Linear Regression

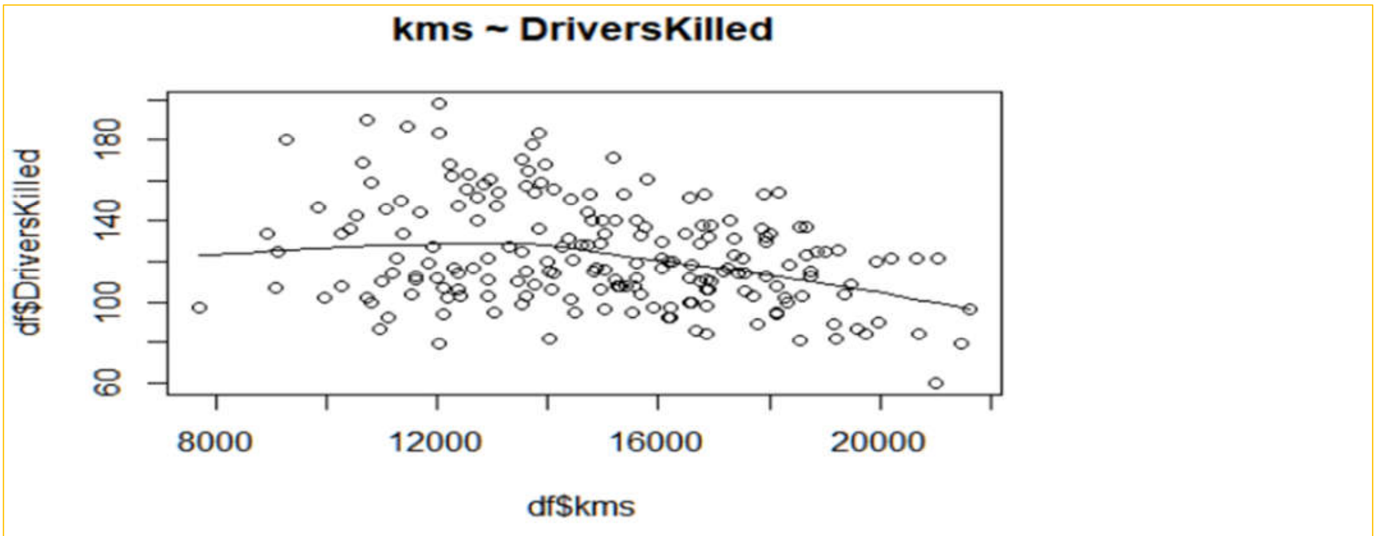
The Seatbelts data set contains the monthly totals of car drivers in Great Britain killed or seriously injured Jan 1969 to Dec 1984.Here is the general structure of Seatbelts data set.

```
> data(Seatbelts)
> #View structure of data
> dat <-Seatbelts
> print(dat)
         DriversKilled drivers front rear   kms PetrolPrice VanKilled law
Jan 1969           107    1687   867  269  9059      0.1030        12   0
Feb 1969            97    1508   825  265  7685      0.1024         6   0
Mar 1969           102    1507   806  319  9963      0.1021        12   0
Apr 1969            87    1385   814  407 10955      0.1009         8   0
May 1969           119    1632   991  454 11823      0.1010        10   0
Jun 1969           106    1511   945  427 12391      0.1006        13   0
Jul 1969           110    1559  1004  522 13460      0.1038        11   0
Aug 1969           106    1630  1091  536 14055      0.1041         6   0
Sep 1969           107    1579   958  405 12106      0.1038        10   0
Oct 1969           134    1653   850  437 11372      0.1030        16   0
Nov 1969           147    2152  1109  434  9834      0.1027        13   0
Dec 1969           180    2148  1113  437  9267      0.1020        14   0
Jan 1970           125    1752   925  316  9130      0.1013        14   0
```

Building a simple regression model that we can use to predict DriversKilled by establishing a linear relationship with kms (Kilometers travelled). First we need to understand these variables graphically and visualize the following behavior:

Scatter plot: Visualize the linear relationship between the predictor and response

```
> #Scatter Plot analysis
> scatter.smooth(x=df$kms, y=df$DriversKilled, main="kms ~ DriversKilled" )
> |
```

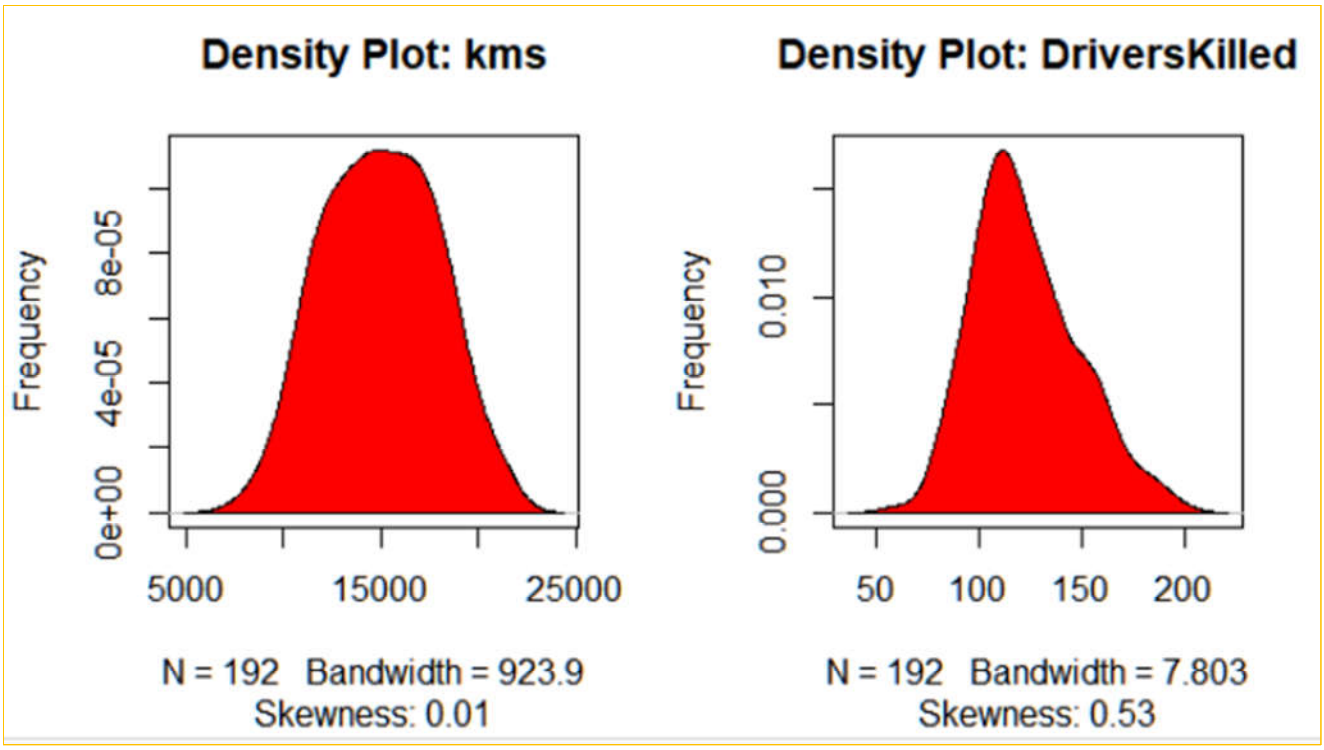


Box plot: To spot any outlier observations in the variable.

```
> #BoxPlot - Check for outliers
> par(mfrow=c(1, 2))  # divide graph area in 2 columns
> # box plot for 'DriversKilled'
> boxplot(df$DriversKilled, main="DriversKilled", sub=paste("Outlier rows: ",
+                                                      boxplot.stats(df$DriversKilled)$out))
> # box plot for 'kms'
> boxplot(df$kms, main="kms", sub=paste("Outlier rows: ", boxplot.stats(df$kms)$out))
> 
```

Density plot: To see the distribution of the predictor variable. Ideally, a close to normal distribution (a bell shaped curve), without being skewed to the left or right is preferred. Let us see how to make each one of them.

```
> #Density plot - Correlation
> # divide graph area in 2 columns
> par(mfrow=c(1, 2))
> # density plot for 'kms'
> plot(density(df$kms), main="Density Plot: kms",
+      ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(df$kms), 2)))
> polygon(density(df$kms), col="red")
> # density plot for 'DriversKilled'
> plot(density(df$DriversKilled), main="Density Plot: DriversKilled",
+   ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(df$DriversKilled), 2)))
> polygon(density(df$DriversKilled), col="red")
> 
```




```
> ## calculate correlation between DriversKilled and kms
> cor(df$DriversKilled, df$kms)
[1] -0.3211016
```

Build Linear Model

```
> ## build linear regression model on full data
> linearMod <- lm(DriversKilled ~ kms, data=Seatbelts)
> print(linearMod)

Call:
lm(formula = DriversKilled ~ kms, data = Seatbelts)

Coefficients:
(Intercept)          kms
 164.391144    -0.002774
```

Now that we have built the linear model, we also have established the relationship between the predictor and response in the form of a mathematical formula for DriversKilled as a function for kms. For the above output, you can notice the 'Coefficients' part having two components: *Intercept*: 164.39, *kms*: -0.002774 these are also called the beta coefficients. In other words,

$$DriversKilled = Intercept + (\beta * kms)$$

$$DriversKilled = 164.391 + (-0.00277 * kms)$$

For example the number of kilometers travelled is 16,000 so according to the formula DriversKilled will be 120.

Linear Regression Diagnostics

```
> #Linear Regression Diagnostics
> summary(linearMod)  # model summary

Call:
lm(formula = DriversKilled ~ kms, data = Seatbelts)

Residuals:
    Min      1Q  Median      3Q     Max
-52.028 -19.021  -1.974  16.719  66.964

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.644e+02  9.067e+00  18.130  < 2e-16 ***
kms         -2.774e-03  5.935e-04  -4.674  5.6e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 24.1 on 190 degrees of freedom
Multiple R-squared:  0.1031,    Adjusted R-squared:  0.09839
F-statistic: 21.84 on 1 and 190 DF,  p-value: 5.596e-06
```

R-Squared and Adj R-Squared

```
> summary(linearMod)  # model summary

Call:
lm(formula = DriversKilled ~ kms, data = Seatbelts)

Residuals:
    Min      1Q  Median      3Q     Max 
-52.028 -19.021  -1.974  16.719  66.964 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.644e+02  9.067e+00  18.130  < 2e-16 ***
kms         -2.774e-03  5.935e-04  -4.674  5.6e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 24.1 on 190 degrees of freedom
Multiple R-squared:  0.1031,	Adjusted R-squared:  0.09839 
F-statistic: 21.84 on 1 and 190 DF,  p-value: 5.596e-06
```

AIC and BIC

For model comparison, the model with the lowest AIC and BIC score is preferred.

```
> #Model comparision
> AIC(linearMod)
[1] 1770.816
> BIC(linearMod)
[1] 1780.589
>
```

Predicting Linear Models

So far we have seen how to build a linear regression model using the whole dataset. If we build it that way, there is no way to tell how the model will perform with new data. So the preferred practice is to split your dataset into a 80:20 sample (training:test), then, build the model on the 80% sample and then use the model thus built to predict the dependent variable on test data.

Step 1: Create the training (development) and test (validation) data samples from original data.

```
> #Making predictions
> # Create Training and Test data -
> set.seed(100)  # setting seed to reproduce results of random sampling
> trainingRowIndex <- sample(1:nrow(Seatbelts), 0.8*nrow(Seatbelts))  # row indices for training data
> trainingData <- Seatbelts[trainingRowIndex, ]  # model training data
> testData  <- Seatbelts[-trainingRowIndex, ]   # test data
>
```

Step 2: Develop the model on the training data and use it to predict the DriversKilled on test data

```
> #Making predictions
> # Step 1 Create Training and Test data -
> set.seed(100)  # setting seed to reproduce results of random sampling
> trainingRowIndex <- sample(1:nrow(Seatbelts), 0.8*nrow(Seatbelts))  # row indices for training data
> trainingData <- as.data.frame.matrix(Seatbelts[trainingRowIndex, ])  # model training data
> testData  <- as.data.frame.matrix(Seatbelts[-trainingRowIndex, ])   # test data
> #Step 2 Build the model on training data
> lmMod <- lm(DriversKilled ~ kms, data=trainingData)  # build the model
> distPred <- predict(lmMod, testData)  # predict distance
```

Step 3: Review diagnostic measures.

```
> #Step 3: Review diagnostic measures.
> summary (lmMod)

Call:
lm(formula = DriversKilled ~ kms, data = trainingData)

Residuals:
    Min      1Q  Median      3Q     Max 
-52.110 -19.364  -2.137  16.959  66.881 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.654e+02  1.002e+01  16.504  < 2e-16 ***
kms         -2.853e-03  6.583e-04  -4.334 2.66e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25.14 on 151 degrees of freedom
Multiple R-squared:  0.1106,	Adjusted R-squared:  0.1047 
F-statistic: 18.78 on 1 and 151 DF,  p-value: 2.664e-05
```

From the model summary, the model p value and predictor's p value are less than the significance level, so we know we have a statistically significant model. Also, the R-Sq and Adj R-Sq are comparative to the original model built on full data.

Step 4: Calculate prediction accuracy and error rates

```
> #Step 4 Calculate prediction accuracy and error rates
> # make actuals_predicteds dataframe.
> actuals_preds <- data.frame(cbind(actuals=testData$DriversKilled, predicteds=distPred))
> correlation_accuracy <- cor(actuals_preds)
> head(actuals_preds)
  actuals predicteds
1     106   130.0743
2     103   128.5879
3     117   130.3881
4     157   126.6622
5     136   125.9860
6     140   129.1100
```

Now let's calculate the Min Max accuracy and MAPE:

```
> #Min Max accuracy and MAPE:
> min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))
> print(min_max_accuracy) # min_max accuracy
[1] 0.8782031
> mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actuals))/actuals_preds$actuals)
> print(mape) # mean absolute percentage deviation
[1] 0.136092
>
```

R CODE

```
#Install neccessary packages
install.packages("dplyr")
install.packages("e1071")
install.packages("DAAG")

#Import neccessary packages
library(dplyr)
library(e1071)
library(DAAG)

data(Seatbelts)
dat <-Seatbelts
#View structure of data
print(dat)

#Limit variables to two columns of interest
df <-data.frame(Seatbelts[,c("DriversKilled","kms")])

#Scatter Plot analysis
scatter.smooth(x=df$kms, y=df$DriversKilled, main="kms ~ DriversKilled" )

#BoxPlot - Check for outliers
par(mfrow=c(1, 2))  # divide graph area in 2 columns

#box plot for 'DriversKilled'
boxplot(df$DriversKilled, main="DriversKilled", sub=paste("Outlier rows: ",

boxplot.stats(df$DriversKilled)$out))

# box plot for 'kms'
boxplot(df$kms, main="kms", sub=paste("Outlier rows: ",
boxplot.stats(df$kms)$out))

#Density plot - Correlation
#divide graph area in 2 columns
par(mfrow=c(1, 2))
```

```
#density plot for 'kms'
plot(density(df$kms), main="Density Plot: kms",
     ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(df$kms), 2)))
polygon(density(df$kms), col="red")

#density plot for 'DriversKilled'
plot(density(df$DriversKilled), main="Density Plot: DriversKilled",
  ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(df$DriversKilled), 2)))
polygon(density(df$DriversKilled), col="red")

#calculate correlation between DriversKilled and kms
cor(df$kms, df$DriversKilled)

#build linear regression model on full data
linearMod <- lm(DriversKilled ~ kms, data=Seatbelts)
print(linearMod)

#Linear Regression Diagnostics
summary(linearMod)  # model summary

#The p Value: Checking for statistical significance
modelSummary <- summary(linearMod)  # capture model summary as an object
modelCoeffs <- modelSummary$coefficients  # model coefficients
print(modelCoeffs)
beta.estimate <- modelCoeffs["kms", "Estimate"]  # get beta estimate for speed
std.error <- modelCoeffs["kms", "Std. Error"]  # get std.error for speed
t_value <- beta.estimate/std.error  # calc t statistic
p_value <- 2*pt(-abs(t_value), df=nrow(Seatbelts)-ncol(Seatbelts)) # calc p Value
f_statistic <- linearMod$fstatistic[1]  # fstatistic
f <- summary(linearMod)$fstatistic  # parameters for model p-value calc
model_p <- pf(f[1], f[2], f[3], lower=FALSE)

print(t_value)
print(p_value)
print(f_statistic)
print(model_p)

#AIC and BIC
AIC(linearMod)
BIC(linearMod)
```

```
#Making predictions
#Step 1 Create Training and Test data -
set.seed(100)  # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(Seatbelts), 0.8*nrow(Seatbelts))  # row indices
for training data
trainingData <- as.data.frame.matrix(Seatbelts[trainingRowIndex, ])  # model
training data
testData  <- as.data.frame.matrix(Seatbelts[-trainingRowIndex, ])   # test data

#Step 2 Build the model on training data
lmMod <- lm(DriversKilled ~ kms, data=trainingData)  # build the model
distPred <- predict(lmMod, testData)  # predict distance

#Step 3: Review diagnostic measures.
summary (lmMod)

#Step 4 Calculate prediction accuracy and error rates
#make actuals_predicteds dataframe.
actuals_preds <- data.frame(cbind(actuals=testData$DriversKilled,
predicteds=distPred))
correlation_accuracy <- cor(actuals_preds)
head(actuals_preds)

#Min Max accuracy and MAPE:
min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1,
max))
print(min_max_accuracy) # min_max accuracy
mape <- mean(abs((actuals_preds$predicteds -
actuals_preds$actuals))/actuals_preds$actuals)
print(mape) # mean absolute percentage deviation
```