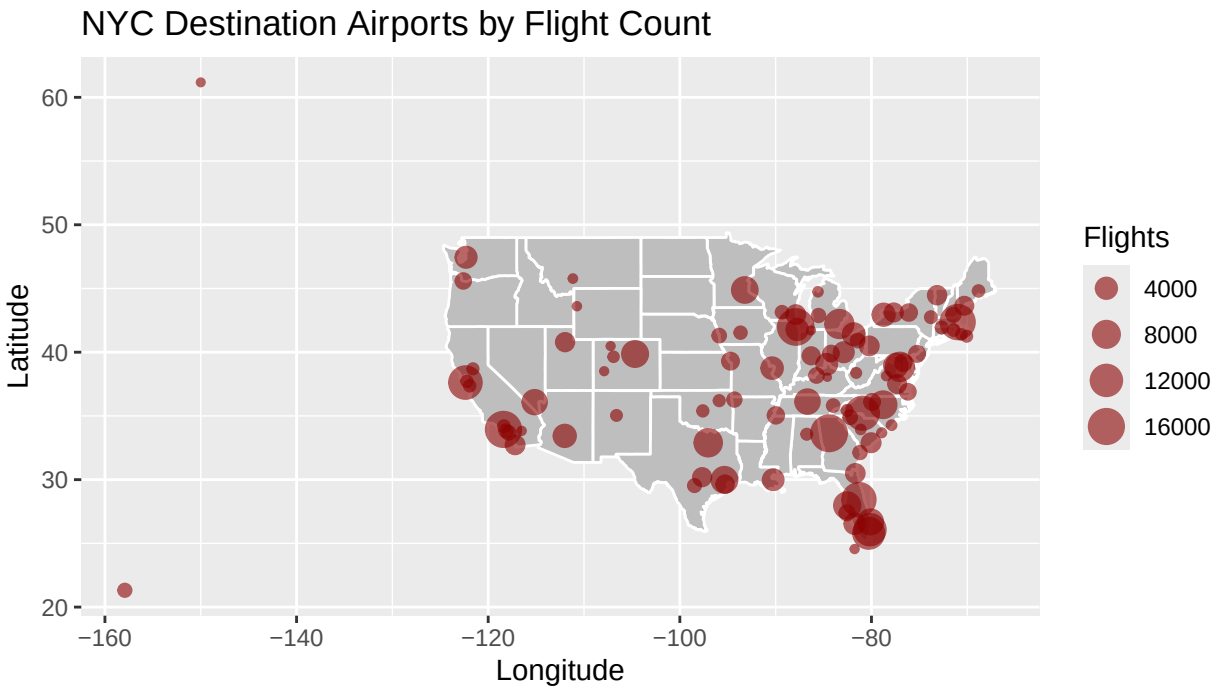
Hint: Suppose that you have saved your list of unique destinations and their corresponding flight counts in a tibble called `per_dest`. Then you can *join* `per_dest` and `airports` using destination airport FAA code as the *key*.

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

```
# I keep running into errors and id where state and region are not found even though im running previou
states_map <- map_data("state")

destination_info <- left_join(per_dest, airports, by = c("dest" = "faa"))

ggplot() +
  geom_map(data = states_map, map = states_map,
           aes(x = long, y = lat, map_id = region),
           fill = "gray", color = "white") +
  expand_limits(x = states_map$long, y = states_map$lat) +
  geom_point(data = destination_info,
             aes(x = lon, y = lat, size = flight_count),
             color = "darkred", alpha = 0.6) +
  coord_quickmap() +
  labs(title = "NYC Destination Airports by Flight Count",
       x = "Longitude", y = "Latitude", size = "Flights")
```

```
## Warning in geom_map(data = states_map, map = states_map, aes(x = long, y = lat,
## : Ignoring unknown aesthetics: x and y
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
missing_values <- destination_info %>% filter(is.na(lon) | is.na(lat))
missing_values
```

```
## # A tibble: 4 x 9
##   dest  flight_count name    lat   lon   alt    tz dst   tzone
##   <chr>        <int> <chr> <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 SJU           5773 <NA>     NA    NA    NA    NA <NA>  <NA>
## 2 BQN            888 <NA>     NA    NA    NA    NA <NA>  <NA>
## 3 STT            518 <NA>     NA    NA    NA    NA <NA>  <NA>
## 4 PSE            358 <NA>     NA    NA    NA    NA <NA>  <NA>
```

Pay attention to the warning message. Which destination airports have missing values?

SJU, BQN, STT, PSE

Question 2 [55pt] Simple linear regression

(a) [10pt] Population regression line vs least squares regression line

In this question, we will reproduce the left panel of Figure 3.3 in the **ISLR** textbook.

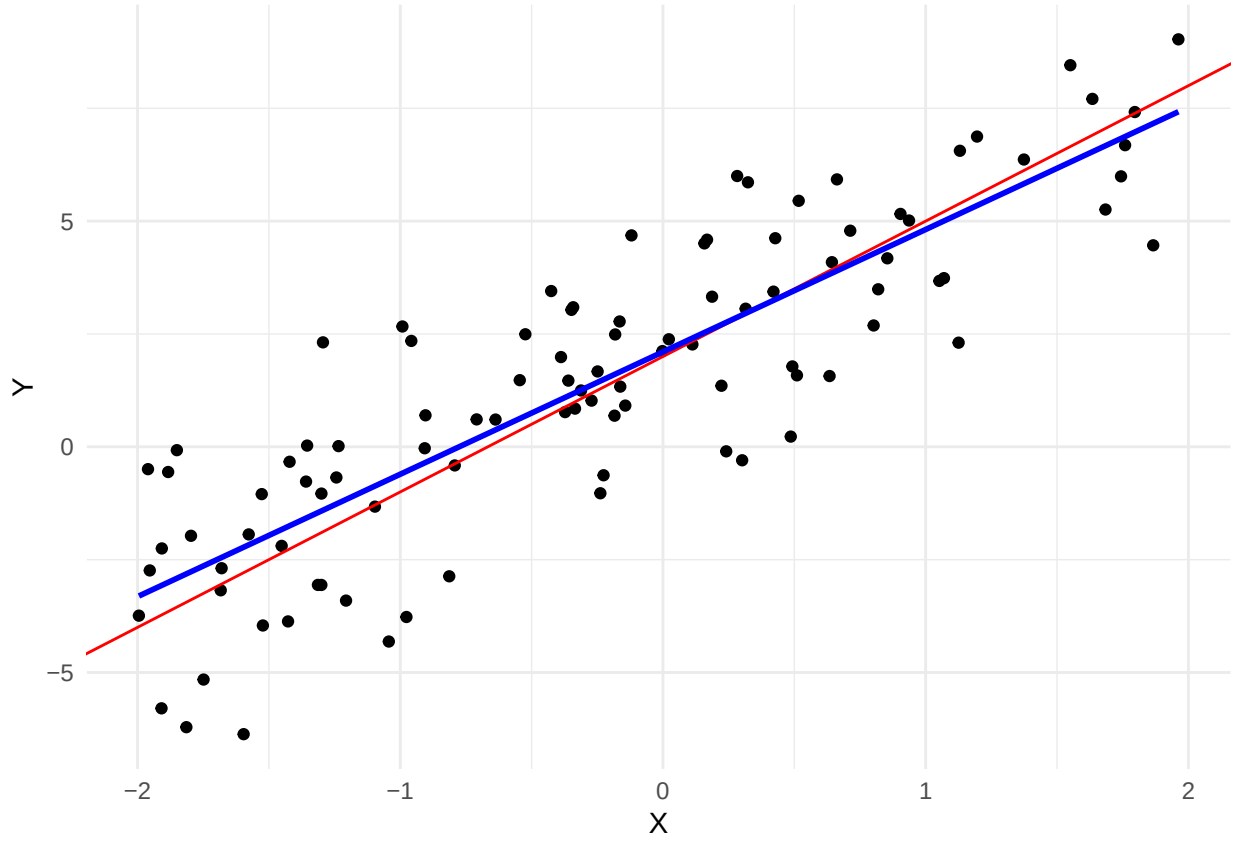
Write your own R code the simulate 100 data points from the following model.

$$Y = 2 + 3X + \epsilon$$
$$X \sim Uniform(A = -2, B = 2)$$
$$\epsilon \sim N(\mu = 0, \sigma = 2)$$

Then use `ggplot2` to make a scatter plot of the 100 data points $(x_1, y_1), \cdots (x_{100}, y_{100})$. Add the population regression line $Y = 2 + 3X$ in red color; add the least squares fit of your simulated data as a blue line.

```
set.seed(167)
X <- runif(100, min = -2, max = 2)
epsilon <- rnorm(100, mean = 0, sd = 2)
Y <- 2 + 3 * X + epsilon
data <- data.frame(X, Y)

library(ggplot2)

ggplot(data, aes(x = X, y = Y)) +
  geom_point(color = "black") +
  geom_abline(slope = 3, intercept = 2, color = "red") +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
# Always remember to set the random seed for simulation studies
set.seed(167) # feel free to change 167 to your lucky number
```

(b) [10pt] Estimate the coefficients β_0 and β_1

In the lecture, we have learned that the analytical solution of the least squares regression $Y = \beta_0 + \beta_1 X + \epsilon$ is

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Write your own R code to calculate $\hat{\beta}_0$ and $\hat{\beta}_1$ using the above equations. Compare your results with the coefficients calculated by the `lm` function. Are they the same?

```r
X_bar <- mean(X)
Y_bar <- mean(Y)
beta_1 <- sum((X - X_bar) * (Y - Y_bar)) / sum((X - X_bar)^2)
beta_0 <- Y_bar - beta_1 * X_bar

lm_model <- lm(Y ~ X)
summary(lm_model)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1423 -1.2995  0.2294  1.1943  3.7195
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1035     0.1855   11.34   <2e-16 ***
## X             2.7112     0.1672   16.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.8 on 98 degrees of freedom
## Multiple R-squared:  0.7286, Adjusted R-squared:  0.7258
## F-statistic:   263 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
beta_1
```

```
## [1] 2.711247
```

```
beta_0
```

```
## [1] 2.103547
```

```
# They are not the same. Similar though
```

(c) [20pt] Assess accuracy of the regression model

Recall that the accuracy of a linear regression fit is typically assessed using the **residual standard error (RSE)** and the R^2 **statistic**.

(i) [10pt] Calculate the residual standard error (RSE) Residual standard error (RSE) is an estimate of σ, the standard deviation of ϵ.

$$\begin{aligned}\text{RSE} = \hat{\sigma} &= \sqrt{\frac{1}{n-2}\text{RSS}} \\ &= \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\end{aligned}$$

Write your own R code to calculate RSE using the above equation. Compare your result with the residual standard error calculated by the `lm` function. Are they the same? How does the RSE value relate to our noise model $\epsilon \sim N(\mu = 0, \sigma = 2)$?

```r
y_hat <- predict(lm_model)
RSS <- sum((Y - y_hat)^2)
RSE <- sqrt(RSS / (length(Y) - 2))

summary(lm_model)$sigma
```

```
## [1] 1.79954
```

```r
# The RSE from both methods should be very close to each other.

# Since we get N(0,2), the RSE value should be roughly 2 since it should reflect the standard deviation
```

(ii) [10pt] Calculate the R^2 statistics Write your own R code to compute the R^2 value using the following equations.

$$\text{TSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \hat{f}(x_i)\right)^2$$

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

Compare your result with the R-squared value calculated by the `lm` function. Are they the same?

```r
TSS <- sum((Y - Y_bar)^2)
RSS <- sum((Y - y_hat)^2)
R_squared <- 1 - (RSS / TSS)

summary(lm_model)$r.squared
```

```
## [1] 0.7285655
```

```r
# They are the same
```

(d) [15pt] Simulations of least squares regression

Now, let's repeat the simulation 10 times, and reproduce the right panel of Figure 3.3 in the **ISLR** textbook.

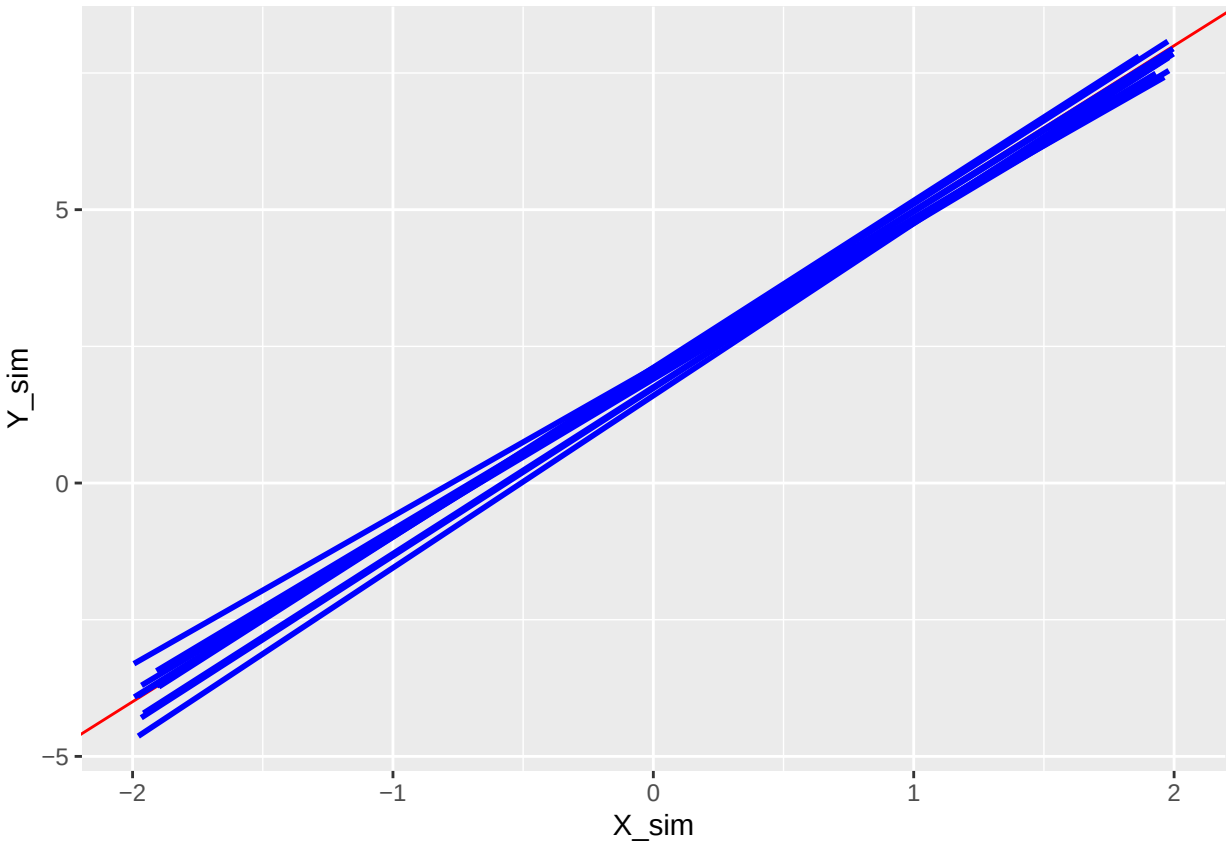
Notes:

- You can draw all the least squares lines using the same blue color.
- Write your code using a `for` loop, rather than manually copying and pasting your code 10 times.
- You can save your ggplot to a variable (e.g. `gg`) and keep adding new layers inside your for loop. So the pseudocode will look like:

```r
gg <- ggplot() +
  geom_abline(draw the population regression line)

for ( every simulation ) {
  ## add your simulation code here
  gg <- gg +
    geom_smooth(draw the sample regression line) ## alternatively you can use geom_abline()
}

gg ## call gg at the end to plot it.
```

```r
gg <- ggplot() +
  geom_abline(slope = 3, intercept = 2, color = "red")

for (i in 1:10) {
  X_sim <- runif(100, -2, 2)
  epsilon_sim <- rnorm(100, mean = 0, sd = 2)
  Y_sim <- 2 + 3 * X_sim + epsilon_sim
  model_sim <- lm(Y_sim ~ X_sim)
  gg <- gg + geom_smooth(data = data.frame(X_sim, Y_sim), aes(x = X_sim, y = Y_sim), method = "lm", se =
}

gg
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```


Y_sim
5
0
−5
−2
−1
0
1
2
X_sim