project

2024-05-04

```
 #install.packages("olsrr")
 #install.packages("car")
 #install.packages("broom")
  #install.packages("tidyverse")
  #install.packages("caret")

library(tidyverse)          # Pipe operator (%>%) and other commands
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'tibble' was built under R version 4.3.3
```

```
## Warning: package 'tidyr' was built under R version 4.3.3
```

```
## Warning: package 'readr' was built under R version 4.3.3
```

```
## Warning: package 'purrr' was built under R version 4.3.3
```

```
## Warning: package 'dplyr' was built under R version 4.3.3
```

```
## Warning: package 'stringr' was built under R version 4.3.3
```

```
## Warning: package 'forcats' was built under R version 4.3.3
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```r
library(caret)              # Random split of data/cross validation
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(olsrr)              # Heteroscedasticity Testing (ols_test_score)
```

```
## Warning: package 'olsrr' was built under R version 4.3.3
```

```
##
## Attaching package: 'olsrr'
##
## The following object is masked from 'package:datasets':
##
##     rivers
```

```r
library(car)                # Muticolinearity detection (vif)
```

```
## Warning: package 'car' was built under R version 4.3.3
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.3.3
```

```
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(broom)              # Diagnostic Metric Table (augment)
```

```
## Warning: package 'broom' was built under R version 4.3.3
```

Exploring Data set

```
data = read.csv("advertising.csv" , header = T)
```

```
head(data)
```

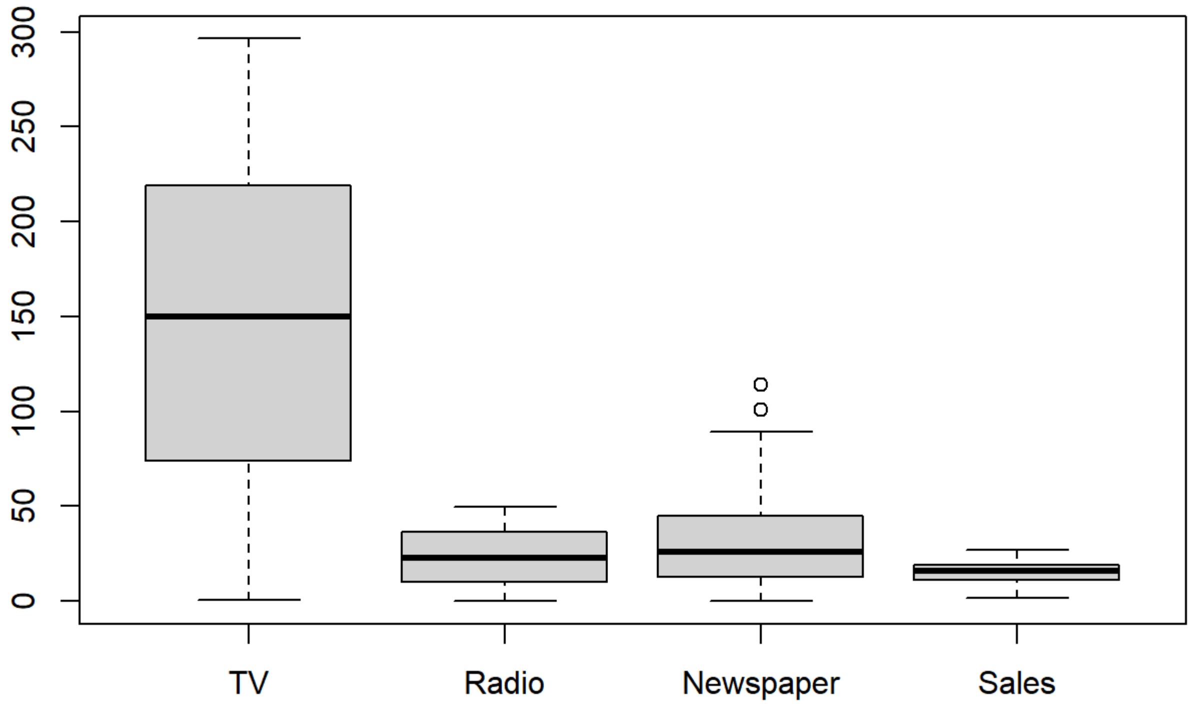
```
##      TV Radio Newspaper Sales
## 1 230.1  37.8      69.2  22.1
## 2  44.5  39.3      45.1  10.4
## 3  17.2  45.9      69.3  12.0
## 4 151.5  41.3      58.5  16.5
## 5 180.8  10.8      58.4  17.9
## 6   8.7  48.9      75.0   7.2
```

```
# Getting Structure of whole data set
str(data)
```

```
## 'data.frame':    200 obs. of  4 variables:
##  $ TV       : num  230.1 44.5 17.2 151.5 180.8 ...
##  $ Radio    : num  37.8 39.3 45.9 41.3 10.8 48.9 32.8 19.6 2.1 2.6 ...
##  $ Newspaper: num  69.2 45.1 69.3 58.5 58.4 75 23.5 11.6 1 21.2 ...
##  $ Sales    : num  22.1 10.4 12 16.5 17.9 7.2 11.8 13.2 4.8 15.6 ...
```

1- There are 200 rows and 4 variables. 2- Variables are : TV,Radio,Newspaper,Sales 3- All are numeric variables.

```
# Checking Outliers
boxplot(data)
```

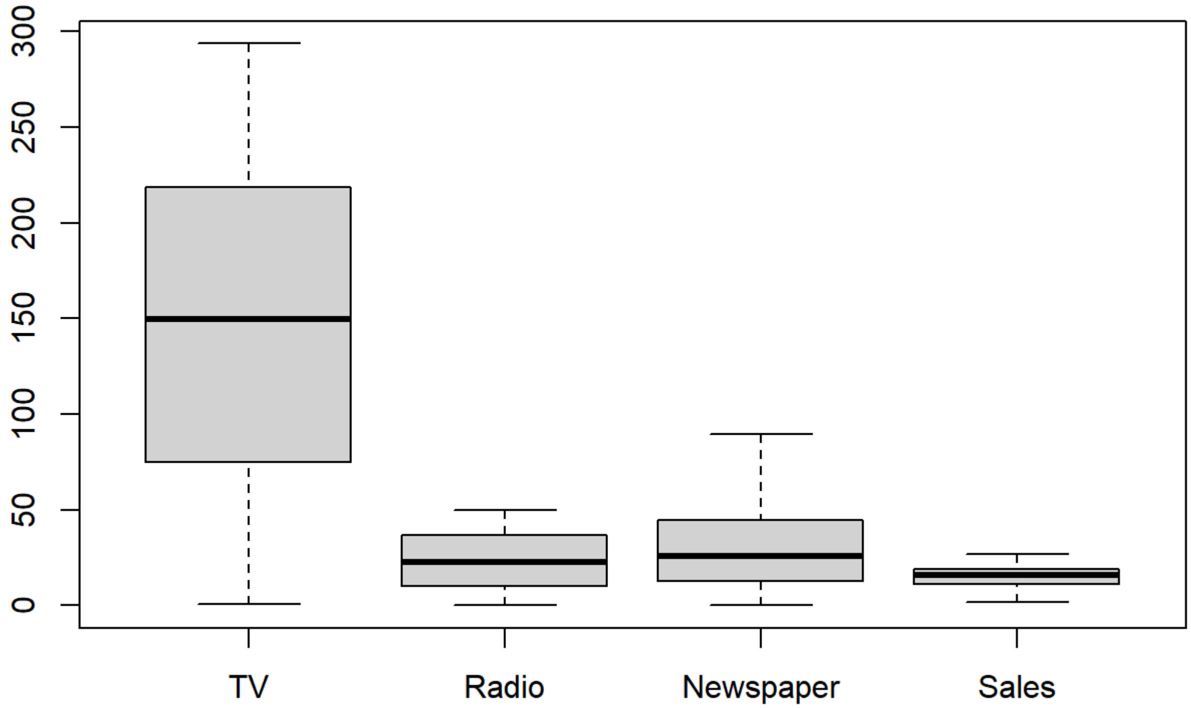


The

above plot shows that two outliers are present in the variable “Newspaper”. Just remove these outliers by the following command

```
data <- data[-which(data$Newspaper %in% boxplot.stats(data$Newspaper)$out), ]
```

```
# Again Checking Outliers
boxplot(data)
```

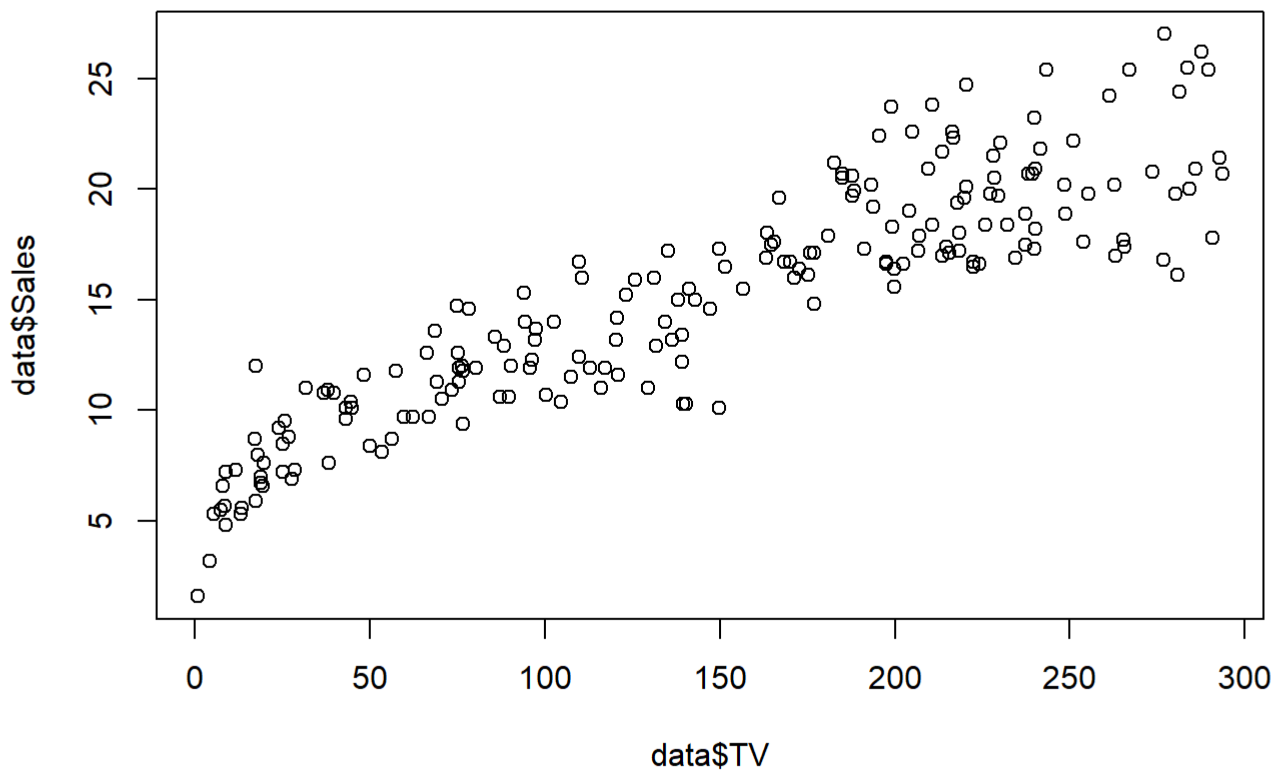



```
# Checking Missing Values
table(is.na(data))
```

```
##
## FALSE
##   792
```

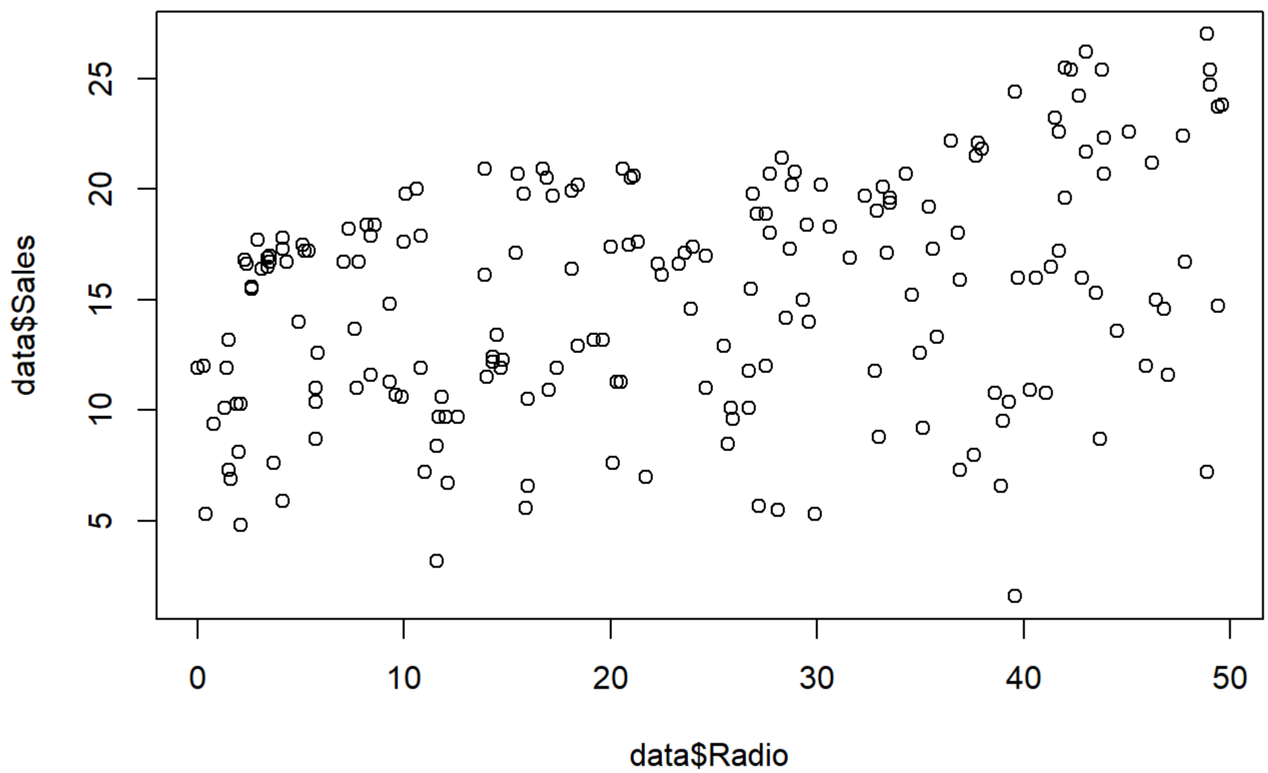
The above output shows that there is no missing value in the given data set.

```
# Scatter Plot between TV and Sales
plot(data$TV , data$Sales)
```



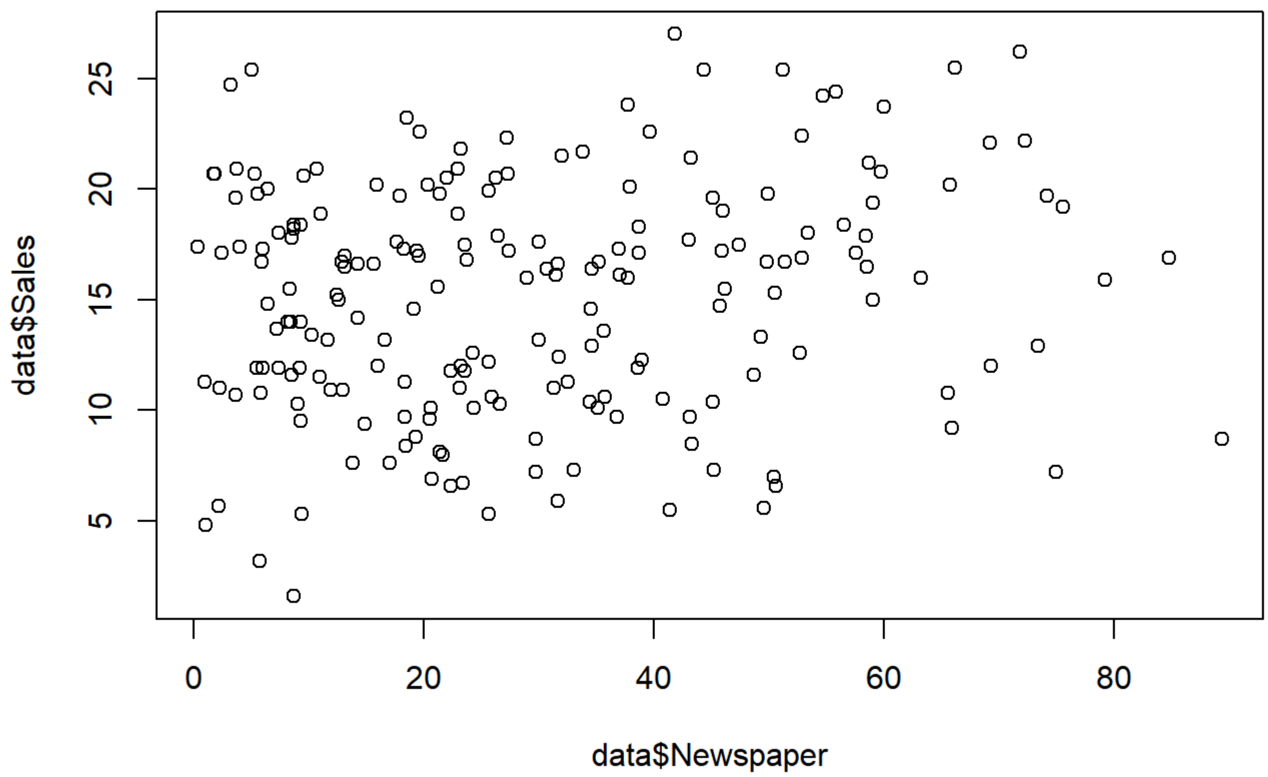
Notice, there is a small curvilinear relationship between TV and Sales.

```
# Scatter Plot between Radio and Sales
plot(data$Radio , data$Sales)
```



Notice, there is a curvilinear relationship between Radio and Sales.

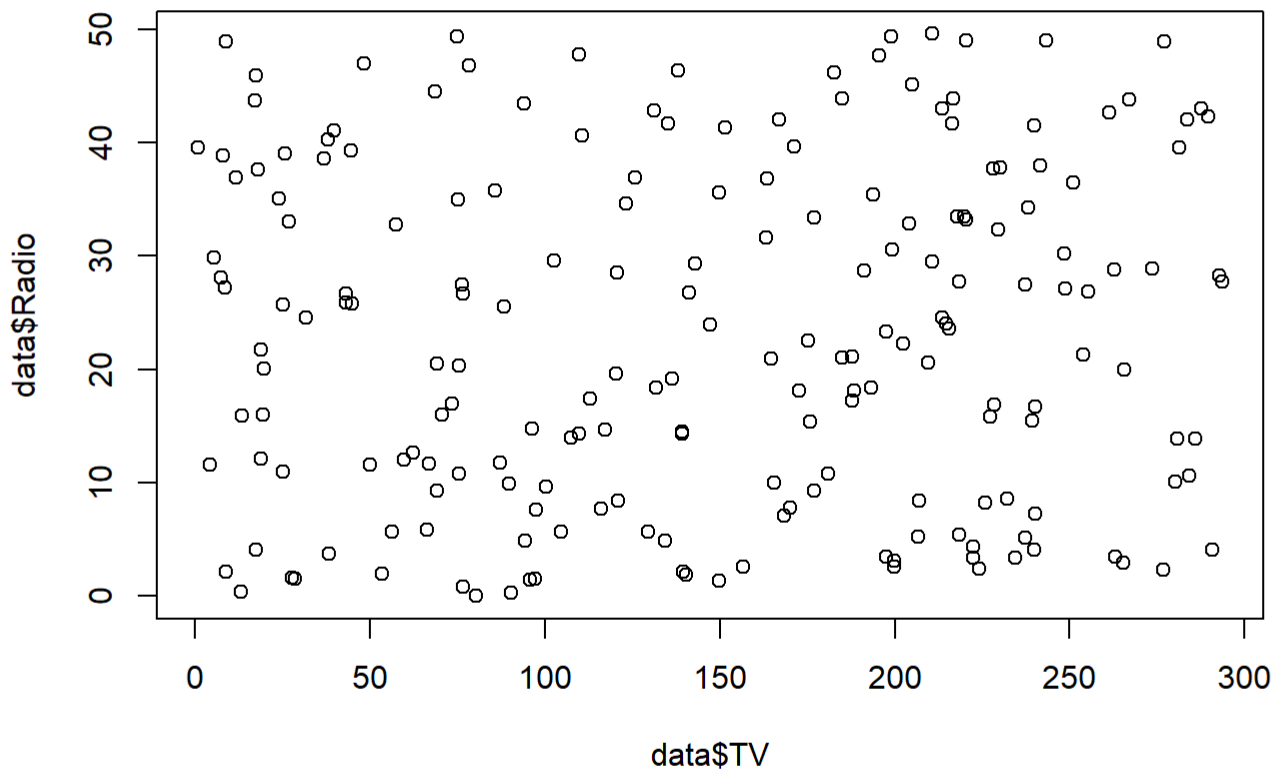
```
# Scatter Plot between Newspaper and Sales
plot(data$Newspaper , data$Sales)
```



Low

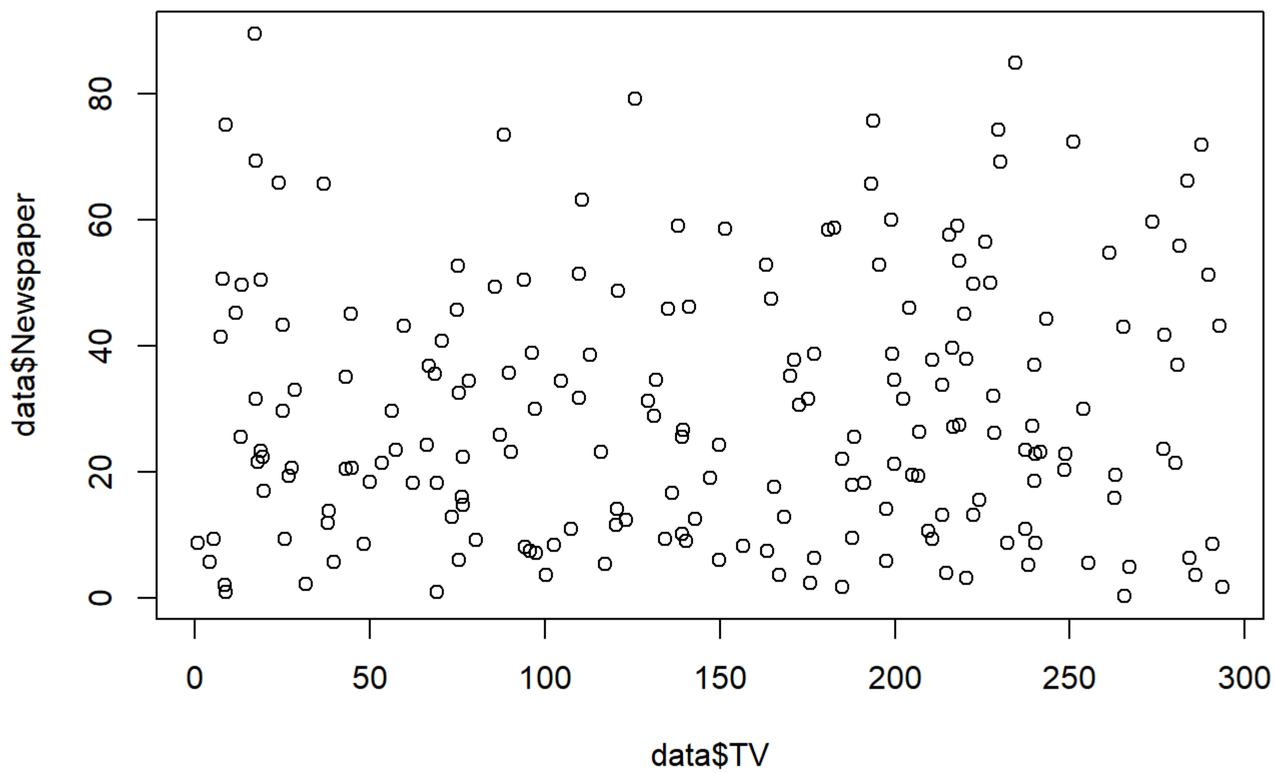
linear relationship between Newspaper and Sales variable

```
# Scatter Plot between TV and Radio
plot(data$TV , data$Radio)
```



No linear relationship between TV and Radio variable.

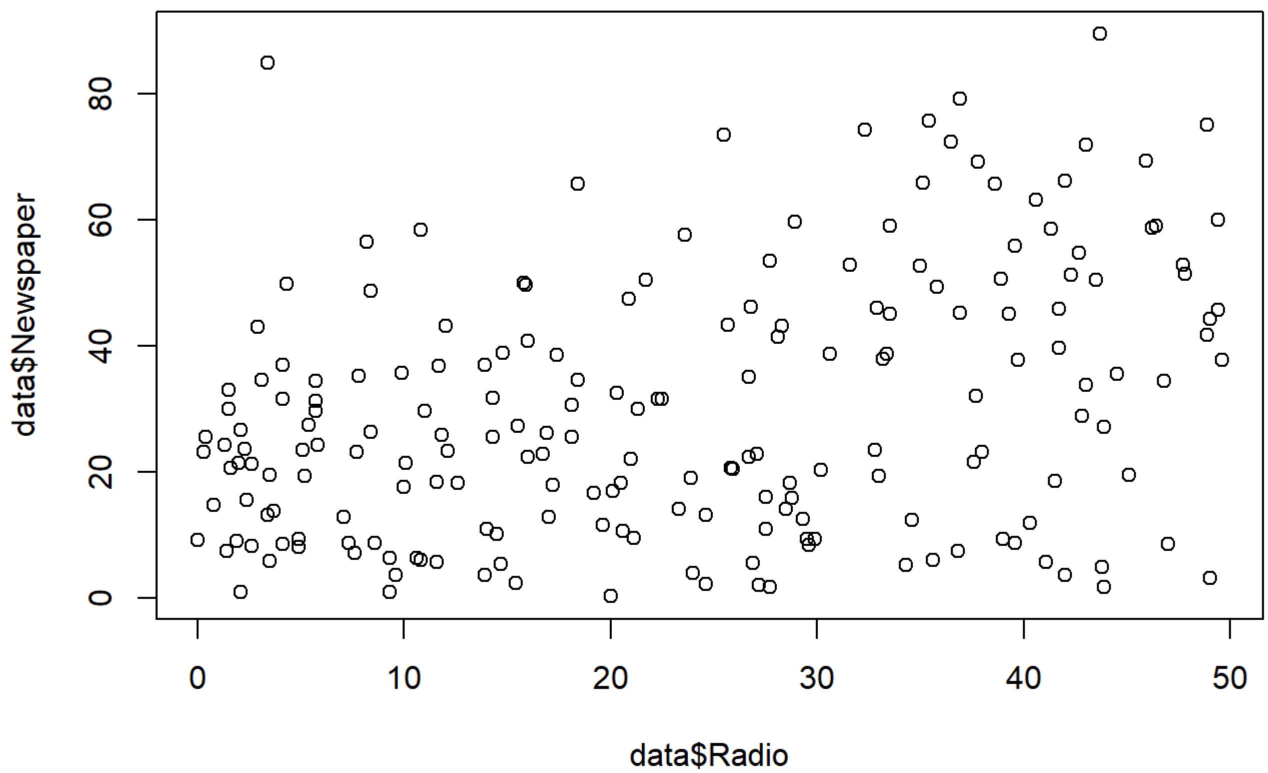
```
# Scatter Plot between Newspaper and TV
plot(data$TV , data$Newspaper)
```



No

linear relationship between TV and Newspaper variable.

```
plot(data$Radio , data$Newspaper)
```



Moderate linear relationship between Radio and Newspaper variable.

split the whole data set into two parts. One part is known as train data set and other is test data set. We do this because first we train/fit the model using train data set and then use the test data set to check the performance of the obtained model on new data set that has not been used during training period. Splitting is done by the following code

```
# Randomly Split the data into training and test set
set.seed(123)
training.samples <- data$Sales %>%
  createDataPartition(p = 0.75, list = FALSE)
train.data  <- data[training.samples, ]
test.data <- data[-training.samples, ]
```

Fitting Simple Linear Regression

```
# Fitting Sales ~ TV
sm1 <- lm(Sales ~ TV , data = train.data)

# Take a look on summary of the model
summary(sm1)
```

```
##
## Call:
## lm(formula = Sales ~ TV, data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4255 -1.5228 -0.0426  1.5328  5.7753
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.737411   0.381264   17.67   <2e-16 ***
## TV          0.056246   0.002221   25.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.29 on 148 degrees of freedom
## Multiple R-squared:  0.8125, Adjusted R-squared:  0.8112
## F-statistic: 641.2 on 1 and 148 DF,  p-value: < 2.2e-16
```

1- This model with TV as predictor explains approximately 81% variability of target (Sales).

2- Residual standard error for the model is 2.29

```
# Fitting Sales ~ Radio
sm2 <- lm(Sales ~ Radio , data = train.data)

# Take a look on summary of the model
summary(sm2)
```

```
##
## Call:
## lm(formula = Sales ~ Radio, data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.8285  -3.2876   0.6236   4.1403   8.3602
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.27086    0.72029  17.036  < 2e-16 ***
## Radio        0.13024    0.02704   4.817 3.58e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.917 on 148 degrees of freedom
## Multiple R-squared:  0.1355, Adjusted R-squared:  0.1297
## F-statistic:  23.2 on 1 and 148 DF,  p-value: 3.577e-06
```

This model with TV as predictor explains approximately 13% variability of target (Sales).

Residual standard error for the model is 4.917

```
# Fitting Sales ~ Newspaper
sm3 <- lm(Sales ~ Newspaper , data = train.data)

# Take a look on summary of the model
summary(sm3)
```

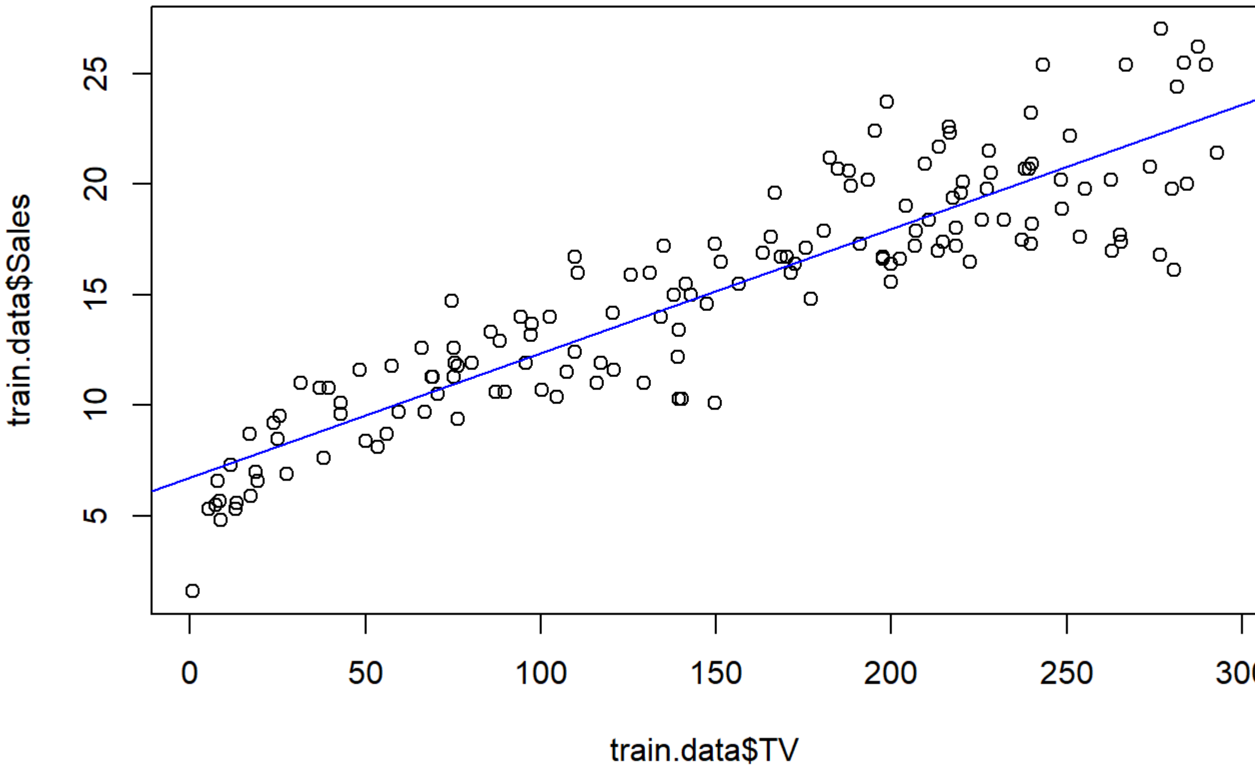
```
##
## Call:
## lm(formula = Sales ~ Newspaper, data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.6285  -3.9253   0.6376   3.7326  11.3377
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.83771    0.75380  18.357   <2e-16 ***
## Newspaper    0.04492    0.02128   2.111   0.0365 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.21 on 148 degrees of freedom
## Multiple R-squared:  0.02923,    Adjusted R-squared:  0.02267
## F-statistic: 4.456 on 1 and 148 DF,  p-value: 0.03646
```

1- This model with TV as predictor explains approximately 2% variability of target (Sales). 2- Residual standard error for the model is 5.21

Till now, we have obtained that Simple Linear Regression Model with TV as predictor is explaining more variability of target (Sales).

Just draw Scatter plot between TV and Sales and also draw the Simple Linear Regression Line in the plot as follows -

```
# Scatter plot with Simple Linear Regression Line
plot(train.data$TV , train.data$Sales)

# Adding Regression Line
abline(lm(train.data$Sales ~ train.data$TV) , col = "blue")
```




```
# Predicting on the test data
test.data$predicted_sales_simple <- predict(sm1, newdata = test.data)

# Calculating residual errors
test.data$residuals_simple <- test.data$Sales - test.data$predicted_sales_simple

# Comparing actual vs predicted values
head(test.data[, c("Sales", "predicted_sales_simple", "residuals_simple")])
```

```
##    Sales predicted_sales_simple residuals_simple
## 1   22.1              19.679513       2.42048710
## 2   10.4               9.240338       1.15966192
## 3   12.0               7.704834       4.29516554
## 6    7.2               7.226747      -0.02674725
## 8   13.2              13.498127      -0.29812651
## 25   9.7              10.241509      -0.54150894
```

if we use single predictor then we completely neglect the effect of rest two other predictors on Sales, that may not be the case in real. So, why not extend this model ?

Fitting Multiple Linear Regression with Diagnostic Plot

include the predictor Radio

Why we include Radio at this stage ?

Because it explains more variability (13%) of Sales in comparison to Newspaper (2%) after TV (81%). —> Results from Simple Linear Regression has been used here.

So, Fit a Multiple Linear Regression model with two predictors TV and Radio and obtain summary of the model

```
# Fitting MLR model with predictors TV and Radio
mm1 <- lm(Sales ~ TV + Radio , data = train.data)

# Take a look on summary of the model
summary(mm1)
```

```
##
## Call:
## lm(formula = Sales ~ TV + Radio, data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2088 -0.8338  0.0356  1.0480  3.6797
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.71593    0.34128   13.82   <2e-16 ***
## TV           0.05462    0.00167   32.70   <2e-16 ***
## Radio        0.10239    0.00947   10.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.715 on 147 degrees of freedom
## Multiple R-squared:  0.8955, Adjusted R-squared:  0.8941
## F-statistic: 630.2 on 2 and 147 DF,  p-value: < 2.2e-16
```

This model with TV and Radio as predictors explains approximately 89% variability of target (Sales) that is a better indication with respect to the model with TV alone as predictor.

Residual standard error for the model is 1.715

Hence, Adopt the model Sales ~ 0.05462 TV + 0.10239 Radio at this stage.

Include the third predictor Newspaper also in your multiple linear regression model

```
# Extending further the MLR including the predictor Newspaper
mm2 <- lm(Sales ~ TV + Radio + Newspaper , data = train.data)

# Take a look on summary of the model
summary(mm2)
```

```
##
## Call:
## lm(formula = Sales ~ TV + Radio + Newspaper, data = train.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1204 -0.7978  0.0033  1.0006  3.6731
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.664309   0.365891  12.748   <2e-16 ***
## TV          0.054607   0.001675  32.595   <2e-16 ***
## Radio       0.100791   0.010306   9.779   <2e-16 ***
## Newspaper   0.003046   0.007634   0.399     0.69
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.72 on 146 degrees of freedom
## Multiple R-squared:  0.8957, Adjusted R-squared:  0.8935
## F-statistic: 417.8 on 3 and 146 DF,  p-value: < 2.2e-16
```

Adjusted R-squared has been reduced 89.41 to 89.35

Residual standard error has been increased from 1.715 to 1.72

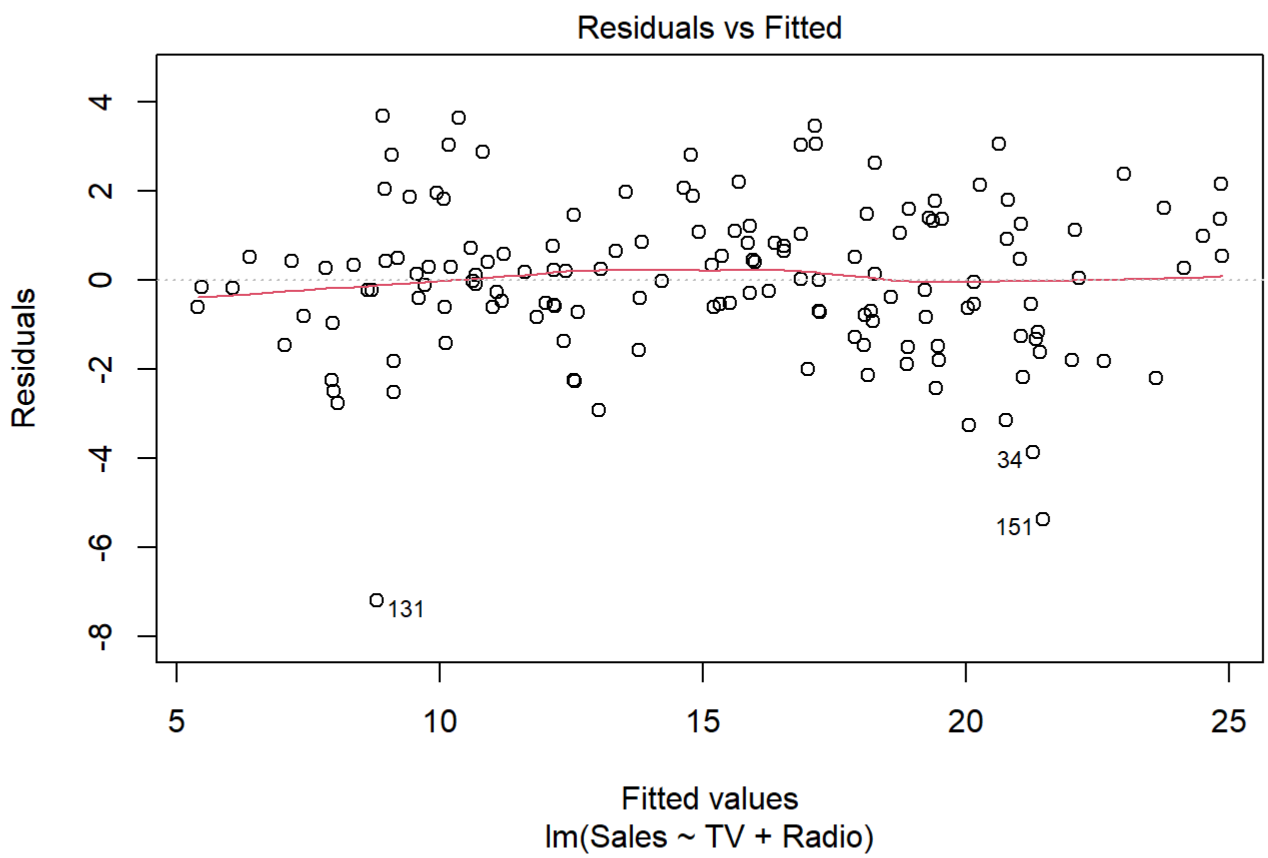
So, we have sufficient evidence from the data for not to include the Newspaper as predictor in the model.

Hence, Remove it from the model and we get the model as in previously fitted multiple linear regression model already stored in R-object mm1

Diagnostic Plots

Residual plot is used to check the first assumption, Linearity

```
# Residual Plot
plot(mm1 , 1)
```



Making predictions with the multiple

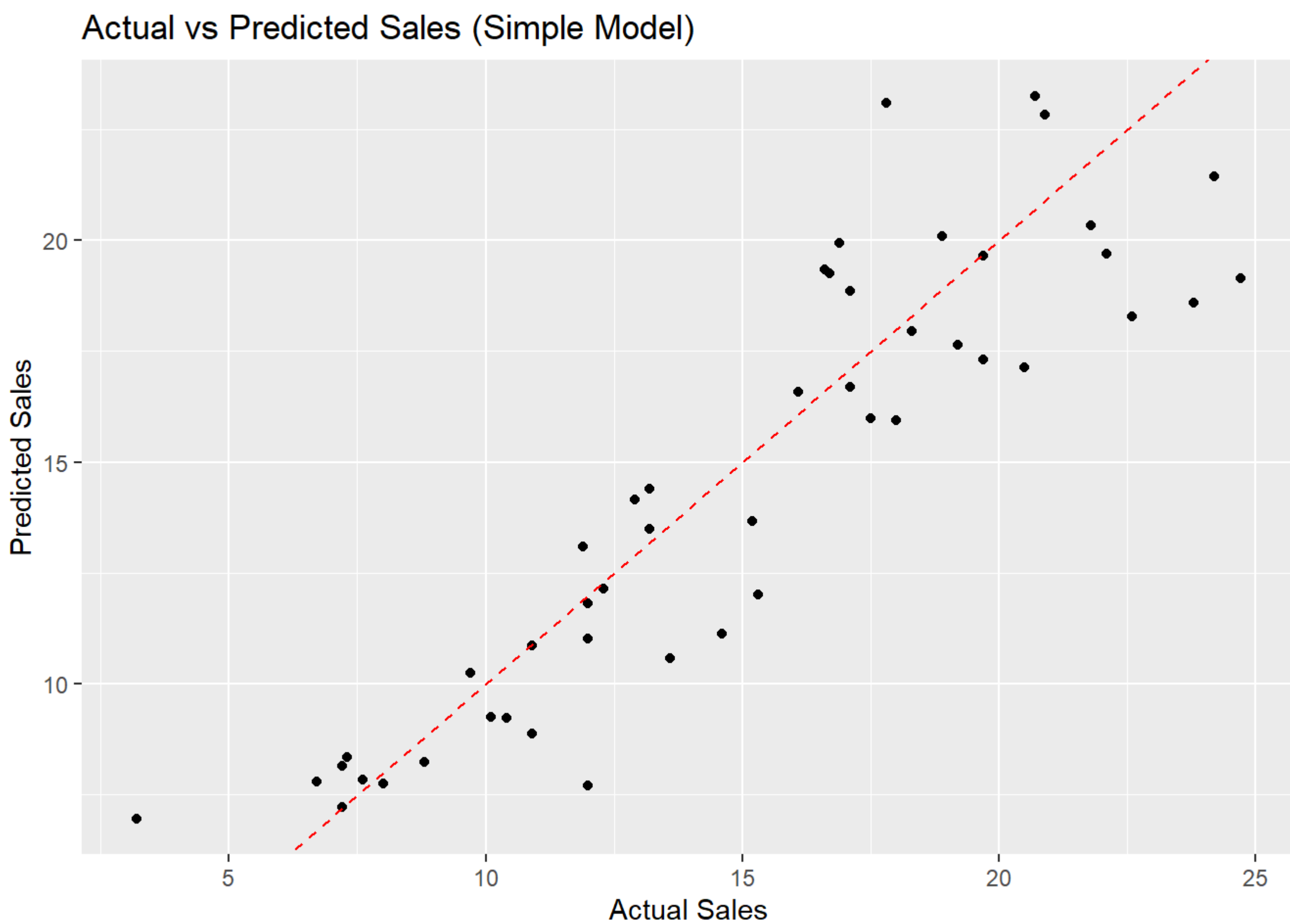
```
# Predicting on the test data
test.data$predicted_sales_simple <- predict(sm1, newdata = test.data)

# Calculating residual errors
test.data$residuals_simple <- test.data$Sales - test.data$predicted_sales_simple

# Comparing actual vs predicted values
sample(test.data[, c("Sales", "predicted_sales_simple", "residuals_simple")])
```

```
##     Sales predicted_sales_simple residuals_simple
## 1    22.1              19.679513       2.42048710
## 2    10.4               9.240338       1.15966192
## 3    12.0               7.704834       4.29516554
## 6     7.2               7.226747      -0.02674725
## 8    13.2              13.498127      -0.29812651
## 25    9.7              10.241509      -0.54150894
## 32   11.9              13.087534      -1.18753397
## 36   17.8              23.087993      -5.28799347
## 42   17.1              16.692874       0.40712602
## 43   20.7              23.251106      -2.55110557
## 46   16.1              16.586007      -0.48600743
## 58   13.2              14.398055      -1.19805538
## 59   23.8              18.593974       5.20602630
## 62   24.2              21.434374       2.76562582
## 69   18.9              20.090105      -1.19010544
## 71   18.3              17.935901       0.36409928
## 73    8.8               8.244792       0.55520823
## 84   13.6              10.584607       3.01539318
## 87   12.0              11.028947       0.97105331
## 92    7.3               8.346034      -1.04603377
## 98   20.5              17.137214       3.36278614
## 101  16.7              19.246422      -2.54642213
## 104  19.7              17.305951       2.39404948
## 107   7.2               8.143550      -0.94354978
## 108  12.0              11.822009       0.17799099
## 112  21.8              20.331961       1.46803868
## 115  14.6              11.135813       3.46418675
## 123  16.6              19.336415      -2.73641502
## 124  15.2              13.661239       1.53876138
## 125  19.7              19.645766       0.05423444
## 129  24.7              19.128306       5.57169353
## 141  10.9              10.865835       0.03416541
## 142  19.2              17.632175       1.56782527
## 145  12.3              12.148233       0.15176678
## 149  10.9               8.874742       2.02525802
## 150  10.1               9.251587       0.84841281
## 156   3.2               6.968018      -3.76801770
## 157  15.3              12.018868       3.28113156
## 160  12.9              14.144950      -1.24495038
## 164  18.0              15.933559       2.06644100
## 166  16.9              19.926993      -3.02699333
## 167   8.0               7.744206       0.25579366
## 169  17.1              18.852703      -1.75270325
## 172  17.5              15.989805       1.51019545
## 173   7.6               7.839824      -0.23982379
## 186  22.6              18.267749       4.33225051
## 189  20.9              22.823639      -1.92363936
## 190   6.7               7.789203      -1.08920279
```

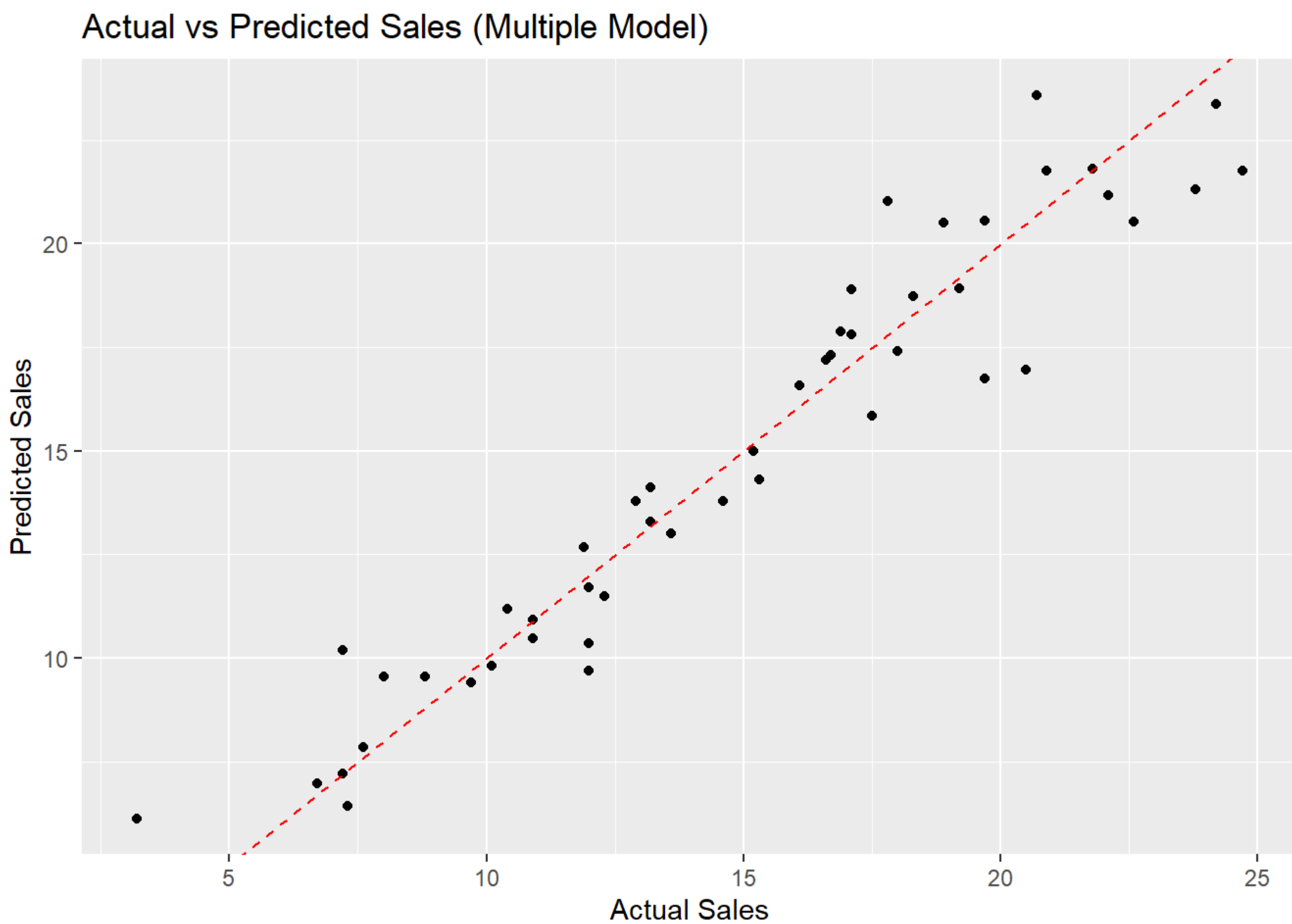
```r
# Predicting on the test data
test.data$predicted_sales_multiple <- predict(mm1, newdata = test.data)

# Calculating residual errors
test.data$residuals_multiple <- test.data$Sales - test.data$predicted_sales_multiple

# Comparing actual vs predicted values
sample(test.data[, c("Sales", "predicted_sales_multiple", "residuals_multiple")])
```

```
##     predicted_sales_multiple residuals_multiple Sales
## 1                  21.154631        0.945368609  22.1
## 2                  11.170472       -0.770472342  10.4
## 3                  10.355074        1.644925631  12.0
## 6                  10.197959       -2.997959172   7.2
## 8                  13.288252       -0.088252128  13.2
## 25                  9.408946        0.291053766   9.7
## 32                 12.664260       -0.764259538  11.9
## 36                 21.014180       -3.214180233  17.8
## 42                 17.803720       -0.703720068  17.1
## 43                 23.588964       -2.888964421  20.7
## 46                 16.583898       -0.483898477  16.1
## 58                 14.121240       -0.921239834  13.2
## 59                 21.308628        2.491371773  23.8
## 62                 23.360527        0.839472689  24.2
## 69                 20.498761       -1.598760655  18.9
## 71                 18.724165       -0.424164910  18.3
## 73                  9.558621       -0.758621415   8.8
## 84                 13.008348        0.591651662  13.6
## 87                 11.699244        0.300756266  12.0
## 92                  6.431684        0.868315542   7.3
## 98                 16.965605        3.534394831  20.5
## 101                17.304012       -0.604012449  16.7
## 104                16.740391        2.959608908  19.7
## 107                 7.207743       -0.007743336   7.2
## 108                 9.684424        2.315576302  12.0
## 112                21.808718       -0.008718119  21.8
## 115                13.779133        0.820866520  14.6
## 123                17.196868       -0.596867556  16.6
## 124                14.982490        0.217509651  15.2
## 125                20.558719       -0.858718652  19.7
## 129                21.766099        2.933901360  24.7
## 141                10.465756        0.434243689  10.9
## 142                18.920677        0.279323447  19.2
## 145                11.485870        0.814130386  12.3
## 149                10.917822       -0.017821930  10.9
## 150                 9.799144        0.300855758  10.1
## 156                 6.127588       -2.927588299   3.2
## 157                14.298806        1.001193544  15.3
## 160                13.793532       -0.893532019  12.9
## 164                17.414453        0.585546794  18.0
## 166                17.872782       -0.972781934  16.9
## 167                 9.543480       -1.543480141   8.0
## 169                18.897771       -1.797771216  17.1
## 172                15.841089        1.658911486  17.5
## 173                 7.844528       -0.244527963   7.6
## 186                20.531073        2.068927028  22.6
## 189                21.760872       -0.860872227  20.9
## 190                 6.976256       -0.276256122   6.7
```

```r
library(ggplot2)

# Plot for simple model (sm1)
plot_simple <- ggplot(test.data, aes(x = Sales, y = predicted_sales_simple)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(title = "Actual vs Predicted Sales (Simple Model)",
       x = "Actual Sales",
       y = "Predicted Sales")

# Plot for multiple model (mm1)
plot_multiple <- ggplot(test.data, aes(x = Sales, y = predicted_sales_multiple)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(title = "Actual vs Predicted Sales (Multiple Model)",
       x = "Actual Sales",
       y = "Predicted Sales")



# Displaying plots
plot_simple
```




```r
plot_multiple
```



Actual vs Predicted Sales (Multiple Model)
Predicted Sales
20
15
10
5
10
15
20
25
Actual Sales