# STATS 769 - Lab 03 - bole001

*Bernard O'Leary*

*18 August 2019*

## Read in data and manipulate using Linux commands

Data format is CSV (Comma Separated Values), reading into R using the read.csv function. Becuase there are a large number of files an alternative approach has been taken, to list all files in the Lab02 directory accroding to the file structure we are looking for ("trips*.csv") and then pick up each file and add to a dataframe (called"scooter_trips.df").

The machine that was used to create the report is a Microsoft Windows 10 machine that has an Ubuntu Linux Bash shell built into it as part of the WSL (Windows Subsystem for Linux) feature that is an optional feature of the Windows 10 OS (https://docs.microsoft.com/en-us/windows/wsl/install-win10). Once access was gained to the STATS 769 Linux machines, the files were copied to the Windows 10 machine used to create the report using the following scp command:

```
bernardo@PKS10198:/mnt/d/Study/UoA-STATS-769$ scp bole001@sc-cer00014-04.its.auckland.ac.nz:/course/Lab
```

A shell script called "STATS769_2019_S2_bole001_lab03.sh" was created to import the data and to filter out the records that contained the word "scooter", then create a new file for each corresponding original file prefixed with the word scooter, and push those reocrds into those files. The code in the scheel script is as follows:

```
#!/bin/bash
# file: STATS769_2019_S2_bole001_lab03.sh

# Copy all of the .csv files to the local directory
cp /course/Labs/Lab02/*.csv .

# Count the total number of bicycle trips
echo "Total bicycle trips:"
grep "bicycle" trips*.csv | wc -l

# Get the scooter strips for each file and put into another file
for i in trips*.csv
do
    head -1 $i > "scooter-"$i
    grep "scooter" $i >> "scooter-"$i
    echo "$i created"
done

# Run the R code for the lab
echo "Running RMD script"
Rscript -e 'library("rmarkdown"); render("STATS769_2019_S2_bole001_lab03.Rmd")'

# Exit
echo "Done"
```

Whereas on the Windows 10 machine used to create the report, it was set to the local filesystem where the csv files for the lab were copied to. Code used to create the initial dataframe for the scooter Trips data follows.

```
# Files are already in our local directory
directory = '.'
scooter_files.ls <- intersect(list.files(path=directory, pattern="scooter"), list.files(path=directory,
# First apply read.csv, then rbind
scooter_trips.df = do.call(rbind, lapply(scooter_files.ls, function(x) read.csv(x, stringsAsFactors = F
```

## Get rid of non-positive and non-finite values in duration and distance

Remove problematic data from the entire dataset.

```
scooter_trips.df <- subset(scooter_trips.df, scooter_trips.df$Trip.Duration > 0)
scooter_trips.df <- subset(scooter_trips.df, scooter_trips.df$Trip.Distance > 0)
scooter_trips.df <- subset(scooter_trips.df, is.finite(scooter_trips.df$Trip.Duration))
scooter_trips.df <- subset(scooter_trips.df, is.finite(scooter_trips.df$Trip.Distance))
```

## Add logged duration and distance variables

Name the variables Trip.Duration.Logged and Trip.Distance.Logged and add to the end of the dataframe. Note that because the data are not cleansed by this point NaNs are produced here.

```
scooter_trips.df$Trip.Duration.Logged <- log(scooter_trips.df$Trip.Duration)
scooter_trips.df$Trip.Distance.Logged <- log(scooter_trips.df$Trip.Distance)
```

## Create the linear model and measure using k-fold cross-validation

Fit a simple linear regression model via k-fold cross-validation (with k = 10) to predict the trip duration based on trip distance and measure the test MSE.

```
# Define MSE function
MSE <- function(m, o) {
    mean((m - o)^2)
}

# Randomly shuffle the data
scooter_trips.df <- scooter_trips.df[sample(nrow(scooter_trips.df)),]

MSE_for_fold.df <- data.frame(fold=integer(),mse_pred_mean=double(),mse_pred_lm=double())

# Create 10 equally size folds
folds <- cut(seq(1,nrow(scooter_trips.df)),breaks=10,labels=FALSE)

# Perform 10 fold cross validation
for(i in 1:10){

  # Segement your data by fold using the which() function
  test_indexes <- which(folds==i,arr.ind=TRUE)
  test_set <- scooter_trips.df[test_indexes, ]
  training_set <- scooter_trips.df[-test_indexes, ]

  # Make the model
```

```
  y_train = training_set$Trip.Duration.Logged
  x_train = training_set$Trip.Distance.Logged
  fit_mean = mean(y_train)
  fit_lm <- lm(y ~ x, data.frame(y=y_train, x=x_train))

  # Test results
  y_test <- test_set$Trip.Duration.Logged
  x_test <- test_set$Trip.Distance.Logged

  # Make a vector that is populated with the fit_mean
  pred_mean <- rep(fit_mean, length(y_test))

  # Get a vector of predictions based on test data
  pred_lm <- predict(fit_lm, data.frame(x=x_test))

  # get the MSE
  MSE_for_fold.df[i,] = c(i,MSE(pred_mean, y_test),MSE(pred_lm, y_test))
}
```

## Calculate MSE for the linear model

Calculate for k-fold cross-validation (with k = 10), look at difference between average MSE for the model and for the average across the dataset.

```
# Print the result
MSE_for_fold.df
```

```
##    fold mse_pred_mean mse_pred_lm
## 1     1     0.9297038   0.3663421
## 2     2     0.8828283   0.3889273
## 3     3     0.9574209   0.3949431
## 4     4     0.9273046   0.3797911
## 5     5     0.9089837   0.3969958
## 6     6     0.9491380   0.4010092
## 7     7     0.9093470   0.3667877
## 8     8     0.9328663   0.3940092
## 9     9     0.8852049   0.3626099
## 10   10     0.9407331   0.3827002
```

```
# Print the average MSE across all folded data
mean(MSE_for_fold.df$mse_pred_mean)
```

```
## [1] 0.9223531
```

```
mean(MSE_for_fold.df$mse_pred_lm)
```

```
## [1] 0.3834116
```

## Create the polynomial model and measure using k-fold cross-validation

Fit a polynomial regression model via k-fold cross-validation to predict the trip duration based on trip distance and the square of trip distance and measure the test MSE.

```
# Define MSE function
MSE <- function(m, o) {
    mean((m - o)^2)
}

# Randomly shuffle the data
scooter_trips.df <- scooter_trips.df[sample(nrow(scooter_trips.df)),]

MSE_for_fold.df <- data.frame(fold=integer(),mse_pred_mean=double(),mse_pred_lm2=double())

# Create 10 equally size folds
folds <- cut(seq(1,nrow(scooter_trips.df)),breaks=10,labels=FALSE)

# Perform 10 fold cross validation
for(i in 1:10){

  # Segement your data by fold using the which() function
  test_indexes <- which(folds==i,arr.ind=TRUE)
  test_set <- scooter_trips.df[test_indexes, ]
  training_set <- scooter_trips.df[-test_indexes, ]

  # Make the model
  y_train = training_set$Trip.Duration.Logged
  x_train = training_set$Trip.Distance.Logged
  fit_mean = mean(y_train)
  fit_lm2 <- lm(y ~ x + I(x^2), data.frame(y=y_train, x=x_train))

  # Test results
  y_test <- test_set$Trip.Duration.Logged
  x_test <- test_set$Trip.Distance.Logged

  # Make a vector that is populated with the fit_mean
  pred_mean <- rep(fit_mean, length(y_test))

  # Get a vector of predictions based on test data
  pred_lm2 <- predict(fit_lm, data.frame(x=x_test))

  # get the MSE
  MSE_for_fold.df[i,] = c(i,MSE(pred_mean, y_test),MSE(pred_lm2, y_test))
}
```

## Calculate MSE for the polynomial model

Calculate for k-fold cross-validation (with k = 10), look at difference between average MSE for the model and for the average across the dataset.

```
# Print the result
MSE_for_fold.df
```

```
##    fold mse_pred_mean mse_pred_lm2
## 1     1     0.9279840    0.3808498
## 2     2     0.9315983    0.3797486
## 3     3     0.9288501    0.3866514
## 4     4     0.9341868    0.4230339
## 5     5     0.8937310    0.3504983
## 6     6     0.8993206    0.3612690
## 7     7     0.9326331    0.3899970
## 8     8     0.9310895    0.3877520
## 9     9     0.9233137    0.3947605
## 10   10     0.9206670    0.3780015
```

```
# Print the average MSE across all folded data
mean(MSE_for_fold.df$mse_pred_mean)
```

```
## [1] 0.9223374
```

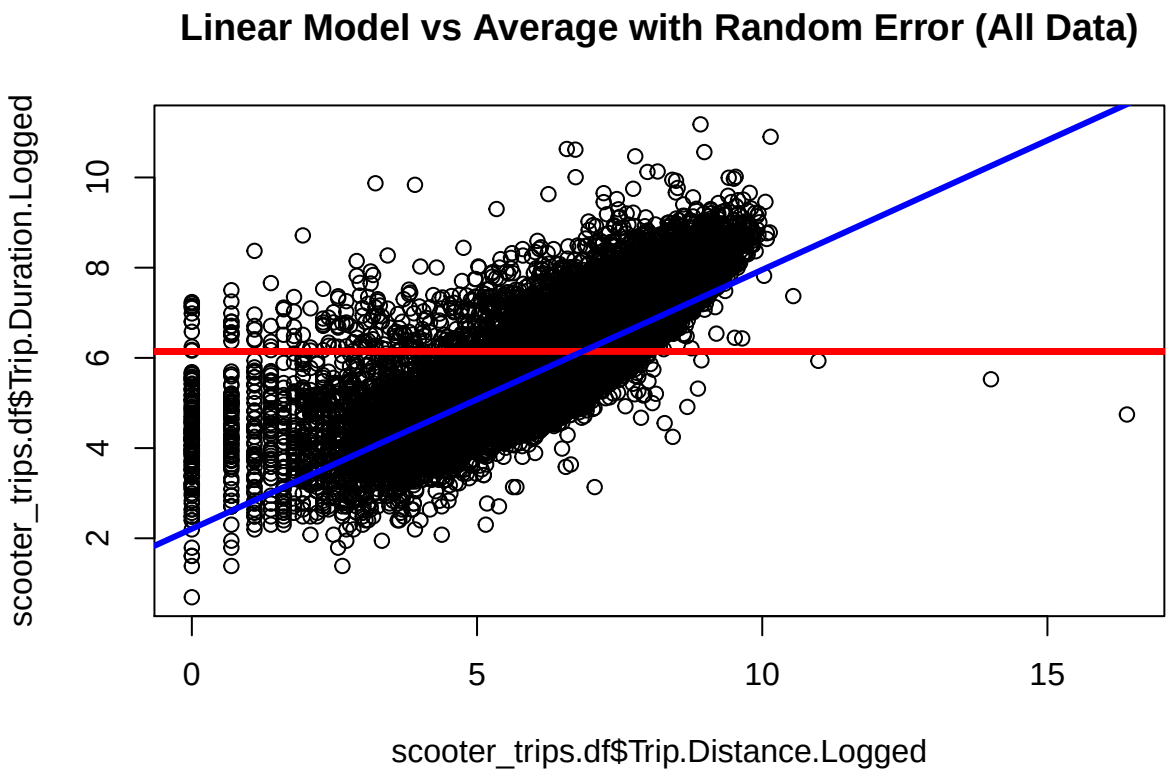
```
mean(MSE_for_fold.df$mse_pred_lm2)
```

```
## [1] 0.3832562
```

## Visualise the Linear and Polynomial models (across all data)

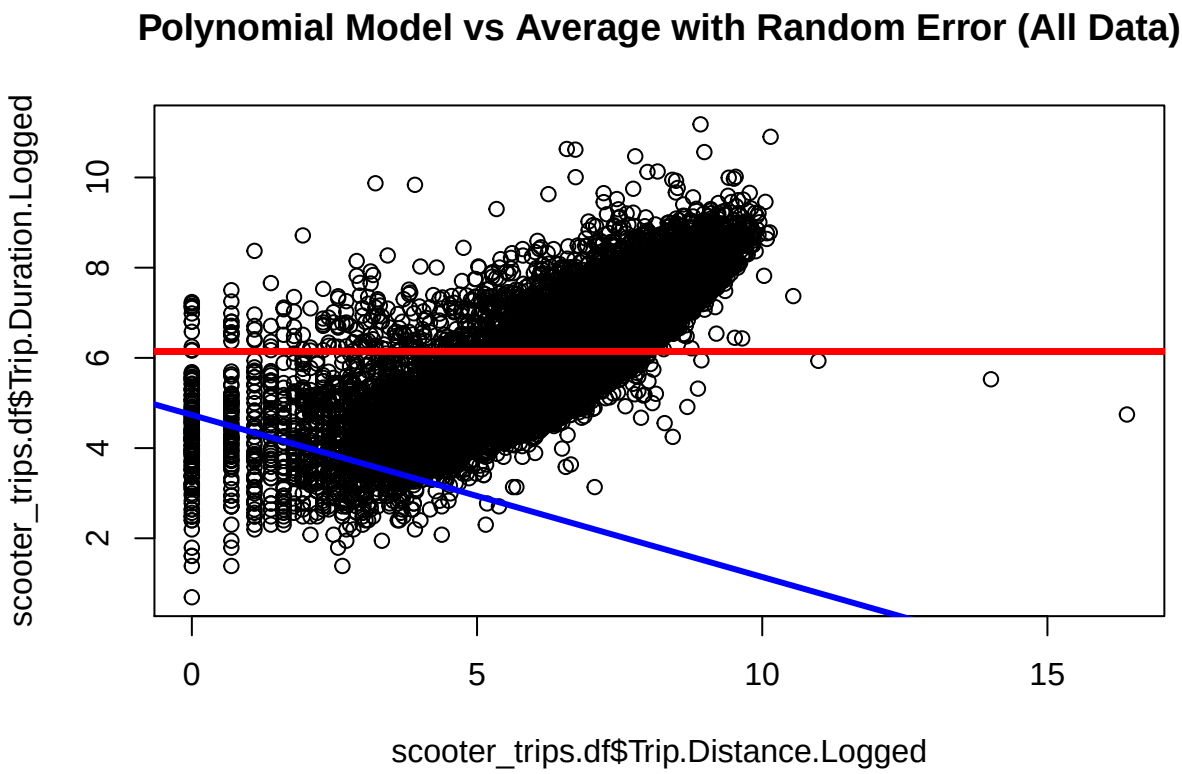
Plot Trip.Duration.Logged vs Trip.Distance.Logged and overlay the average with random error model and our derived linear model. This is to help us visualise the data and the derived model as it applies to the test dataset.

The charting approach that was used unfortunately was unable to successfully chat the line for the polynomial model as is indicated by the R message "only using the first two of 3 regression coefficients".




Plot scooter_trips.df$Trip.Duration vs scooter_trips.df$Trip.Distance and overlay the average with random error model and our derived polynomial linear model. This is to help us visualise the data and the derived model as it applies to the test dataset.

```
## Warning in abline(fit_lm2, col = "blue", lwd = 3): only using the first two
## of 3 regression coefficients
```




### Summary and analysis

Our model has reduced the variability that we see by simply taking the average model with random error significantly - on average it decreases the variablity by about 60 percent. This is a good model as it explains more of the variability in the data than the average model with random error. The model would be useful for prediction. Both the linear and the polynomial models exhibit the same level of accuracy - i.e. no advantage is gained by adding a polynomial term.