Homework 2 - Regression

Sicily Xie 54385315

Due 18 October at 11pm

Instructions

- Use the space inside of

```
::: {.solbox data-latex=""}
:::
```

to answer the following questions.

- Do not move this file or copy it and work elsewhere. Work in the same directory.
- Use a new branch named whatever you want. Create it now! Can't come up with something, try here. Make a small change, say by adding your name. Commit and push now!
- Try to Knit your file now. Are there errors? Fix them now, not at 11pm on the due date.
- There MUST be some text between `::: {.solbox}` and the next `:::` or this will fail to Knit.
- If your code or figures run off the .pdf, you'll lose 2 points automatically.

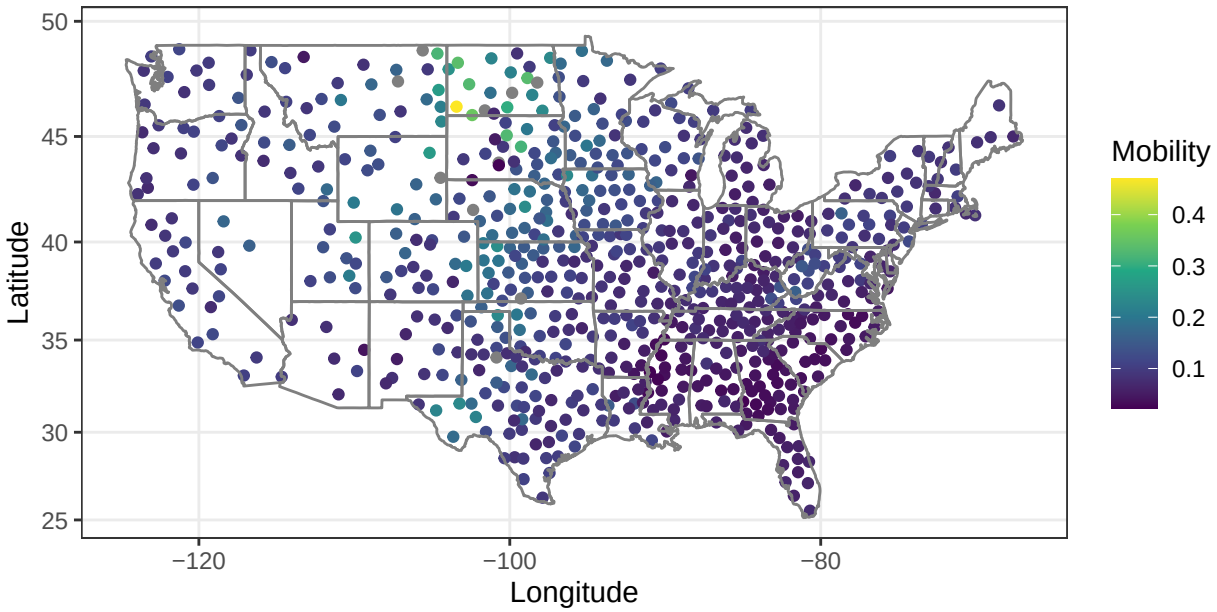
Regularized methods

In this section we will conduct an analysis of the `mobility` data first visited during the first week of class. There is a file `mobility.html` in your repo which gives descriptions of all the variables. Load it in a web browser to find out about the covariates available to you.

This assignment will look at economic mobility across generations in the contemporary USA. The data come from a large study, based on tax records, which allowed researchers to link the income of adults to the income of their parents several decades previously. For privacy reasons, we don't have that individual-level data, but we do have aggregate statistics about economic mobility for several hundred communities, containing most of the American population, and covariate information about those communities. We are interested in predicting economic mobility from the characteristics of communities. Note that some observations are missing values for some covariates.

1. The following code generates a map of `Mobility` ignoring Alaska and Hawaii. Describe the geographic pattern in words.

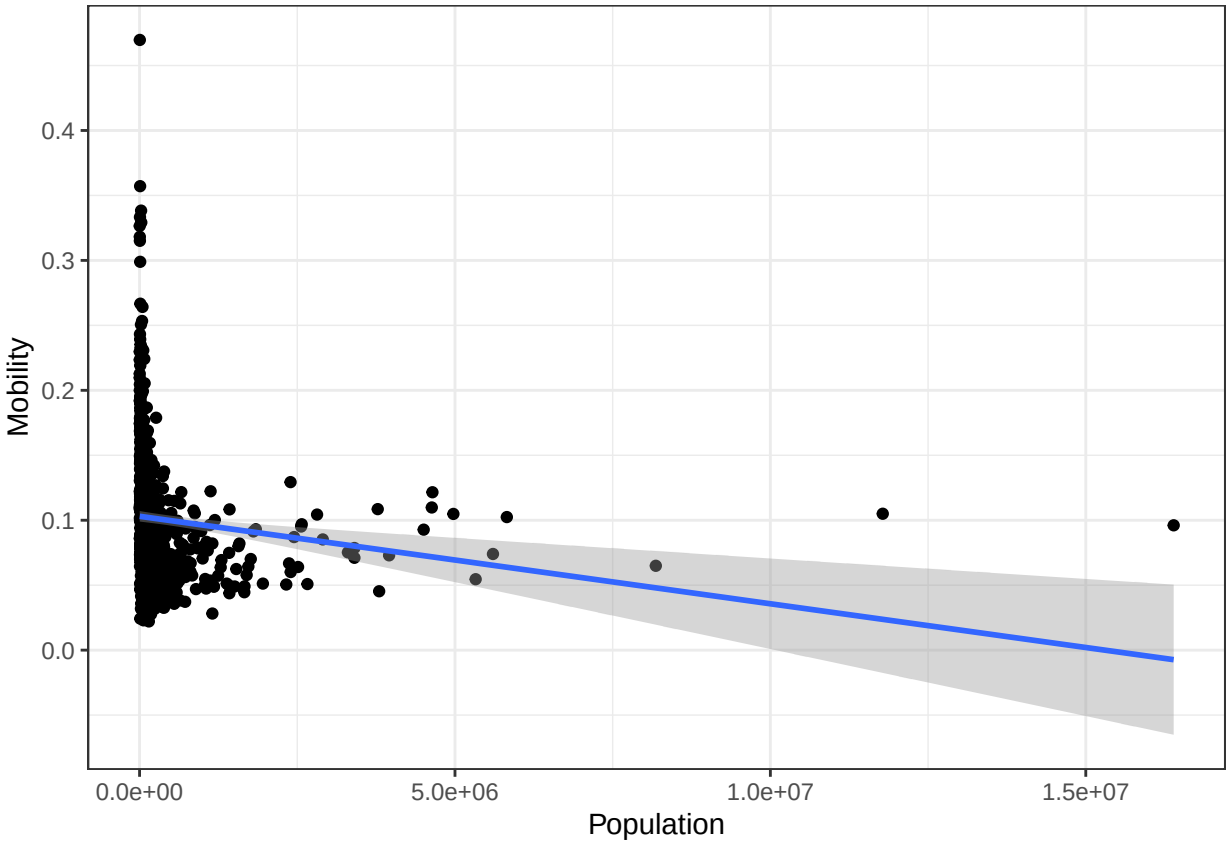
```
ggplot(
  mobility %>% filter(!(State %in% c("AK", "HI") )),
  aes(x=Longitude, y=Latitude, color = Mobility)) +
  geom_point() +
  coord_map() +
  borders("state") +
  scale_color_viridis_c()
```

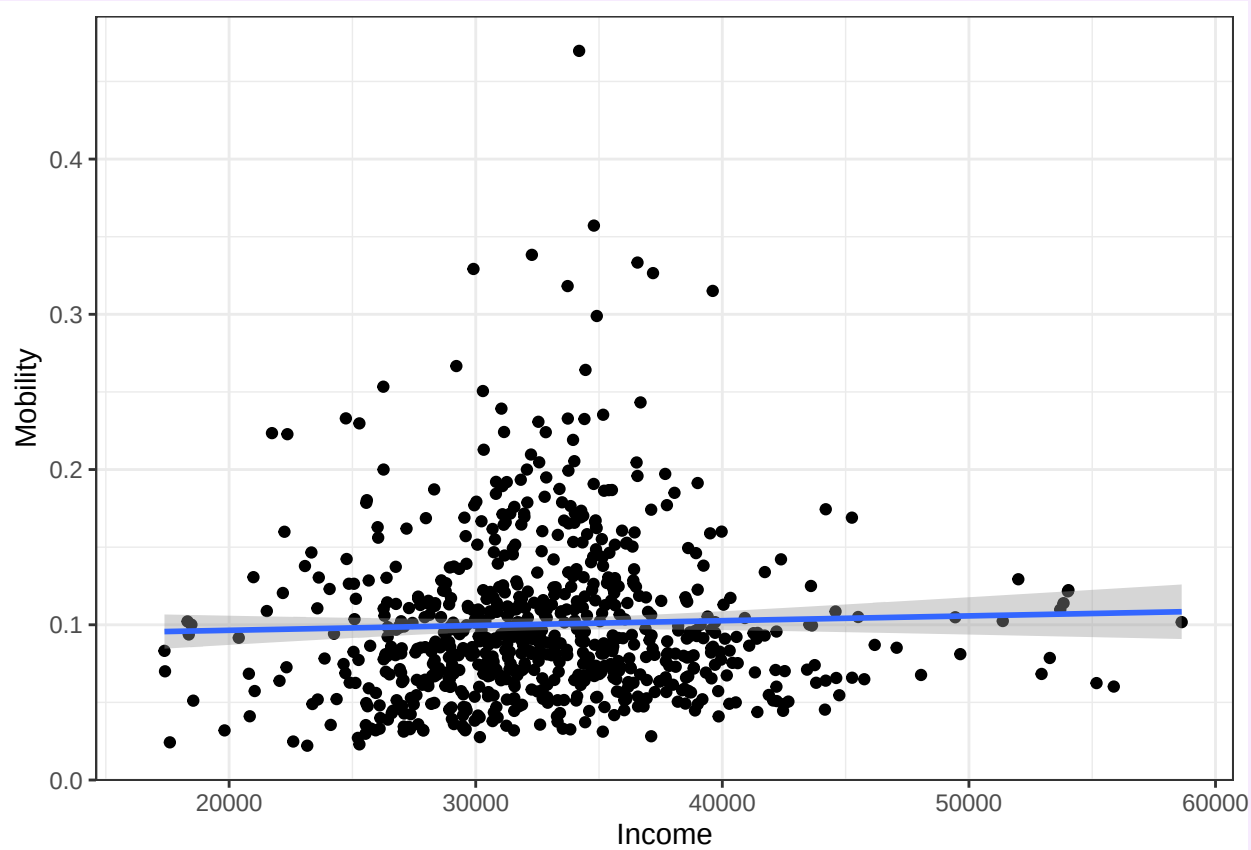


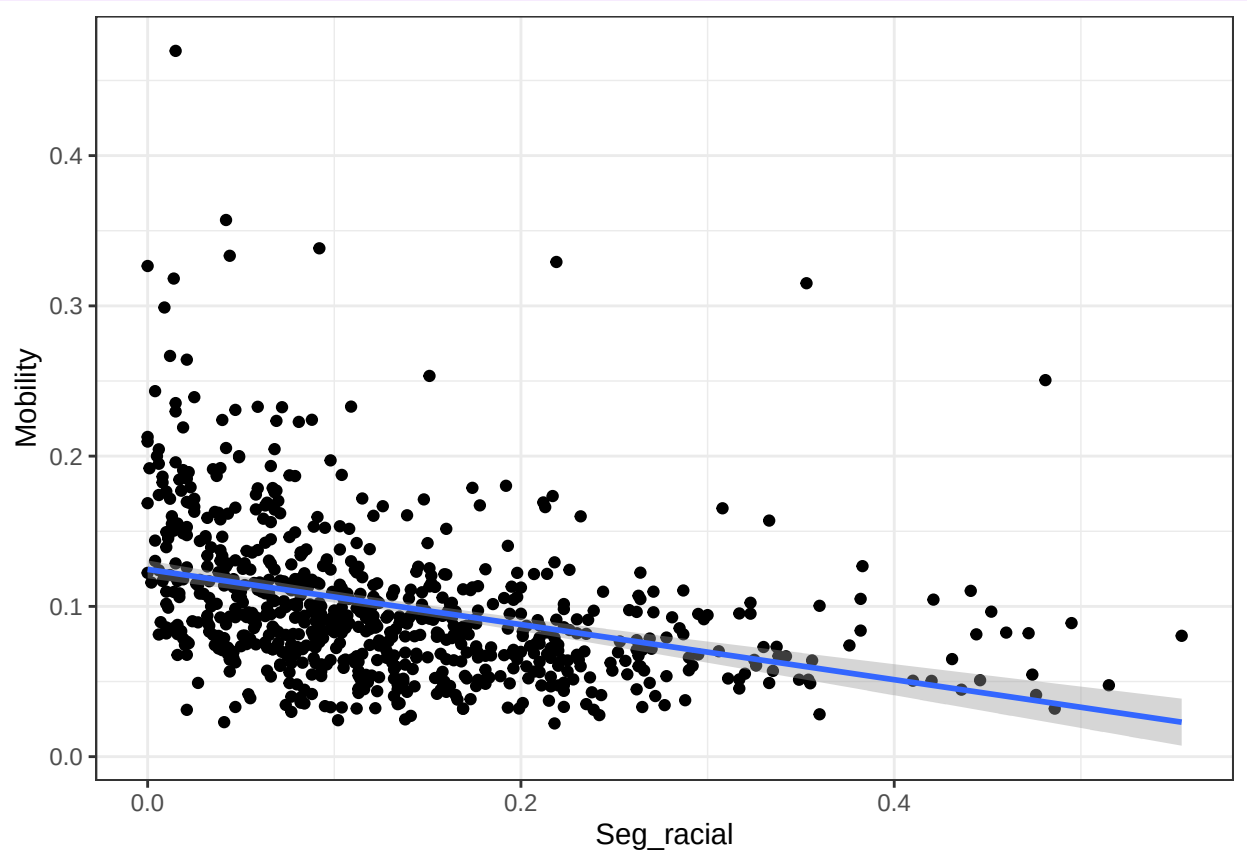
As the longitude gets larger and larger and the latitude gets lower and lower, the value of mobility is getting smaller as well, which means that economic mobility across generations moves slower.

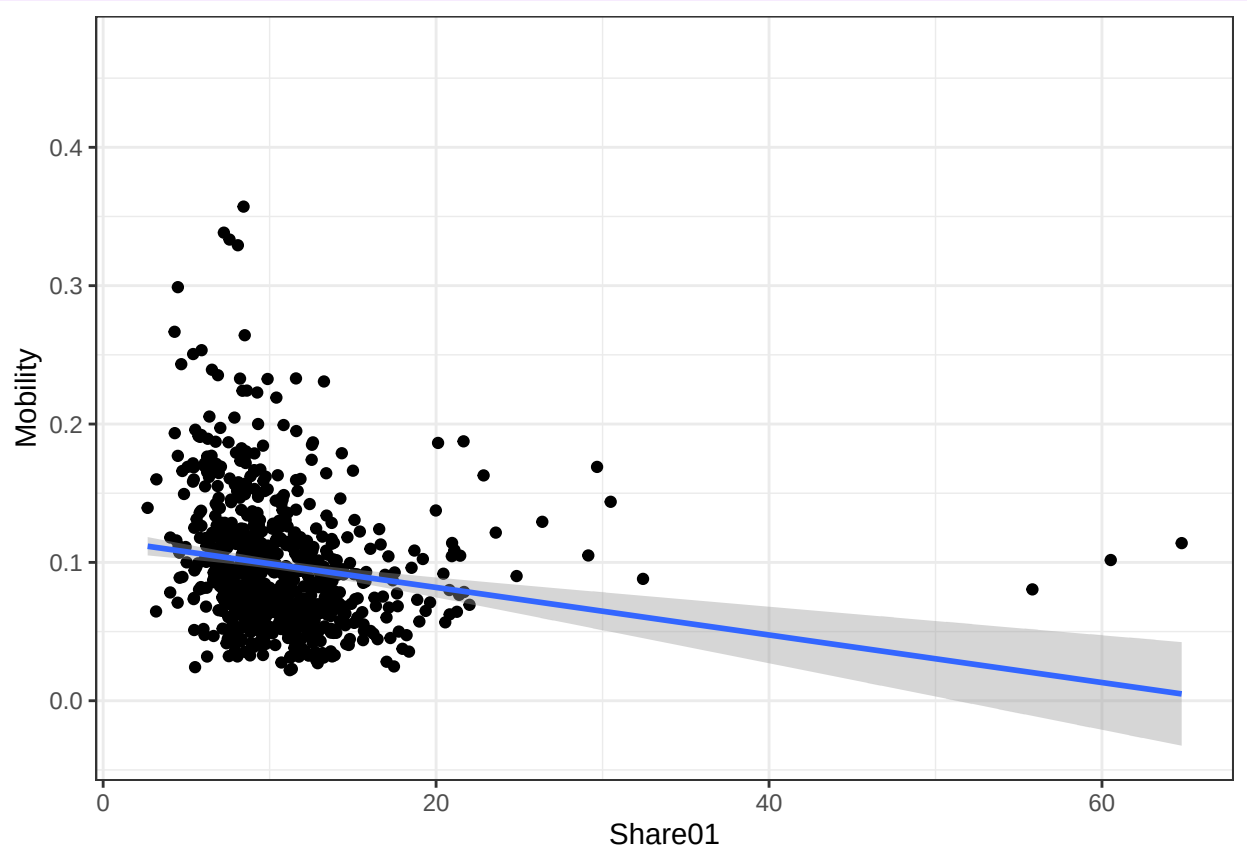
2. Make scatter plots of mobility against each of the following variables (by "plot mobility against B" we mean "put mobility on the y-axis and B on the x-axis"): `Population`, `Income`, `Seg_racial`, `Share01`, `School_spending`, `Violent_crime`, and `Commute`. Include on each plot a line for the univariate regression of mobility on the variable, and give a table of the slope coefficients. Carefully explain the interpretation of each coefficient in the context of the problem. You likely need to look at the documentation to determine what these variables mean. **Hint:** This can be done easily with `ggplot() + geom_smooth()`. See the Examples here. If you choose a different route, I suggest writing a function.

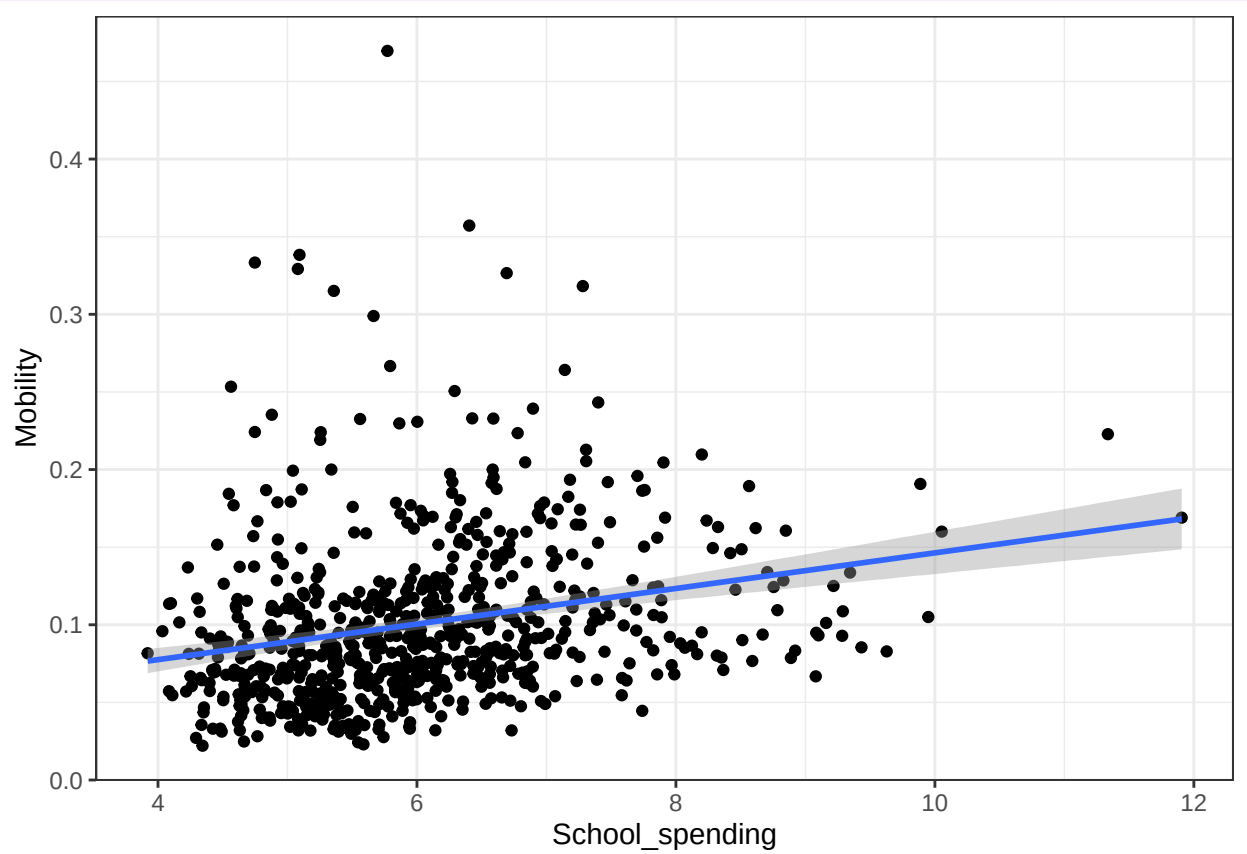
```
x <- c("Population", "Income", "Seg_racial", "Share01", "School_spending",
"Violent_crime", "Commute")
slope = c()
for (i in 1:7) {
p <- ggplot(mobility, aes_string(x = x[i], y = "Mobility")) +
geom_point() + geom_smooth(method = "lm")
print(p)
slope[i] <- coef(lm(paste("Mobility ~", x[i]), data = mobility))[2]
}
```

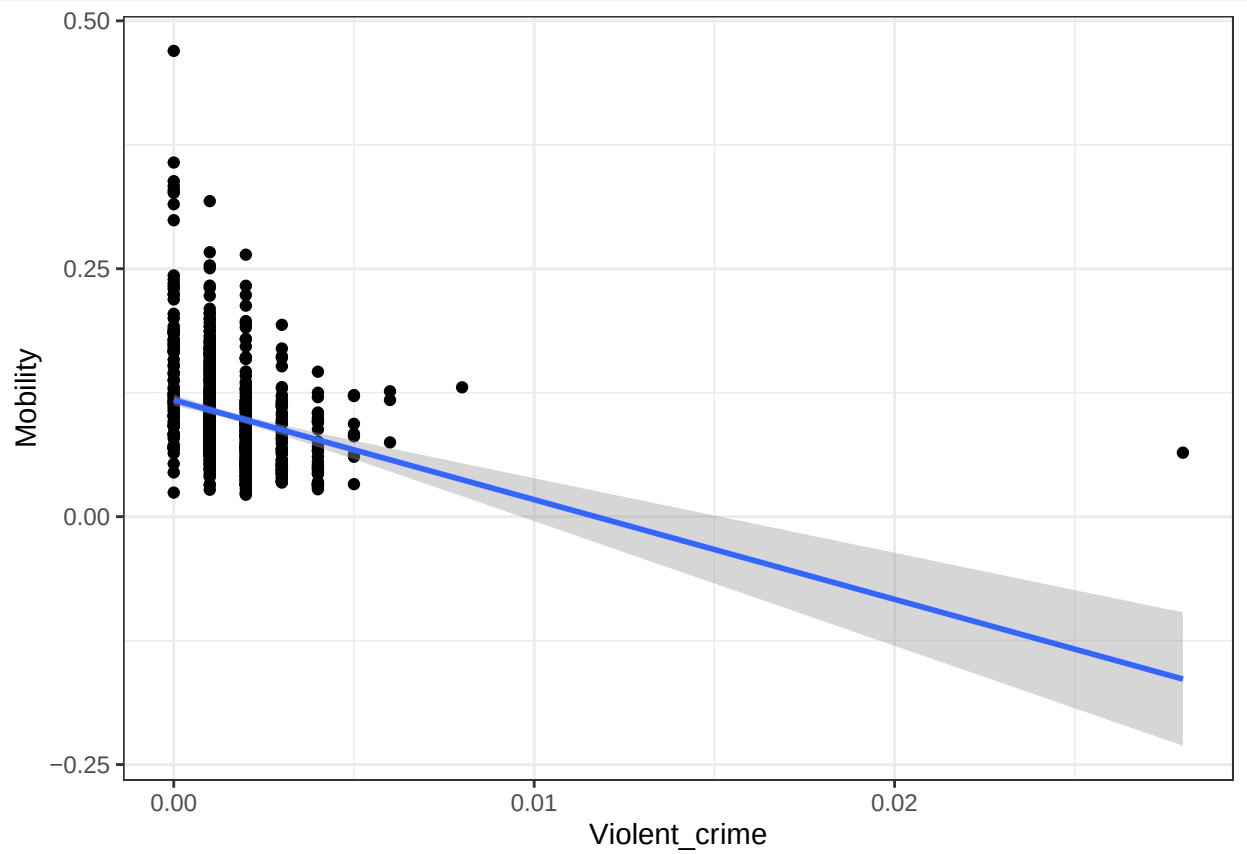


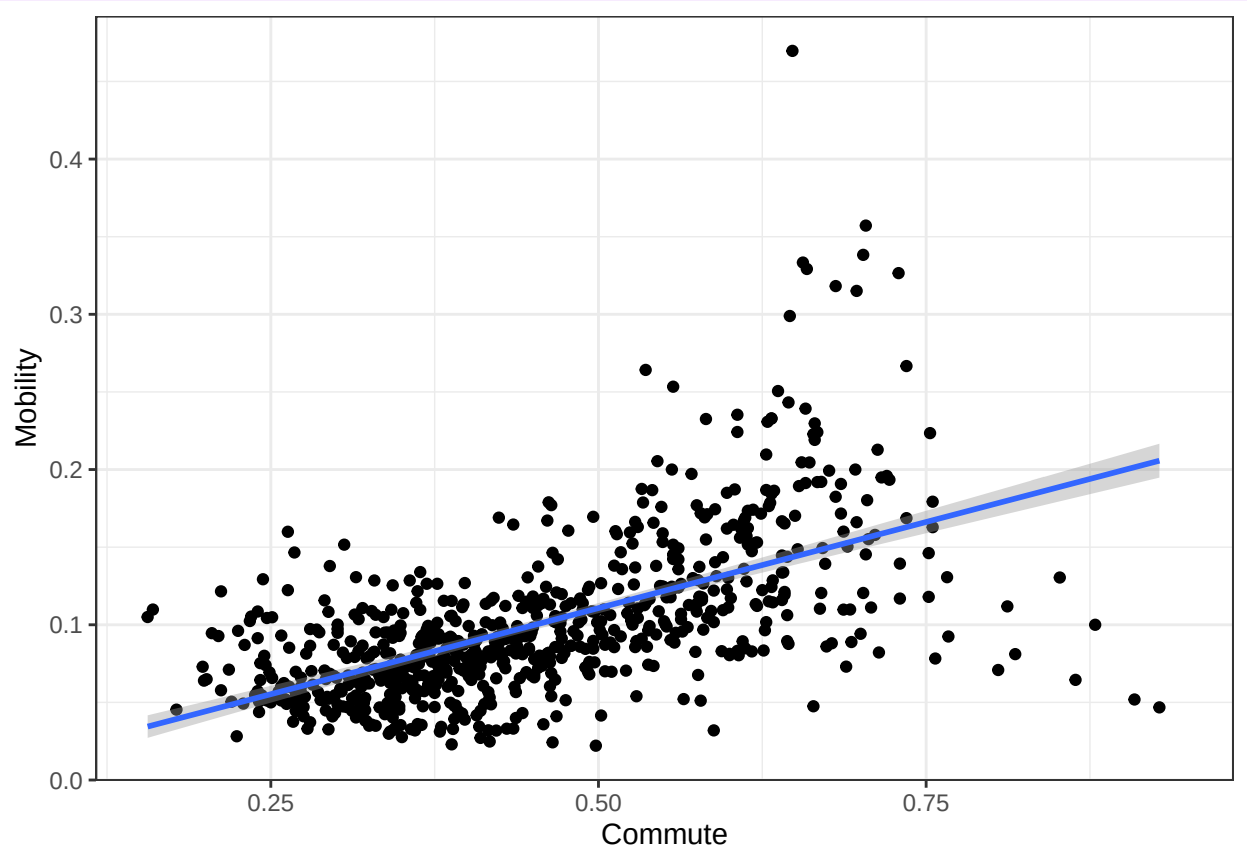

Mobility
0.0
0.1
0.2
0.3
0.4
20000
30000
40000
50000
60000
Income


Mobility
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
Seg_racial


Mobility
0.4
0.3
0.2
0.1
0.0
0
20
40
60
Share01


Mobility
0.0
0.1
0.2
0.3
0.4
4
6
8
10
12
School_spending


0.50
0.25
0.00
−0.25
Mobility
0.00
0.01
0.02
Violent_crime




```
names(slope) <- x
as.table(slope)
```

```
##      Population          Income      Seg_racial         Share01 School_spending
##   -6.732228e-09    3.094097e-07   -1.835019e-01   -1.718199e-03    1.146429e-02
##   Violent_crime         Commute
##   -1.005057e+01    2.218695e-01
```

Slope of "Population": -6.732228e-09 means that if we increase one unit of population, the predicted mobility would decrease by 6.732228e-09 unit. Slope of "Seg_racial": -1.835019e-01 means that if we increase one unit of Seg_racial, the predicted mobility would decrease by 1.835019e-01 unit. Slope of "Share01": -1.718199e-03 means that if we increase one unit of Share01, the predicted mobility would decrease by 1.718199e-03 unit. Slope of "Violent_crime": -1.005057e+01 means that if we increase one unit of Violent_crime, the predicted mobility would decrease by 1.005057e+01 unit. Slope of "Income": 3.094097e-07 means that if we increase one unit of Income, the predicted mobility would increase by 3.094097e-07 unit. Slope of "School_spending": 1.146429e-02 means that if we increase one unit of School_spending, the predicted mobility would increase by 1.146429e-0 unit. Slope of "Commute": 2.218695e-01 means that if we increase one unit of Commute, the predicted mobility would increase by 2.218695e-01 unit.

3. Run a linear regression of `mobility` against all appropriate covariates.
    a. Report all regression coefficients and their standard errors to reasonable precision; you may use either a table or a figure as you prefer. Do not just paste in R's output.
    b. Explain why the `ID` variable must be excluded.
    c. Explain which other variables, if any, you excluded from the regression, and why. (If you think they can all be used, explain why.)
    d. Compare the coefficients you found in problem 2 to the coefficients for the same variables in this regression. Are they much different? Have any changed sign?

```
temp <- subset(mobility, select = -c(ID, State, Name))
ols <- lm(Mobility ~ . , data = temp) # YOU NEED TO CHANGE THIS LINE!!
```

a.

```
temp2 <- coef(summary(ols))[,c(1,2)]
temp2
```

```
##                                 Estimate   Std. Error
## (Intercept)                 1.766119e-01 7.220590e-02
## Population                  1.560626e-09 2.171557e-09
## Urban                       1.568268e-03 3.514804e-03
## Black                       8.856449e-02 2.539824e-02
## Seg_racial                 -4.837401e-02 1.654400e-02
## Seg_income                  1.064474e+00 8.313054e-01
## Seg_poverty                -8.582205e-01 4.470597e-01
## Seg_affluence              -3.178434e-01 4.163419e-01
## Commute                     7.548455e-02 2.551385e-02
## Income                      3.058754e-07 5.984096e-07
## Gini                        2.929105e+00 2.887916e+00
## Share01                    -2.936911e-02 2.888869e-02
## Gini_99                    -3.032795e+00 2.888089e+00
## Middle_class                8.649399e-02 4.264554e-02
## Local_tax_rate              1.329218e-01 2.378940e-01
## Local_gov_spending          9.928755e-07 2.761021e-06
## Progressivity               5.601685e-03 1.119211e-03
## EITC                       -5.896938e-04 4.092169e-04
## School_spending            -1.286334e-03 2.066442e-03
## Student_teacher_ratio      -5.020417e-04 1.021261e-03
## Test_scores                 4.603429e-04 2.757986e-04
## HS_dropout                 -1.917857e-01 7.678662e-02
## Colleges                   -1.052767e-01 7.219088e-02
## Tuition                    -3.329211e-08 4.002036e-07
## Graduation                 -1.385688e-02 1.264213e-02
## Labor_force_participation -6.894593e-02 4.756330e-02
## Manufacturing              -1.727035e-01 2.528346e-02
## Chinese_imports            -8.122153e-04 6.989214e-04
## Teenage_labor              -2.125416e+00 1.927565e+00
## Migration_in               -8.819434e-02 2.763284e-01
```

```
## Migration_out             -5.249081e-01 3.379638e-01
## Foreign_born               1.070508e-01 4.982707e-02
## Social_capital            -2.021243e-03 2.430426e-03
## Religious                  6.082179e-02 1.157155e-02
## Violent_crime             -3.193596e+00 1.481136e+00
## Single_mothers            -3.469003e-01 8.330516e-02
## Divorced                   7.964103e-02 1.417140e-01
## Married                   -8.914393e-02 6.706034e-02
## Longitude                  1.129333e-04 2.048594e-04
## Latitude                   1.423615e-03 5.311832e-04
```

b. ID variable in our data set represents a numerical code to identify the community. Each observation has the unique one, and it is a categorical variable. So we have n-1 dummy variables and need to compute n-1 beta and use all the degree of freedoms, so the ID variable must be excluded.

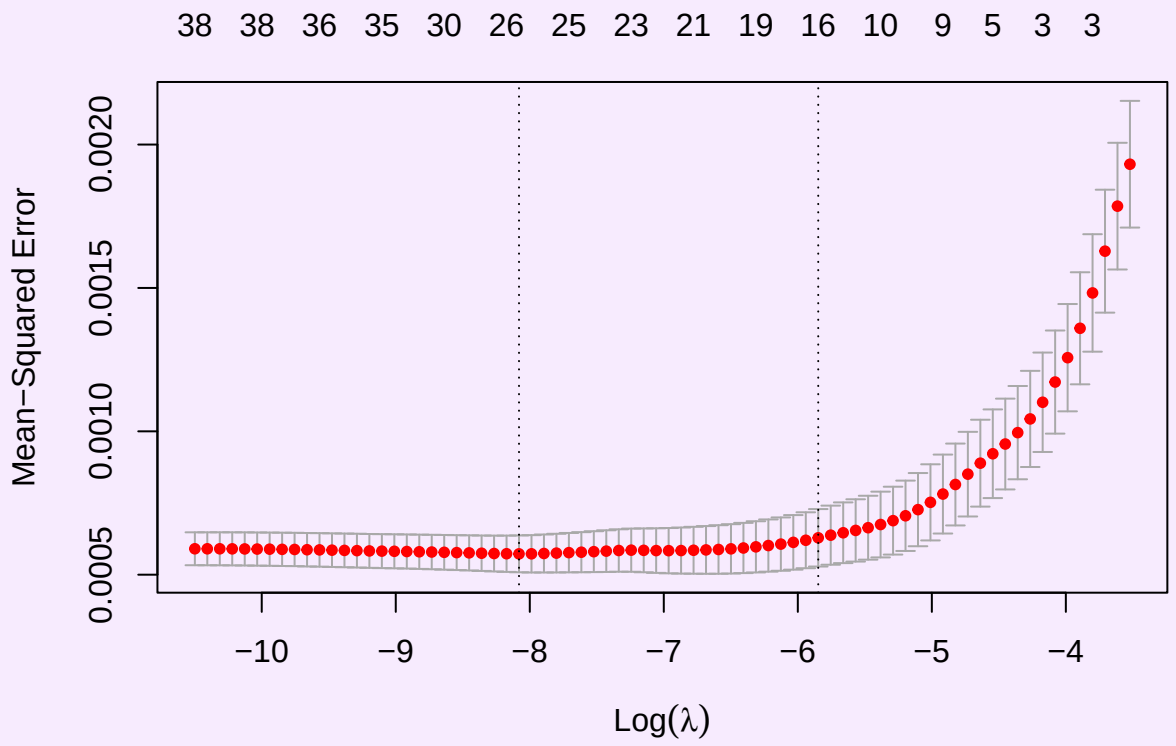
c. Name and State variables should exclude from the regression as well since they are categorical variables and the reason that we need to exclude is the same as question 3b.

d. The values of the coefficients are different from problem 2, whereas the signs are the same. In problem 2, we compute each single x variable against y, the correlation between each x and y. In problem 3, we compute the coefficients by y and all the x variables.

4. With all the covariates you used in the previous question, use ridge regression and lasso (with the `{glmnet}` package). Use cross validation as implemented in `cv.glmnet()`.
    a. Plot the CV curve for Lasso. Explain the difference between the two vertical lines shown on the figure. What are the numbers on the top of the plot?
    b. Plot the coefficient trace for ridge regression. What does "L1 norm" on the x-axis mean? Are any coefficient estimates exactly 0 for any value of the penalty parameter? As $\lambda \to 0$, which way do I move on the x-axis?

a.

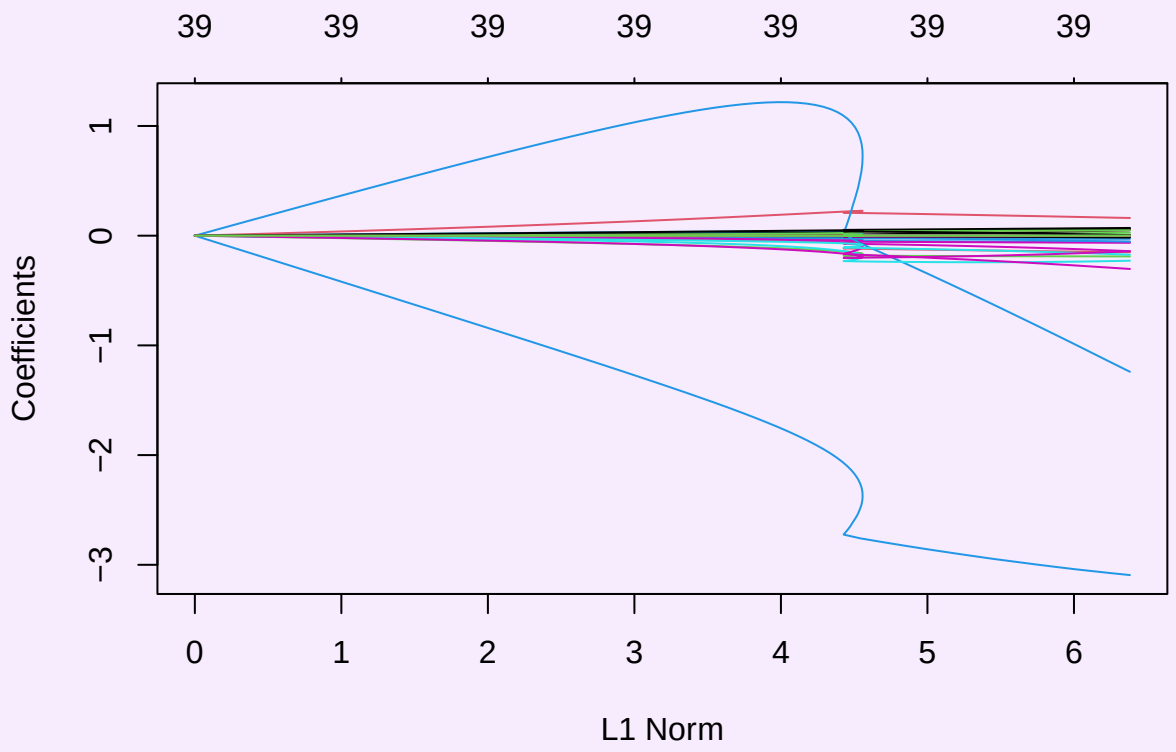
```
library(glmnet)
x <- model.matrix(ols) # grabs the design matrix from the internal lm() output
x <- x[, -1] # remove the intercept column, glmnet() will do this for us
y <- mobility$Mobility[-ols$na.action] # remove those observations with
# missing values. glmnet() hates those. They are already dropped in x.
set.seed(01101101)
lasso <- cv.glmnet(x, y, alpha = 1)
plot(lasso)
```



The left vertical line shows the log value of λ when the MSE is small. The right vertical line delivers the log value of λ when the MSE is 1 unit standard error more significant than the smallest MSE. The top right numbers of the plot are the numbers of covariates of the model.

b.

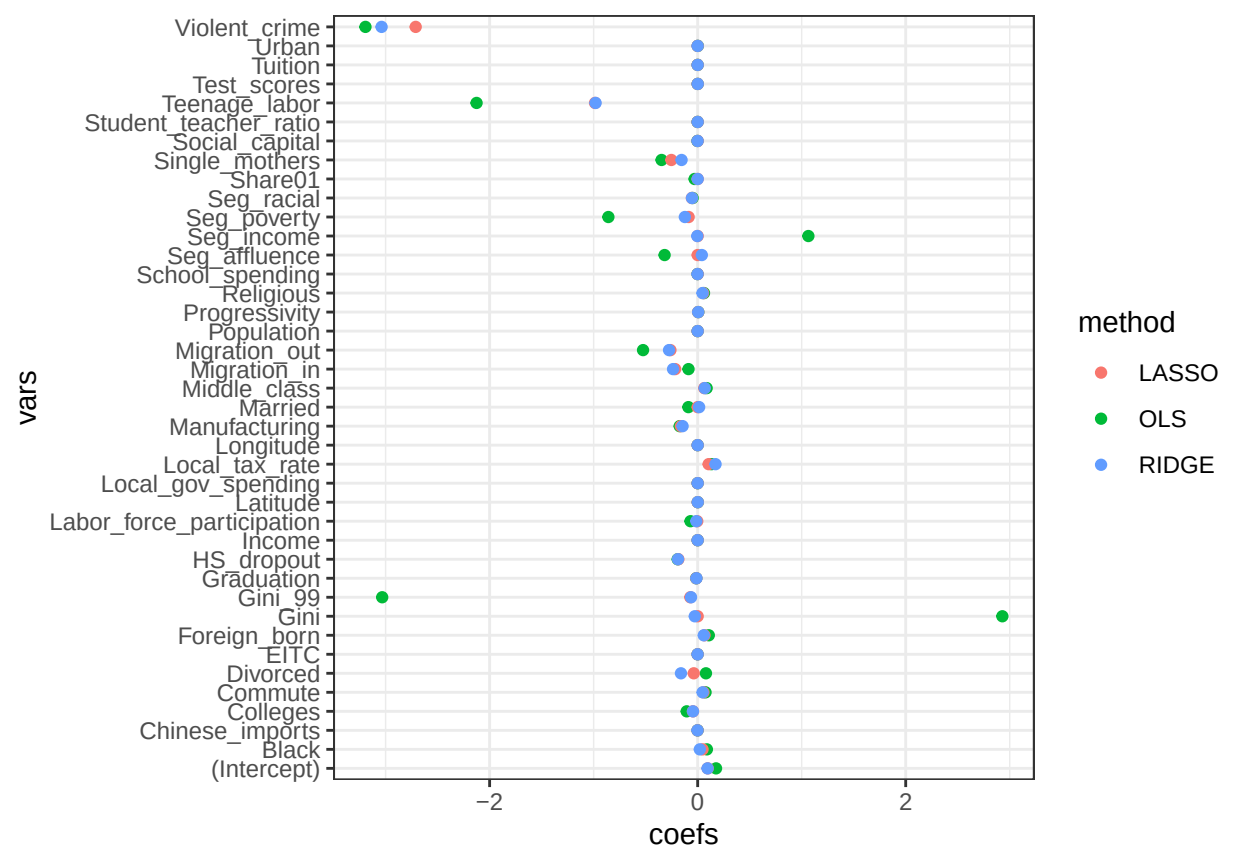
```
ridge <- glmnet(x, y, alpha = 0)
ridge_cv <- cv.glmnet(x, y, alpha = 0)
plot(ridge)
```



L1 Norm on the x-axis means the sum of absolute estimated beta vector values. There are no coefficient estimates of 0 for any value of the penalty parameter. As λ approaches 0, we should move right on the x-axis.

5. For both the Ridge regression and Lasso regression in the previous section, choose the set of coefficient estimates using `lambda.min`. Compare these coefficient estimates with those in problem 3 by producing a graph (put the Variable names on the y-axis, the coefficient estimate on the x-axis, and use color or shape to denote the three methods). Describe what you see.

```
lasso_min <- glmnet(x, y, alpha = 1, lambda = lasso$lambda.min)
ridge_min <- glmnet(x, y, alpha = 0, lambda = ridge_cv$lambda.min)
temp <- data.frame(coefs = c(coef(ols),coef(lasso_min)[,1], coef(ridge_min)[,1]),
vars = rep(c(rownames(coef(ridge_min))),3), method = rep(c("OLS", "LASSO", "RIDGE"),
each = ncol(x) + 1))
p <- ggplot(temp, aes(coefs, vars, color = method)) + geom_point()
p
```



As shown above, all the coefficient estimates by ridge regression are close to 0, while some of the coefficient estimates by lasso are 0, and all the absolute values of coefficients by OLS are larger than ridge regression and lasso.

6. Calculate the LOOCV score for your OLS model in problem 3. Compare this score with the scores from problem 4. Select 1 of these three models to use for out-of-sample prediction. Explain why you chose this model. For your chosen model, make a map of the residuals (borrow code from problem 1). Describe any patterns you see. But change the coloring so that 0 is white (use `scale_color_gradient2()`)

```
set.seed(123456)
loocv <- mean((residuals(ols) / (1 - ls.diag(ols)$hat))^2)
loocv
```

```
## [1] 0.0005750473
```

```
min(ridge_cv$cvm)
```

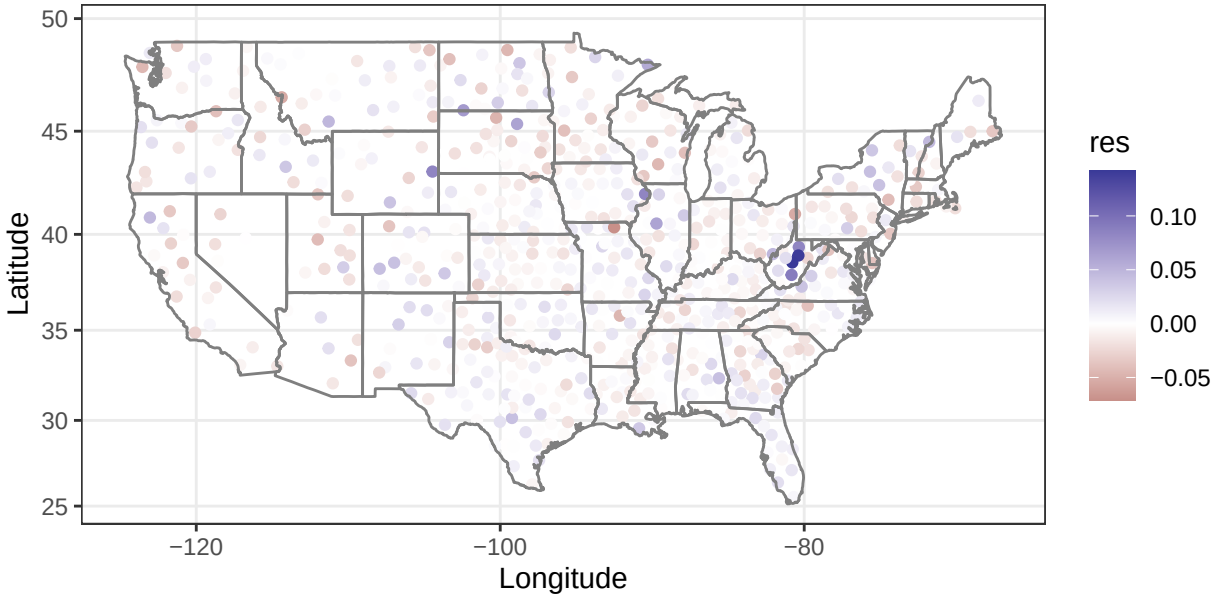
```
## [1] 0.0005680506
```

```
min(lasso$cvm)
```

```
## [1] 0.0005726274
```

Ridge regression has the minimumn CV, we use ridge for out-of-sample prediction.

```
mobility$res[(1:nrow(mobility))] <- y-predict(ridge_min, newx = x)[, 1]
mobility %>%
filter(!(State %in% c("AK", "HI"))) %>%
ggplot(aes(Longitude, Latitude, color = res)) +
geom_point() + coord_map() + borders("state") +
scale_color_gradient2()
```



The color of the middle of the map is lighter than the color in the west and east. Therefore, the prediction for the center of the map is better than it for the west and east. However, it may cause by the high mobility in the middle of the map.

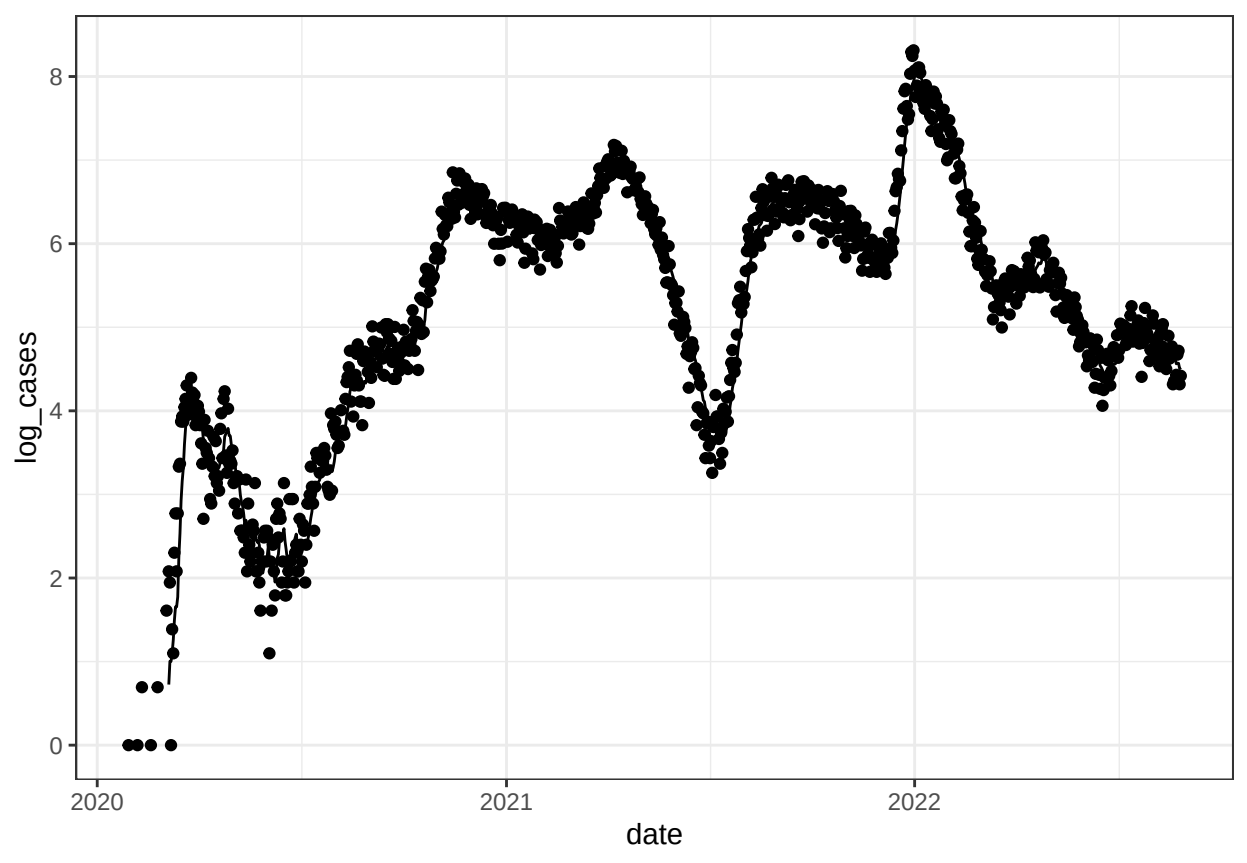
Nonparametric regression

It may be helpful to read through the Kernel Regression section of the worksheets before starting this section.

```
data("bccovid")
```

1. Plot the data on the log-scale along with a 7-day trailing average: $\overline{y}_i = \sum_{j=i-6}^{i} y_j$. This is one of the most common ways to "smooth" this data (see e.g. CBC). If you install the `R` package `{zoo}`, there's a function to do this easily with appropriately chosen arguments (though you can write your own!). How well does this smoother "track" the data? Do you notice any discrepancies? Issues? It may be helpful to look at the most recent 2 months of data to answer these questions.

```
library("zoo")
bccovid$log_cases <- log(bccovid$cases)
bccovid$avg_log_cases <- rollmean(bccovid$log_cases, 7, align = "right", fill = NA)
ggplot(bccovid) + geom_point(aes(date, log_cases)) + geom_line(aes(date, avg_log_cases))
```



The smoother tracks the data well, whereas for some peaks and valleys, the smoother is a little bit rough. In the most recent two months of data, the smoother is not very smooth.

2. The 7-day trailing average is a "linear smoother". Complete the function below that creates the smoothing matrix for any dimension `n` and any "window" of days `k`. You may assume that the data is equally spaced and arranged in increasing order. Think carefully about how to handle the first few rows. Your resulting matrix must be square. Evaluate your function for `n = 8`, and `k = 3` (round the entries to 2 decimals so it prints nicely).

```
trail_mean_mat <- function(n, k = 3) {
  stopifnot(n == floor(n), k == floor(k), n > 0, k > 0) # check for valid inputs
  mat <- matrix(0, nrow = n, ncol = n) # preallocate space
  for (i in 1:n) { # loop over rows
    idx <- max(1, (i - k + 1)):i # which columns are nonzero?
    denom <- length(idx)
    mat[i, idx] <- 1 / denom
  }
  mat
}
round(trail_mean_mat(8, 3), 2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,] 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## [2,] 0.50 0.50 0.00 0.00 0.00 0.00 0.00 0.00
## [3,] 0.33 0.33 0.33 0.00 0.00 0.00 0.00 0.00
## [4,] 0.00 0.33 0.33 0.33 0.00 0.00 0.00 0.00
## [5,] 0.00 0.00 0.33 0.33 0.33 0.00 0.00 0.00
## [6,] 0.00 0.00 0.00 0.33 0.33 0.33 0.00 0.00
## [7,] 0.00 0.00 0.00 0.00 0.33 0.33 0.33 0.00
## [8,] 0.00 0.00 0.00 0.00 0.00 0.33 0.33 0.33
```

3. What is the effective degrees of freedom of the "trailing average" for $n = 50$ and $k = 10$? For n large relative to k, what does this look like for arbitrary k? (I'm being a bit hand-wavy here. You can examine $lim_{n\to\infty} edf(n,k)/n$ and convert to get the answer in terms of n and k.)

```
sum(diag(round(trail_mean_mat(50,10),2)))
```

```
## [1] 6.92
```

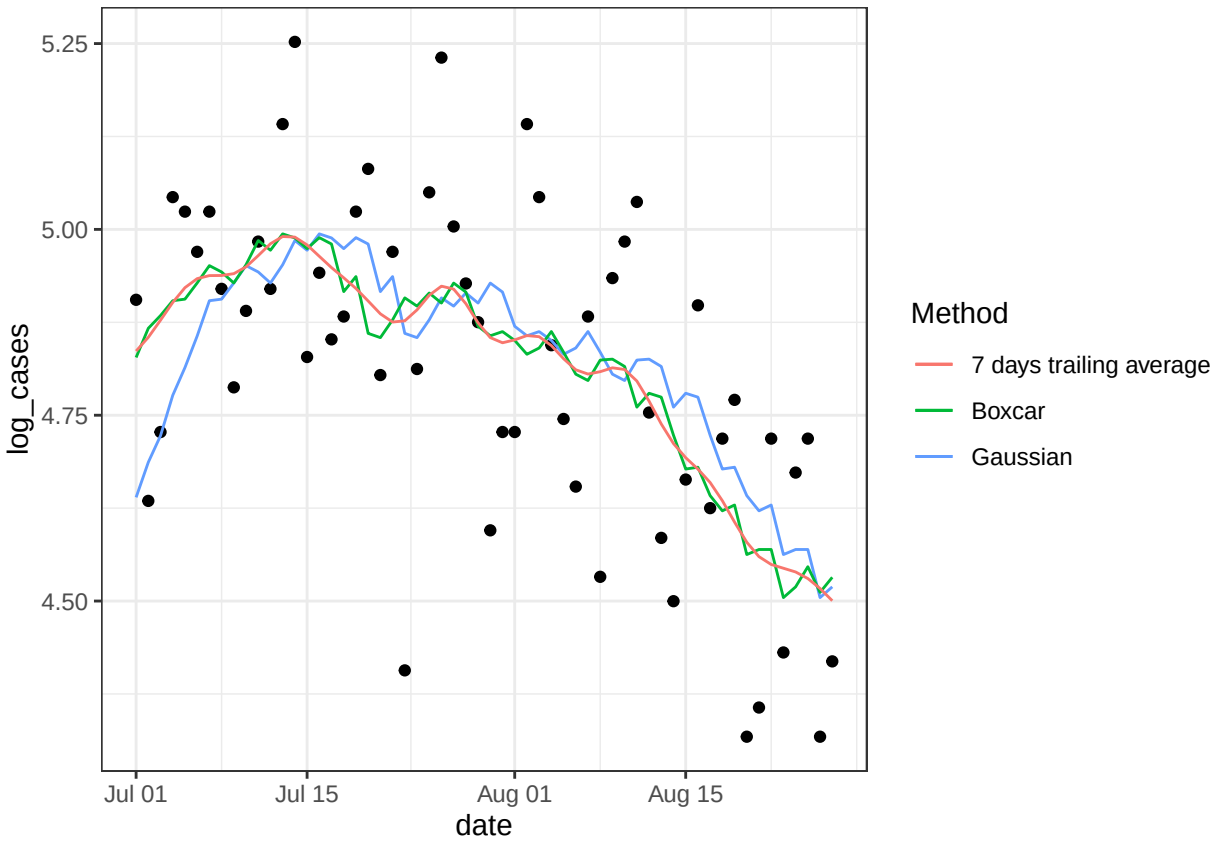
The effective degrees of freedom of the "trailing average" for n = 50 and k = 10 is 6.92. For n large relative to k, the edf will close to n/k.

4. The following function generates the smoothing matrices and returns fitted nonparametric regression estimates and the EDF for Gaussian and Boxcar kernels with different bandwidths.

```
kernel_smoother <- function(x, y, kern = c("boxcar", "gaussian"), band = 7) {
  dmat <- as.matrix(dist(x))
  kern <- match.arg(kern)
  W <- switch(kern,
              boxcar = dmat <= (band * 0.5),
              gaussian = dnorm(dmat, 0, sd = band * 0.3706506))
  W <- sweep(W, 1, rowSums(W), '/')
  fit <- W %*% y
  out <- list(fit = fit, edf = sum(diag(W)))
  out
}
```

Use just the data on or after 1 July 2022. Use the function to plot the data (as points) along with (as lines) (1) the trailing 7-day average, (2) the boxcar smoother with `band = 7` and (3) the Gaussian smoother with `band = 7`. How do the results from the two new smoothers compare with those of the 7-day trailing average? Explain.

```
data <- filter(bccovid, date >= as.Date("2022-07-01"))
data$log_cases <- log(data$cases)
data$boxcar <- kernel_smoother(data$date, data$log_cases, kern = "boxcar", band = 7)$fit
data$gaussian <- kernel_smoother(data$date, data$log_cases, kern = "gaussian", band = 7)$fit
ggplot(data) + geom_point(aes(date, log_cases)) +
geom_line(aes(date, avg_log_cases, color = "red")) +
geom_line(aes(date, boxcar, color = "green")) +
geom_line(aes(date, gaussian, color = "blue")) +
scale_color_discrete(name = "Method",
                     labels = c("7 days trailing average", "Boxcar", "Gaussian"))
```



The Boxcar fits more smoothly than the 7-day trailing average and Gaussian methods.

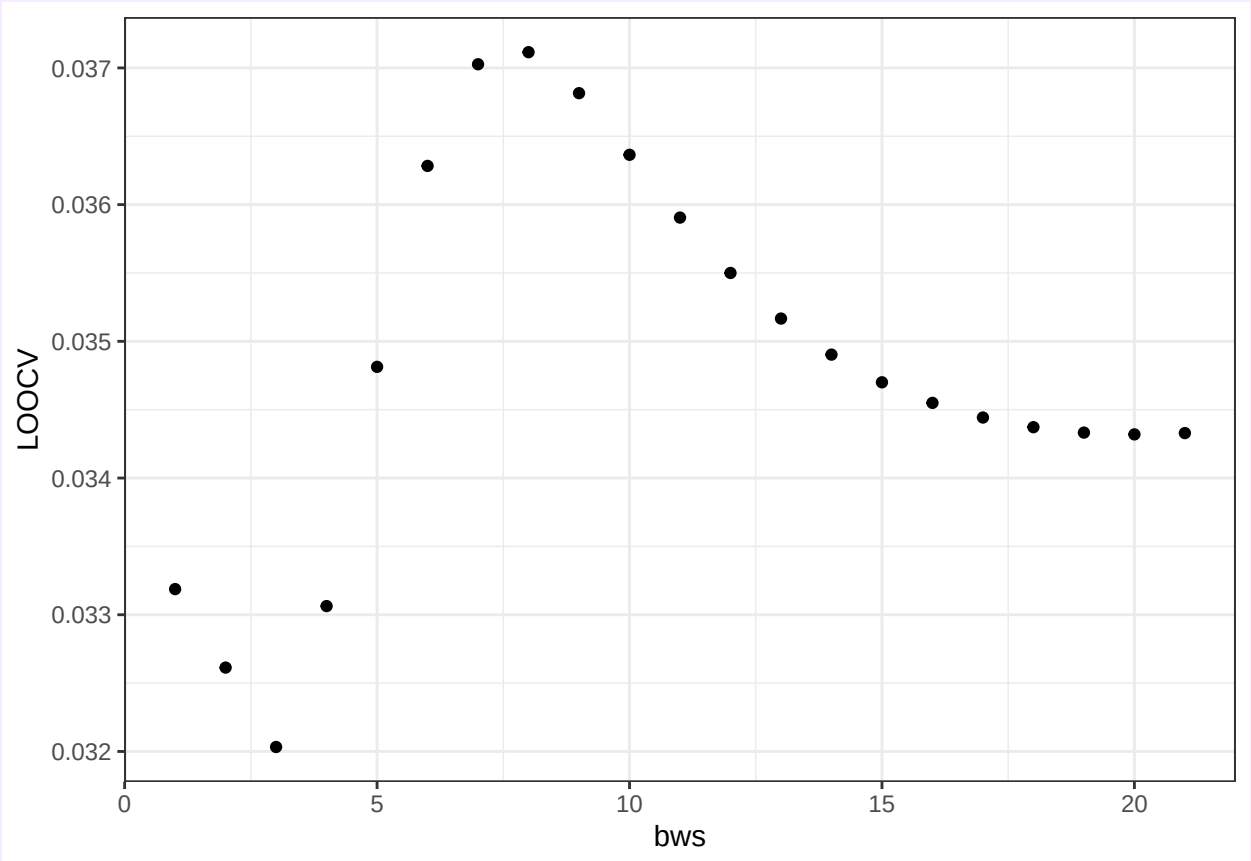
5. Adjust the `kernel_smoother()` function so that it also computes and returns the LOO-CV score. Compute the the LOO-CV score for each integer value of `band` from 1 to 21 for the Gaussian kernel. Plot the scores against `band`. Which value is best? Will the resulting plot be smoother or wigglier than with `band = 7`? Can you think of any reasons that this is best bandwidth by this metric? Should we use it?

```
kernel_smoother <- function(x, y, kern = c("boxcar", "gaussian"), band = 7) {
  # copy the code from above and add a few lines
  dmat <- as.matrix(dist(x))
  kern <- match.arg(kern)
  W <- switch(kern,
              boxcar = dmat <= (band * 0.5),
              gaussian = dnorm(dmat, 0, sd = band * 0.3706506))
  W <- sweep(W, 1, rowSums(W), '/')
  fit <- W %*% y
  out <- list(fit = fit, edf = sum(diag(W)),loocv = mean((y-fit)^2/(1-diag(W))^2))
  out
}
```

```r
bws <- 1:21
## some code
temp <- sapply(bws, function(x)
kernel_smoother(data$date, data$log_cases, kern = "gaussian", band = x)$loocv)
temp
```

```
##  [1] 0.03318756 0.03261346 0.03203272 0.03306338 0.03481317 0.03628321
##  [7] 0.03702672 0.03711508 0.03681507 0.03636444 0.03590515 0.03549995
## [13] 0.03516654 0.03490282 0.03470030 0.03454964 0.03444252 0.03437201
## [19] 0.03433245 0.03431926 0.03432876
```

```r
ggplot(data.frame(bws,temp),aes(x = bws,y = temp))+geom_point()+ylab("LOOCV")
```



When band = 3, the LOO-CV is smallest. When band = 7, the band gets smaller, and the range for taking the average becomes smaller, so that the plot would be wigglier.