DVD Sales Dataset

Satyam Prakash

9 January 2018

library(readr)  
#sales\_dataset<-read.csv(file.choose())  
sales\_dataset<- read.csv("G:/Data/DVD sales data set/Sales\_dataset.csv")  
dim(sales\_dataset)

## [1] 200 4

str(sales\_dataset)

## 'data.frame': 200 obs. of 4 variables:  
## $ advertise : num 10.3 985.7 1445.6 1188.2 574.5 ...  
## $ sales : int 330 120 360 270 220 170 70 210 200 300 ...  
## $ plays : int 43 28 35 33 44 19 20 22 21 40 ...  
## $ attractiveness: int 10 7 7 7 5 5 1 9 7 7 ...

colnames(sales\_dataset)

## [1] "advertise" "sales" "plays" "attractiveness"

library(caTools)

## Warning: package 'caTools' was built under R version 3.4.3

set.seed(2)  
split<-sample.split(sales\_dataset,SplitRatio = 0.75)  
split

## [1] TRUE FALSE TRUE TRUE

train\_data<-subset(sales\_dataset,split=="TRUE")  
dim(train\_data)

## [1] 150 4

test\_data<-subset(sales\_dataset,split=="FALSE")  
dim(test\_data)

## [1] 50 4

##Descriptive Analysis of the training dataset

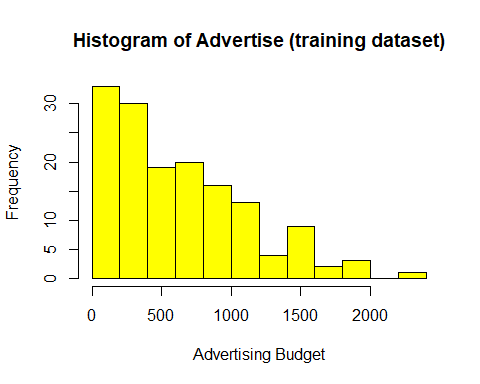
library(psych)

## Warning: package 'psych' was built under R version 3.4.3

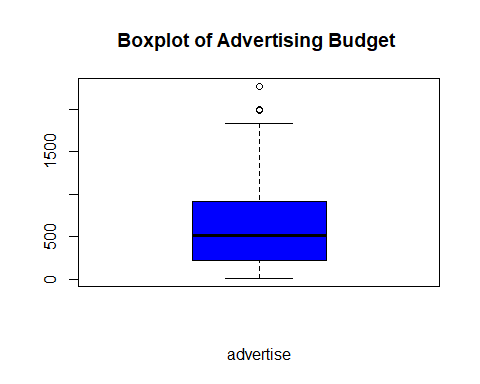
describe(train\_data$ advertise)

## vars n mean sd median trimmed mad min max range skew  
## X1 1 150 624.53 505.3 520.11 566.85 502.01 9.1 2271.86 2262.76 0.87  
## kurtosis se  
## X1 0.15 41.26

hist(train\_data$advertise,xlab = "Advertising Budget",ylab="Frequency",main="Histogram of Advertise (training dataset)",col="yellow")



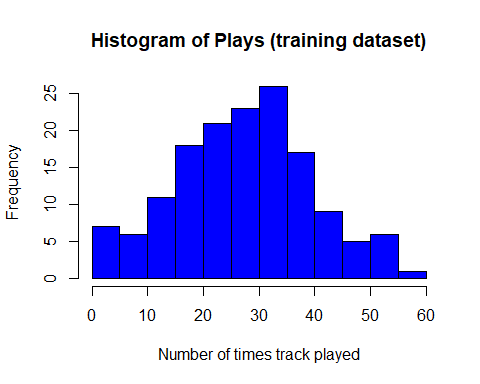
boxplot(train\_data$advertise,col="blue",main="Boxplot of Advertising Budget ",xlab="advertise") # we can see 2-3 Outliers in the dats



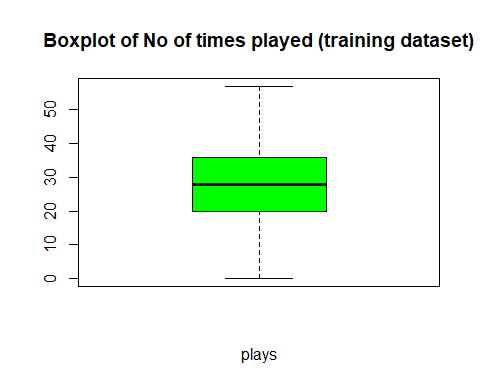
describe(train\_data$plays)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 150 27.78 12.35 28 27.66 11.86 0 57 57 0.05 -0.32  
## se  
## X1 1.01

hist(train\_data$plays,xlab = "Number of times track played",ylab="Frequency",main="Histogram of Plays (training dataset)",col="blue")



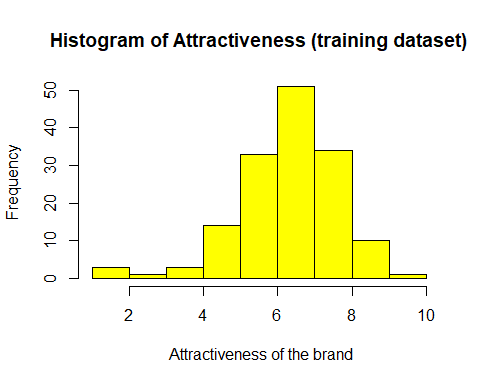
boxplot(train\_data$plays,col="green",main="Boxplot of No of times played (training dataset)",xlab="plays")



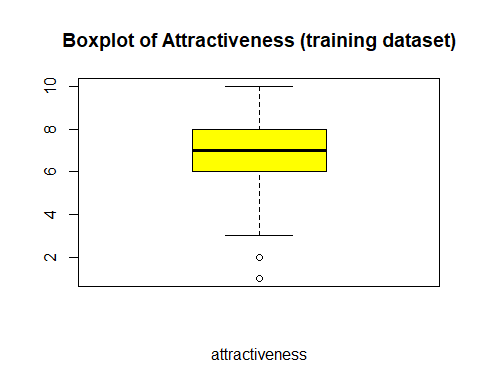
describe(sales\_dataset$attractiveness)

## vars n mean sd median trimmed mad min max range skew kurtosis se  
## X1 1 200 6.77 1.4 7 6.88 1.48 1 10 9 -1.27 3.56 0.1

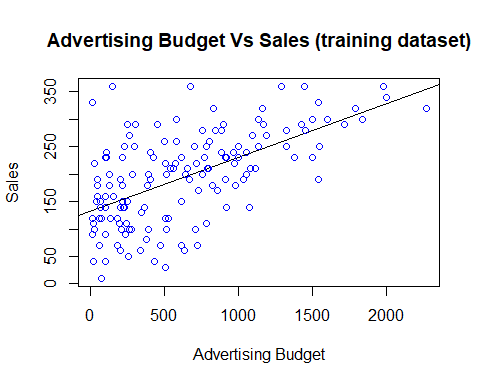
hist(train\_data$attractiveness,xlab = " Attractiveness of the brand",ylab="Frequency",main="Histogram of Attractiveness (training dataset)",col="yellow")



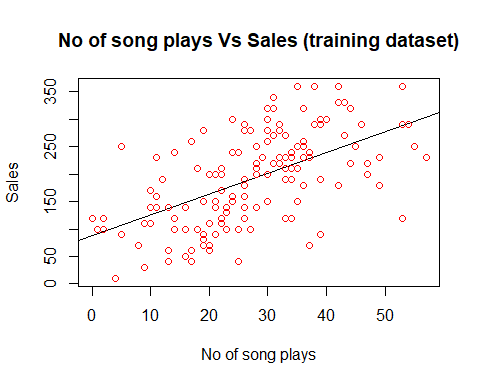
boxplot(train\_data$attractiveness,col="yellow",main="Boxplot of Attractiveness (training dataset)",xlab="attractiveness") # we can see 2 outliers in our data



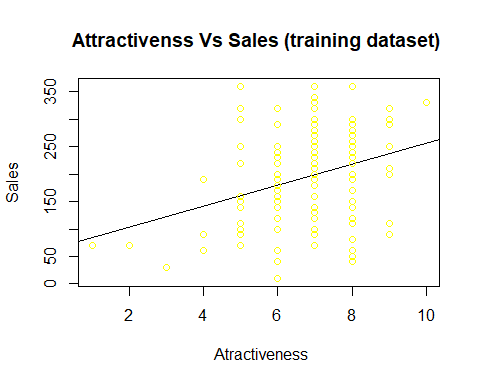
##Predictor Variable Vs Response Variable to check the relationship between them  
  
plot(train\_data$advertise,train\_data$sales,main= "Advertising Budget Vs Sales (training dataset)" , xlab = "Advertising Budget", ylab = "Sales",col="blue",abline(lm(sales~advertise,data=train\_data)))



plot(train\_data$plays,train\_data$sales,main= "No of song plays Vs Sales (training dataset)" , xlab = "No of song plays", ylab = "Sales",col="red",abline(lm(sales~plays,data=train\_data)))



plot(train\_data$attractiveness,train\_data$sales,main="Attractivenss Vs Sales (training dataset) ",xlab= " Atractiveness",ylab="Sales",col="yellow",abline(lm(sales~attractiveness,data=train\_data)))



#Correlation in training dataset  
  
#Checking correalation of each of predictors with Response Variable in training dataset  
cor(train\_data$advertise,train\_data$sales)

## [1] 0.596413

cor(train\_data$plays,train\_data$sales)

## [1] 0.5629677

cor(train\_data$attractiveness,train\_data$sales)

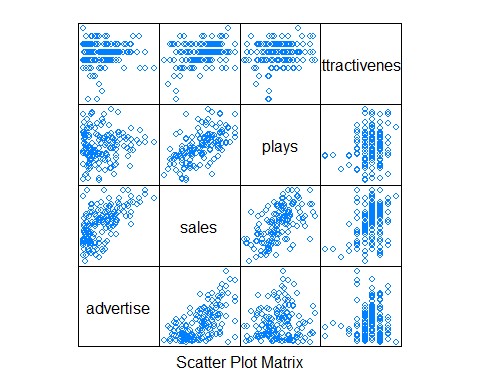
## [1] 0.3285217

# We can see that there is not much of correlation between the predictor variables

#storing all the correlation between the variables in the cr table  
cr<-cor(train\_data)  
cr

## advertise sales plays attractiveness  
## advertise 1.00000000 0.5964130 0.08297099 0.06158975  
## sales 0.59641303 1.0000000 0.56296767 0.32852169  
## plays 0.08297099 0.5629677 1.00000000 0.16786093  
## attractiveness 0.06158975 0.3285217 0.16786093 1.00000000

##Scatter plot Matrix to interpret Relationship  
  
#plotting a scatter matrix to understand the pattern or the relationship between the variables and the response variable  
library(lattice)  
splom(~train\_data[c(1:150),],groups=NULL,data = train\_data,axis.line.tck=0,axis.text.alpha=0)



#Multi-Collinearity in training dataset  
  
# finding multicollinearity by removing sales from the train\_data & checking correlation between predictor variables only  
#we created a new dataset sales\_dataset\_a to find multicollinearity among the variable

#install.packages("caret")  
library(caret)

## Warning: package 'caret' was built under R version 3.4.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.2

##   
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':  
##   
## %+%, alpha

train\_data\_a<-subset(train\_data,select = c(sales))  
numericdata<-train\_data\_a[sapply(train\_data\_a,is.numeric)]  
descrCor<-cor(numericdata)  
descrCor

## sales  
## sales 1

# we can see that predictor variables i.e. advertise,plays,attractiveness are not highly correlated   
# meaning multi collinarity doesnot exist among the variables

#Model building  
  
#building the model  
  
#install.packages("CARS")  
  
library(CARS)

## Warning: package 'CARS' was built under R version 3.4.3

fit<-lm(sales~advertise+plays+attractiveness,data=train\_data)  
summary(fit)

##   
## Call:  
## lm(formula = sales ~ advertise + plays + attractiveness, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -118.00 -30.29 -3.70 35.08 147.66   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -36.634675 20.420869 -1.794 0.0749 .   
## advertise 0.089398 0.007912 11.300 < 2e-16 \*\*\*  
## plays 3.244344 0.327668 9.901 < 2e-16 \*\*\*  
## attractiveness 12.461907 2.827921 4.407 2.02e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 48.57 on 146 degrees of freedom  
## Multiple R-squared: 0.6657, Adjusted R-squared: 0.6588   
## F-statistic: 96.9 on 3 and 146 DF, p-value: < 2.2e-16

#Verifying the Variation Inflation Factor for our model "fit"  
  
#install.packages("car")  
  
library(car)

## Warning: package 'car' was built under R version 3.4.2

##   
## Attaching package: 'car'

## The following object is masked from 'package:psych':  
##   
## logit

vif(fit)

## advertise plays attractiveness   
## 1.009308 1.034632 1.031422

# all the variables in the dataset has VIF less then 5 so all the variable are not correlated highly  
# Also the model which we built has all the three predictors as significant p-value hence we can continue with our model "fit" for predicting  
  
#Optimizing the fit model  
  
#Trying to optimize the model "fit" by getting Multiple R-squared value as high along with other parameters satisfied like p-value significance,F-value significance etc  
model1<-lm(sales~advertise,data=train\_data)  
summary(model1)

##   
## Call:  
## lm(formula = sales ~ advertise, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -152.274 -42.230 2.471 37.544 213.274   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 132.43645 8.71122 15.203 < 2e-16 \*\*\*  
## advertise 0.09815 0.01086 9.039 8.08e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 66.97 on 148 degrees of freedom  
## Multiple R-squared: 0.3557, Adjusted R-squared: 0.3514   
## F-statistic: 81.71 on 1 and 148 DF, p-value: 8.082e-16

model2<-lm(sales~plays,data=train\_data)  
summary(model2)

##   
## Call:  
## lm(formula = sales ~ plays, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -169.311 -49.010 1.802 53.642 142.598   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 88.4536 13.8964 6.365 2.31e-09 \*\*\*  
## plays 3.7898 0.4573 8.287 6.45e-14 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 68.96 on 148 degrees of freedom  
## Multiple R-squared: 0.3169, Adjusted R-squared: 0.3123   
## F-statistic: 68.67 on 1 and 148 DF, p-value: 6.454e-14

model3<-lm(sales~attractiveness,data=train\_data)  
summary(model3)

##   
## Call:  
## lm(formula = sales ~ attractiveness, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -177.183 -58.950 1.934 52.597 200.166   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 64.253 31.268 2.055 0.0416 \*   
## attractiveness 19.116 4.518 4.232 4.05e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 78.8 on 148 degrees of freedom  
## Multiple R-squared: 0.1079, Adjusted R-squared: 0.1019   
## F-statistic: 17.91 on 1 and 148 DF, p-value: 4.054e-05

model4<-lm(sales~advertise+plays,data=train\_data)  
summary(model4)

##   
## Call:  
## lm(formula = sales ~ advertise + plays, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -105.504 -36.203 4.269 33.966 160.399   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 40.154174 11.294927 3.555 0.000509 \*\*\*  
## advertise 0.091090 0.008383 10.866 < 2e-16 \*\*\*  
## plays 3.480609 0.342906 10.150 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 51.53 on 147 degrees of freedom  
## Multiple R-squared: 0.6212, Adjusted R-squared: 0.616   
## F-statistic: 120.5 on 2 and 147 DF, p-value: < 2.2e-16

model5<-lm(sales~plays+attractiveness,data=train\_data)  
summary(model5)

##   
## Call:  
## lm(formula = sales ~ plays + attractiveness, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -179.637 -44.670 -0.648 41.922 165.717   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.1036 27.4885 0.040 0.968031   
## plays 3.5177 0.4459 7.889 6.37e-13 \*\*\*  
## attractiveness 14.0122 3.8541 3.636 0.000383 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 66.28 on 147 degrees of freedom  
## Multiple R-squared: 0.3733, Adjusted R-squared: 0.3648   
## F-statistic: 43.78 on 2 and 147 DF, p-value: 1.215e-15

model6<-lm(sales~advertise+attractiveness,data=train\_data)  
summary(model6)

##   
## Call:  
## lm(formula = sales ~ advertise + attractiveness, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -137.023 -35.760 0.624 37.809 190.946   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 18.84973 25.30129 0.745 0.457   
## advertise 0.09518 0.01017 9.363 < 2e-16 \*\*\*  
## attractiveness 17.04342 3.59454 4.741 4.97e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 62.58 on 147 degrees of freedom  
## Multiple R-squared: 0.4412, Adjusted R-squared: 0.4336   
## F-statistic: 58.03 on 2 and 147 DF, p-value: < 2.2e-16

###Now we have build all the models possible with the 3 predictors available . All have R-Squared value less than "fit" .So we can say that 'fit' is the bes model that can be built.  
  
#Summary of "fit Model built for predicting"  
  
summary(fit)

##   
## Call:  
## lm(formula = sales ~ advertise + plays + attractiveness, data = train\_data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -118.00 -30.29 -3.70 35.08 147.66   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -36.634675 20.420869 -1.794 0.0749 .   
## advertise 0.089398 0.007912 11.300 < 2e-16 \*\*\*  
## plays 3.244344 0.327668 9.901 < 2e-16 \*\*\*  
## attractiveness 12.461907 2.827921 4.407 2.02e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 48.57 on 146 degrees of freedom  
## Multiple R-squared: 0.6657, Adjusted R-squared: 0.6588   
## F-statistic: 96.9 on 3 and 146 DF, p-value: < 2.2e-16

# value of Multiple R sqaure for "fit" 0.6657 meaning that the model is able to explain 66.57% of the variance in the sales in training dataset  
  
# final equation of our model 'fit' is :  
  
 # Checking Durban Watson statistics   
dwt(fit) # durban Watson Statistics for our model is 2.060746

## lag Autocorrelation D-W Statistic p-value  
## 1 -0.06005864 2.060746 0.668  
## Alternative hypothesis: rho != 0

# The DWS statistics is the number that tests the autocorrelation in the errors/residuals from a regression analysis.  
#It's value will always lie between "0 & 4"" .DWS value of '2' means there is no autocorrelation in the sample,  
#whereas value towards "zero" from '2' means there is +ve autocorrelation & "towards 4"" from '2' means -ve autocorrelation.  
#Autocorrelation is a characteristic of data in which the correlation between the values of the same variables is based on related objects.

#Comparing Predicted and Actual sales Values in training dataset  
  
#Finding the predicted Value of sales through the model made "fit"  
train\_data$predictedsales <- predict(fit)  
train\_data$predictedsales

## [1] 228.40804 293.38140 263.88431 219.78640 82.89342 194.93634 164.68663  
## [8] 308.25460 107.84112 163.38094 164.83135 130.50955 257.52634 269.25331  
## [15] 293.55500 228.49621 332.84505 219.56381 264.43491 108.18321 324.97227  
## [22] 182.77697 220.80242 158.48892 193.88747 258.94291 229.77649 140.16916  
## [29] 233.24815 161.00059 243.33318 329.96969 225.93536 221.28343 153.33856  
## [36] 153.23961 281.80601 225.04098 88.43296 85.62109 308.00533 240.89309  
## [43] 147.19310 96.03463 176.67151 199.71923 199.81875 115.05322 181.91970  
## [50] 121.18203 175.72570 137.77707 119.19719 269.87302 147.57247 262.56019  
## [57] 167.34643 228.06388 248.20684 98.41937 117.24896 43.38935 168.75068  
## [64] 116.43775 316.69291 319.71919 134.13113 226.02135 242.10896 296.42382  
## [71] 266.88348 210.28435 91.82541 330.31471 156.21857 155.22941 123.35851  
## [78] 145.47506 240.62173 103.01067 87.46751 163.60806 174.18664 221.25501  
## [85] 75.34403 194.55853 100.92021 143.05562 304.62669 144.01258 214.38917  
## [92] 123.20591 283.21413 57.45444 213.67063 262.21600 205.42061 175.03597  
## [99] 247.09604 200.10417 156.00660 261.57791 102.74813 205.57747 212.94981  
## [106] 192.73413 185.70497 165.02176 139.69984 114.52249 289.57383 228.85283  
## [113] 72.09267 153.15653 132.54743 107.06578 150.07562 244.01905 83.51868  
## [120] 158.50909 169.55605 213.96860 235.82467 166.40534 131.68748 167.40643  
## [127] 212.33805 224.63847 216.85641 209.45805 292.21540 160.93672 190.11577  
## [134] 265.04172 217.92254 53.47749 280.40894 329.34966 291.01892 158.98270  
## [141] 186.62942 267.80677 153.46524 150.28642 238.70430 251.73629 216.38003  
## [148] 162.99403 232.84563 210.64897

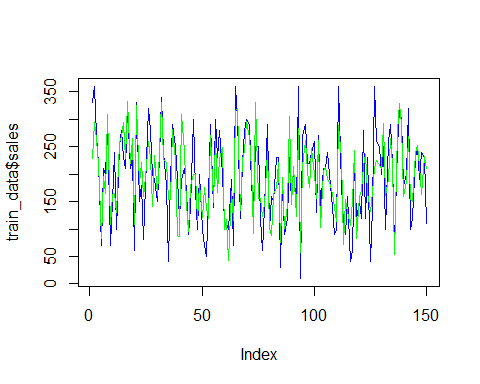
# error values of sales through the model is below. These values are giving us the residuals/errors   
#i.e difference between predicted and Actual sales  
  
train\_data$error<- residuals(fit)  
train\_data$error

## [1] 101.5919626 66.6185976 6.1156946 0.2136047 -12.8934174  
## [6] 15.0636615 35.3133663 -18.2545951 -37.8411213 -13.3809353  
## [11] 75.1686480 -30.5095485 -7.5263364 10.7466939 -63.5550046  
## [16] -18.4962143 -12.8450548 -9.5638105 -34.4349056 -48.1832069  
## [21] 5.0277348 -32.7769691 -40.8024249 -78.4889170 -13.8874651  
## [26] 61.0570864 50.2235085 59.8308373 -43.2481482 -11.0005885  
## [31] -13.3331776 10.0303128 14.0646397 -41.2834276 -113.3385636  
## [36] 36.7603853 8.1939866 24.9590221 101.5670398 34.3789054  
## [41] -118.0053265 -30.8930890 22.8068968 -6.0346257 -36.6715101  
## [46] 100.2807745 -29.8187514 -15.0532197 18.0803048 -21.1820304  
## [51] -105.7256966 -87.7770744 40.8028137 20.1269781 -7.5724736  
## [56] 37.4398084 62.6535742 51.9361185 -48.2068436 11.5806264  
## [61] -7.2489625 56.6106484 21.2493192 -46.4377474 43.3070868  
## [66] -19.7191915 -14.1311343 -6.0213473 37.8910436 3.5761763  
## [71] 23.1165246 -30.2843463 48.1745931 -80.3147146 93.7814270  
## [76] -35.2294137 -63.3585062 -5.4750631 49.3782750 -3.0106705  
## [81] 72.5324862 -13.6080613 55.8133560 8.7449927 -45.3440317  
## [86] -4.5585309 -10.9202143 -23.0556222 -74.6266928 5.9874199  
## [91] -4.3891698 16.7940933 76.7858675 -47.4544394 56.3293697  
## [96] 27.7840029 14.5793940 44.9640296 -7.0960364 59.8958259  
## [101] -26.0065951 8.4220873 37.2518713 4.4225317 -2.9498102  
## [106] 47.2658731 14.2950327 -25.0217617 -49.6998422 -14.5224942  
## [111] 70.4261729 -48.8528269 37.9073320 -63.1565298 27.4525661  
## [116] -67.0657771 -90.0756176 -14.0190520 36.4813190 -8.5090902  
## [121] -49.5560549 66.0313965 -115.8246726 63.5946638 -91.6874846  
## [126] -27.4064344 147.6619538 35.3615340 33.1435877 -9.4580530  
## [131] -42.2154018 -60.9367235 69.8842301 24.9582834 2.0774572  
## [136] 16.5225118 -30.4089436 -9.3496626 8.9810777 21.0173047  
## [141] 13.3705828 52.1932343 -53.4652448 -30.2864153 -8.7043038  
## [146] -1.7362914 -26.3800300 77.0059703 -2.8456319 -100.6489743

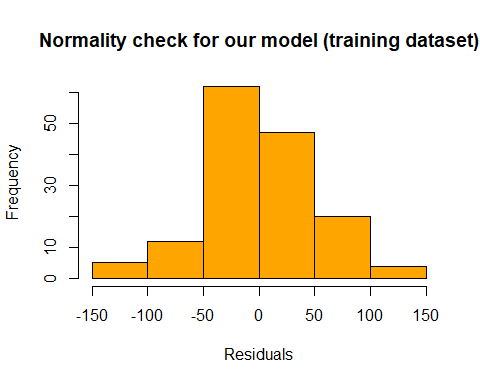
# Adding observation No. Column n our dataset  
train\_data$obsno<-c(1:150)  
train\_data$obsno

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
## [18] 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34  
## [35] 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51  
## [52] 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68  
## [69] 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85  
## [86] 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102  
## [103] 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119  
## [120] 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136  
## [137] 137 138 139 140 141 142 143 144 145 146 147 148 149 150

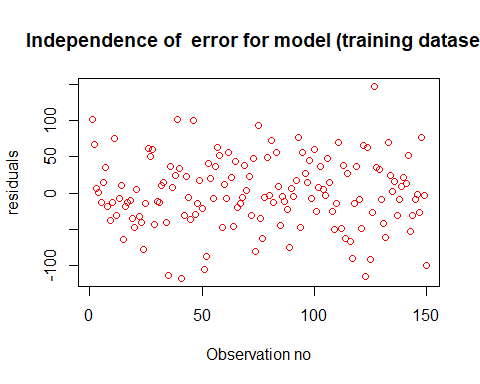
View(train\_data)  
  
#comparing model predicted values with actual values of sales using the graph for train\_data  
#we can see the line curve as below  
plot(train\_data$sales,type="l",lty=1.8,col="blue")  
lines(train\_data$predictedsales,type="l",col="green")



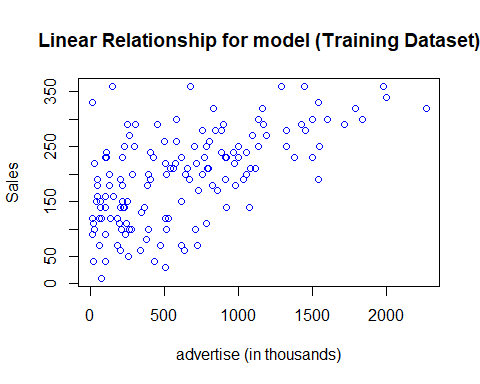
#most of the lines are overlapping meaning our model "fit" is a good model based on the predictors given to us in sales\_dataset for train\_data  
  
#Assumptions Test for Model "fit"  
  
  
####1. Normality test for model  
hist(train\_data$error,main = "Normality check for our model (training dataset)", xlab="Residuals",col="orange")



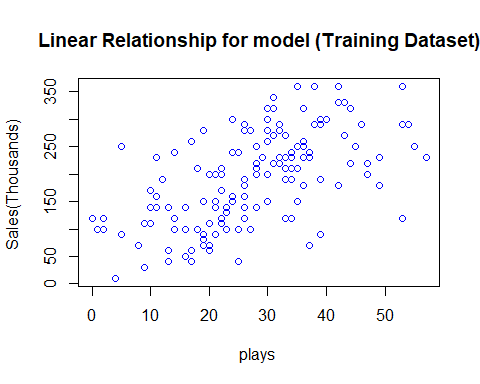
####2. Independence of observations   
plot(train\_data$obsno,train\_data$error,main="Independence of error for model (training dataset)",xlab= "Observation no", ylab="residuals",col="red")



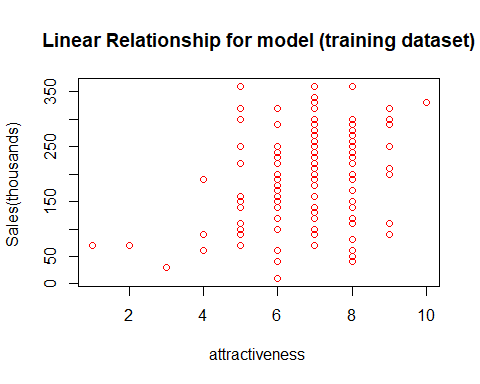
####3 Check of linear relationship   
plot(train\_data$advertise,train\_data$sales,main="Linear Relationship for model (Training Dataset)",xlab="advertise (in thousands)",ylab="Sales",col="blue")



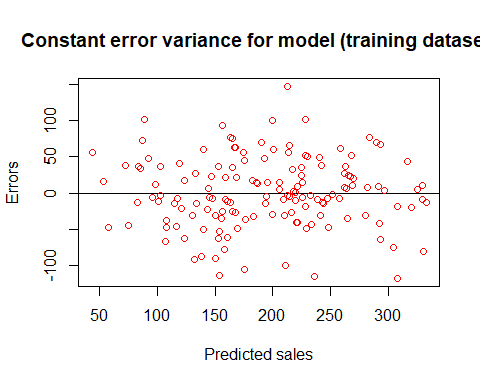
plot(train\_data$plays,train\_data$sales,main="Linear Relationship for model (Training Dataset)",xlab="plays ",ylab="Sales(Thousands) ",col="blue")



plot(train\_data$attractiveness,train\_data$sales,main="Linear Relationship for model (training dataset)",xlab="attractiveness ",ylab="Sales(thousands) ",col="red")



####4 Check of Constant Error Variance : Homoscedacity  
plot(train\_data$predictedsales,train\_data$error,main="Constant error variance for model (training dataset)",xlab="Predicted sales",ylab="Errors",abline(h=0),col="red")



# for finding the confidence intervals & the predited values for model for training dataset  
  
confint(fit)

## 2.5 % 97.5 %  
## (Intercept) -76.99337075 3.7240214  
## advertise 0.07376214 0.1050342  
## plays 2.59675886 3.8919283  
## attractiveness 6.87295802 18.0508560

fitted(fit)

## 1 3 4 5 7 8 9   
## 228.40804 293.38140 263.88431 219.78640 82.89342 194.93634 164.68663   
## 11 12 13 15 16 17 19   
## 308.25460 107.84112 163.38094 164.83135 130.50955 257.52634 269.25331   
## 20 21 23 24 25 27 28   
## 293.55500 228.49621 332.84505 219.56381 264.43491 108.18321 324.97227   
## 29 31 32 33 35 36 37   
## 182.77697 220.80242 158.48892 193.88747 258.94291 229.77649 140.16916   
## 39 40 41 43 44 45 47   
## 233.24815 161.00059 243.33318 329.96969 225.93536 221.28343 153.33856   
## 48 49 51 52 53 55 56   
## 153.23961 281.80601 225.04098 88.43296 85.62109 308.00533 240.89309   
## 57 59 60 61 63 64 65   
## 147.19310 96.03463 176.67151 199.71923 199.81875 115.05322 181.91970   
## 67 68 69 71 72 73 75   
## 121.18203 175.72570 137.77707 119.19719 269.87302 147.57247 262.56019   
## 76 77 79 80 81 83 84   
## 167.34643 228.06388 248.20684 98.41937 117.24896 43.38935 168.75068   
## 85 87 88 89 91 92 93   
## 116.43775 316.69291 319.71919 134.13113 226.02135 242.10896 296.42382   
## 95 96 97 99 100 101 103   
## 266.88348 210.28435 91.82541 330.31471 156.21857 155.22941 123.35851   
## 104 105 107 108 109 111 112   
## 145.47506 240.62173 103.01067 87.46751 163.60806 174.18664 221.25501   
## 113 115 116 117 119 120 121   
## 75.34403 194.55853 100.92021 143.05562 304.62669 144.01258 214.38917   
## 123 124 125 127 128 129 131   
## 123.20591 283.21413 57.45444 213.67063 262.21600 205.42061 175.03597   
## 132 133 135 136 137 139 140   
## 247.09604 200.10417 156.00660 261.57791 102.74813 205.57747 212.94981   
## 141 143 144 145 147 148 149   
## 192.73413 185.70497 165.02176 139.69984 114.52249 289.57383 228.85283   
## 151 152 153 155 156 157 159   
## 72.09267 153.15653 132.54743 107.06578 150.07562 244.01905 83.51868   
## 160 161 163 164 165 167 168   
## 158.50909 169.55605 213.96860 235.82467 166.40534 131.68748 167.40643   
## 169 171 172 173 175 176 177   
## 212.33805 224.63847 216.85641 209.45805 292.21540 160.93672 190.11577   
## 179 180 181 183 184 185 187   
## 265.04172 217.92254 53.47749 280.40894 329.34966 291.01892 158.98270   
## 188 189 191 192 193 195 196   
## 186.62942 267.80677 153.46524 150.28642 238.70430 251.73629 216.38003   
## 197 199 200   
## 162.99403 232.84563 210.64897

predicted\_training<- predict(fit, interval="confidence") #predicted with lower & upper limit of prediction by model "fit"  
dim(predicted\_training)

## [1] 150 3

predicted\_training<-as.data.frame(predicted\_training)  
  
# comparing the predicted values from "fit & the actual values of sales in train\_data in the dataframe  
predicted\_training$actualtrainingsales<-train\_data$sales  
predicted\_training # now with actual sales values from train\_data

## fit lwr upr actualtrainingsales  
## 1 228.40804 204.98449 251.83159 330  
## 3 293.38140 277.97581 308.78700 360  
## 4 263.88431 251.83179 275.93682 270  
## 5 219.78640 202.31825 237.25454 220  
## 7 82.89342 50.16392 115.62292 70  
## 8 194.93634 179.17842 210.69426 210  
## 9 164.68663 155.40145 173.97182 200  
## 11 308.25460 289.43482 327.07437 290  
## 12 107.84112 80.38374 135.29850 70  
## 13 163.38094 150.90009 175.86178 150  
## 15 164.83135 155.84111 173.82160 240  
## 16 130.50955 117.02967 143.98942 100  
## 17 257.52634 239.66490 275.38777 250  
## 19 269.25331 254.29166 284.21495 280  
## 20 293.55500 278.11778 308.99223 230  
## 21 228.49621 219.64634 237.34608 210  
## 23 332.84505 310.07332 355.61679 320  
## 24 219.56381 210.76123 228.36640 210  
## 25 264.43491 248.73227 280.13755 230  
## 27 108.18321 90.52895 125.83747 60  
## 28 324.97227 306.81216 343.13237 330  
## 29 182.77697 168.38783 197.16611 150  
## 31 220.80242 202.80750 238.79735 180  
## 32 158.48892 145.53830 171.43954 80  
## 33 193.88747 178.91005 208.86488 180  
## 35 258.94291 245.38193 272.50390 320  
## 36 229.77649 220.87460 238.67838 280  
## 37 140.16916 128.41371 151.92461 200  
## 39 233.24815 222.51340 243.98290 190  
## 40 161.00059 151.35610 170.64508 150  
## 41 243.33318 233.54056 253.12580 230  
## 43 329.96969 307.17035 352.76903 340  
## 44 225.93536 215.30105 236.56967 240  
## 45 221.28343 208.49298 234.07387 180  
## 47 153.33856 139.71558 166.96155 40  
## 48 153.23961 142.86387 163.61536 190  
## 49 281.80601 268.55470 295.05733 290  
## 51 225.04098 214.02727 236.05469 250  
## 52 88.43296 69.57388 107.29204 190  
## 53 85.62109 64.16293 107.07926 120  
## 55 308.00533 290.64105 325.36960 190  
## 56 240.89309 229.93346 251.85272 210  
## 57 147.19310 132.45135 161.93485 170  
## 59 96.03463 77.93200 114.13725 90  
## 60 176.67151 161.79127 191.55175 140  
## 61 199.71923 191.64157 207.79688 300  
## 63 199.81875 188.26676 211.37074 170  
## 64 115.05322 102.21621 127.89023 100  
## 65 181.91970 165.75824 198.08115 200  
## 67 121.18203 109.62208 132.74198 100  
## 68 175.72570 162.16455 189.28685 70  
## 69 137.77707 123.13303 152.42112 50  
## 71 119.19719 104.54707 133.84730 160  
## 72 269.87302 248.15019 291.59585 290  
## 73 147.57247 135.74815 159.39680 140  
## 75 262.56019 246.41399 278.70639 300  
## 76 167.34643 153.39679 181.29606 230  
## 77 228.06388 217.58764 238.54012 280  
## 79 248.20684 232.32296 264.09073 200  
## 80 98.41937 82.99699 113.84175 110  
## 81 117.24896 100.03795 134.45997 110  
## 83 43.38935 22.97088 63.80782 100  
## 84 168.75068 153.55147 183.94989 190  
## 85 116.43775 99.43361 133.44188 70  
## 87 316.69291 291.09877 342.28706 360  
## 88 319.71919 295.14768 344.29070 300  
## 89 134.13113 122.56944 145.69283 120  
## 91 226.02135 215.14288 236.89982 220  
## 92 242.10896 225.53529 258.68262 280  
## 93 296.42382 275.16345 317.68419 300  
## 95 266.88348 254.67824 279.08871 290  
## 96 210.28435 199.59021 220.97848 180  
## 97 91.82541 77.22341 106.42741 140  
## 99 330.31471 310.11642 350.51301 250  
## 100 156.21857 137.96178 174.47536 250  
## 101 155.22941 141.92033 168.53849 120  
## 103 123.35851 107.23117 139.48585 60  
## 104 145.47506 135.56950 155.38062 140  
## 105 240.62173 219.93883 261.30462 290  
## 107 103.01067 84.36607 121.65528 100  
## 108 87.46751 72.55633 102.37870 160  
## 109 163.60806 149.46455 177.75157 150  
## 111 174.18664 161.56697 186.80632 230  
## 112 221.25501 209.21199 233.29802 230  
## 113 75.34403 51.54332 99.14475 30  
## 115 194.55853 177.75111 211.36595 190  
## 116 100.92021 77.41114 124.42929 90  
## 117 143.05562 130.93530 155.17595 120  
## 119 304.62669 282.84349 326.40989 230  
## 120 144.01258 130.11243 157.91273 150  
## 121 214.38917 205.18503 223.59331 210  
## 123 123.20591 110.71614 135.69567 140  
## 124 283.21413 265.28149 301.14678 360  
## 125 57.45444 38.82378 76.08510 10  
## 127 213.67063 199.64012 227.70114 270  
## 128 262.21600 247.33196 277.10004 290  
## 129 205.42061 197.36899 213.47223 220  
## 131 175.03597 161.68563 188.38631 220  
## 132 247.09604 236.99757 257.19450 240  
## 133 200.10417 192.03353 208.17482 260  
## 135 156.00660 146.43429 165.57890 130  
## 136 261.57791 249.12040 274.03543 270  
## 137 102.74813 87.95765 117.53860 140  
## 139 205.57747 192.83530 218.31964 210  
## 140 212.94981 204.57563 221.32400 210  
## 141 192.73413 178.29525 207.17301 240  
## 143 185.70497 173.85368 197.55625 200  
## 144 165.02176 149.90501 180.13851 140  
## 145 139.69984 128.83180 150.56788 90  
## 147 114.52249 101.06142 127.98357 100  
## 148 289.57383 275.35438 303.79327 360  
## 149 228.85283 217.68003 240.02563 180  
## 151 72.09267 54.91374 89.27159 110  
## 152 153.15653 135.01202 171.30104 90  
## 153 132.54743 120.30794 144.78693 160  
## 155 107.06578 89.63962 124.49193 40  
## 156 150.07562 139.15561 160.99563 60  
## 157 244.01905 233.96153 254.07657 230  
## 159 83.51868 64.12723 102.91013 120  
## 160 158.50909 143.15261 173.86557 150  
## 161 169.55605 160.54152 178.57059 120  
## 163 213.96860 204.94174 222.99546 280  
## 164 235.82467 214.35994 257.28940 120  
## 165 166.40534 152.98851 179.82216 230  
## 167 131.68748 120.55852 142.81645 40  
## 168 167.40643 154.96669 179.84618 140  
## 169 212.33805 196.72327 227.95283 360  
## 171 224.63847 213.17675 236.10018 260  
## 172 216.85641 201.60905 232.10377 250  
## 173 209.45805 197.66155 221.25456 200  
## 175 292.21540 274.71233 309.71847 250  
## 176 160.93672 151.40291 170.47053 100  
## 177 190.11577 176.63390 203.59764 260  
## 179 265.04172 247.07247 283.01097 290  
## 180 217.92254 199.31507 236.53002 220  
## 181 53.47749 20.37555 86.57943 70  
## 183 280.40894 258.02069 302.79719 250  
## 184 329.34966 300.17636 358.52296 320  
## 185 291.01892 277.06641 304.97144 300  
## 187 158.98270 144.79370 173.17169 180  
## 188 186.62942 177.15825 196.10058 200  
## 189 267.80677 254.60915 281.00438 320  
## 191 153.46524 142.00513 164.92536 100  
## 192 150.28642 137.02719 163.54565 120  
## 193 238.70430 227.99651 249.41210 230  
## 195 251.73629 236.96175 266.51083 250  
## 196 216.38003 207.15491 225.60515 190  
## 197 162.99403 149.90419 176.08387 240  
## 199 232.84563 211.60558 254.08568 230  
## 200 210.64897 194.29687 227.00108 110

#Verifying for our testing dataset  
  
# Firstly Checking basics details about test\_data  
test\_data

## advertise sales plays attractiveness  
## 2 985.685 120 28 7  
## 6 568.954 170 19 5  
## 10 174.093 300 40 7  
## 14 97.972 190 38 6  
## 18 196.650 210 36 8  
## 22 957.167 230 28 6  
## 26 313.362 250 40 8  
## 30 785.692 150 8 6  
## 34 759.862 130 6 7  
## 38 236.598 130 25 8  
## 42 50.000 310 63 7  
## 46 1507.972 220 37 7  
## 50 689.547 340 46 7  
## 54 607.258 230 29 8  
## 58 836.331 310 38 7  
## 62 1500.000 340 38 8  
## 66 233.999 80 20 7  
## 70 566.501 240 32 8  
## 74 642.786 210 32 7  
## 78 51.229 160 19 7  
## 82 256.894 70 1 4  
## 86 1126.461 360 38 7  
## 90 237.703 150 27 8  
## 94 268.598 140 1 7  
## 98 746.024 210 34 6  
## 102 1351.254 290 37 9  
## 106 263.268 160 18 7  
## 110 102.568 140 22 7  
## 114 233.291 80 2 7  
## 118 624.538 150 20 5  
## 122 474.760 180 22 5  
## 126 1567.548 240 29 6  
## 130 777.237 230 37 8  
## 134 923.373 170 15 7  
## 138 30.425 60 28 1  
## 142 893.355 210 26 6  
## 146 456.897 120 18 9  
## 150 564.158 150 32 7  
## 154 42.568 230 45 7  
## 158 1177.047 230 23 6  
## 162 26.895 60 19 6  
## 166 169.583 230 28 7  
## 170 985.968 210 17 6  
## 174 68.093 150 15 7  
## 178 801.577 210 32 8  
## 182 345.687 110 22 8  
## 186 601.434 180 21 6  
## 190 56.894 140 27 7  
## 194 303.172 150 21 7  
## 198 800.615 250 34 6

dim(test\_data)

## [1] 50 4

summary(test\_data)

## advertise sales plays attractiveness  
## Min. : 26.89 Min. : 60.0 Min. : 1.00 Min. :1.00   
## 1st Qu.: 233.47 1st Qu.:142.5 1st Qu.:19.25 1st Qu.:6.00   
## Median : 565.33 Median :185.0 Median :27.50 Median :7.00   
## Mean : 560.79 Mean :191.6 Mean :26.66 Mean :6.76   
## 3rd Qu.: 801.34 3rd Qu.:230.0 3rd Qu.:35.50 3rd Qu.:7.00   
## Max. :1567.55 Max. :360.0 Max. :63.00 Max. :9.00

#Verifying the "fit" model for our testing dataset  
  
  
#Verifying our model on testing data set so that we get to know the accuracy and predicted values of our testing dataset and compare them with the actual sales values in testing dataset  
#Finding the predicted Value of sales through the model made "fit"  
predict\_testing<-predict(fit,test\_data)  
predict\_testing

## 2 6 10 14 18 22 26   
## 229.55874 138.18084 195.93602 170.18034 197.43710 214.54738 220.84832   
## 30 34 38 42 46 50 54   
## 134.33095 137.99502 165.32060 259.46223 305.44934 261.48273 211.43431   
## 58 62 66 70 74 78 82   
## 248.65020 320.44291 136.40463 217.52373 211.88157 116.82098 39.42315   
## 86 90 94 98 102 106 110   
## 274.58729 171.90807 77.85519 215.13764 316.36285 132.53254 131.14363   
## 114 118 122 126 130 134 138   
## 77.94315 146.39429 139.49310 272.35867 252.58487 181.81169 69.38879   
## 142 146 150 154 158 162 166   
## 202.35401 174.76643 204.85237 200.39964 217.98253 102.18366 156.60071   
## 170 174 178 182 186 190 194   
## 181.43435 105.35122 238.53910 165.33993 160.03509 143.28217 145.83291   
## 198   
## 220.01797

test\_data$predicted\_testing\_sales <- predict\_testing  
dim(test\_data)

## [1] 50 5

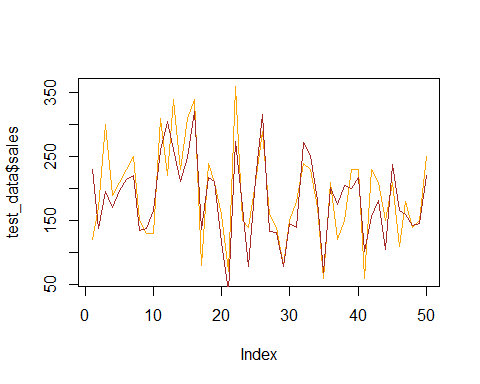
View(test\_data)  
  
# error values of sales through the model is below. These values are giving us the residuals/errors   
#i.e difference between predicted and Actual sales  
  
test\_data$error\_sales<-test\_data$sales - test\_data$predicted\_testing\_sales  
test\_data$error\_sales

## [1] -109.558741 31.819159 104.063985 19.819658 12.562897  
## [6] 15.452624 29.151682 15.669049 -7.995016 -35.320602  
## [11] 50.537770 -85.449341 78.517273 18.565694 61.349799  
## [16] 19.557091 -56.404631 22.476265 -1.881568 43.179018  
## [21] 30.576847 85.412705 -21.908074 62.144810 -5.137638  
## [26] -26.362851 27.467461 8.856374 2.056848 3.605707  
## [31] 40.506900 -32.358671 -22.584868 -11.811695 -9.388793  
## [36] 7.645988 -54.766433 -54.852368 29.600362 12.017470  
## [41] -42.183660 73.399293 28.565646 44.648782 -28.539102  
## [46] -55.339929 19.964912 -3.282171 4.167085 29.982026

# Adding observation No. Column n our dataset  
test\_data$obsno<-c(1:50)  
test\_data$obsno

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23  
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46  
## [47] 47 48 49 50

#comparing model predicted values with actual values of sales using the graph for train\_data  
#we can see the line curve as below  
plot(test\_data$sales,type="l",lty=1.8,col="orange")  
lines(test\_data$predicted\_testing\_sales,type="l",col="brown")



"brown line: Actual sales in testing dataset,blue line : Predicted sales by model fit for testing datase"

## [1] "brown line: Actual sales in testing dataset,blue line : Predicted sales by model fit for testing datase"