Assignment 1

Statistics and Data Science 365/565

Due: September 18 (midnight)

This homework treats linear regression and classification, and gives you a chance to practice using R. If you have forgotten some definitions or terms from previous classes, see the file "notation.pdf" under the "Files" tab on Canvas. It should provide all you need to know to do this assignment. Remember that you are allowed to collaborate on the homework with classmates, but you must write your final solutions by yourself and acknowledge any collaboration at the top of your homework.

Problem 1: Two views of linear regression (10 points)

Recall that in linear regression we model each response Y_i as a linear combination of input variables $X_{i,p}$ and noise. That is

$$Y_i = \beta_0 + \beta_1 X_{i,1} + ... + \beta_p X_{i,p} + \epsilon_i$$

which can be written in matrix form as

$$Y = X\beta + \epsilon$$

where $Y \in \mathbb{R}^n$ is the vector of responses (outcomes), $X \in \mathbb{R}^{n\times(p+1)}$ is the design matrix, where each row is a data point, and $\beta \in \mathbb{R}^{p+1}$ is the vector of parameters, including the intercept, and $\epsilon \in \mathbb{R}^n$ is a noise vector. Assume throughout this problem that the matrix $X^T X$ is invertible.

View 1: $\widehat{\beta}$ minimizes the Euclidean distance between Y and $X\beta$.

Suppose we make no assumptions about ϵ. We simply want to find the β that minimizes the Euclidean distance between Y and $X\beta$, i.e., the ℓ_2 norm of $Y - X\beta$. That is, we seek

$$\widehat{\beta} = \underset{\beta\in\mathbb{R}^p}{\arg\min} \|Y - X\beta\|^2.$$

Derive an explicit form for the minimzer $\widehat{\beta}$. Your derivation should involve calculating the gradient of the objective function $f(\beta) = \|Y - X\beta\|^2$, and solving for the β that makes the gradient zero. Express your solution as a function of the matrix X and the vector Y. (If you get stuck, try to first find a clean way to write the gradient with respect to β of the ℓ_2 norm function $g(\beta) = \|\beta\|^2$.)

Solution:

To minimize the objective function, or the equation for the sum of squared residuals, we need to differentiate the function with respect to β and solve for the derivative equal to zero. We can expand the objective function to the following:

$$f(\beta) = \|Y - X\beta\|^2 = (Y - X\beta)^T(Y - X\beta) = YY^T - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta = YY^T - 2\beta^T X^T Y + \beta^T X^T X\beta$$

The second to third line holds true because $\beta^T X^T Y$ is a scalar and thus $\beta^T X^T Y = (\beta^T X^T Y)^T = Y^T X\beta$. If we differentiated with respect to β and set the derivative equal to zero, then we can solve for the β that minimizes our initial objective function:

$$\frac{\partial f(\beta)}{\partial\beta} = -2X^T Y + 2X^T X\beta = 0(X^T X)^{-1}X^T X\beta = (X^T X)^{-1}X^T Y\widehat{\beta} = (X^T X)^{-1}X^T Y$$

The movement from the first to second line comes from moving terms in line one and left-multiplying both sides by $(X^T X)^{-1}$. Then, $(X^T X)^{-1}X^T X = I$, so we get rid of those terms on the left and we have the solution for $\widehat{\beta}$ that minimizes the objective function.

View 2: $\widehat{\beta}$ is the MLE in a normal model.

Suppose we assume the same linear regression model as above, but now we assume that the ϵ_i are uncorrelated and identically distributed as $N(0, \sigma^2)$. Therefore, we can write

$$Y \sim N(X\beta, \sigma^2 I_n),$$

meaning that Y has a multivariate normal distribution with mean $X\beta$ and diagonal covariance matrix $\sigma^2 I_n$. Recall that for a vector $X \sim N(\mu, \Sigma)$, the density is

$$f(x) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\Big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\Big).$$

To derive the maximum likelihood estimator under this model, maximize the log density of Y as a function of β, assuming that σ^2 is known. Show that the maximizer is the same as that obtained under View 1.

Solution:

We can first take the log of the density function f(x) and then subsitute $X\beta = \mu$ and $\sigma^2 I = \Sigma$.

$$log(f(x)) = -\frac{1}{2}log(2\pi\Sigma) - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)log(Y) = -\frac{1}{2}log(2\pi\sigma^2 I) - \frac{1}{2}(Y - X\beta)^T(\sigma^2 I)^{-1}(Y - X\beta) = -\frac{1}{2}log(2\pi\sigma^2 I) - \frac{1}{2\sigma^2}$$

The second to third line comes from the fact that the inverse of the identity matrix is still the identity matrix. Then, from the third line to the fourth line, we expanded the $(Y - X\beta)^T(Y - X\beta)$ the same as above. We know that the log-likelihood function is concave down, so we differentiate with respect to β and set equal to zero to find the maximum likelihood estimate $\widehat{\beta}$.

$$\frac{\partial log(Y)}{\partial \beta} = -\frac{1}{2\sigma^2}(-2X^TY + 2X^TX\beta) = 0(X^TX)^{-1}X^TX\beta = (X^TX)^{-1}X^TY\widehat{\beta} = (X^TX)^{-1}X^TY$$

Again, we find that $\widehat{\beta} = (X^TX)^{-1}X^TY$ and have shown that the solution to the maximum likelihood estimator is equal to the solution to View 1.

Problem 2: Linear regression and classification (30 points)

Citi Bike is a public bicycle sharing system in New York City. There are hundreds of bike stations scattered throughout the city. Customers can check out a bike at any station and return it at any other station. Citi Bike caters to both commuters and tourists. Details on this program can be found at https://www.citibikenyc.com/

For this problem, you will build models to predict Citi Bike usage, in number of trips per day. The dataset consists of Citi Bike usage information and weather data recorded from Central Park.

In the citibike_*.csv files, we see:

1. date
2. trips: the total number of Citi Bike trips. This is the outcome variable.
3. n_stations: the total number of Citi Bike stations in service
4. holiday: whether or not the day is a work holiday
5. month: taken from the date variable
6. dayofweek: taken from the date variable

In the weather.csv file, we have:

1. date

2. PRCP: amount precipitation (i.e. rainfall amount) in inches
3. SNWD: snow depth in inches
4. SNOW: snowfall in inches
5. TMAX: maximum temperature for the day, in degrees F
6. TMIN: minimum temperature for the day, in degrees F
7. AWND: average windspeed

You are provided a training set consisting of data from 7/1/2013 to 3/31/2016, and a test set consisting of data after 4/1/2016. The weather file contains weather data for the entire year.

Part a: Read in and merge the data.

To read in the data, you can run, for example:

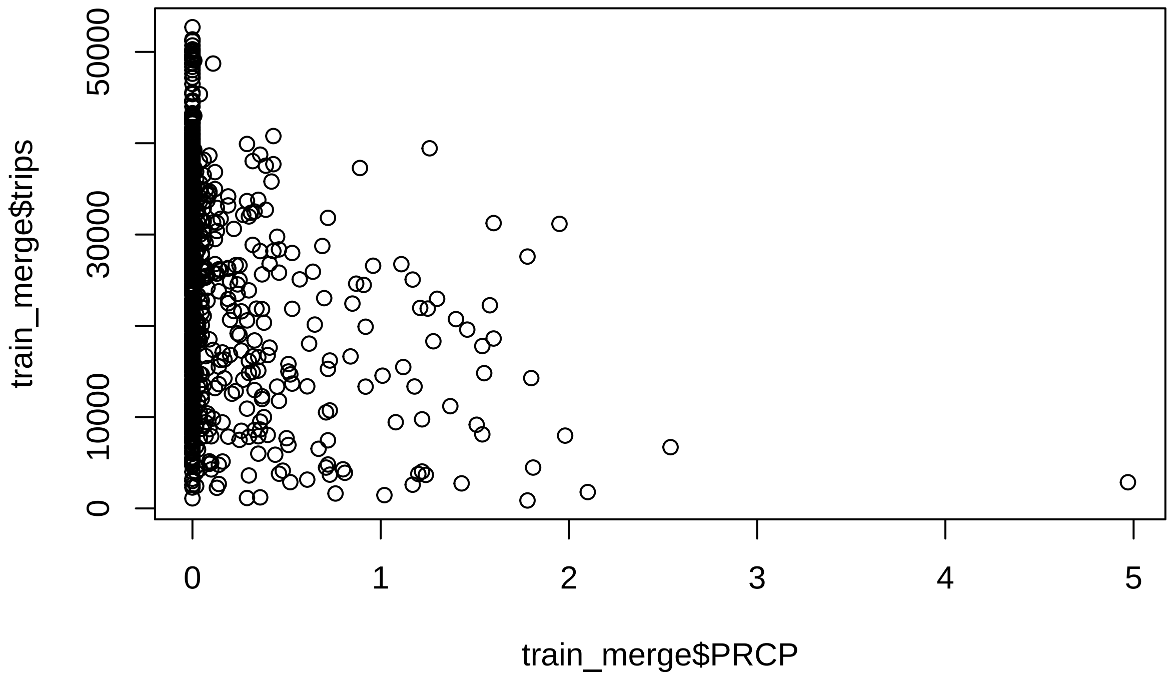
```
train <- read.csv("citibike_train.csv")
test <- read.csv("citibike_test.csv")
weather <- read.csv("weather.csv")
```

Merge the training and test data with the weather data, by date. Once you have successfully merged the data, you may drop the "date" variable; we will not need it for the rest of this assignment.

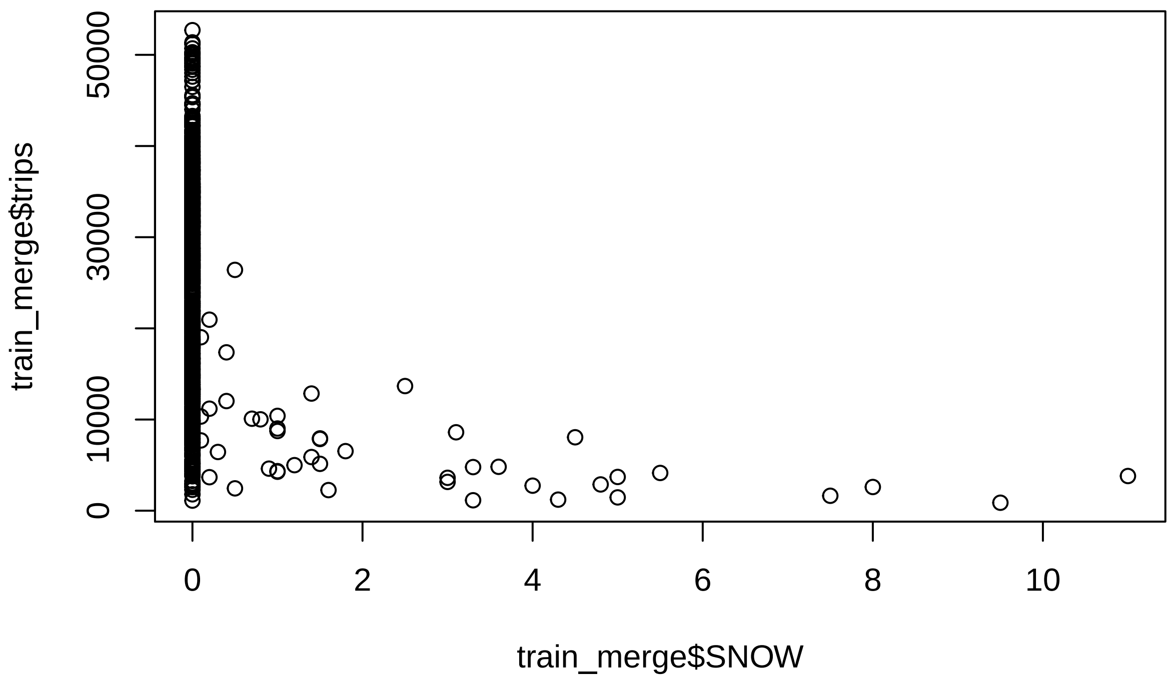
```
train_merge <- merge(train,weather)
test_merge <- merge(test,weather)
drop <- "date"
train_merge <- train_merge[,!(names(train_merge) %in% drop)]
test_merge <- test_merge[,!(names(test_merge) %in% drop)]
```

As always, before you start any modeling, you should look at the data. Make scatterplots of some of the numeric variables. Look for outliers and strange values. Comment on any steps you take to remove entries or otherwise process the data. Also comment on whether any predictors are strongly correlated with each other.

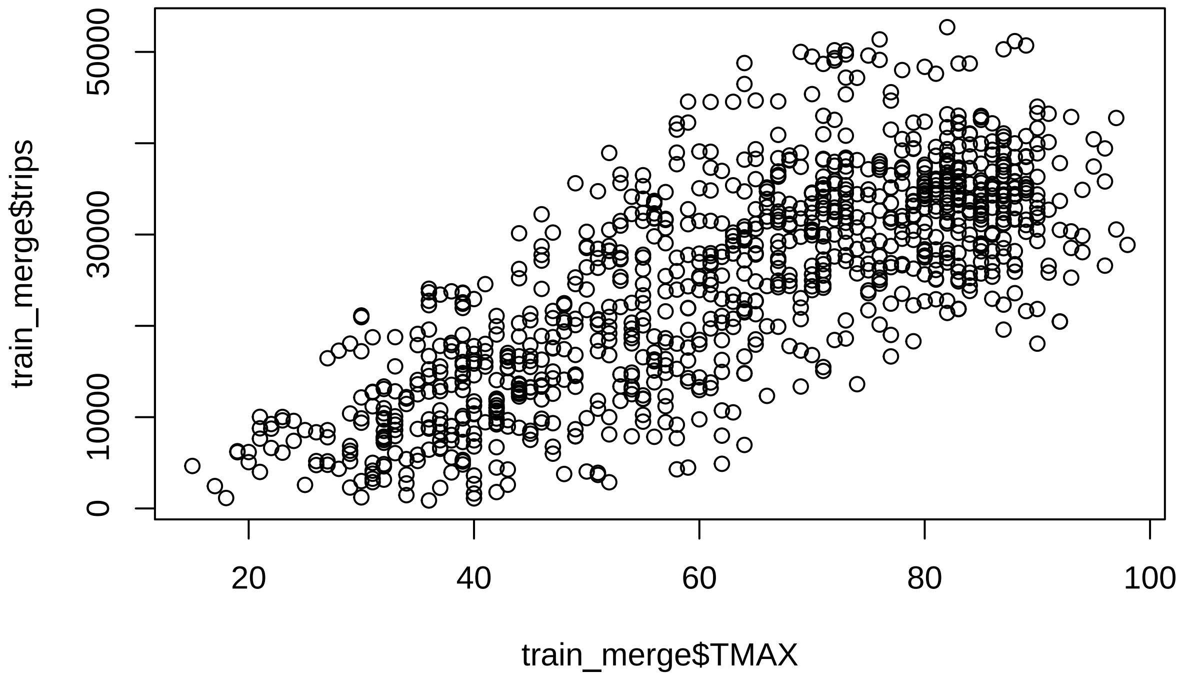
```
train_merge <- train_merge[train_merge$AWND > 0,]
plot(train_merge$PRCP,train_merge$trips)
```



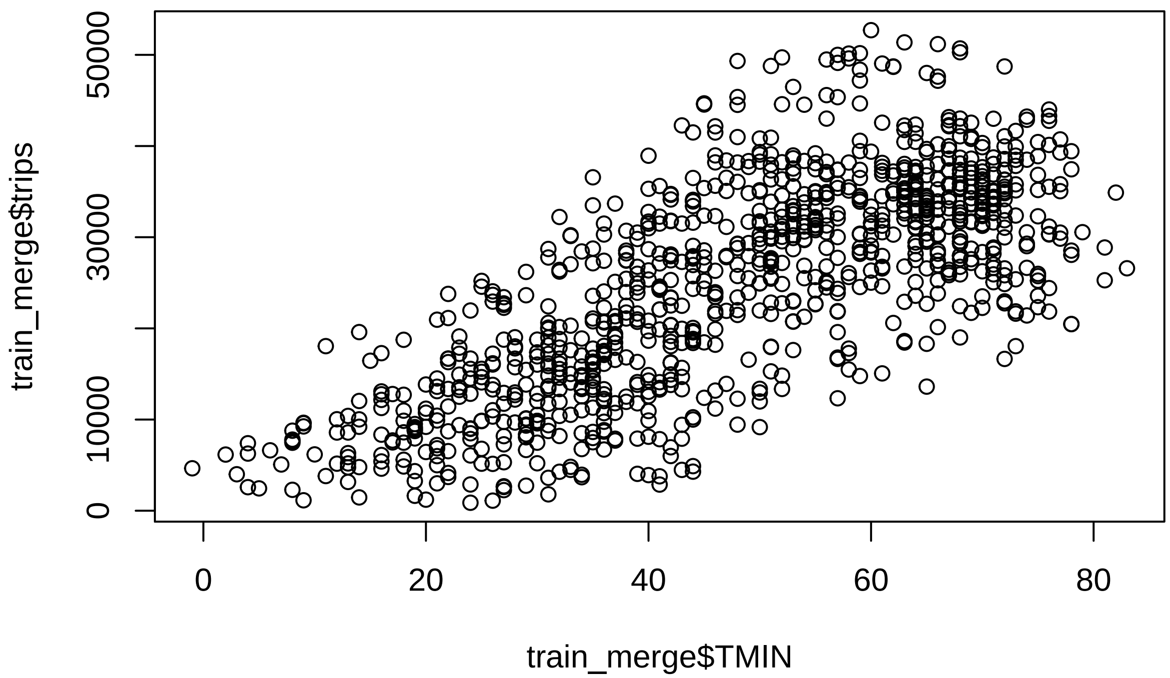

```
plot(train_merge$SNOW,train_merge$trips)
```



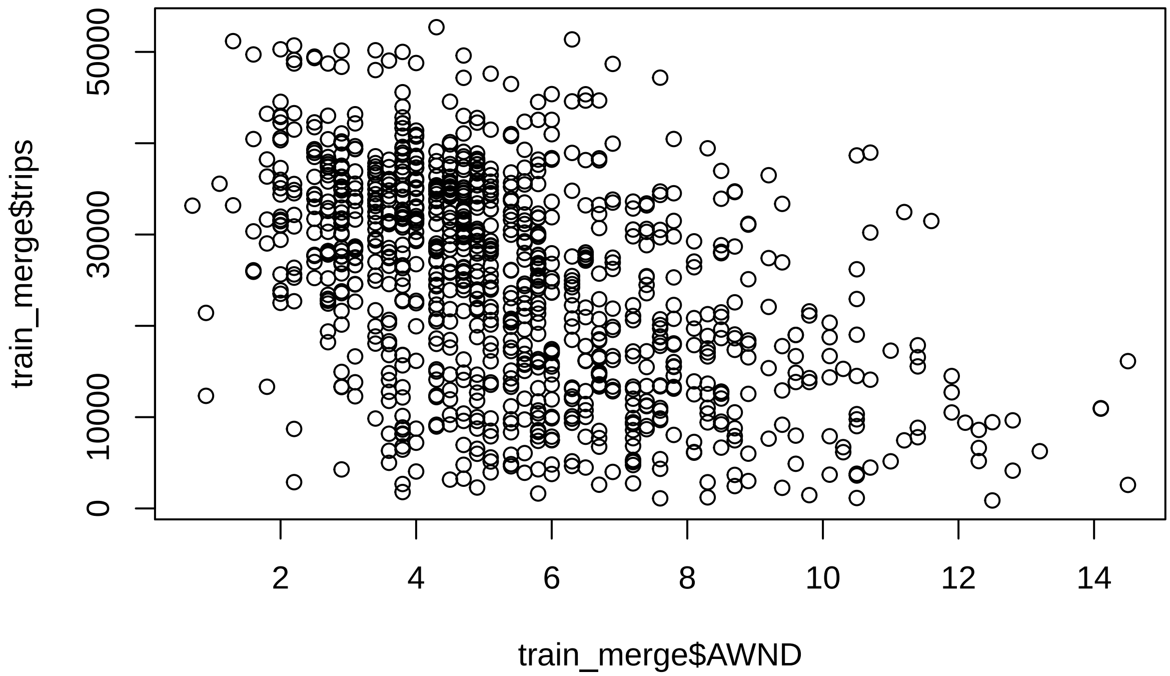

```
plot(train_merge$TMAX,train_merge$trips)
```



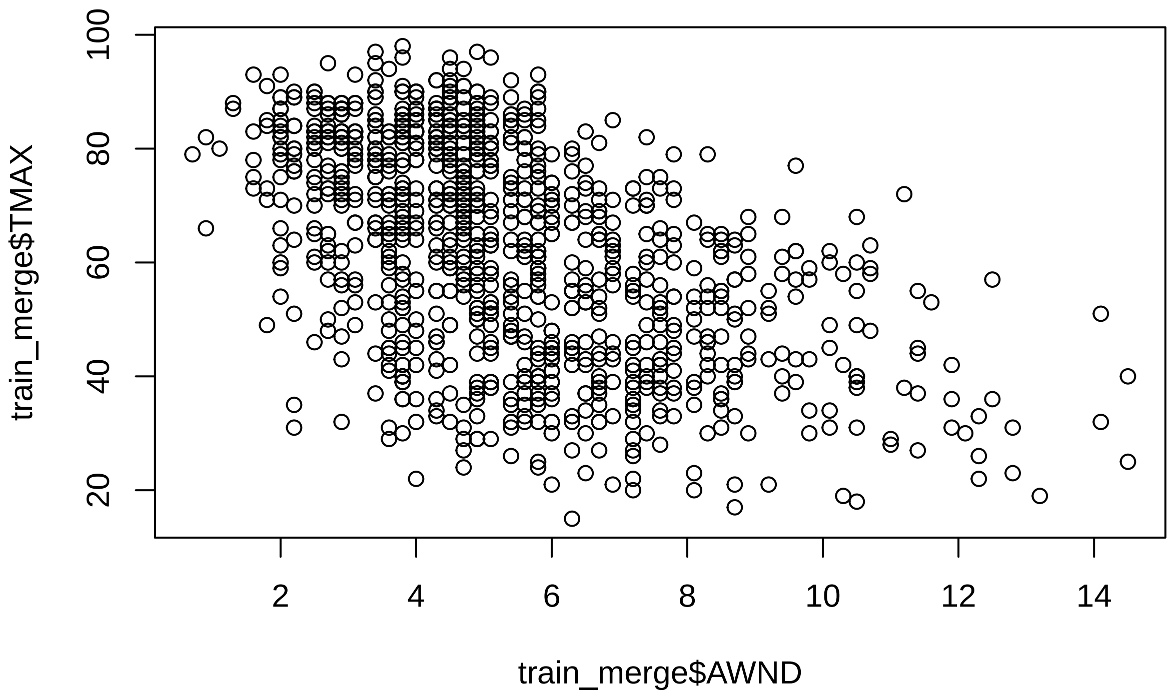

```
plot(train_merge$TMIN,train_merge$trips)
```



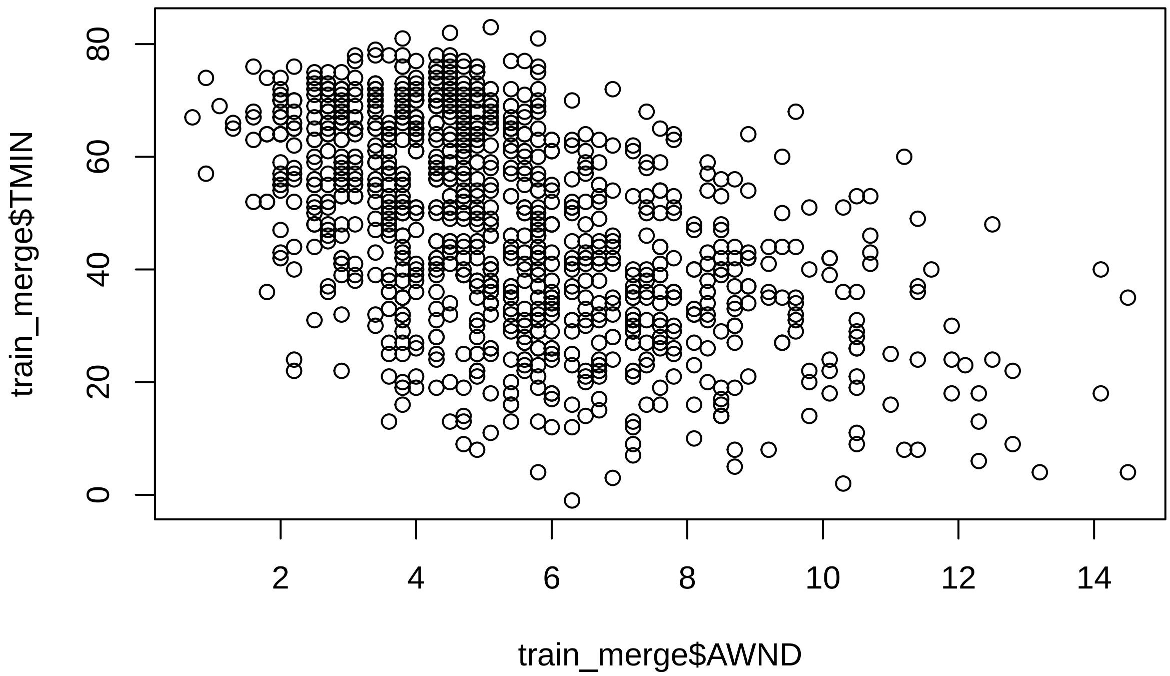

```
plot(train_merge$AWND,train_merge$trips)
```



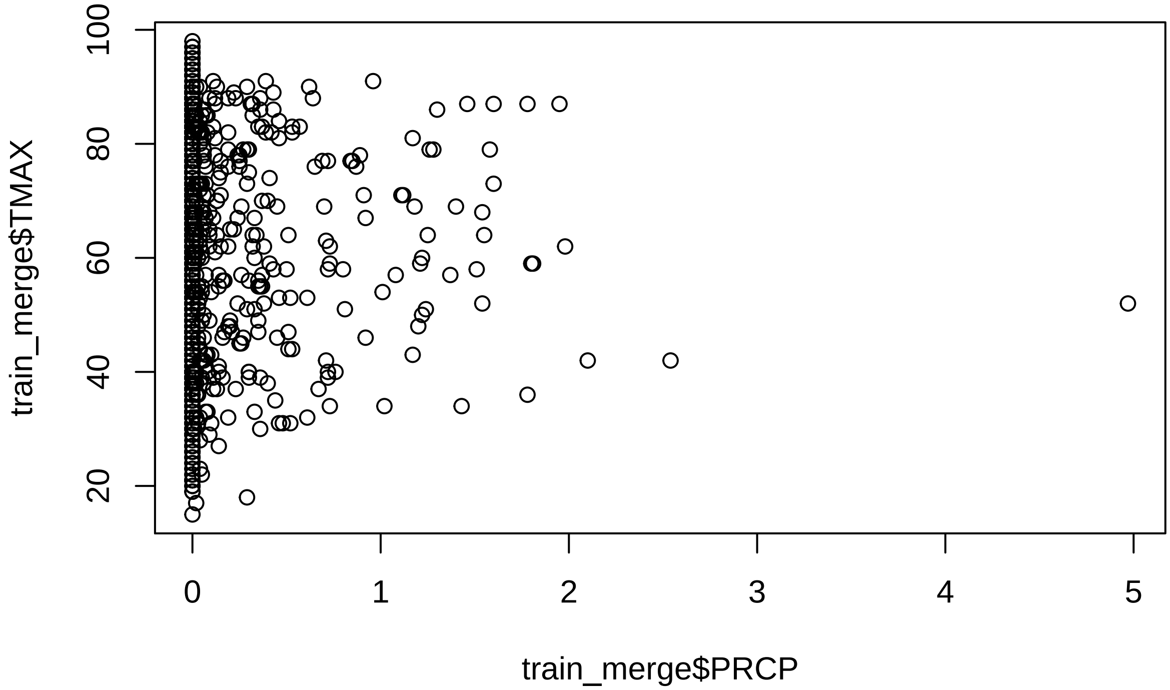

```
plot(train_merge$AWND,train_merge$TMAX)
```




```
plot(train_merge$AWND,train_merge$TMIN)
```




```r
plot(train_merge$PRCP,train_merge$TMAX)
```



Comments:

For average windspeed, there are several extremely low outliers at -10,000. I will only keep data where the the average windspeed is positive. The plots then show the training data variables after these outliers have been removed. Temperature (max and min) are both stongly negatively correlated with average windspeed. Otherwise, we do not see much strong correlation with any of the snow/participation variables because most data points show no snow or participation regardless of other weather factors.

For the rest of this problem, you will train your models on the training data and evaluate them on the test data.

Part b: Linear regression

Fit a linear regression model to predict the number of trips. Include all the covariates in the data. Print the summary of your model using the R `summary` command. Next, find the "best" linear model that uses only q variables (where including the intercept counts as one of the variables), for each $q = 1, 2, 3, 4, 5$. It is up to you to choose how to select the "best" subset of variables. (A categorical variable or factor such as "month" corresponds to a single variable.) Describe how you selected each model. Give the R^2 and the mean squared error (MSE) on the training and test set for each of the models. Which model gives the best fit to the data? Comment on your findings.

```r
#Model with all predictive variables.
modall <- lm(trips ~ ., data=train_merge)
summary(modall)
```

```
##
## Call:
## lm(formula = trips ~ ., data = train_merge)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16372.1  -2199.8    216.2   2496.3  20776.4
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -12589.146   1476.054  -8.529  < 2e-16 ***
## n_stations         69.688      2.848  24.468  < 2e-16 ***
## holidayTRUE    -10687.551    792.633 -13.484  < 2e-16 ***
## monthAug         4020.636    796.919   5.045 5.40e-07 ***
## monthDec        -3270.991    749.933  -4.362 1.43e-05 ***
## monthFeb        -5578.753    909.155  -6.136 1.23e-09 ***
## monthJan        -5338.133    827.200  -6.453 1.72e-10 ***
## monthJul         2419.923    825.049   2.933  0.00344 **
## monthJun         4201.243    823.687   5.101 4.07e-07 ***
## monthMar        -3707.483    749.673  -4.945 8.95e-07 ***
## monthMay         3342.001    779.435   4.288 1.99e-05 ***
## monthNov         1643.865    725.365   2.266  0.02365 *
## monthOct         6235.137    707.044   8.819  < 2e-16 ***
## monthSep         6480.139    751.764   8.620  < 2e-16 ***
## dayofweekMon     -711.143    487.412  -1.459  0.14488
## dayofweekSat    -5102.204    487.483 -10.466  < 2e-16 ***
## dayofweekSun    -6262.104    489.472 -12.794  < 2e-16 ***
## dayofweekThurs    623.638    483.343   1.290  0.19727
## dayofweekTues    -182.295    486.709  -0.375  0.70808
## dayofweekWed      818.113    485.799   1.684  0.09249 .
## PRCP            -7937.539    393.399 -20.177  < 2e-16 ***
## SNWD             -246.061     62.101  -3.962 7.97e-05 ***
## SNOW               76.392    186.040   0.411  0.68144
## TMAX              328.904     29.631  11.100  < 2e-16 ***
## TMIN              -72.314     32.290  -2.240  0.02535 *
## AWND             -359.990     68.306  -5.270 1.68e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4087 on 972 degrees of freedom
## Multiple R-squared:  0.8771, Adjusted R-squared:  0.874
```

```
## F-statistic: 277.6 on 25 and 972 DF,  p-value: < 2.2e-16
```

```r
print(paste("Train R^2:",summary(modall)$r.squared))
```

```
## [1] "Train R^2: 0.877133614114182"
```

```r
print(paste("Train MSE:",mean(summary(modall)$residuals^2)))
```

```
## [1] "Train MSE: 16265864.1512502"
```

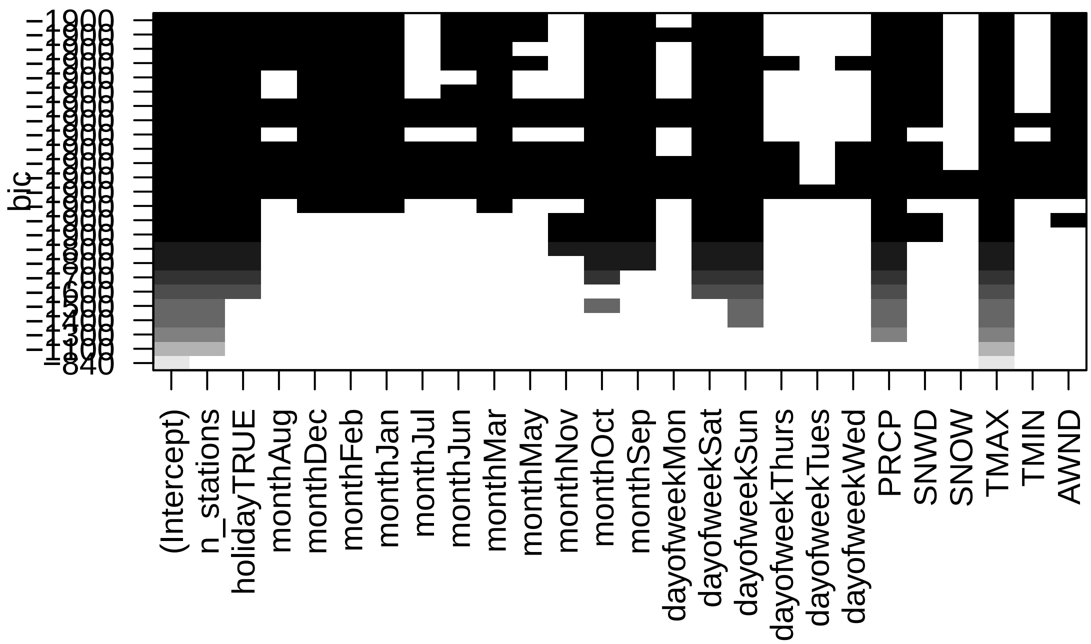
```r
pred <- predict(modall,test_merge)
print(paste("Test R_squared:",cor(test_merge$trips,pred)^2))
```

```
## [1] "Test R_squared: 6.53888939873337e-05"
```

```r
print(paste("Test MSE:",mean((test_merge$trips-pred)^2)))
```

```
## [1] "Test MSE: 566456063752.122"
```

```r
#Use regsubsets function to choose best variables.
library(leaps)
mod <- regsubsets(trips ~ .,data=train_merge,nvmax = 40)
modsum <- summary(mod)
plot(mod)
```




```r
#For the following model we fit with the q=1 predictor variables (always including the intercept).
#Model q=1
mod1 <- lm(trips ~ TMAX, data=train_merge)
summary(mod1)
```

```
##
## Call:
## lm(formula = trips ~ TMAX, data = train_merge)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19864.5  -4889.0    -54.1   4379.8  23113.5
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3609.64     815.53  -4.426 1.07e-05 ***
## TMAX          457.49      12.45  36.745  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7504 on 996 degrees of freedom
## Multiple R-squared:  0.5755, Adjusted R-squared:  0.5751
## F-statistic:  1350 on 1 and 996 DF,  p-value: < 2.2e-16
```

```r
print(paste("Train R_squared:",summary(mod1)$r.squared))
```

```
## [1] "Train R_squared: 0.575480974246267"
```

```r
print(paste("Train MSE:",summary(mod1)$sigma^2))
```

```
## [1] "Train MSE: 56313486.8292588"
```

```r
pred <- predict(mod1,test_merge)
print(paste("Test R_squared:",cor(test_merge$trips,pred)^2))
```

```
## [1] "Test R_squared: 0.347789859122436"
```

```r
print(paste("Test MSE:",mean((test_merge$trips-pred)^2)))
```

```
## [1] "Test MSE: 262154197.585209"
```

```r
#Model q=2
mod2 <- lm(trips ~ TMAX + n_stations, data=train_merge)
summary(mod2)
```

```
##
## Call:
## lm(formula = trips ~ TMAX + n_stations, data = train_merge)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21679.8  -4077.5    349.8   4406.0  16103.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -28955.890   1578.288  -18.35   <2e-16 ***
## TMAX           479.043     10.889   43.99   <2e-16 ***
## n_stations      67.782      3.771   17.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6523 on 995 degrees of freedom
## Multiple R-squared:  0.6795, Adjusted R-squared:  0.6789
## F-statistic:  1055 on 2 and 995 DF,  p-value: < 2.2e-16
```

```r
print(paste("Train R_squared:",summary(mod2)$r.squared))
```

```
## [1] "Train R_squared: 0.679538094016834"
```

```r
print(paste("Train MSE:",summary(mod1)$sigma^2))
```

```
## [1] "Train MSE: 56313486.8292588"
```

```r
pred <- predict(mod2,test_merge)
print(paste("Test R_squared:",cor(test_merge$trips,pred)^2))
```

```
## [1] "Test R_squared: 0.476711865453968"
```

```r
print(paste("Test MSE:",mean((test_merge$trips-pred)^2)))
```

```
## [1] "Test MSE: 79785749.6730557"
```

```r
#Model q=3
mod3 <- lm(trips ~ TMAX + n_stations + PRCP, data=train_merge)
summary(mod3)
```

```
##
## Call:
## lm(formula = trips ~ TMAX + n_stations + PRCP, data = train_merge)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22100.7  -3800.1    244.7   3830.1  24579.3
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -27065.390   1430.944  -18.91   <2e-16 ***
## TMAX           476.108      9.836   48.40   <2e-16 ***
## n_stations      65.702      3.409   19.28   <2e-16 ***
## PRCP         -8108.175    539.485  -15.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5891 on 994 degrees of freedom
## Multiple R-squared:  0.7389, Adjusted R-squared:  0.7381
## F-statistic: 937.5 on 3 and 994 DF,  p-value: < 2.2e-16
```

```r
print(paste("Train R_squared:",summary(mod3)$r.squared))
```

```
## [1] "Train R_squared: 0.738877704901058"
```

```r
print(paste("Train MSE:",summary(mod3)$sigma^2))
```

```
## [1] "Train MSE: 34708206.1520892"
```

```r
pred <- predict(mod3,test_merge)
print(paste("Test MSE:",mean((test_merge$trips-pred)^2)))
```

```
## [1] "Test MSE: 70905799.4178131"
```

```r
#Model q=4
mod4 <- lm(trips ~ TMAX + n_stations + PRCP + dayofweek, data=train_merge)
summary(mod4)
```

```
##
## Call:
## lm(formula = trips ~ TMAX + n_stations + PRCP + dayofweek, data = train_merge)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -23442.1  -2790.9    115.9   3100.5  23844.2
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -25295.866   1358.185 -18.625  < 2e-16 ***
## TMAX              477.420      8.929  53.470  < 2e-16 ***
## n_stations         64.588      3.093  20.885  < 2e-16 ***
## PRCP            -8453.394    491.207 -17.209  < 2e-16 ***
## dayofweekMon    -1276.700    632.437  -2.019   0.0438 *
## dayofweekSat    -4591.487    634.464  -7.237 9.21e-13 ***
## dayofweekSun    -5922.417    635.639  -9.317  < 2e-16 ***
## dayofweekThurs    630.477    631.540   0.998   0.3184
## dayofweekTues     109.959    632.593   0.174   0.8620
## dayofweekWed      967.276    632.811   1.529   0.1267
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5343 on 988 degrees of freedom
## Multiple R-squared:  0.7865, Adjusted R-squared:  0.7845
## F-statistic: 404.4 on 9 and 988 DF,  p-value: < 2.2e-16
```

```r
print(paste("Train R_squared:",summary(mod4)$r.squared))
```

```
## [1] "Train R_squared: 0.786484458990826"
```

```r
print(paste("Train MSE:",summary(mod4)$sigma^2))
```

```
## [1] "Train MSE: 28552697.5538818"
```

```r
pred <- predict(mod4,test_merge)
print(paste("Test R_squared:",cor(test_merge$trips,pred)^2))
```

```
## [1] "Test R_squared: 0.708879544709358"
```

```r
print(paste("Test MSE:",mean((test_merge$trips-pred)^2)))
```

```
## [1] "Test MSE: 54998775.108795"
```

```r
#Model q=5
mod5 <- lm(trips ~ TMAX + n_stations + PRCP + dayofweek + month, data=train_merge)
summary(mod5)
```

```
##
## Call:
## lm(formula = trips ~ TMAX + n_stations + PRCP + dayofweek + month,
##     data = train_merge)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21605.6  -2200.7    488.1   2922.4  20764.8
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -17561.567   1359.247 -12.920  < 2e-16 ***
## TMAX              302.657     17.469  17.325  < 2e-16 ***
## n_stations         72.086      2.989  24.115  < 2e-16 ***
## PRCP            -8089.246    417.167 -19.391  < 2e-16 ***
## dayofweekMon     -912.075    536.434  -1.700 0.089401 .
```

```
## dayofweekSat    -4486.881    537.099  -8.354 2.25e-16 ***
## dayofweekSun    -5579.635    539.030 -10.351  < 2e-16 ***
## dayofweekThurs    691.059    534.545   1.293 0.196387    
## dayofweekTues     402.139    536.273   0.750 0.453509    
## dayofweekWed     1205.900    536.060   2.250 0.024699 *  
## monthAug         3778.132    835.669   4.521 6.91e-06 ***
## monthDec        -3828.821    812.677  -4.711 2.82e-06 ***
## monthFeb        -6919.278    902.888  -7.663 4.35e-14 ***
## monthJan        -5871.509    900.340  -6.521 1.12e-10 ***
## monthJul         1600.997    851.750   1.880 0.060452 .  
## monthJun         3986.444    880.649   4.527 6.73e-06 ***
## monthMar        -3966.835    809.071  -4.903 1.11e-06 ***
## monthMay         3186.141    854.035   3.731 0.000202 ***
## monthNov         1169.151    789.357   1.481 0.138891    
## monthOct         6136.249    760.048   8.073 2.00e-15 ***
## monthSep         5970.346    803.862   7.427 2.42e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4522 on 977 degrees of freedom
## Multiple R-squared:  0.8488, Adjusted R-squared:  0.8457 
## F-statistic: 274.2 on 20 and 977 DF,  p-value: < 2.2e-16
```

```r
print(paste("Train R_squared:",summary(mod5)$r.squared))
```

```
## [1] "Train R_squared: 0.848796278481776"
```

```r
print(paste("Train MSE:",summary(mod5)$sigma^2))
```

```
## [1] "Train MSE: 20447608.2083175"
```

```r
pred <- predict(mod5,test_merge)
print(paste("Test R_squared:",cor(test_merge$trips,pred)^2))
```

```
## [1] "Test R_squared: 0.725478801249959"
```

```r
print(paste("Test MSE:",mean((test_merge$trips-pred)^2)))
```

```
## [1] "Test MSE: 46503209.8587607"
```

Comments:

I selected variables based on which ones were included in the model using the regsubsets function. Most of these choices make sense logically. Higher temperatures and less precipatiation would signify better weather and thus more people are likely to take bike trips. Also, with less open stations, less trips will be made because less bikes are available. Also, on the weekends you see less trips because people are not using bikes to commute to work (and tourists likely use cabs/Uber more because they do not know the city). The model that best fits the data is the q=5 model. This makes sense because neither R_squared nor MSE penalizes when using more predictor variables. We know that R_squared will always increase and MSE will always decrease as more predictors are added to the model.

Part c: KNN Classification

Now we will transform the outcome variable to allow us to do classification. Create a new vector Y with entries:

$$Y[i] = \mathbf{1}\{trips[i] > median(trips)\}$$

Use the median of the variable from the full data (training and test combined). After computing the binary outcome variable Y, you should drop the original trips variable from the data.

```
tot_trips <- c(train_merge$trips,test_merge$trips)
med_trips <- median(tot_trips)
test_merge$trips_bi <- as.factor(ifelse(test_merge$trips >= med_trips, 1, 0))
train_merge$trips_bi <- as.factor(ifelse(train_merge$trips >= med_trips, 1, 0))
drop <- "trips"
train_merge <- train_merge[,!(names(train_merge) %in% drop)]
test_merge <- test_merge[,!(names(test_merge) %in% drop)]
```

Recall that in k-nearest neighbors classification, the predicted value $\widehat{Y}$ of X is the majority vote of the labels for the k nearest neighbors X_i to X. We will use the Euclidean distance as our measure of distance between points. Note that the Euclidean distance doesn't make much sense for factor variables, so just drop the predictors that are categorical for this problem. Standardize the numeric predictors so that they have mean zero and constant standard deviation—the R function `scale` can be used for this purpose.

```
#Drop factor columns
drop <- c("holiday","month","dayofweek")
train_merge <- train_merge[,!(names(train_merge) %in% drop)]
test_merge <- test_merge[,!(names(test_merge) %in% drop)]
#Convert INT columns to numeric
train_merge$TMAX <- as.numeric(train_merge$TMAX)
train_merge$TMIN <- as.numeric(train_merge$TMIN)
test_merge$TMAX <- as.numeric(test_merge$TMAX)
test_merge$TMIN <- as.numeric(test_merge$TMIN)
#Scale all numeric predictor variables.
means <- colMeans(train_merge[1:7])
sds <- apply(train_merge[1:7], 2, sd)
for (x in names(train_merge[1:7])){
     train_merge[x] <- (train_merge[x]-means[x])/sds[x]
     test_merge[x] <- (test_merge[x]-means[x])/sds[x]
}
```

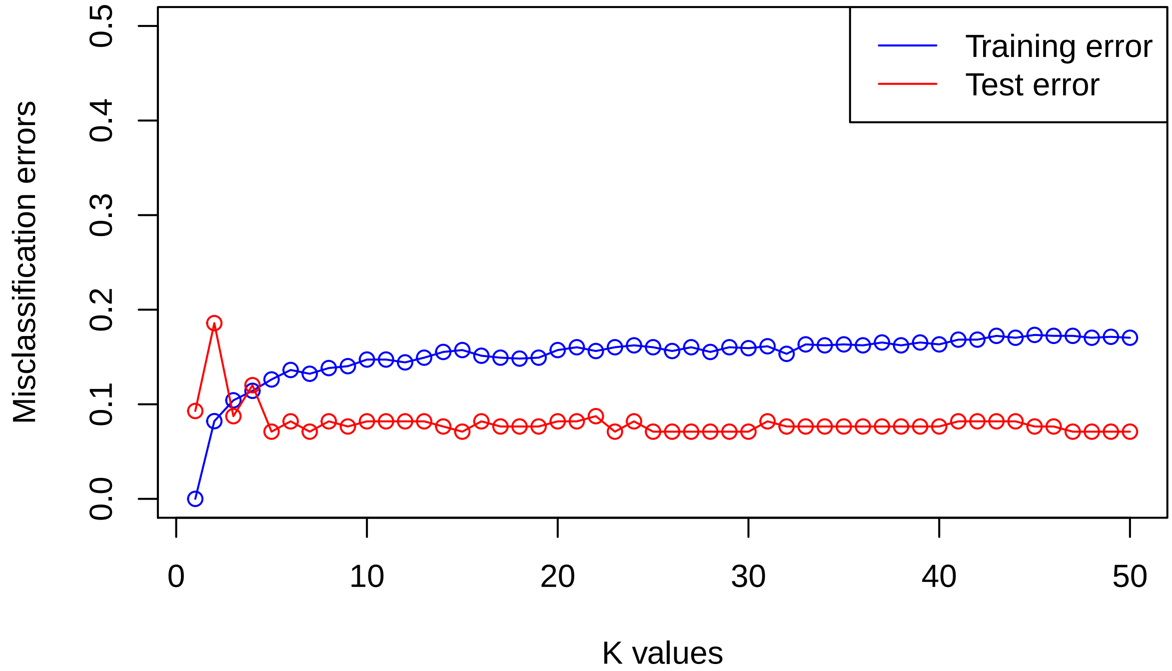
Use the FNN library to perform k-nearest neighbor classification, using as the neighbors the labeled points in the training set. Fit a classifier for $k = 1 : 50$, and find the mis-classification rate on both the training and test sets for each k. On a single plot, show the training set error and the test set error as a function of k. How would you choose the optimal k? Comment on your findings, and in particular on the possibility of overfitting.

```
set.seed(0)
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 3.4.4
```

```
test_error <- rep(0,50)
for (i in c(1:50)){
  m1 <- knn(train=train_merge[1:7], test=test_merge[1:7],cl=train_merge$trips_bi,k=i)
  test_error[i] <- (table(test_merge$trips_bi,m1)[2]+table(test_merge$trips_bi,m1)[3])/length(test_merg
}

train_error <- rep(0,50)
for (i in c(1:50)){
  m2 <- knn(train=train_merge[1:7], test=train_merge[1:7],cl=train_merge$trips_bi,k=i)
  train_error[i] <- (table(train_merge$trips_bi,m2)[2]+table(train_merge$trips_bi,m2)[3])/length(train_
}
```

```
plot(train_error, type="o", ylim=c(0,0.5), col="blue", xlab = "K values", ylab = "Misclassification erro
lines(test_error, type = "o", col="red")
legend("topright", legend=c("Training error","Test error"), col = c("blue","red"), lty=1:1)
```



Comments:

We can choose our optimal K where we achieve the minimum test error. According to this graph, the minimum test error is acheived for k~23. After that, we see some fluctation but then the test error begins to rise. If we were to use a larger k value, then we start to see an increase in test error, an indicator of overfitting. Also, we see a very low test error because the test data is not balanced (168 of 183 trip indicators are 1), thus our prediction is very accurate.

Problem 3: Classification for a Gaussian Mixture (25 points)

A Gaussian mixture model is a random combination of multiple Gaussians. Specifically, we can generate n data points from such a distribution in the following way. First generate labels $Y_1, \ldots, Y_n$ according to

$$Y_i = \begin{cases} 0 & \text{with probability } 1/2 \\ 1 & \text{with probability } 1/2. \end{cases}$$

Then, generate the data $X_1, \ldots, X_n$ according to

$$X_i \sim \begin{cases} N(\mu_0, \sigma_0^2) & \text{if } Y_i = 0 \\ N(\mu_1, \sigma_1^2) & \text{if } Y_i = 1. \end{cases}$$

Given such data $\{X_i\}$, we may wish to recover the true labels Y_i, which is a classification task.

Part a.

Suppose we have a mixture of two Gaussians, $N(\mu_0, \sigma_0^2)$ and $N(\mu_1, \sigma_1^2)$, with $\mu_0 = 0, \mu_1 = 3$, and $\sigma_0^2 = \sigma_1^2 = 1$. Consider the loss function $\mathbf{1}\{f(X) \neq Y\}$. What is the classifier that minimizes the expected loss? Your classifier will be a function $f : \mathbb{R} \to \{0, 1\}$, so write it as an indicator function. Show your work, and simplify your answer as much as possible.

What is the Bayes error rate? Again, show your work.

Solution:

From Bayes theorem, we know the classifier that minimizes the expected loss function is $\mathbf{1}\{m(x) > 1/2\}$. Then, from Bayes rule and the slides in class, we know $m(x) > 1/2$ is equivalent to $\frac{p_1(x)}{p_0(x)} > 1$, which is to say the ratio of the probability density functions for the Gaussian distributions is greater than one.

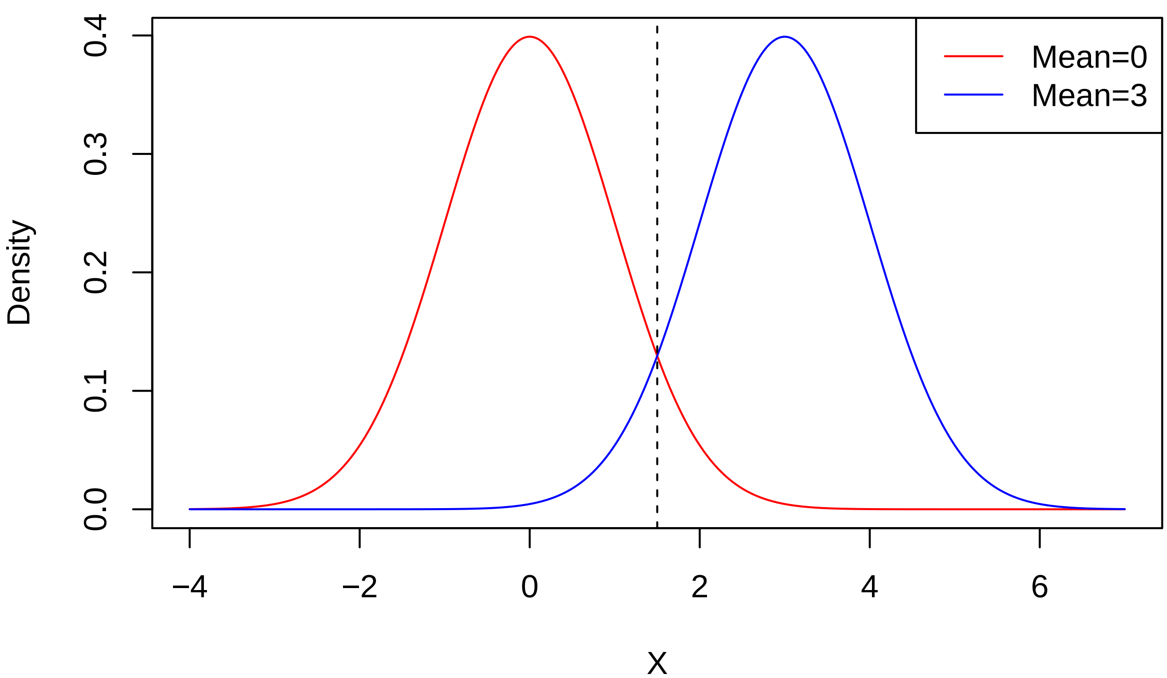
$$\frac{p_1(x)}{p_0(x)} = \frac{1/\sqrt{2\pi\sigma_1^2} * exp(-1/2)exp((x-\mu_1)^2/\sigma_1^2)}{1/\sqrt{2\pi\sigma_0^2} * exp(-1/2)exp((x-\mu_0)^2/\sigma_0^2)} = exp((x-\mu_1)^2 - (x-\mu_0)^2) = exp((x-3)^2 - (x)^2) > 1$$

The transition to line two comes from the fact that $\mu_0 = \mu_1 = 1$ so terms divide out and then we can simplify the exponent term. Now we can solve for x below:

$$exp((x-3)^2 - (x)^2) > 1(x-3)^2 - x^2 = x^2 - 6x + 9 - x^2 > 0x > 3/2$$

Thus our classifier that minimizes the loss function is $\mathbf{1}\{m(x) > 1/2\} = \mathbf{1}\{\frac{p_1(x)}{p_0(x)} > 1\}$when $x = 3/2$.

```
set.seed(0)
x0seq<-seq(-4,7,.01)
densities0<-dnorm(x0seq,0,1)
x1seq<-seq(-4,7,.01)
densities1<-dnorm(x1seq,3,1)
plot(x0seq,densities0,type="l",col="Red",xlab="X",ylab="Density")
lines(x1seq,densities1,col="Blue")
legend("topright", legend=c("Mean=0","Mean=3"), col = c("red","blue"), lty=1:1)
abline(v=3/2,col="Black",lty="dashed")
```



Now, because of the symmetry of these distributions due to their equal variance, we know that the Bayes error rate is equivalent to 2 times the the probability of Y=1 times the CDF of the normal distribution for $N(3, 1)$ below x=3/2. Since the probability of Y=1 is 1/2, then this is just equal to $\Phi(\frac{3/2-3}{1}) = \Phi(-\frac{3}{2}) = 1 - \Phi(\frac{3}{2})$. Again because of symmetry, this is equivalent to 1 minus the CDF of the normal distribution for $N(0, 1)$ below x=3/2 (or the CDF above x=3/2).

Part b.

Suppose we have the same mixture as in Part a, but now $\sigma_0^2 \neq \sigma_1^2$. What classifier minimizes the expected loss in this case?

Solution:

Again, we can solve for x using the fact that $p_1(x)/p_0(x) > 1$ for the classifier that minimizes the expected loss.

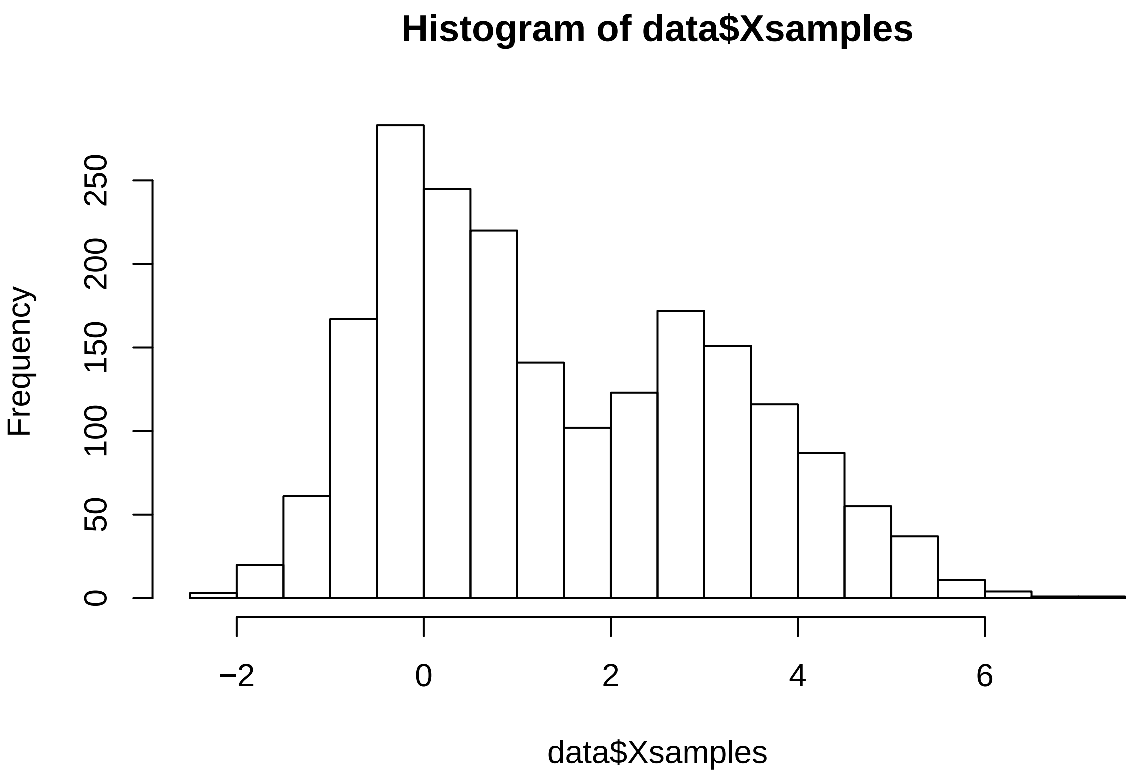
$$\frac{p_1(x)}{p_0(x)} = \frac{1/\sqrt{2\pi\sigma_1^2} * exp(-(x-\mu_1)^2/2\sigma_1^2)}{1/\sqrt{2\pi\sigma_0^2} * exp(-(x-\mu_0)^2/2\sigma_0^2)} = \frac{\sigma_0}{\sigma_1} * exp(-\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_0)^2}{2\sigma_0^2}) > 1 - \frac{(x-3)^2}{2\sigma_1^2} + \frac{(x)^2}{2\sigma_0^2} > ln(\frac{\sigma_1}{\sigma_0})$$

Thus our minimizing classifier is $\mathbf{1}\{m(x) > 1/2\} = \mathbf{1}\{\frac{p_1(x)}{p_0(x)} > 1\}$ when we solve for x above. For computational complexity, I did not bother solving explicitly for X here. In problem 3c, we can use this form in our classifier function, so we do not need to solve explicitly there either.

Part c.

Now generate $n = 2000$ data points from the mixture where $\mu_0 = 0, \mu_1 = 3$, and $\sigma_0^2 = 0.5, \sigma_1^2 = 1.5$. Plot a histogram of the X's. This histogram is meant to be a sanity check for you; it should help you verify that you've generated the data properly.

```r
set.seed(1)
N=2000
Ysamples <- rbinom(N,1,1/2)
Xsamples <- rep(0,length(Ysamples))
for (i in c(1:length(Ysamples))) {
  if (Ysamples[i]==0) {
    x_i <- rnorm(1,mean=0,sd=sqrt(0.5))
  }
  if (Ysamples[i]==1) {
    x_i <- rnorm(1,mean=3,sd=sqrt(1.5))
  }
  Xsamples[i] <- x_i
}
data <- data.frame(cbind(Ysamples,Xsamples))
hist(data$Xsamples,breaks=30)
```



Set aside a randomly-selected test set of $n/5$ points. We will refer to the rest of the data as the training data. Use the labels of the training data to calculate the group means. That is, calculate the mean value of all the X_i's in the training data with label $Y_i = 0$. Call this sample mean $\widehat{\mu}_0$. Do the same thing to find $\widehat{\mu}_1$. To be explicit, let $C_j = \{i : Y_i = j\}$, and define

$$\widehat{\mu}_j = \frac{1}{|C_j|} \sum_{i \in C_j} X_i$$

Now classify the data in your test set. To do this, recall that your rule in Part b. depended on the true data means $\mu_0 = 0$ and $\mu_1 = 3$. Plug in the sample means $\widehat{\mu}_j$ instead. You should be able to do the classification in a single line of code, but there is no penalty for using more lines. Evaluate the estimator's performance using the loss:

$$\frac{1}{n} \sum_{i=1}^{n} 1\{\widehat{Y}_i \neq Y_i\}$$

```
set.seed(1)
sample_size <- N/5
test_rows <- sample(nrow(data), N/5,replace=FALSE)
test_data <- data[test_rows,]
train_data <- data[-test_rows,]
mu0_hat <- mean(train_data[train_data$Ysamples == 0,]$Xsamples)
mu1_hat <- mean(train_data[train_data$Ysamples == 1,]$Xsamples)

Ypred <- ifelse((((-(test_data$Xsamples-mu1_hat)^2/3 + (test_data$Xsamples-mu0_hat)^2)>log(sqrt(1.5/0.5)
#The error is the mean of the sum of the indicators where it is 1 if Ypred[i]=Ytest[i]
error_rate <- sum(ifelse(Ypred!=test_data$Ysamples,1,0))/sample_size
print(paste("Error Rate:",error_rate))
```

```
## [1] "Error Rate: 0.0625"
```

Part d.

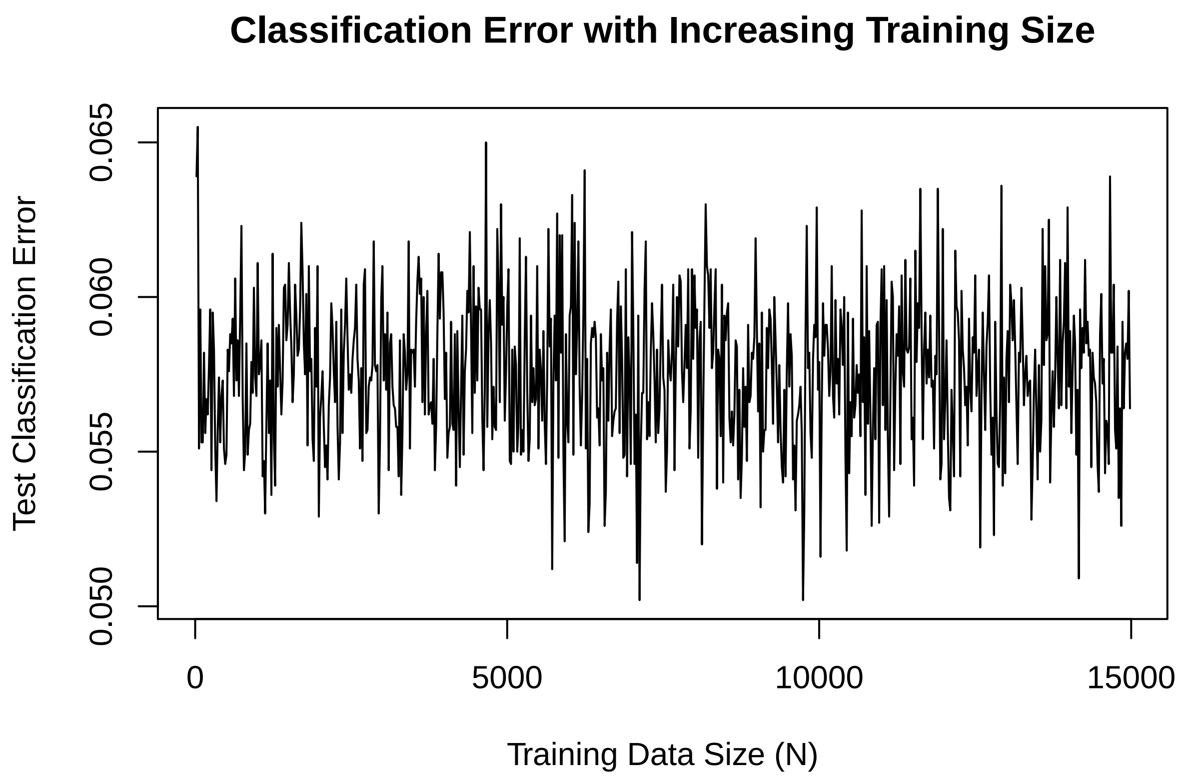
Now you train and evaluate classifiers for training sets of increasing size n, as specified below. For each n, you should

1. Generate a training set of size n from the mixture model in Part c.
2. Generate a test set of size 10,000. Note that the test set itself will change on each round, but the size will always be the same: 10,000.
3. Compute the sample means on the training data.
4. Classify the test data as described in Part c.
5. Compute the error rate.

Plot the error rate as a function of n. Comment on your findings. What is happening to the error rate as n grows?

```
seq.n <- seq(from = 1, to = 15000, by = 20)

error <- rep(0,length(seq.n))

for (j in c(1:length(seq.n))) {
  N = seq.n[j]
  #Train Data
  Ysamples <- rbinom(N,1,1/2)
  Xsamples <- rep(0,N)
  for (i in c(1:length(Ysamples))) {
    if (Ysamples[i]==0) {
      x_i <- rnorm(1,mean=0,sd=sqrt(0.5))
    }
    if (Ysamples[i]==1) {
      x_i <- rnorm(1,mean=3,sd=sqrt(1.5))
    }
    Xsamples[i] <- x_i
  }
  train_data <- data.frame(cbind(Ysamples,Xsamples))

  #Test Data
  Ysamples_test <- rbinom(10000,1,1/2)
  Xsamples_test <- rep(0,10000)
  for (i in c(1:length(Ysamples_test))) {
    if (Ysamples_test[i]==0) {
      x_i <- rnorm(1,mean=0,sd=sqrt(0.5))
    }
    if (Ysamples_test[i]==1) {
      x_i <- rnorm(1,mean=3,sd=sqrt(1.5))
    }
    Xsamples_test[i] <- x_i
  }
  test_data <- data.frame(cbind(Ysamples_test,Xsamples_test))

  #Make predictions according to formula derived in part 3b.
  mu0_hat <- mean(train_data[train_data$Ysamples == 0,]$Xsamples)
  mu1_hat <- mean(train_data[train_data$Ysamples == 1,]$Xsamples)
  Ypred <- ifelse(((-(test_data$Xsamples_test-mu1_hat)^2/3 + (test_data$Xsamples_test-mu0_hat)^2)>log(s
```

```
  #The test error is the mean of the sum of the indicators where it is 1 if Ypred[i]=Ytest[i]
  error_rate[j] <- sum(ifelse(Ypred!=test_data$Ysamples,1,0))/10000
}

plot(seq.n,error_rate,type="l",xlab="Training Data Size (N)",ylab="Test Classification Error",main="Cla
```



Comments:

After a quick decrease where the training size is extremely small, the error rate appears to be randomly oscillating. However, the error rate itself is gradually decreasing. Talking to Professor Lafferty office hours this makes sense because of the given test data size. If we were to increase the size of the test data, we would see a more significant decrease in classification error as N increases.