Homework 2 - Regression

Instructions

Due 17 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 edge of the `.pdf`, you'll lose 2 points automatically.
- Be sure to add your name in the `author` field at the top.

1 Regularized methods (5 points)

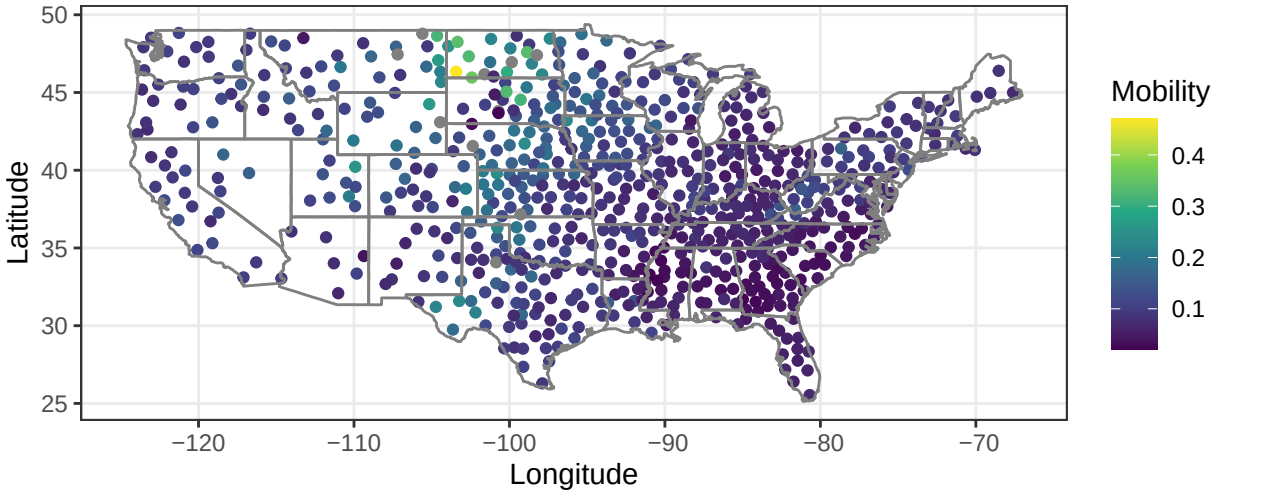
In this section we will conduct an analysis of the `mobility` data first visited during the first week of class. Use `?mobility` in the console or visit the package website for information on the variables contained here.

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.

1.1 (0.5 points)

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

```
data("mobility")
mobility %>%
  filter(!(State %in% c("AK", "HI"))) %>%
  ggplot(aes(x = Longitude, y = Latitude, colour = Mobility)) +
  geom_point() +
  coord_sf() +
  borders("state") +
  scale_colour_viridis_c()
```



(placeholder text)

1.2 (1 points)

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.

```
#some code
```

(placeholder text)

1.3 (1 point)

In this question, we will run a linear regression of `mobility` against all appropriate covariates and answer some questions. By "appropriate", we mean

> use any covariate that could potentially be predictive of mobility. Some are obviously useless or colinear with others. Don't use those. All that is necessary to make this choice is to examine the documentation for the data `?mobility` or on the web.

a. (.25 points) 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. (.25 points) Explain why the `ID` variable must be excluded.
c. (.25 points) Explain which other variables, if any, you excluded from the regression, and why. (If you think they can all be used, explain why.)
d. (.25 points) 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?

```
ols <- lm(Mobility ~ . , data = mobility) # YOU NEED TO CHANGE THIS LINE!!

# some code
```

a.
b.
c.
d.

1.4 (1 point)

Continuing 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. (.5 points) 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. (.5 points) 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)
```

b.

```
# more code
```

1.5 (0.5 points)

For both the Ridge regression and Lasso regression in Question 1.4, choose the set of coefficient estimates using `lambda.min`. Compare these coefficient estimates with those in Question 1.3 by producing a graph (put the Variable names on the y-axis, the coefficient estimate on the x-axis, and use colour or shape to denote the three methods). Describe what you see.

```
# some code
```

(placeholder text)

1.6 (1 point)

Calculate the LOOCV score for your OLS model in Question 1.3. Is it justifiable to compare this score with the scores `lambda.min` values from Question 1.4 to select between these 3 methods? Explain why or why not.

```
# some code
```

(placeholder text)

2 Nonparametric regression (5 points)

It may be helpful to read through the Kernel Regression section of the worksheets before starting this section. See also the documentation for the `arcuate` data.

```
data("arcuate")
```

2.1 (1 point)

Plot the `fa` against `position` as points, along with two curves calculated by averaging the points in a local neighbourhood of the k nearest observations. If $d_{ij} = |x_i - x_j|$, then the average at the i^{th} location is:

$$\overline{y}_i = \frac{1}{k} \sum_{j=1}^{n} y_j I(d_{ij} \leq d_{i,(k)}),$$

where $d_{i,(k)}$ is the "kth order statistic for distances of x to x_i".

Here's a function (perhaps hard to understand) that does this for arbitrarily spaced data.

```
knn_smoother <- function(x, y, k = 10) {
  kth_smallest <- function(z) sort(z, partial = k)[k]
  Dmat <- as.matrix(dist(x))
  q <- apply(Dmat, 1, kth_smallest)
  as.vector(Matrix::Matrix((Dmat <= q) / k, sparse = TRUE) %*% y)
}
```

This is one of the most common ways to "smooth" data. Use `k = 10` and `k = 70`.

Describe two features you notice about the curves.

```
# some code
```

(placeholder text)

2.2 (1 point)

If it isn't obvious from the function above, you should recognize that this method is a "linear smoother". Let's imagine a slightly simpler version where the x-values are a sequence (regular and evenly spaced) and the y-values are arranged so that x is increasing. This is also a linear smoother.

Complete the function below that creates the smoothing matrix for any dimension `n` and any "window" of days `k` (where `k` is odd). Think carefully about how to handle the first few and last few rows. Your resulting matrix must be square. Evaluate your function for `n = 7`, and `k = 3` (round the entries to 2 decimals so it prints nicely).

```r
window_mat <- function(n, k = 3) {
  # first check for valid inputs
  stopifnot(n == floor(n), k == floor(k), n > 0, k > 0, k %% 2 == 1L)
  mat <- matrix(0, nrow = n, ncol = n) # preallocate space
  left_boundary <-
  right_boundary <-
  mat[1:(k - 1), left_boundary] <-
  mat[(n - k + 2):n, right_boundary] <-
  lr <-
  for (i in k:(n - k + 1)) mat[i, (i - lr):(i + lr)] <-
  mat / k
}
```

2.3 (1 point)

a. What is the degrees of freedom of the "window smoother" for $n = 50$ and $k = 11$?
b. What is the degrees of freedom in general (for any k and any n)?

(placeholder text)

2.4 (1 point)

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

```
kernel_smoother <- function(x, y, kern = c("boxcar", "gaussian"), band = 10) {
  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
  list(fit = fit, df = sum(diag(W)))
}
```

a. Explain how the `kernel_smoother()` with `kern = "boxcar"` is or is not different from the from the `knn_smoother()` used above.

b. Use the `arcuate` data again. Plot the data (as points, same as in 1) along with (as lines) (1) the `window_smoother` with `k = 71`, (2) the boxcar smoother with `band = 30` and (3) the Gaussian smoother with `band = 30`. How do the results from the two new smoothers compare with those of the `window_smoother`? Explain.

```
# some code
```

(placeholder text)

2.5 (1 point)

Adjust the `kernel_smoother()` function so that it also computes and returns the GCV score. Compute the the GCV score for each integer value of `band` from 2 to 20 for the Gaussian kernel.

a. Plot the scores against `band`. Which value is best?
b. Will the resulting plot be smoother or wigglier than with `band = 35`?
c. Plot both over the data by modifying your code from Question 2.4. Comment on the differences.

```
kernel_smoother <- function(x, y, kern = c("boxcar", "gaussian"), band = 7) {
  # copy the code from above and add a few lines

}
```

```
bws <- 2:20
## some code
```

Some explanation.