NAME: Hammad Ansari **ROLL NO: 2018450002**

Experiment No. 4

Title: To Implement Linear Regression in R

Concept :

Linear regression answers a simple question: Can you measure an exact relationship between one target variable and a set of predictors?

The simplest of probabilistic models is the straight-line model:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

where

- y = Dependent variable
- x = Independent variable
- = random error component
- = intercept
- = Coefficient of x

Example Problem

For this analysis, we will use the *cars* dataset that comes with R by default. cars is a standard built-in dataset. Here we need to find out how distance and speed id related to each other. Before we begin building the regression model, it is a good practice to analyze and understand the variables. The graphical analysis and correlation study below will help with this.

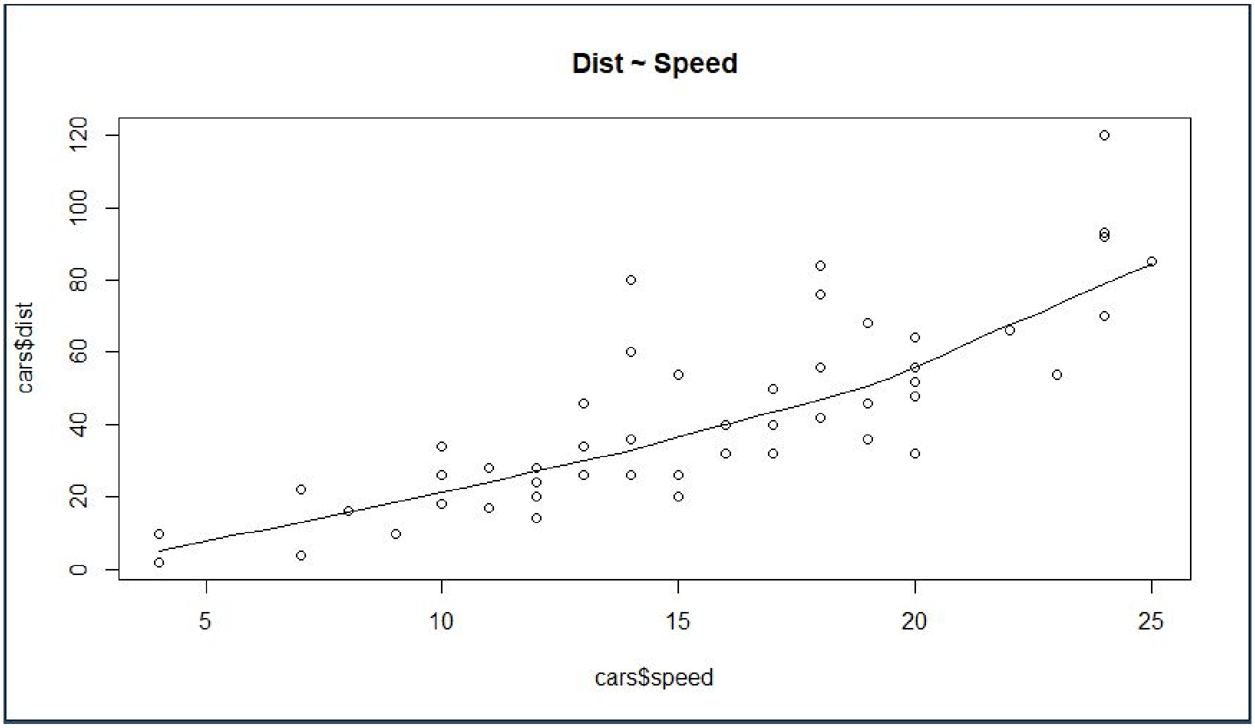
The following plots are drawn to visualize the following behavior:

1. **Scatter plot**: Visualize the linear relationship between the predictor and response

   Code:

   ```
   scatter.smooth(x=cars$speed, y=cars$dist, main="Dist ~ Speed")  # Scatter plot
   ```

Screenshot:



2. **Box plot**: To spot any outlier observations in the variable. Having outliers in your predictor can drastically affect the predictions as they can easily affect the direction/slope of the line of best fit.

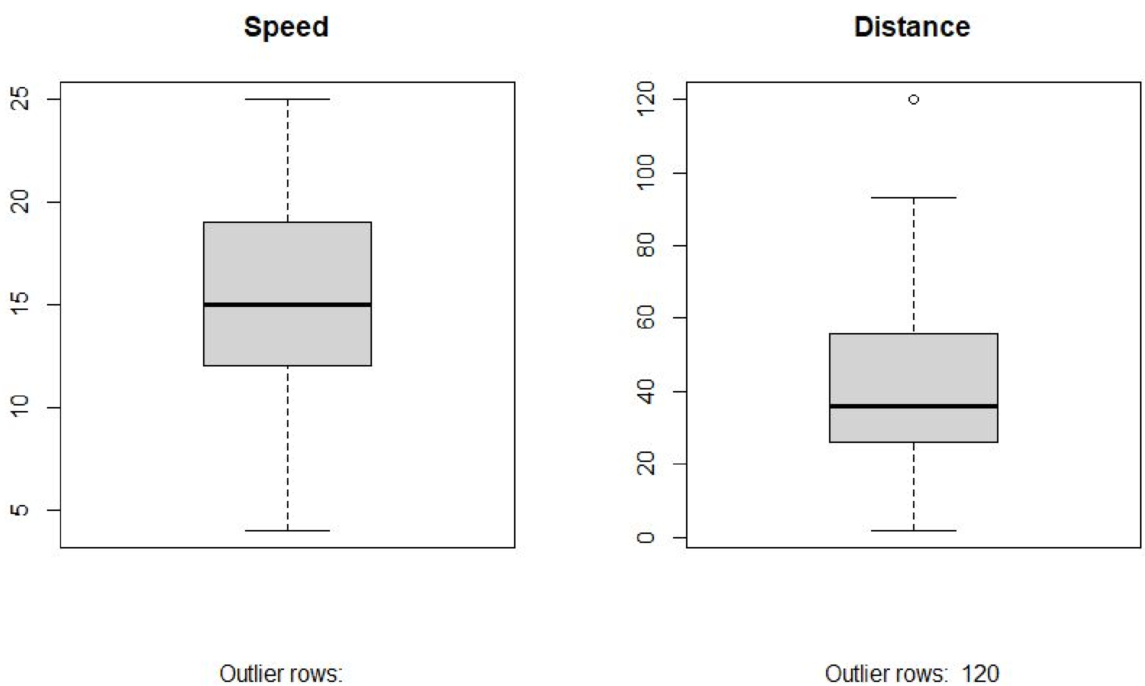
Code:

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

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

boxplot(cars$dist, main="Distance", sub=paste("Outlier rows: ", boxplot.stats(cars$dist)$out))  # box plot for 'distance'
```

Screenshot:



3. **Density Plot:**

Code:

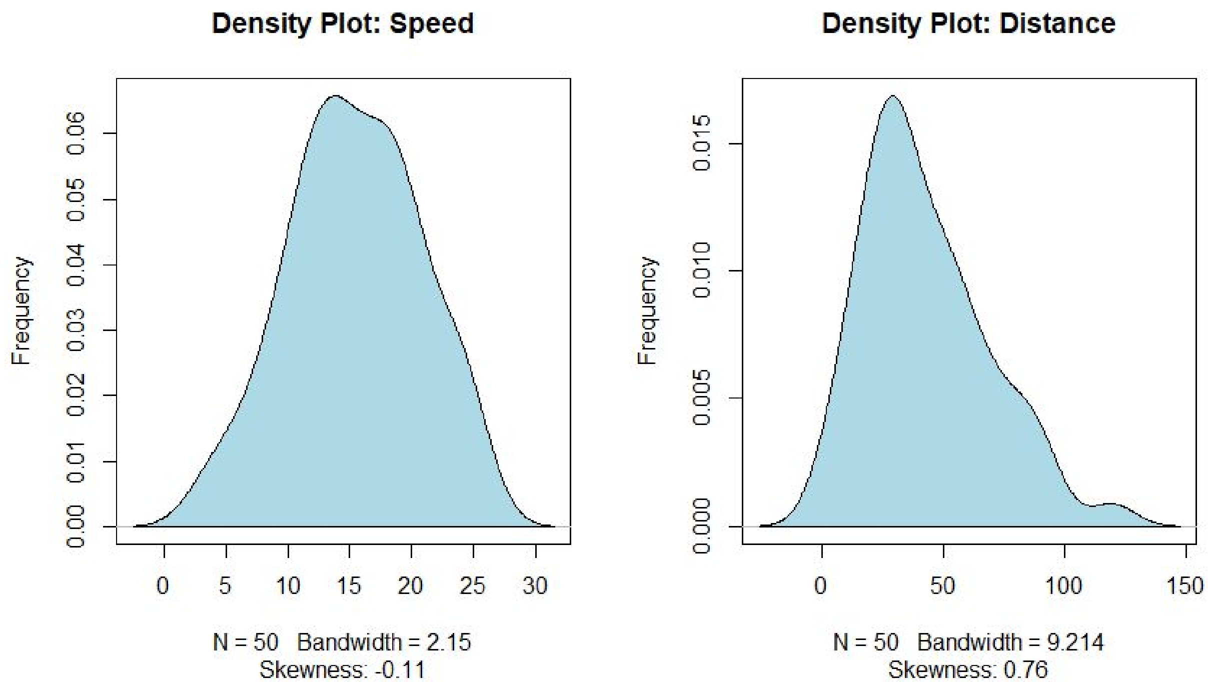
```
plot(density(cars$speed), main="Density Plot: Speed", ylab="Frequency",
sub=paste("Skewness:", round(e1071::skewness(cars$speed), 2)))  # density
plot for 'speed'


polygon(density(cars$speed), col="light blue")


plot(density(cars$dist), main="Density Plot: Distance", ylab="Frequency",
sub=paste("Skewness:", round(e1071::skewness(cars$dist), 2)))  # density plot
for 'dist'


polygon(density(cars$dist), col="light blue")
```

Screenshot:



4. **Find Correlation:** Correlation is a statistical measure that suggests the level of linear dependence between two variables, that occur in pairs – just like what we have here in speed and dist. Correlation can take values between -1 to +1. If we observe for every instance where speed increases, the distance also increases along with it, then there is a high positive correlation between them and therefore the correlation between them will be closer to 1. The opposite is true for an inverse relationship, in which case, the correlation between the variables will be close to -1. A value closer to 0 suggests a weak relationship between the variables.

   Code:

   ```
   cor(cars$speed, cars$dist)  # calculate correlation between speed and distance
   ```

   Screenshot:

```
> cor(cars$speed, cars$dist)  # calculate correlation between speed a
nd distance
[1] 0.8068949
>
```

5. **Build a Linear Model**: The function used for building linear models is lm()

   Code:

   linearModel <- lm(dist ~ speed, data=cars) # build linear regression model on full data

   print(linearModel)

   Screenshot:

```
Call:
lm(formula = dist ~ speed, data = cars)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

6. **Do Linear Regression Diagnostics**

   Code:

   summary(linearModel) # model summary

Screenshot:

```
> summary(linearModel)  # model summary

Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max 
-29.069  -9.525  -2.272   9.215  43.201 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -17.5791     6.7584  -2.601   0.0123 *  
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438 
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

7. **Predicting Linear Models:**

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

Code:

Create Training and Test data -

set.seed(100) # setting seed to reproduce results of random sampling

trainingRowIndex <- sample(1:nrow(cars), 0.8*nrow(cars)) # row indices for training data

trainingData <- cars[trainingRowIndex,] # model training data

testData <- cars[-trainingRowIndex,]   # test data

Screenshot:

```
> # Create Training and Test data -
> set.seed(100)  # setting seed to reproduce results of random sampli
ng

> trainingRowIndex <- sample(1:nrow(cars), 0.8*nrow(cars))  # row ind
ices for training data

> trainingData <- cars[trainingRowIndex, ]  # model training data

> testData  <- cars[-trainingRowIndex, ]   # test data
> View(testData)
> View(trainingData)
```

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

Code:

Build the model on training data

lmMod <- lm(dist ~ speed, data=trainingData) # build the model

distPred <- predict(lmMod, testData) # predict distance

Screenshot:

```
> # Build the model on training data
> lmMod <- lm(dist ~ speed, data=trainingData)  # build the model

> distPred <- predict(lmMod, testData)  # predict distance
> 
```

Step 3: Review diagnostic measures

Code:

summary (lmMod) # model summary

AIC (lmMod) # Calculate akaike information criterion

Screenshot:

```
> summary (lmMod)  # model summary

Call:
lm(formula = dist ~ speed, data = trainingData)

Residuals:
    Min      1Q  Median      3Q     Max 
-24.726 -11.242  -2.564  10.436  40.565 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -20.1796     7.8254  -2.579   0.0139 *  
speed         4.2582     0.4947   8.608 1.85e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.49 on 38 degrees of freedom
Multiple R-squared:  0.661,	Adjusted R-squared:  0.6521 
F-statistic: 74.11 on 1 and 38 DF,  p-value: 1.848e-10


> AIC (lmMod)  # Calculate akaike information criterion
[1] 336.6933
```

Step 4: Calculate prediction accuracy and error rates

Code:

actuals_preds <- data.frame(cbind(actuals=testData$dist, predicteds=distPred))
# make actuals_predicteds dataframe.

correlation_accuracy <- cor(actuals_preds) # 82.7%

head(actuals_preds)

Min-Max Accuracy Calculation

min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))

=> 38.00%, min_max accuracy

```
# MAPE Calculation

mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actuals))/actuals_preds$actuals)

# => 69.95%, mean absolute percentage deviation


DMwR::regr.eval(actuals_preds$actuals, actuals_preds$predicteds)
```

Screenshot:

```
> correlation_accuracy <- cor(actuals_preds)  # 82.7%

> head(actuals_preds)
   actuals predicteds
3        4   9.627845
5       16  13.886057
17      34  35.177120
24      20  43.693545
28      40  47.951757
32      42  56.468182
>
```

```
> # Min-Max Accuracy Calculation
> min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actua
ls_preds, 1, max))

> # => 38.00%, min_max accuracy
>
> # MAPE Calculation
> mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actual
s))/actuals_preds$actuals)  .... [TRUNCATED]

> # => 69.95%, mean absolute percentage deviation
>
> DMwR::regr.eval(actuals_preds$actuals, actuals_preds$predicteds)
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
        mae         mse        rmse        mape
 12.5069370 267.0002421  16.3401420   0.4959096
>
```

8. **K- Fold Cross-validation**

Build Model on a different subset of training data and predicted the remaining data
Split your data into ‘k’ mutually exclusive random sample portions. Keeping each portion as test data, we build the model on the remaining (k-1 portion) data and calculate the mean squared error of the predictions. Then finally, the average of these mean squared errors (for ‘k’ portions) is computed. We can use this metric to compare different linear models.

Code:
library(DAAG)

cvResults <- suppressWarnings(CVlm(cars, form.lm=dist ~ speed, m=5, dots=FALSE, seed=29, legend.pos="topleft", printit=FALSE, main="Small symbols are predicted values while bigger ones are actuals.")); # performs the CV
attr(cvResults, 'ms')

Screenshot:

```
> cvResults <- suppressWarnings(CVlm(cars, form.lm=dist ~ speed, m=5, dots=FAL
SE, seed=29, legend.pos="topleft",  printit=FALSE, main="Small symbols a ..."
 ... [TRUNCATED]

> attr(cvResults, 'ms')
[1] 254.2661
> library(DAAG)
> cvResults <- suppressWarnings(CVlm(cars, form.lm=dist ~ speed, m=5, dots=FAL
SE, seed=29, legend.pos="topleft",  printit=FALSE, main="Small symbols are pre
dicted values while bigger ones are actuals."));  # performs the CV
> attr(cvResults, 'ms')
[1] 254.2661
```