# Homework 1

Applied Data Mining and Machine Learning

*John Brandt*

*January 30, 2018*

## Conceptual Questions

### Number 1

$$MSE = E[(\hat{f} - f)^2] = bias(\hat{f})^2 + var(\hat{f})$$

$$MSE = E[(\hat{f} - E[\hat{f}] + E[\hat{f}] - f)^2]$$

$$MSE = E[((\hat{f} - E(\hat{f}))^2 + 2(\hat{f} - E[\hat{f}])(E[f] - f) + (E[f] - \hat{f})^2$$

$$MSE = E[(\hat{f} - E[\hat{f}])^2] + E[(\hat{f} - E[\hat{f}]) * 2(E[\hat{f}) - f)] + (E[f] - \hat{f})^2)$$

$$MSE = E[(\hat{f} - E[\hat{f}])^2] + (E[(f) - \hat{f})^2$$

### Number 2

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

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

$$V \ \sim N(\mu, \Sigma)$$

$$f(v) = \frac{1}{\sqrt{|2\pi\sigma^2|}} \exp\Big(-\frac{1}{2\sigma^2}(Y-\beta x)^2\Big).$$

$$ln\Big(\frac{1}{\sqrt{2\pi\sigma^2}}\Big) - \frac{1}{2\sigma^2}\Big(Y - \beta x\Big)^2$$

$$\frac{\partial v}{\partial \beta} = \frac{1}{\sigma^2}\Big(x(y - \hat{\beta}x)\Big) = 0$$

$$x(y - \hat{\beta}x) = 0$$

$$xy - \hat{\beta}x^2 = 0$$

$$\hat{\beta}x^2 = xy$$

$$\hat{\beta} = \frac{xy}{x^2}$$

$$\hat{\beta} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2}$$

## Load in data

```
main <- read.csv("citibike_main.csv")
weather <- read.csv("weather.csv")
```

## a) Data pre-Processing

Merge dataframes and convert relevant columns to factors.

```
main <- merge(main, weather)
main$holiday <- factor(main$holiday)
main$month <- factor(main$month)
main$dayofweek <- factor(main$dayofweek)
```

## b) Splitting the dataset for training/validation

Use random number generator to split data frame into 80% training and 20% test sets.

```
set.seed(123)

s <- sample(1:nrow(main), nrow(main)/5, replace=FALSE)
test <- main[s,]
train <- main[-s,]
```

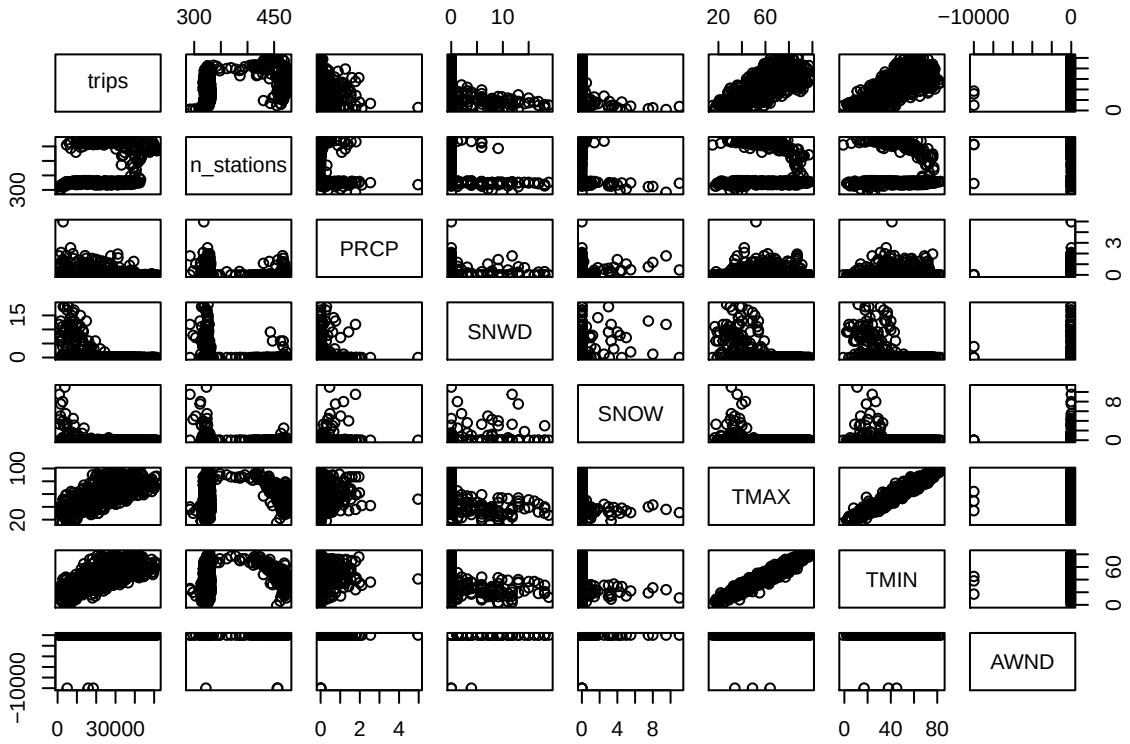
Tmax and Tmin are highly colinear (r=0.97). Tmax will be considered instead of Tmin because of its slightly higher correlation (0.76 vs. 0.74) with trips.

```
round(cor(train[,-c(1,4,5,6)]),2)
```

```
           trips n_stations  PRCP  SNWD  SNOW  TMAX  TMIN  AWND
trips       1.00       0.24 -0.28 -0.45 -0.27  0.76  0.74  0.06
n_stations  0.24       1.00 -0.04 -0.15 -0.09 -0.10 -0.11 -0.06
PRCP       -0.28      -0.04  1.00  0.02  0.23 -0.03 -0.01  0.02
SNWD       -0.45      -0.15  0.02  1.00  0.25 -0.41 -0.43 -0.01
SNOW       -0.27      -0.09  0.23  0.25  1.00 -0.23 -0.23  0.01
TMAX        0.76      -0.10 -0.03 -0.41 -0.23  1.00  0.97  0.04
TMIN        0.74      -0.11 -0.01 -0.43 -0.23  0.97  1.00  0.05
AWND        0.06      -0.06  0.02 -0.01  0.01  0.04  0.05  1.00
```

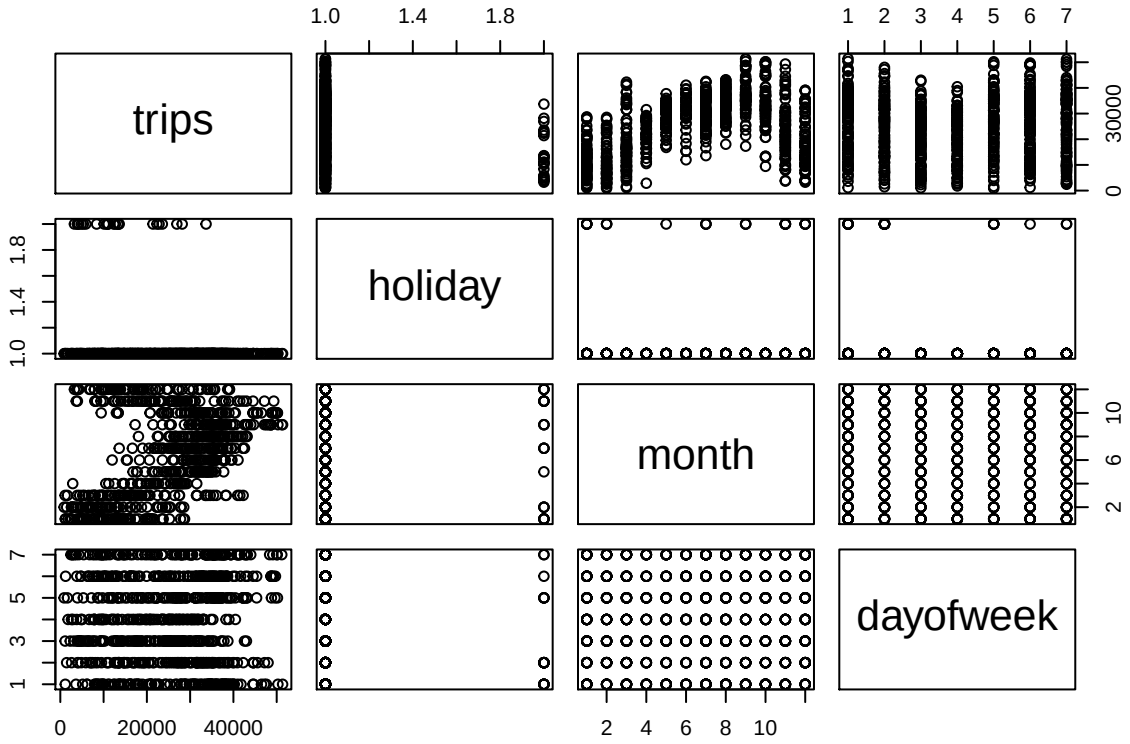
Plots of categorical variables show that holiday, month, and day of week likely are all important predictors of trips. There are less trips on holidays, more trips during the summer, and less trips on tuesday/wednesdays.

```
plot(train[,-c(1,4,5,6,13)])
```




Correlation plots between numerical variables indicate that data transformation is not necessary.

```
plot(train[,c(2,4,5,6)])
```



## c) Linear Regression

### Model with all predictors

```
lm_all <- lm(trips ~ dayofweek + TMIN + month + n_stations + holiday +
               PRCP + SNWD + SNOW + TMAX + AWND, data=train)
summary(lm_all)
```

```
Call:
lm(formula = trips ~ dayofweek + TMIN + month + n_stations +
    holiday + PRCP + SNWD + SNOW + TMAX + AWND, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-19123.2  -2306.0    293.3   2726.6  19404.7

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -2.073e+04  1.397e+03 -14.843  < 2e-16 ***
dayofweekMon   -7.350e+02  5.594e+02  -1.314  0.18921
dayofweekSat   -5.226e+03  5.527e+02  -9.455  < 2e-16 ***
dayofweekSun   -6.020e+03  5.637e+02 -10.678  < 2e-16 ***
dayofweekThurs  6.698e+02  5.652e+02   1.185  0.23633
dayofweekTues  -4.868e+02  5.528e+02  -0.881  0.37885
dayofweekWed    7.498e+02  5.528e+02   1.356  0.17539
TMIN           -6.847e+01  3.724e+01  -1.839  0.06634 .
month2         -3.381e+02  7.930e+02  -0.426  0.66999
month3          1.323e+03  7.225e+02   1.832  0.06741 .
month4          4.965e+03  9.445e+02   5.257 1.89e-07 ***
month5          8.310e+03  1.069e+03   7.776 2.38e-14 ***
month6          9.367e+03  1.175e+03   7.971 5.62e-15 ***
month7          7.697e+03  1.201e+03   6.409 2.55e-10 ***
month8          9.507e+03  1.159e+03   8.200 9.95e-16 ***
month9          1.209e+04  1.051e+03  11.494  < 2e-16 ***
month10         1.169e+04  8.977e+02  13.025  < 2e-16 ***
month11         6.527e+03  7.728e+02   8.446  < 2e-16 ***
month12         2.387e+03  7.410e+02   3.221  0.00133 **
n_stations      6.837e+01  3.259e+00  20.981  < 2e-16 ***
holidayTRUE    -1.026e+04  9.048e+02 -11.337  < 2e-16 ***
PRCP           -7.764e+03  4.256e+02 -18.245  < 2e-16 ***
SNWD           -2.208e+02  6.960e+01  -3.172  0.00157 **
SNOW           -8.802e+01  2.030e+02  -0.434  0.66470
TMAX            3.485e+02  3.380e+01  10.311  < 2e-16 ***
AWND            7.349e-01  2.492e-01   2.949  0.00328 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4231 on 775 degrees of freedom
Multiple R-squared:  0.8731,    Adjusted R-squared:  0.869
F-statistic: 213.3 on 25 and 775 DF,  p-value: < 2.2e-16
```

## Model with 5 predictors

Because only saturday and sunday were significant in the intial regression, I bin the day of week into weekend/non-weekend.

```
test$weekend <- 0
test$weekend[test$dayofweek %in% c("Sat","Sun")] <- 1
test$weekend <- factor(test$weekend)
train$weekend <- 0
train$weekend[train$dayofweek %in% c("Sat","Sun")] <- 1
train$weekend <- factor(train$weekend)
```

### Removing variables

Month is removed, because people likely bike less during the winter because of temperature and snow depth, and not something inherent to what portion of the year it is.

Tmin is removed because of the colinearity with Tmax.

This results in the following model with 8 predictors:

```
lm_8_predictors <- lm(trips ~ weekend + n_stations + holiday + PRCP +
                        SNWD + SNOW + TMAX + AWND, data = train)

summary(lm_8_predictors)
```

```
Call:
lm(formula = trips ~ weekend + n_stations + holiday + PRCP +
    SNWD + SNOW + TMAX + AWND, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-16957.2  -2743.7    -72.7   3032.2  22023.2

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.098e+04  1.494e+03 -14.044  < 2e-16 ***
weekend1    -5.853e+03  3.983e+02 -14.696  < 2e-16 ***
n_stations   6.030e+01  3.269e+00  18.446  < 2e-16 ***
holidayTRUE -1.069e+04  1.048e+03 -10.199  < 2e-16 ***
PRCP        -8.163e+03  4.960e+02 -16.458  < 2e-16 ***
SNWD        -3.958e+02  6.982e+01  -5.669 2.01e-08 ***
SNOW        -1.911e+02  2.370e+02  -0.807   0.4202
TMAX         4.465e+02  1.057e+01  42.257  < 2e-16 ***
AWND         6.166e-01  2.916e-01   2.114   0.0348 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5005 on 792 degrees of freedom
Multiple R-squared:  0.8185,    Adjusted R-squared:  0.8167
F-statistic: 446.6 on 8 and 792 DF,  p-value: < 2.2e-16
```

Snow is removed from the model because the correlation plots suggest it is already modeled by the combination of temperature and precipitation.

Wind direction is removed from the model because it is only marginally significant. Based upon t-values and standard error, snow depth is also removed from the model in order to bring the number of predictors to 5, creating the following model:

$$trips = weekend + n\_stations + holiday + precipitation + max\_temperature$$

```
lm_5_predictors <- lm(trips ~ weekend + n_stations + holiday + PRCP + TMAX, data = train)
summary(lm_5_predictors)
```

```
Call:
lm(formula = trips ~ weekend + n_stations + holiday + PRCP +
    TMAX, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-19449.1  -2994.1     -3.4   3167.1  23151.1

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -24517.938   1387.801 -17.667   <2e-16 ***
weekend1     -5871.616    404.599 -14.512   <2e-16 ***
n_stations      64.083      3.255  19.686   <2e-16 ***
holidayTRUE -10533.802   1071.450  -9.831   <2e-16 ***
PRCP         -8221.387    494.306 -16.632   <2e-16 ***
TMAX           475.303      9.603  49.496   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5120 on 795 degrees of freedom
Multiple R-squared:  0.8094,    Adjusted R-squared:  0.8082
F-statistic: 675.4 on 5 and 795 DF,  p-value: < 2.2e-16
```

```
lm_5 <- lm_5_predictors
```

## 4 Predictor model

Iteratively removing one predictor at a time, and using the AIC to test for model fit, indicates that removing the "holiday" variable results in the best-fit 4 predictor model as follows:

$$trips = weekend + n\_stations + precipitation + max\_temperature$$

```
lm_4_predictors_opt_1 <- lm(trips ~ weekend + n_stations + holiday + PRCP, data = train)
lm_4_predictors_opt_2 <- lm(trips ~ weekend + n_stations + PRCP, data = train)
lm_4_predictors_opt_3 <- lm(trips ~ weekend + n_stations + PRCP + TMAX, data = train)
lm_4_predictors_opt_4 <- lm(trips ~ weekend + holiday + PRCP + TMAX, data =train)
lm_4_predictors_opt_5 <- lm(trips ~ n_stations + holiday + PRCP + TMAX, data =train)

AIC(lm_4_predictors_opt_1, lm_4_predictors_opt_2, lm_4_predictors_opt_3,
    lm_4_predictors_opt_4, lm_4_predictors_opt_5)
```

```
                      df      AIC
lm_4_predictors_opt_1  6 17088.13
```

```
lm_4_predictors_opt_2  5 17130.58
lm_4_predictors_opt_3  6 16053.44
lm_4_predictors_opt_4  6 16279.59
lm_4_predictors_opt_5  6 16149.77
```

```
lm_4 <- lm_4_predictors_opt_3
```

## 3 Predictor model

As before, iteratively remove one predictor at a time, and train for model fit with AIC. Doing so suggests that removing the weekend variable results in the best 3-predictor model as follows:

$$trips = n\_stations + precipitation + maxtemperature$$

```
lm_3_predictors_opt_1 <- lm(trips ~ weekend + n_stations + PRCP, data = train)
lm_3_predictors_opt_2 <- lm(trips ~ weekend + n_stations + TMAX, data = train)
lm_3_predictors_opt_3 <- lm(trips ~ weekend + PRCP + TMAX, data = train)
lm_3_predictors_opt_4 <- lm(trips ~ n_stations + PRCP + TMAX, data = train)

AIC(lm_3_predictors_opt_1, lm_3_predictors_opt_2,
    lm_3_predictors_opt_3, lm_3_predictors_opt_4)
```

```
                     df      AIC
lm_3_predictors_opt_1  5 17130.58
lm_3_predictors_opt_2  5 16259.72
lm_3_predictors_opt_3  5 16339.05
lm_3_predictors_opt_4  5 16200.31
```

```
lm_3 <- lm_3_predictors_opt_4
```

## 2 Predictor model

Best two predictor model, determined by iterative removal and AIC is:

$$trips = n\_stations + max\_temperature$$

```
lm_2_predictors_opt_1 <- lm(trips ~ n_stations + PRCP, data = train)
lm_2_predictors_opt_2 <- lm(trips ~ PRCP + TMAX, data = train)
lm_2_predictors_opt_3 <- lm(trips ~ n_stations + TMAX, data = train)

AIC(lm_2_predictors_opt_1, lm_2_predictors_opt_2, lm_2_predictors_opt_3)
```

```
                     df      AIC
lm_2_predictors_opt_1  4 17172.63
lm_2_predictors_opt_2  4 16456.81
lm_2_predictors_opt_3  4 16365.02
```

```
lm_2 <- lm_2_predictors_opt_3
```

## 1 Predictor model

The best one predictor model, determined by iterative removal and AIC is:

$$trips = \beta_1 max\_temperature + \epsilon$$

```
lm_1_predictor_opt_1 <- lm(trips ~ n_stations, data=train)
lm_1_predictor_opt_2 <- lm(trips ~ TMAX, data = train)
AIC(lm_1_predictor_opt_1, lm_1_predictor_opt_2)
```

```
                     df      AIC
lm_1_predictor_opt_1  3 17234.69
lm_1_predictor_opt_2  3 16592.75
```

```
lm_1 <- lm_1_predictor_opt_2
```

# Testing model on test data set

In order to validate the model, I use the predict function to run the model fit on a previously established smaller subset of the data.

```
lm_all_test <- predict(lm_all, newdata=test)
lm_5_test <- predict(lm_5, newdata=test)
lm_4_test <- predict(lm_4, newdata=test)
lm_3_test <- predict(lm_3, newdata=test)
lm_2_test <- predict(lm_2, newdata=test)
lm_1_test <- predict(lm_2, newdata=test)
```

## Model comparison

### Extract R-squared

### Training models

```
lm_all_r2 <- summary(lm_all)$r.squared
lm_5_r2 <- summary(lm_5)$r.squared
lm_4_r2 <- summary(lm_4)$r.squared
lm_3_r2 <- summary(lm_3)$r.squared
lm_2_r2 <- summary(lm_2)$r.squared
lm_1_r2 <- summary(lm_1)$r.squared
```

### Test models

In order to extract R2 from the test models, I construct a function that calculates

$$R^2 = 1 - (SSresidual/SStotal)$$

```
R2 <- function(model, data) {
  SS.total <- sum((data$trips - mean(data$trips))^2)
  SS.residual <- sum((data$trips - model)^2)
  r.sq <- 1 - SS.residual/SS.total
}
```

```
lm_all_r2_t <- R2(lm_all_test, test)
lm_5_r2_t <- R2(lm_5_test, test)
lm_4_r2_t <- R2(lm_4_test, test)
lm_3_r2_t <- R2(lm_3_test, test)
lm_2_r2_t <- R2(lm_2_test, test)
lm_1_r2_t <- R2(lm_1_test, test)

rsquared <- c(lm_all_r2, lm_5_r2, lm_4_r2, lm_3_r2, lm_2_r2,
              lm_1_r2, lm_all_r2_t, lm_5_r2_t, lm_4_r2_t, lm_3_r2_t, lm_2_r2_t, lm_1_r2_t)
names <- c('all', '5', '4', '3', '2', '1', 'all', '5',
           '4', '3', '2', '1')
```

### Extract Mean Squared Error

```
# Train model mean squared error = mean of residuals squared
mse <- function(model) {
  mean(model$residuals^2)
}

# Test model mean squared errors = mean of residuals squared
msetest <- function(model, data) {
  mean((data$trips - model)^2)
}

mse_all <- mse(lm_all)
mse_5 <- mse(lm_5)
mse_4 <- mse(lm_4)
mse_3 <- mse(lm_3)
mse_2 <- mse(lm_2)
mse_1 <- mse(lm_1)


mse_all_t <- msetest(lm_all_test, test)
mse_5_t <- msetest(lm_5_test, test)
mse_4_t <- msetest(lm_4_test, test)
mse_3_t <- msetest(lm_3_test, test)
mse_2_t <- msetest(lm_2_test, test)
mse_1_t <- msetest(lm_1_test, test)

mse_list <- c(mse_all, mse_5, mse_4, mse_3, mse_2, mse_1, mse_all_t,
              mse_5_t, mse_4_t, mse_3_t, mse_2_t, mse_1_t)
```

### Bind MSE, Rsquared, predictors, and category to a dataframe

```
summary_table <- data.frame(cbind(do.call('rbind', strsplit(names, " ")),
                                  round(rsquared,2)))

summary_table$mse <- mse_list
summary_table$category <- "test"
summary_table[7:12,4] <- "train"
```

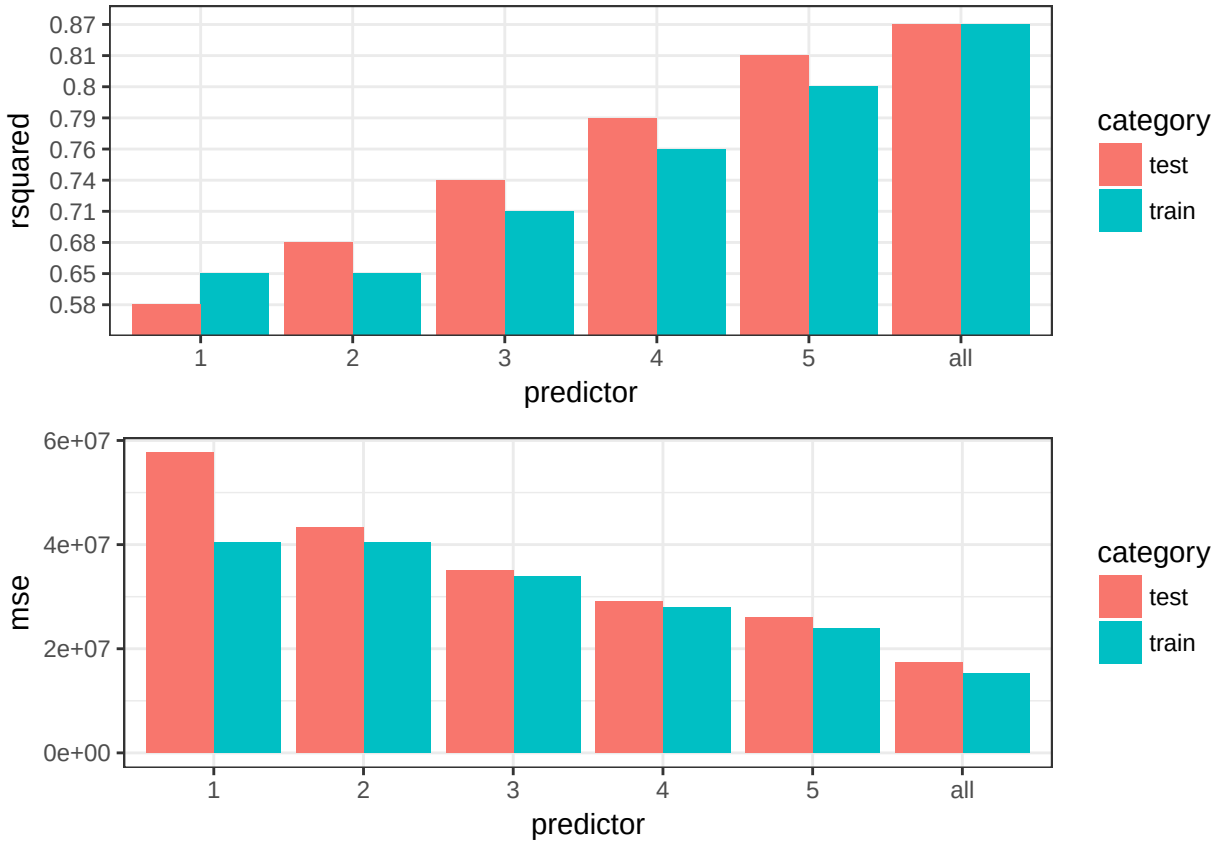
```
colnames(summary_table) <- c("predictor", "rsquared", "mse", "category")
summary_table$predictor <- factor(summary_table$predictor)
```

### Plot R-squared and MSE of test and train fit

```
model_plot <- ggplot(aes(x=predictor, y=rsquared), data=summary_table)+
  geom_bar(aes(fill=category), stat="identity", position="dodge")+
  theme_bw()

mse_plot <- ggplot(aes(x=predictor, y=mse), data=summary_table)+
  geom_bar(aes(fill=category), stat="identity", position="dodge")+
  theme_bw()

multiplot(model_plot, mse_plot, cols=1)
```




## d) K-nearest neighbors regression

### i) Scale variables and set seed

```
# Remove categorical data variables
test_scale <- test
num.vars <- sapply(test_scale, is.numeric)
```

```
test_scale[num.vars] <- lapply(test_scale[num.vars], scale)

# Scale variables
test_scale <- test_scale[, num.vars]

# Do the same for training set
train_scale <- train
num.vars <- sapply(train_scale, is.numeric)

train_scale[num.vars] <- lapply(train_scale[num.vars], scale)
train_scale <- train_scale[, num.vars]

# Set seed for reproducability
set.seed(123)
```

### ii) Train-set k-nearest neighbors regression from k = 1:50

Apparently, you're not supposed to include your response variable in the input dataframe to the knn.reg function, so I subset the columns that are not "trips" in the input, and specify the trips as the y-response. This generates different results than does keeping trips in the KNN function, but it makes more sense not to model trips based upon trips.

```
knn_df <- as.data.frame(matrix(0, 50, 3))


# No matter what I did (and I tried for 2 hours!) I could not get knn.reg to work when i=2,
# so I only was able to run the models from 3:50

for (i in 3:50) {
  model <- knn.reg(train_scale[,-1], y=train_scale$trips, k=i) # remove 'trips' from input
  knn_df[i,1] <- "train"
  knn_df[i,2] <- round(mse(model), 4)
  knn_df[i,3] <- i
}
colnames(knn_df) <- c("category", "mse", "k")

knn_df_test <- as.data.frame(matrix(0, 50, 3))
```

### Test-set k-nearest neighbors regression from k = 1:50

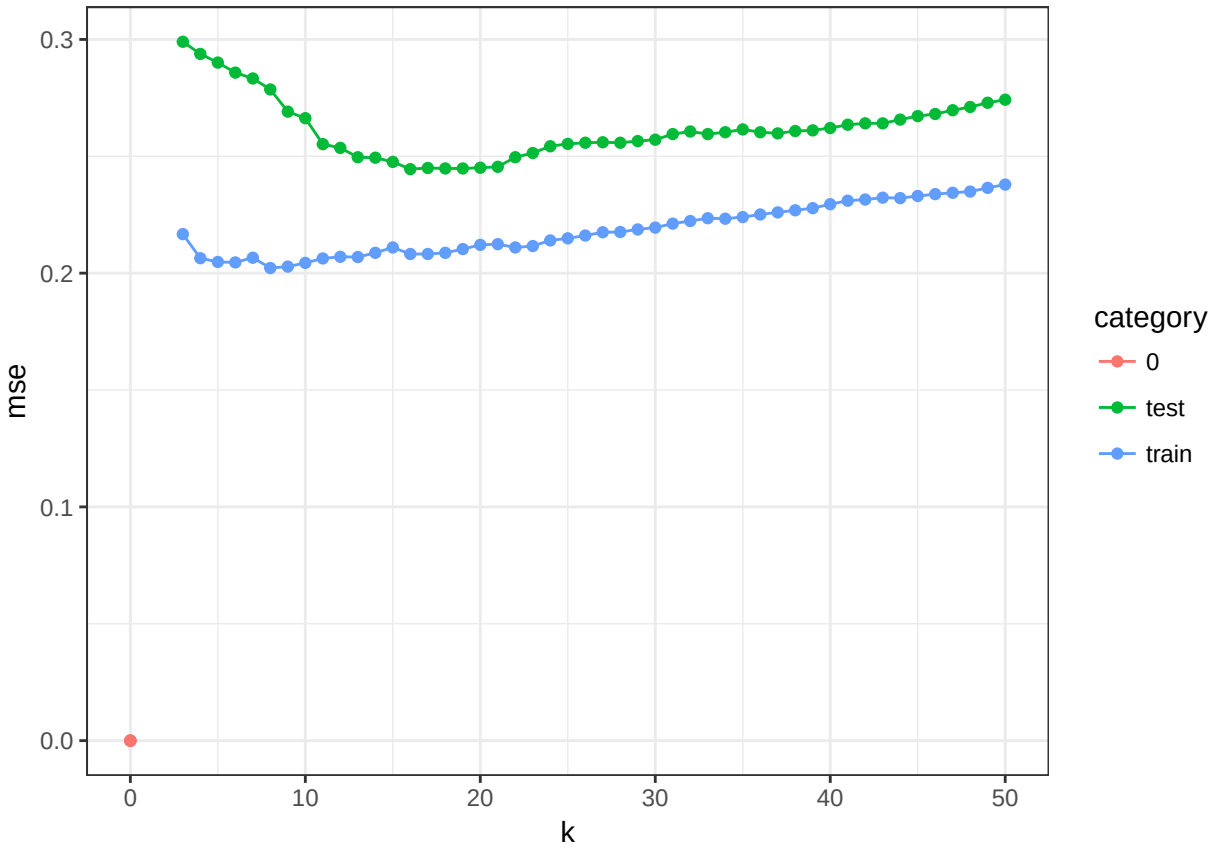
```
for (i in 3:50) {
  model <- knn.reg(train_scale[,-1], test=test_scale[,-1], y=train_scale$trips, k=i)
  knn_df_test[i,1] <- "test"
  knn_df_test[i,2] <- round(mean(abs(model$pred - test_scale$trips)^2), 4)
  knn_df_test[i,3] <- i
}
```

### Plot of MSE for train and test k-nearest neighbors regression

```
colnames(knn_df_test) <- c("category", "mse", "k")
knn_df <- rbind(knn_df, knn_df_test)

knn_plot <- ggplot(data = knn_df, aes(x=k, y=mse))+
  geom_point(aes(color=category))+
  geom_line(aes(color=category))+
  theme_bw()

print(knn_plot)
```




## e) Predictions for test set

Of the 6 models specified by this problem set, I choose the model with 5 predictors to be the best linear model. This model does not have the autocorrelation issue between Tmin and Tmax, while still having nearly as good of test/training R-squared and MSE as does the model with all predictors.

$$trips = weekend + n\_stations + holiday + precipitation + max\_temperature$$

The K-nearest neighbors regression with K = 16 was chosen as the best KNN model, because it is the point where the average of the test and train MSE is the smallest, indicating that the model is adequately balancing variance and bias.

## Reading in final test dataset

```
final_test <- read.csv("citibike_test.csv")
final_test <- merge(final_test, weather)
final_test$holiday <- factor(final_test$holiday)
final_test$month <- factor(final_test$month)
final_test$dayofweek <- factor(final_test$dayofweek)

final_test$weekend <- 0
final_test$weekend[final_test$dayofweek %in% c("Sat","Sun")] <- 1
final_test$weekend <- factor(final_test$weekend)
```

## Linear model predictions

```
lm_5_prediction <- predict(lm_5, newdata=final_test)
```

```
final_test_scale <- final_test
num.vars_final <- sapply(final_test_scale, is.numeric)

final_test_scale[num.vars_final] <- lapply(final_test_scale[num.vars_final], scale)
final_test_scale <- final_test_scale[, num.vars_final]

# This creates an error where snow and snowdepth are set to NaN because they cannot
# be scaled because all data is 0 for the prediction set.
# To deal with this, I reset these columns to 0.

final_test_scale$SNWD = 0
final_test_scale$SNOW = 0


knn_prediction <- knn.reg(train_scale[,-1], test=final_test_scale,
                          y=train_scale$trips, k=16)
```

### Descale the trips predicted data and write to dataframe

```
final_predictions <- as.data.frame(matrix(0, 183, 3))
colnames(final_predictions) <- c("date", "trips_lm", "trips_knn")
```

To convert the KNN prediction from scaled trips to trips, I assume that the $\mu$ and $\sigma$ of the predicted trips is equal to the $\mu$ and $\sigma$ of the training data set.

```
knn_prediction_descaled <- round((knn_prediction$pred*sd(train$trips))+
                                   mean(train$trips),0)
final_predictions$trips_knn <- knn_prediction_descaled
final_predictions$date <- final_test$date
final_predictions$trips_lm <- round(lm_5_prediction, 0)
```

### Write CSV

```
write.csv(final_predictions, file='final_predictions.csv', row.names=FALSE)
```

```
finalplots <- ggplot(data=final_predictions, aes(y=trips_knn, x=trips_lm))+
    geom_point()+
    geom_smooth(method="lm", se=FALSE)+
    theme_bw()+
    geom_abline(slope=1)+
  ylab("Predicted trips (KNN)")+
  xlab("Predicted trips (LM)")+
  theme(panel.grid.minor=element_blank())+
  ggtitle("KNN underpredicts trips relative to LM")

print(finalplots)
```