Group Project 9660

Group 2

05-10-2023

Install packages

```
#install.packages("dplyr")
#install.packages("stargazer")
#install.packages("rmarkdown")
#install.packages("stargazer")
#install.packages("car")
#install.packages("ggplot2")
#install.packages("rlang", dependencies = TRUE)
#install.packages("tinytex")
#tinytex::install_tinytex()
#install.packages("stats")
#install.packages("psych")
#install.packages("GGally")
#install.packages("tidyverse")
#install.packages("tidymodels")
#install.packages("forcats")
#install.packages("corrplot")
#install.packages("mgcv")
```

Original Data preview

```
# Load the dataset
Original_Video.Game <- read.csv("Video_Games_Sales_as_at_22_Dec_2016.csv")

# Filter the data for years between 2000 and 2016
Original_Video.Game <- Original_Video.Game %>%
filter(Year_of_Release >= 2000 & Year_of_Release <= 2016)

# Print the summary of the dataset
summary(Original_Video.Game)
##      Name             Platform         Year_of_Release       Genre
Publisher            NA_Sales          EU_Sales          JP_Sales
##  Length:14470       Length:14470       Length:14470       Length:14470
Length:14470       Min.   : 0.0000   Min.   : 0.0000   Min.   :0.00000
##  Class :character   Class :character   Class :character   Class :character
Class :character   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:0.00000
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
Mode  :character   Median : 0.0800   Median : 0.0200   Median :0.00000
##
```

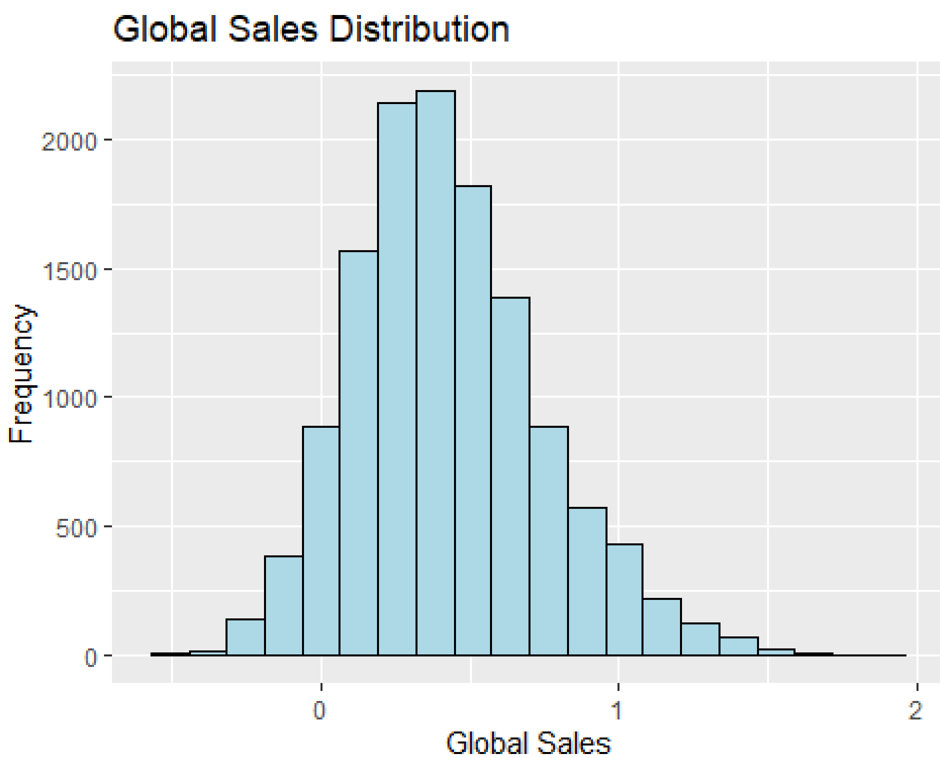
```
Mean   : 0.2439   Mean   : 0.1441   Mean   :0.05638
##
3rd Qu.: 0.2300   3rd Qu.: 0.1100   3rd Qu.:0.03000
##
Max.   :41.3600   Max.   :28.9600   Max.   :6.50000
##
##   Other_Sales        Global_Sales     Critic_Score    Critic_Count
User_Score          User_Count       Developer            Rating
##  Min.   : 0.00000   Min.   : 0.010   Min.   :13.00   Min.   :  3.0
Length:14470       Min.   :    4.0   Length:14470       Length:14470
##  1st Qu.: 0.00000   1st Qu.: 0.060   1st Qu.:60.00   1st Qu.: 12.0   Class
:character   1st Qu.:   10.0   Class :character   Class :character
##  Median : 0.01000   Median : 0.160   Median :71.00   Median : 22.0   Mode
:character   Median :   24.0   Mode  :character   Mode  :character
##  Mean   : 0.05032   Mean   : 0.495   Mean   :68.86   Mean   : 26.6
Mean   :  161.5
##  3rd Qu.: 0.04000   3rd Qu.: 0.440   3rd Qu.:79.00   3rd Qu.: 37.0
3rd Qu.:   81.0
##  Max.   :10.57000   Max.   :82.530   Max.   :98.00   Max.   :113.0
Max.   :10665.0
##                                      NA's   :6583    NA's   :6583
NA's   :7099

# Count the number of missing values in each column
original_missing_values <- summarise_all(Original_Video.Game, list(~
sum(is.na(.))))
print(original_missing_values)
##   Name Platform Year_of_Release Genre Publisher NA_Sales EU_Sales JP_Sales
Other_Sales Global_Sales Critic_Score Critic_Count User_Score User_Count
## 1    0        0               0     0         0        0        0        0
0            0         6583         6583          0       7099
##   Developer Rating
## 1         0      0

# Print the dimensions of the dataset
dim(Original_Video.Game)
## [1] 14470    16

# View the dataset in a separate window
View(Original_Video.Game)

# Apply a function to return the class of each column
sapply(Original_Video.Game,class)
##            Name        Platform Year_of_Release           Genre
Publisher        NA_Sales        EU_Sales        JP_Sales     Other_Sales
##     "character"     "character"     "character"     "character"
"character"       "numeric"       "numeric"       "numeric"       "numeric"
##    Global_Sales    Critic_Score    Critic_Count      User_Score
User_Count       Developer          Rating
```

```
##        "numeric"        "integer"        "integer"      "character" 
"integer"      "character"      "character"

# Print the structure of the dataset
str(Original_Video.Game)
## 'data.frame':    14470 obs. of  16 variables:
##  $ Name           : chr  "Wii Sports" "Mario Kart Wii" "Wii Sports Resort" 
"New Super Mario Bros." ...
##  $ Platform       : chr  "Wii" "Wii" "Wii" "DS" ...
##  $ Year_of_Release: chr  "2006" "2008" "2009" "2006" ...
##  $ Genre          : chr  "Sports" "Racing" "Sports" "Platform" ...
##  $ Publisher      : chr  "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ NA_Sales       : num  41.4 15.7 15.6 11.3 14 ...
##  $ EU_Sales       : num  28.96 12.76 10.93 9.14 9.18 ...
##  $ JP_Sales       : num  3.77 3.79 3.28 6.5 2.93 4.7 1.93 4.13 3.6 0.24 
...
##  $ Other_Sales    : num  8.45 3.29 2.95 2.88 2.84 2.24 2.74 1.9 2.15 1.69 
...
##  $ Global_Sales   : num  82.5 35.5 32.8 29.8 28.9 ...
##  $ Critic_Score   : int  76 82 80 89 58 87 NA 91 80 61 ...
##  $ Critic_Count   : int  51 73 73 65 41 80 NA 64 63 45 ...
##  $ User_Score     : chr  "8" "8.3" "8" "8.5" ...
##  $ User_Count     : int  322 709 192 431 129 594 NA 464 146 106 ...
##  $ Developer      : chr  "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ Rating         : chr  "E" "E" "E" "E" ...

# reference the original variables in the data frame
attach(Original_Video.Game)
```

Original Data statistics

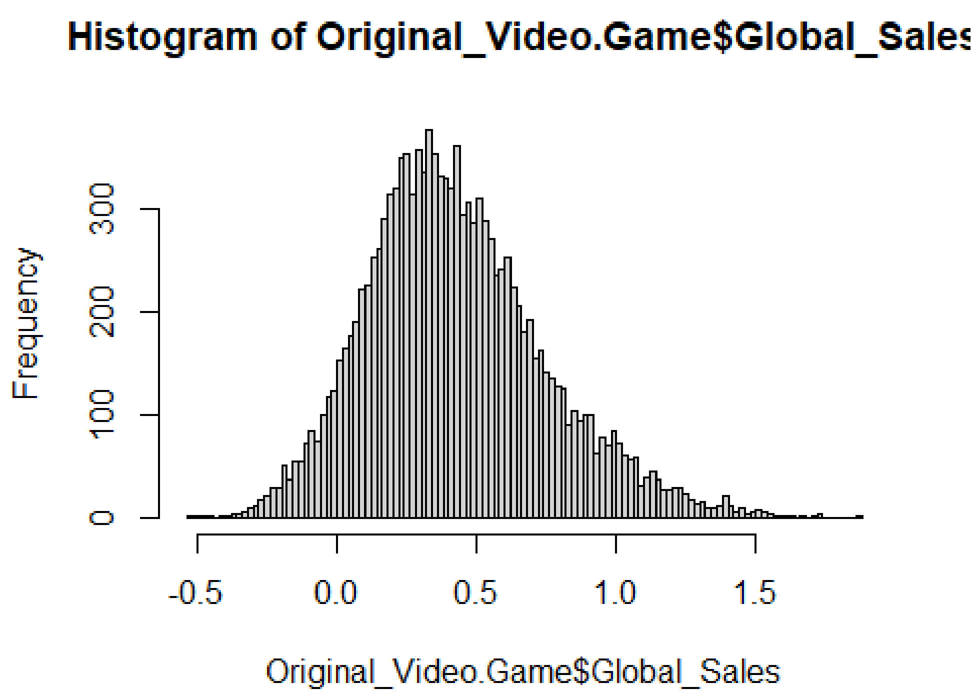
```
# Remove outliers using the IQR method for Global_Sales
q1 <- quantile(Original_Video.Game$Global_Sales, 0.25, na.rm = TRUE)
q3 <- quantile(Original_Video.Game$Global_Sales, 0.75, na.rm = TRUE)
iqr <- q3 - q1
upper <- q3 + 1.5*iqr
lower <- q1 - 1.5*iqr
Original_Video.Game <- subset(Original_Video.Game, Global_Sales >= lower & 
Global_Sales <= upper)

# calculate mean for global sales
original_global_sales_mean <- mean(Original_Video.Game$Global_Sales)
print(original_global_sales_mean)
## [1] 0.2142826

# Calculate variance of a numeric variable
original_variance <- var(Original_Video.Game$Global_Sales, na.rm = TRUE)
print(original_variance)
## [1] 0.05088653
```

```
# Calculate standard deviation of a numeric variable
original_sd <- sd(Original_Video.Game$Global_Sales, na.rm = TRUE)
print(original_sd)
## [1] 0.2255804

# Set the seed for reproducibility
set.seed(3)

# Add random normal values to the Global_Sales variable
Original_Video.Game$Global_Sales <- Original_Video.Game$Global_Sales +
rnorm(nrow(Original_Video.Game), mean = original_global_sales_mean , sd =
original_sd)

# Plot a histogram of the Global_Sales variable
ggplot(Original_Video.Game, aes(x = Global_Sales)) +
  geom_histogram(bins = 20, color = "black", fill = "lightblue") +
  labs(title = "Global Sales Distribution", x = "Global Sales", y =
"Frequency")
```

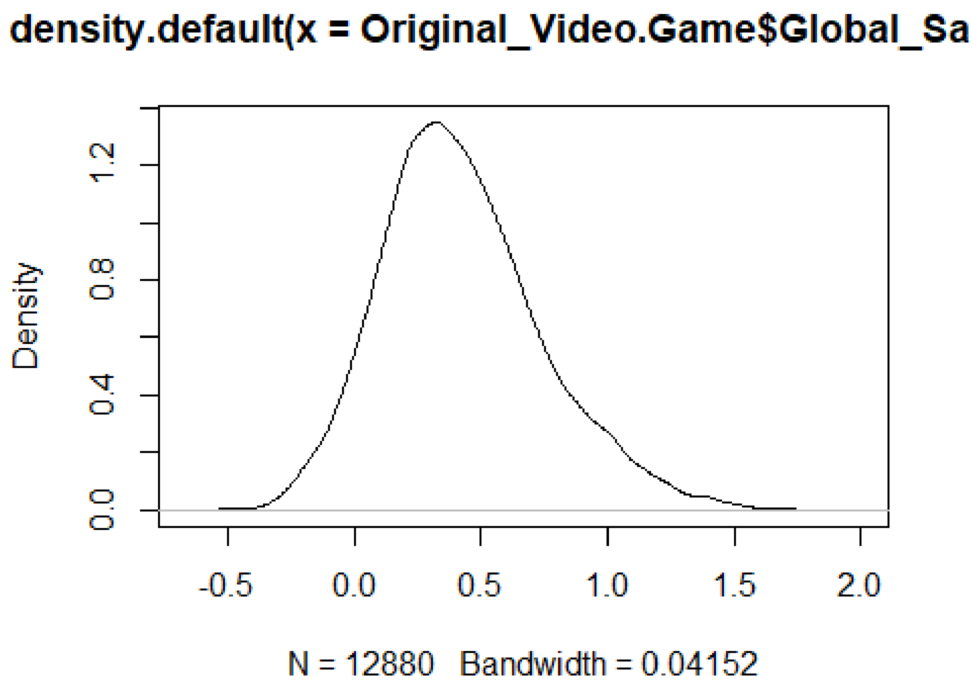



```
# Calculate correlation between numeric variables
original_correlations <- cor(Original_Video.Game[,
sapply(Original_Video.Game, is.numeric)], use = "complete.obs")
print(original_correlations)
##                     NA_Sales   EU_Sales      JP_Sales Other_Sales
```

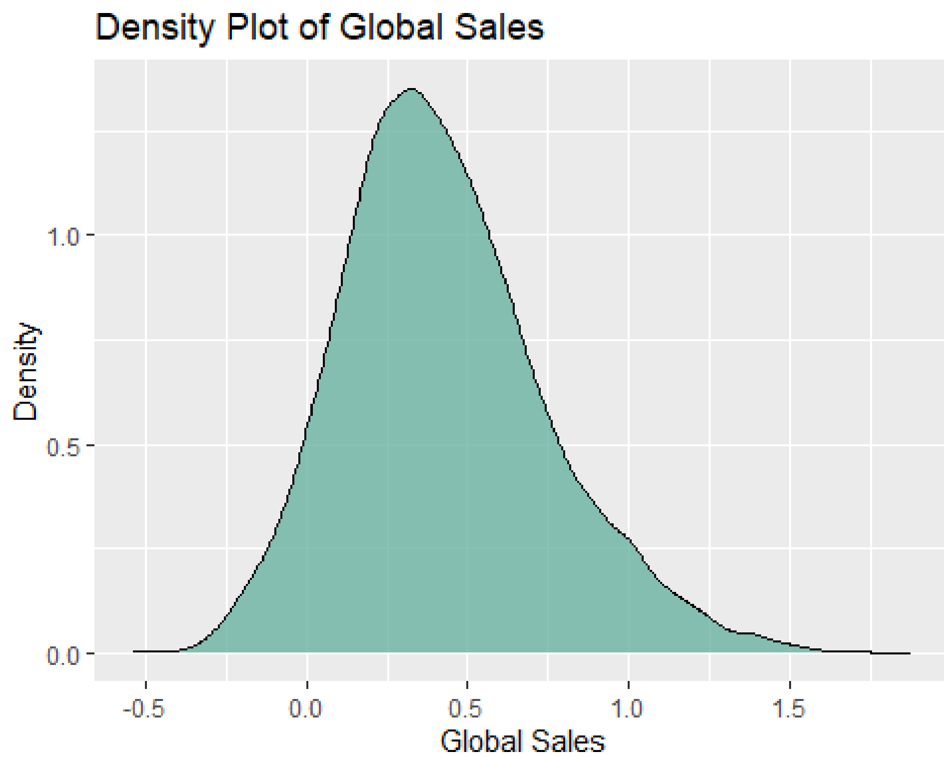
```
Global_Sales Critic_Score Critic_Count  User_Count
## NA_Sales       1.0000000000 0.3379415  0.0005509181  0.49594706
0.61576357   0.17868864    0.2157053 -0.02426960
## EU_Sales       0.3379415234 1.0000000  0.0128536964  0.63192318
0.54957274   0.20083235    0.2654758  0.27087742
## JP_Sales       0.0005509181 0.0128537  1.0000000000  0.04906703
0.21069477   0.08123742    0.1184602 -0.01777846
## Other_Sales    0.4959470639 0.6319232  0.0490670289  1.00000000
0.53433409   0.13560410    0.2259492  0.09626510
## Global_Sales   0.6157635723 0.5495727  0.2106947663  0.53433409
1.00000000   0.16508905    0.2379256  0.08878281
## Critic_Score   0.1786886365 0.2008323  0.0812374237  0.13560410
0.16508905   1.00000000    0.3306252  0.21331815
## Critic_Count   0.2157052746 0.2654758  0.1184601987  0.22594917
0.23792564   0.33062523    1.0000000  0.22654774
## User_Count    -0.0242695983 0.2708774 -0.0177784646  0.09626510
0.08878281   0.21331815    0.2265477  1.00000000

# Plot histogram of a numeric variable
hist(Original_Video.Game$Global_Sales, breaks = 100)
```




```
# Plot a density plot of Global_Sales
plot(density(Original_Video.Game$Global_Sales))
```




```
# Calculate the mean for numeric columns
sapply(Original_Video.Game[sapply(Original_Video.Game, is.numeric)], mean)
##     NA_Sales     EU_Sales     JP_Sales  Other_Sales Global_Sales 
Critic_Score Critic_Count   User_Count 
##   0.11042469   0.05381988   0.03089674   0.01882531   0.42550297 
NA           NA           NA 

# Calculate the standard deviation for numeric columns
sapply(Original_Video.Game[sapply(Original_Video.Game, is.numeric)], sd)
##     NA_Sales     EU_Sales     JP_Sales  Other_Sales Global_Sales 
Critic_Score Critic_Count   User_Count 
##   0.14206790   0.09111080   0.08295832   0.03453205   0.31994910 
NA           NA           NA 

# Create a density plot of Global_Sales
ggplot(Original_Video.Game, aes(Global_Sales)) +
  geom_density(fill = "#69b3a2", alpha = 0.8) +
  ggtitle("Density Plot of Global Sales") +
  xlab("Global Sales") +
  ylab("Density")
```



Revised Data Preview

```
# Remove missing values
Video.Game <- na.omit(Original_Video.Game)

# Convert categorical variables to factors
Video.Game$Name <- as.factor(Video.Game$Name)
Video.Game$Platform <- as.factor(Video.Game$Platform)
Video.Game$Genre <- as.factor(Video.Game$Genre)
Video.Game$Publisher <- as.factor(Video.Game$Publisher)
Video.Game$Developer <- as.factor(Video.Game$Developer)
Video.Game$Rating <- as.factor(Video.Game$Rating)

#Convert numeric variables to numeric
Video.Game$Year_of_Release <- as.numeric(Video.Game$Year_of_Release)
Video.Game$Critic_Score <- as.numeric(Video.Game$Critic_Score)
Video.Game$Critic_Count <- as.numeric(Video.Game$Critic_Count)
Video.Game$User_Score <- as.numeric(Video.Game$User_Score)
Video.Game$User_Count <- as.numeric(Video.Game$User_Count)

# Filter the data for years between 2006 and 2016
Video.Game <- Video.Game %>%
filter(Year_of_Release >= 2000 & Year_of_Release <= 2016)

# Print the summary of the dataset
```

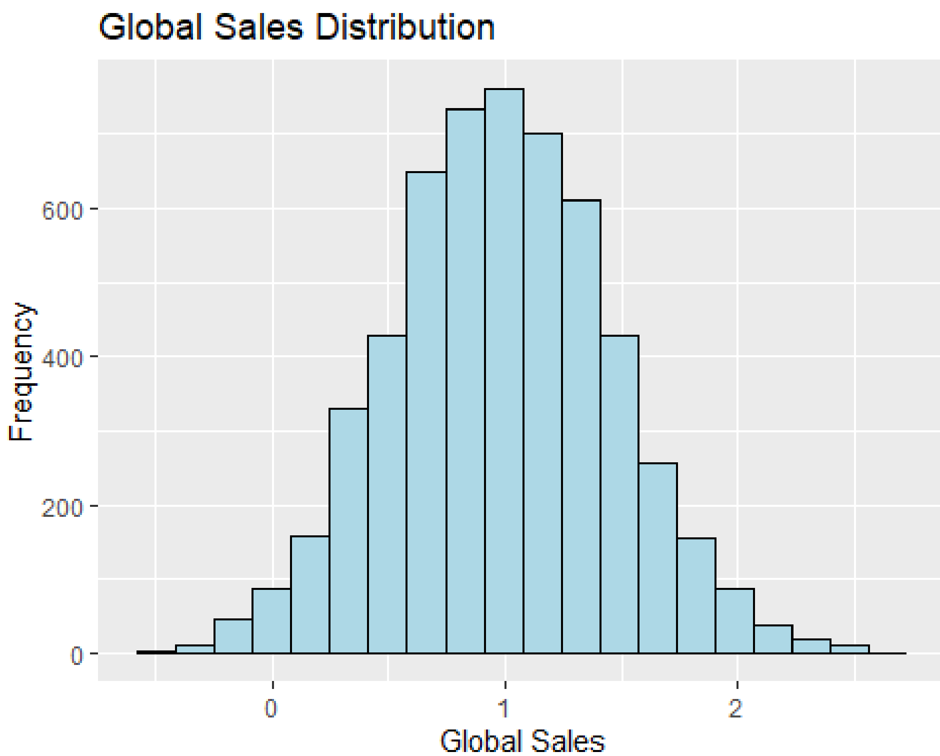
```
summary(Video.Game)
##                                         Name         Platform
Year_of_Release          Genre                            Publisher
##  Harry Potter and the Order of the Phoenix:   7   PS2    : 880   Min.
:2000    Action      :1341   Electronic Arts            : 673
##  Spider-Man 3                             :   7   X360   : 651   1st
Qu.:2004    Sports      : 755   Ubisoft                    : 412
##  Terraria                                 :   7   PC     : 633   Median
:2007    Shooter     : 663   Activision                 : 365
##  Tomb Raider: Legend                      :   7   PS3    : 564   Mean
:2008    Role-Playing: 581   THQ                        : 263
##  FIFA World Cup Germany 2006              :   6   XB     : 519   3rd
Qu.:2011    Racing      : 467   Sega                       : 236
##  Ghostbusters: The Video Game             :   6   DS     : 385   Max.
:2016    Platform    : 310   Sony Computer Entertainment: 221
##  (Other)                                  :5512   (Other):1920
(Other)     :1435   (Other)                    :3382
##     NA_Sales          EU_Sales           JP_Sales         Other_Sales
Global_Sales      Critic_Score    Critic_Count      User_Score
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
Min.   :-0.4531   Min.   :13.00   Min.   :  3.00   Min.   :0.500
##  1st Qu.:0.0400   1st Qu.:0.01000   1st Qu.:0.00000   1st Qu.:0.01000
1st Qu.: 0.2546   1st Qu.:60.00   1st Qu.: 13.00   1st Qu.:6.300
##  Median :0.1100   Median :0.04000   Median :0.00000   Median :0.01500
Median : 0.4704   Median :70.00   Median : 22.00   Median :7.400
##  Mean   :0.1582   Mean   :0.08054   Mean   :0.02185   Mean   :0.02789
Mean   : 0.5009   Mean   :68.03   Mean   : 25.64   Mean   :7.083
##  3rd Qu.:0.2200   3rd Qu.:0.11000   3rd Qu.:0.00000   3rd Qu.:0.04000
3rd Qu.: 0.7143   3rd Qu.:78.00   3rd Qu.: 35.00   3rd Qu.:8.200
##  Max.   :0.9400   Max.   :0.93000   Max.   :0.74000   Max.   :0.54000
Max.   : 1.8782   Max.   :96.00   Max.   :106.00   Max.   :9.600
##
##    User_Count             Developer     Rating
##  Min.   :    4.0   EA Canada   : 108       :  64
##  1st Qu.:    9.0   EA Sports   :  96   E   :1616
##  Median :   21.0   Capcom      :  91   E10+: 784
##  Mean   :  104.1   Ubisoft     :  83   M   :1091
##  3rd Qu.:   57.0   Konami      :  80   RP  :   1
##  Max.   :10665.0   Omega Force:  64   T   :1996
##                    (Other)    :5030

# Count the number of missing values in each column
revised_missing_values <- summarise_all(Video.Game, list(~ sum(is.na(.))))
print(revised_missing_values)
##   Name Platform Year_of_Release Genre Publisher NA_Sales EU_Sales JP_Sales
Other_Sales Global_Sales Critic_Score Critic_Count User_Score User_Count
## 1    0        0               0     0         0        0        0        0
0           0            0            0          0          0
##   Developer Rating
## 1         0      0
```

```
# Print the dimensions of the dataset
dim(Video.Game)
## [1] 5552   16

# View the dataset in a separate window
View(Video.Game)

# Apply a function to return the class of each column
sapply(Video.Game,class)
##            Name        Platform Year_of_Release           Genre
Publisher        NA_Sales        EU_Sales        JP_Sales     Other_Sales
##        "factor"        "factor"       "numeric"        "factor"
"factor"       "numeric"       "numeric"       "numeric"       "numeric"
##    Global_Sales    Critic_Score    Critic_Count      User_Score
User_Count       Developer          Rating
##       "numeric"       "numeric"       "numeric"       "numeric"
"numeric"        "factor"        "factor"

# Print the structure of the dataset
str(Video.Game)
## 'data.frame':    5552 obs. of  16 variables:
##  $ Name           : Factor w/ 3793 levels " Tales of Xillia 2",..: 3599
2064 34 3296 2254 185 2531 2432 3780 2408 ...
##  $ Platform       : Factor w/ 17 levels "3DS","DC","DS",..: 13 13 15 13 15
15 10 15 14 14 ...
##  $ Year_of_Release: num  2008 2007 2007 2007 2008 ...
##  $ Genre          : Factor w/ 12 levels "Action","Adventure",..: 5 3 10 1
1 1 5 1 1 3 ...
##  $ Publisher      : Factor w/ 256 levels "10TACLE Studios",..: 229 231 23
64 224 236 236 236 236 155 ...
##  $ NA_Sales       : num  0.51 0.44 0.69 0.45 0.65 0.58 0.22 0.54 0.52 0.5
...
##  $ EU_Sales       : num  0.4 0.46 0.04 0.46 0.22 0.34 0.64 0.34 0.36 0.26
...
##  $ JP_Sales       : num  0 0 0.22 0 0.05 0 0 0.02 0.05 0.17 ...
##  $ Other_Sales    : num  0.11 0.11 0.06 0.11 0.1 0.09 0.16 0.1 0.08 0.08
...
##  $ Global_Sales   : num  1.007 1.158 0.964 1.268 1.476 ...
##  $ Critic_Score   : num  51 74 80 64 81 72 90 81 77 76 ...
##  $ Critic_Count   : num  18 23 54 19 74 32 16 73 70 74 ...
##  $ User_Score     : num  3.6 8 7.9 6.1 8 7.3 8.5 7 7.7 8 ...
##  $ User_Count     : num  8 27 101 38 302 318 525 189 758 273 ...
##  $ Developer      : Factor w/ 1224 levels "","10tacle Studios,
Fusionsphere Systems",..: 481 28 832 356 1066 1128 1135 1136 1135 115 ...
##  $ Rating         : Factor w/ 6 levels "","E","E10+",..: 2 6 6 6 4 4 3 6 4
3 ...
##  - attr(*, "na.action")= 'omit' Named int [1:7328] 3 6 7 11 18 19 20 22 23
24 ...
```

```
##   ..- attr(*, "names")= chr [1:7328] "1593" "1596" "1597" "1601" ...

# reference the revised variables in the data frame
attach(Video.Game)
## The following objects are masked from Original_Video.Game:
##
##     Critic_Count, Critic_Score, Developer, EU_Sales, Genre, Global_Sales,
JP_Sales, NA_Sales, Name, Other_Sales, Platform, Publisher,
##     Rating, User_Count, User_Score, Year_of_Release
```

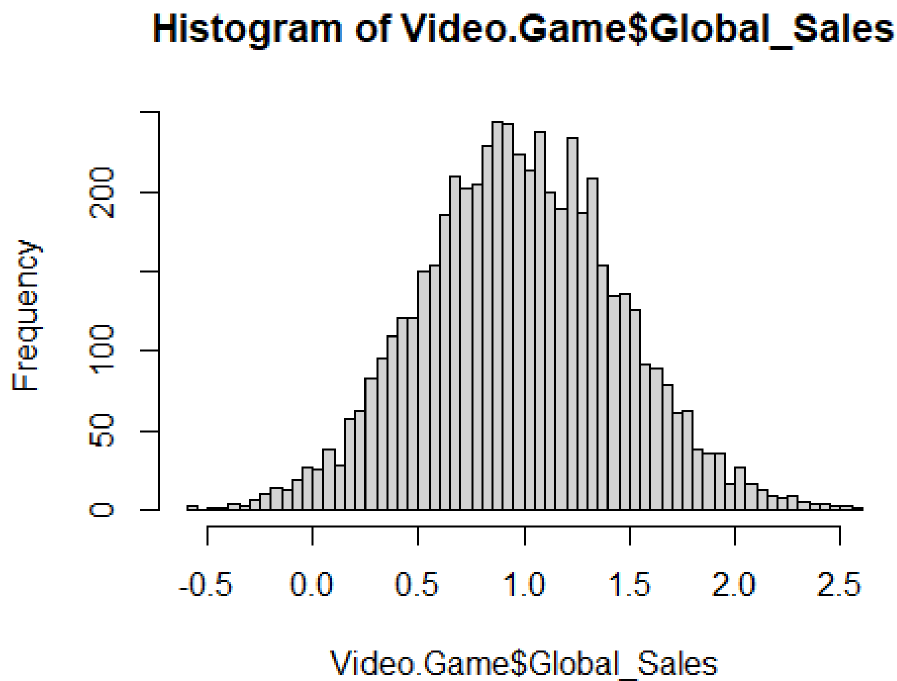
Revised Data statistics

```
# Remove outliers using the IQR method for Global_Sales
q1 <- quantile(Video.Game$Global_Sales, 0.25, na.rm = TRUE)
q3 <- quantile(Video.Game$Global_Sales, 0.75, na.rm = TRUE)
iqr <- q3 - q1
upper <- q3 + 1.5*iqr
lower <- q1 - 1.5*iqr
Video.Game <- subset(Video.Game, Global_Sales >= lower & Global_Sales <=
upper)

# calculate mean for global sales
revised_global_sales_mean <- mean(Video.Game$Global_Sales)
print(revised_global_sales_mean)
## [1] 0.4935089

# Calculate variance of a numeric variable
revised_variance <- var(Video.Game$Global_Sales, na.rm = TRUE)
print(revised_variance)
## [1] 0.1088451

# Calculate standard deviation of a numeric variable
revised_sd <- sd(Video.Game$Global_Sales, na.rm = TRUE)
print(revised_sd)
## [1] 0.3299168

# Set the seed for reproducibility
set.seed(3)

# Add random normal values to the Global_Sales variable
Video.Game$Global_Sales <- Video.Game$Global_Sales + rnorm(nrow(Video.Game),
mean = revised_global_sales_mean , sd = revised_sd)

# Plot a histogram of the Global_Sales variable
ggplot(Video.Game, aes(x = Global_Sales)) +
  geom_histogram(bins = 20, color = "black", fill = "lightblue") +
  labs(title = "Global Sales Distribution", x = "Global Sales", y =
"Frequency")
```

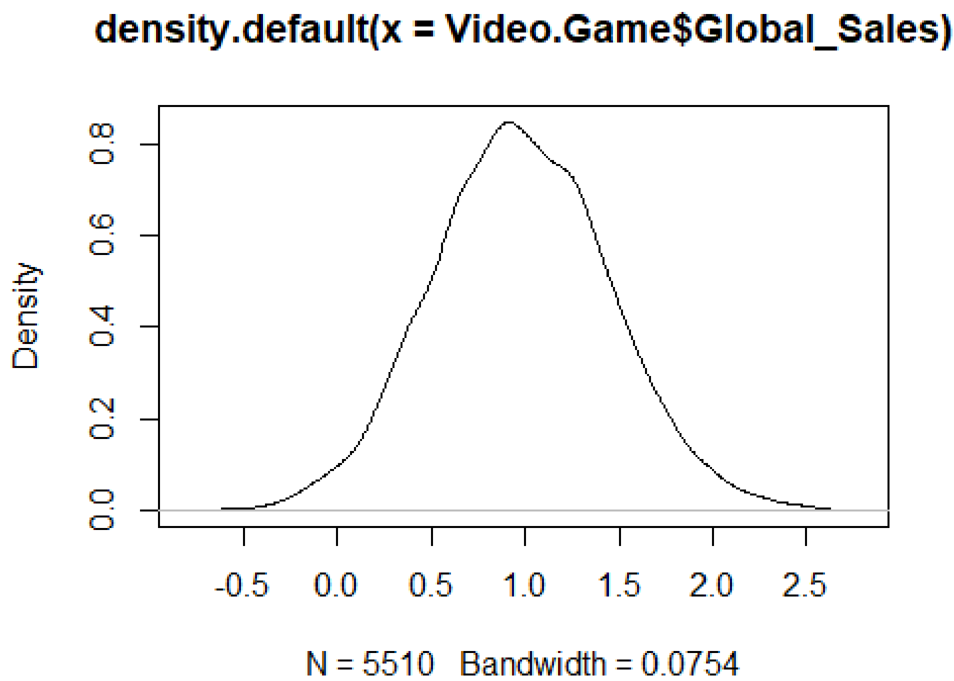



```
# Calculate correlation between numeric variables
revised_correlations <- cor(Video.Game[, sapply(Video.Game, is.numeric)], use
= "complete.obs")
print(revised_correlations)
##                  Year_of_Release     NA_Sales   EU_Sales     JP_Sales
Other_Sales Global_Sales Critic_Score Critic_Count  User_Score  User_Count
## Year_of_Release     1.000000000 -0.101666790 0.09731540  0.050232518
0.10942815  -0.01339470  0.002958213    0.1356116 -0.22342259  0.18560796
## NA_Sales           -0.101666790  1.000000000 0.33073898  0.003311067
0.49309080   0.43473854  0.172473322    0.2119618  0.07768621 -0.03019462
## EU_Sales            0.097315396  0.330738982 1.00000000  0.011276202
0.62099581   0.37426514  0.194451819    0.2646900  0.03791376  0.26882240
## JP_Sales            0.050232518  0.003311067 0.01127620  1.000000000
0.04698147   0.15200756  0.080851048    0.1195634  0.13178419 -0.01729629
## Other_Sales         0.109428152  0.493090798 0.62099581  0.046981468
1.00000000   0.37918641  0.128752641    0.2231947  0.03404031  0.08994499
## Global_Sales       -0.013394702  0.434738538 0.37426514  0.152007564
0.37918641   1.00000000  0.103088284    0.1752382  0.04843587  0.07126432
## Critic_Score        0.002958213  0.172473322 0.19445182  0.080851048
0.12875264   0.10308828  1.000000000    0.3292032  0.58866427  0.21316760
## Critic_Count        0.135611580  0.211961787 0.26468996  0.119563431
0.22319474   0.17523816  0.329203202    1.0000000  0.18925007  0.23206714
## User_Score         -0.223422592  0.077686207 0.03791376  0.131784191
0.03404031   0.04843587  0.588664269    0.1892501  1.00000000  0.02890343
## User_Count          0.185607961 -0.030194616 0.26882240 -0.017296294
```

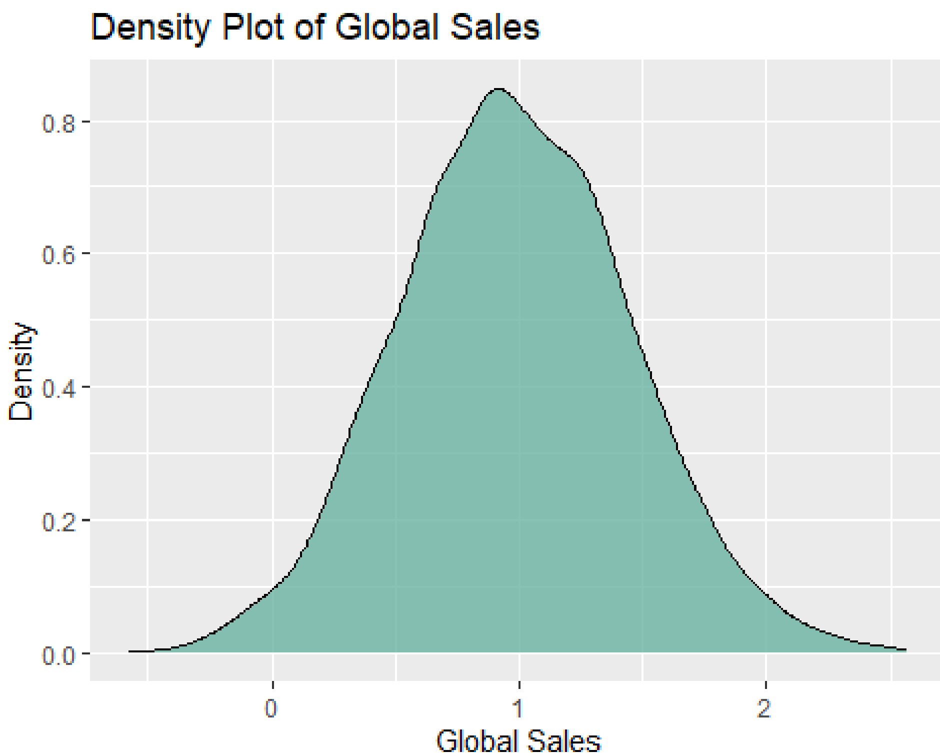
```
0.08994499   0.07126432  0.213167603    0.2320671  0.02890343  1.00000000

# Plot histogram of a numeric variable
hist(Video.Game$Global_Sales, breaks = 100)
```




```
# Plot a density plot of Global_Sales
plot(density(Video.Game$Global_Sales))
```

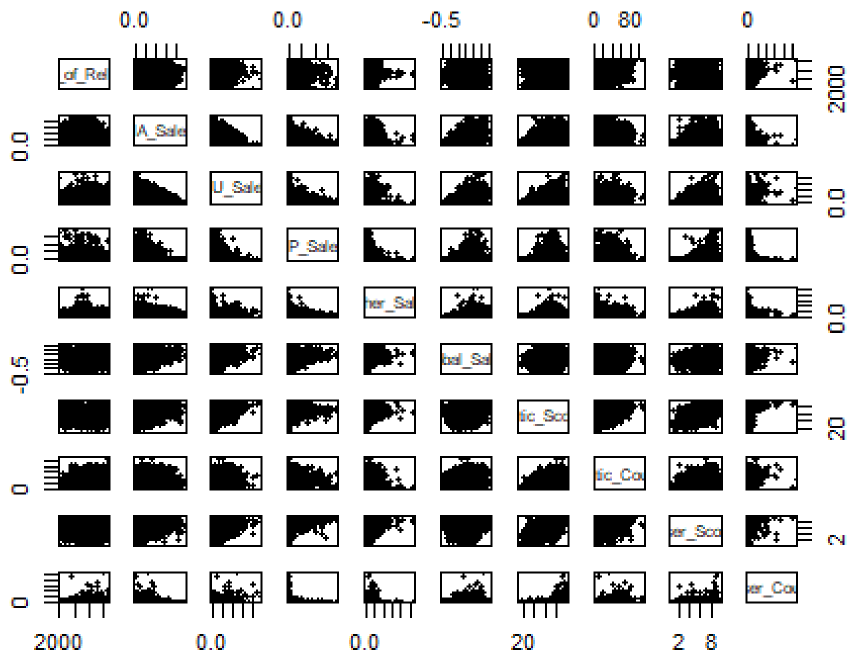



```
# Calculate the mean for numeric columns
sapply(Video.Game[sapply(Video.Game, is.numeric)], mean)
## Year_of_Release         NA_Sales         EU_Sales         JP_Sales
Other_Sales     Global_Sales     Critic_Score     Critic_Count       User_Score
##    2.007553e+03     1.559837e-01     7.898004e-02     2.173684e-02
2.738475e-02     9.812912e-01     6.796225e+01     2.557423e+01     7.081579e+00
##      User_Count
##    1.027194e+02

# Calculate the standard deviation for numeric columns
sapply(Video.Game[sapply(Video.Game, is.numeric)], sd)
## Year_of_Release         NA_Sales         EU_Sales         JP_Sales
Other_Sales     Global_Sales     Critic_Score     Critic_Count       User_Score
##      4.11060136       0.15507885       0.10081054       0.06792289
0.03595749       0.46918224      13.57073499      16.22036452       1.46796618
##      User_Count
##    388.33043602

# Create a density plot of Global_Sales
ggplot(Video.Game, aes(Global_Sales)) +
  geom_density(fill = "#69b3a2", alpha = 0.8) +
  ggtitle("Density Plot of Global Sales") +
  xlab("Global Sales") +
  ylab("Density")
```

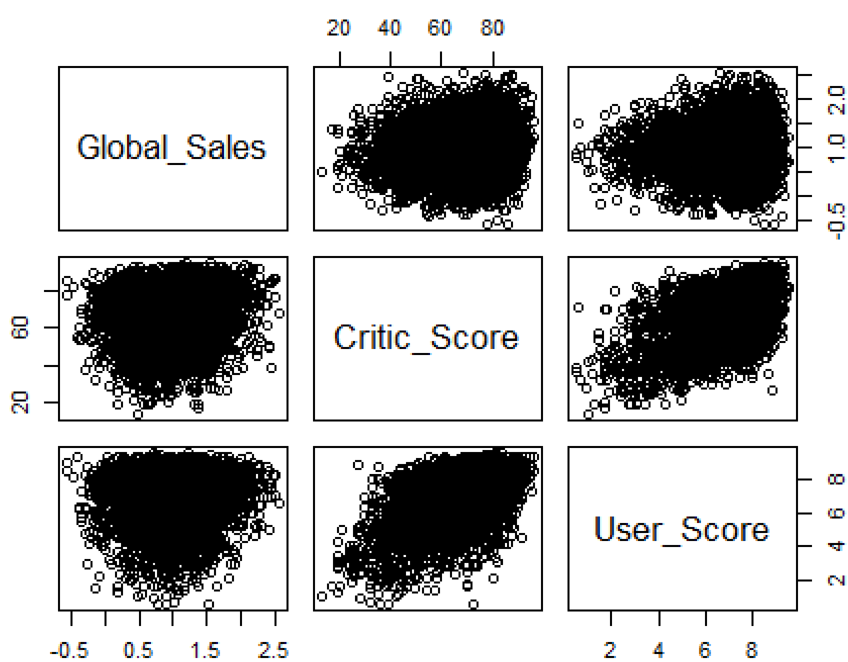


Revised Data statistics

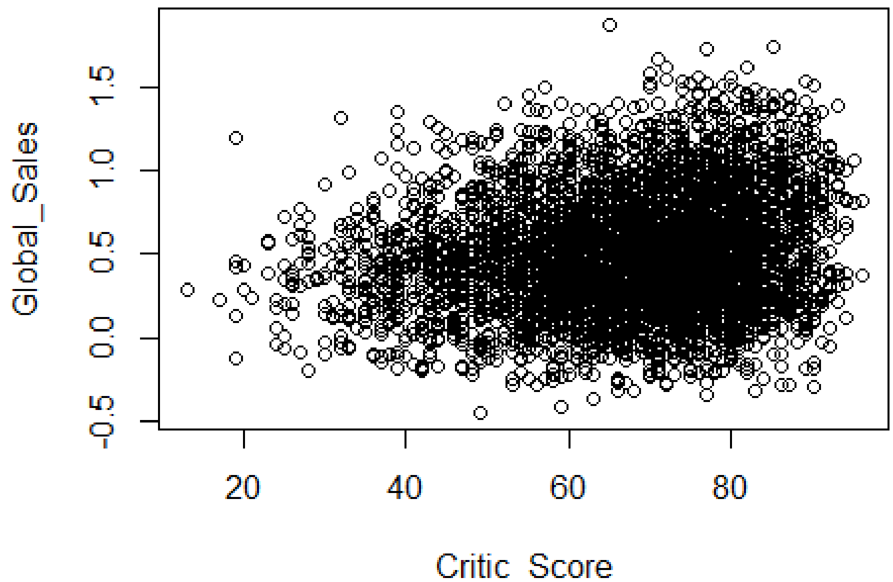
```
# Select only the numeric columns
num_data <- Video.Game %>% select_if(is.numeric)

# Create scatterplot matrix using pairs() function
pairs(num_data, pch = 20)
```



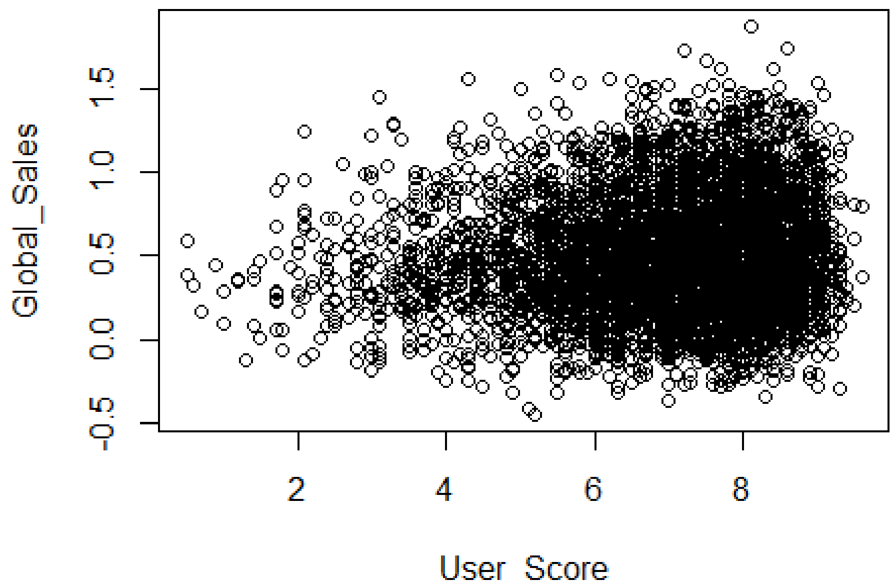
```
# Create scatterplot matrix using subset of variables and pairs() function
pairs(~Global_Sales + Critic_Score + User_Score, Video.Game)
```



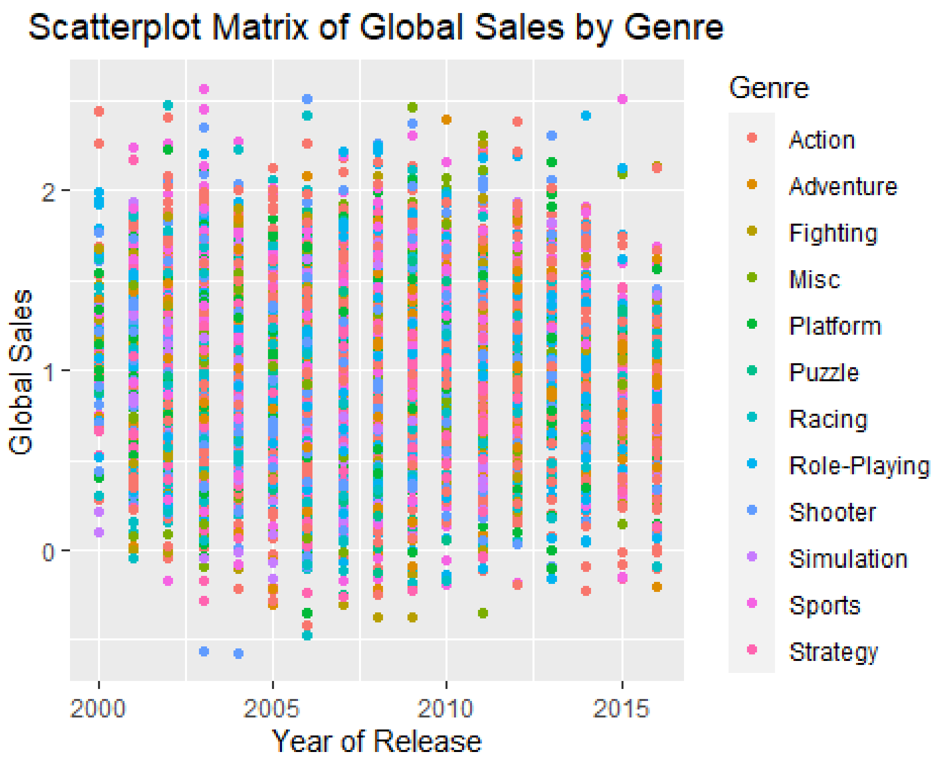
```
# Plot using a subset of variables
plot(Critic_Score,Global_Sales)
```



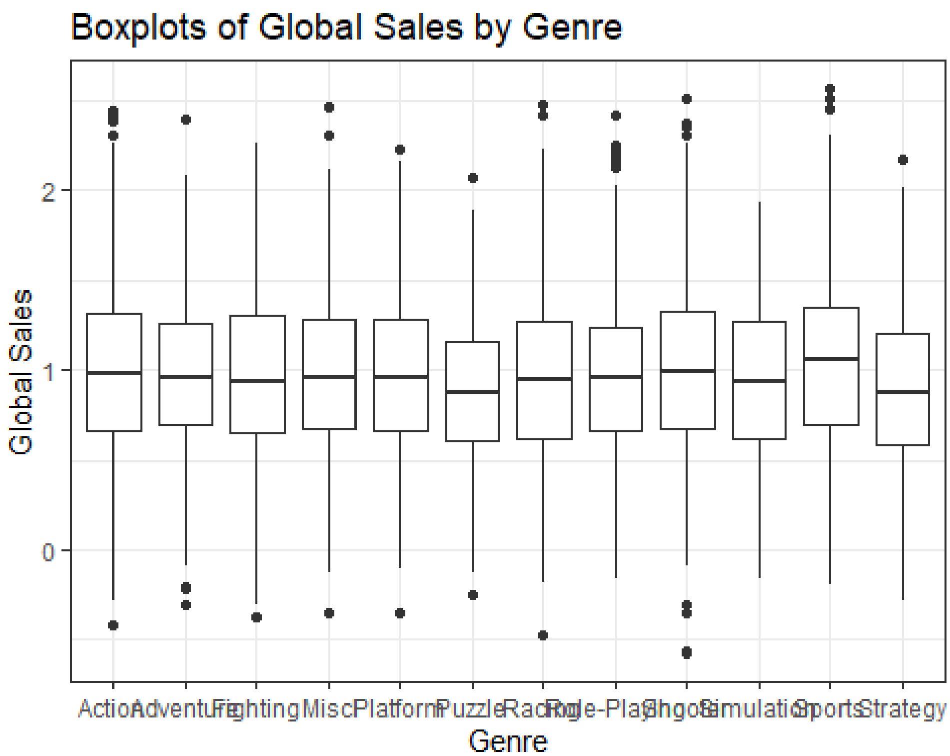

```
plot(User_Score,Global_Sales)
```



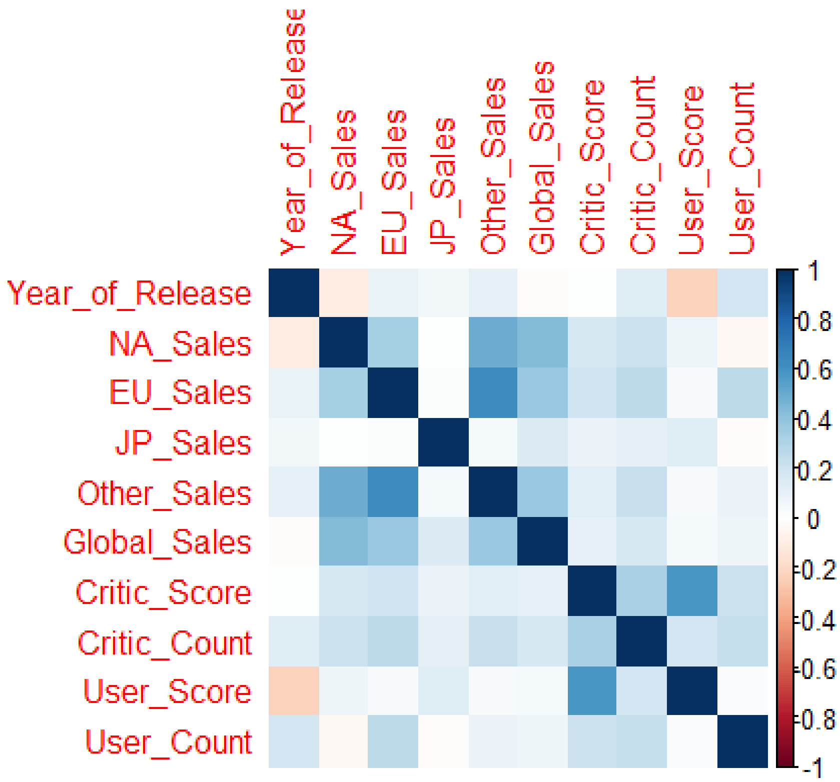

```
# Create scatterplot matrix using ggplot() function
ggplot(Video.Game, aes(x = Year_of_Release, y = Global_Sales, color = Genre))
+
  geom_point() +
  labs(x = "Year of Release", y = "Global Sales", color = "Genre") +
  ggtitle("Scatterplot Matrix of Global Sales by Genre") +
  theme(plot.title = element_text(hjust = 0.5))
```



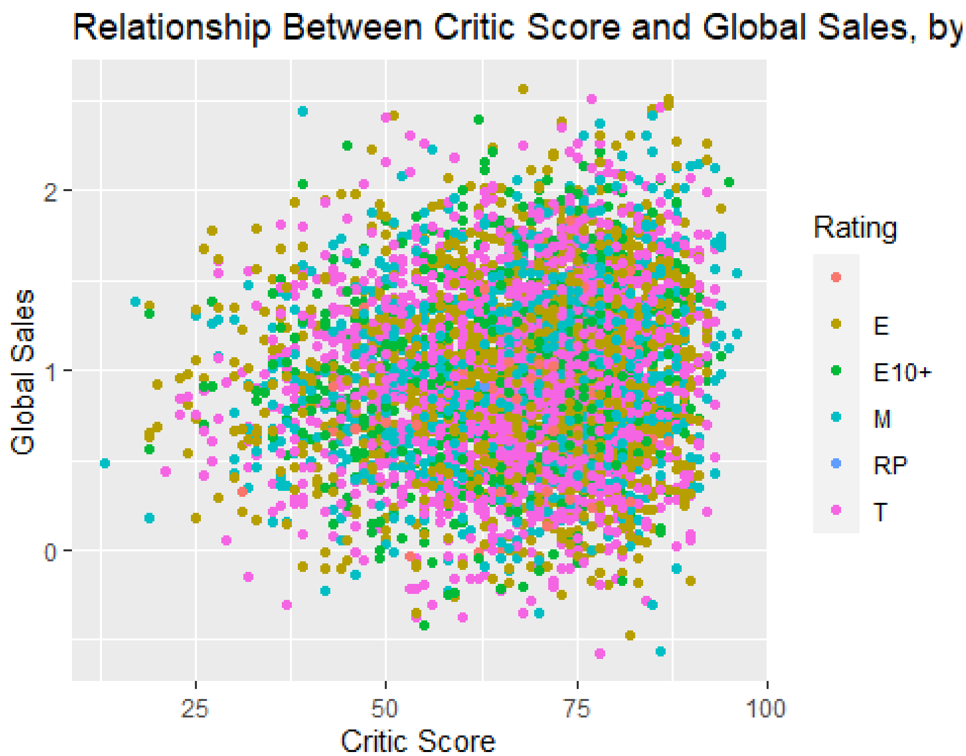

```r
# Generate boxplots for numeric variables
ggplot(Video.Game, aes(x = Genre, y = Global_Sales)) +
  geom_boxplot() +
  labs(x = "Genre", y = "Global Sales") +
  ggtitle("Boxplots of Global Sales by Genre") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_bw()
```




```r
# Pairwise correlation between all numeric variables:
corrplot(cor(Video.Game[, sapply(Video.Game, is.numeric)], use =
"complete.obs"), method = "color")
```




```
# Create scatterplot matrix using ggplot() function
ggplot(Video.Game, aes(x = Critic_Score, y = Global_Sales, color = Rating)) +
  geom_point() +
  ggtitle("Relationship Between Critic Score and Global Sales, by Rating") +
  xlab("Critic Score") +
  ylab("Global Sales")
```

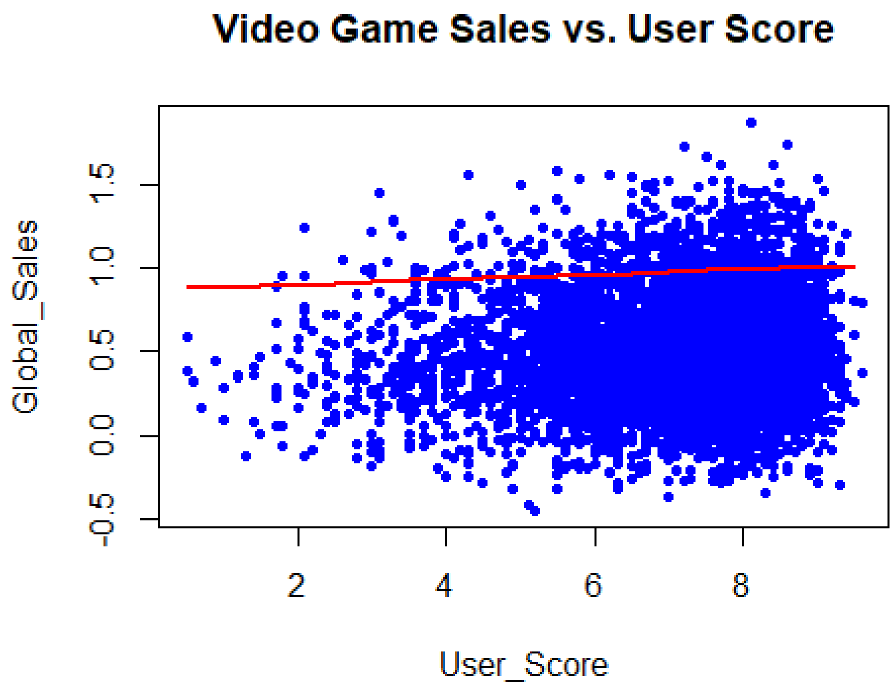



```
stargazer(Video.Game, title = "Summary Statistics",
          type = "text", summary.stat = c("mean", "sd", "min", "max"))
##
## Summary Statistics
## ==================================================
## Statistic         Mean    St. Dev.  Min    Max
## --------------------------------------------------
## Year_of_Release 2,007.553  4.111   2,000  2,016
## NA_Sales          0.156    0.155   0.000  0.940
## EU_Sales          0.079    0.101   0.000  0.930
## JP_Sales          0.022    0.068   0.000  0.740
## Other_Sales       0.027    0.036   0.000  0.540
## Global_Sales      0.981    0.469   -0.575 2.568
## Critic_Score     67.962   13.571     13     96
## Critic_Count     25.574   16.220     3     106
## User_Score        7.082    1.468   0.500  9.600
## User_Count       102.719  388.330    4    10,665
## --------------------------------------------------
```

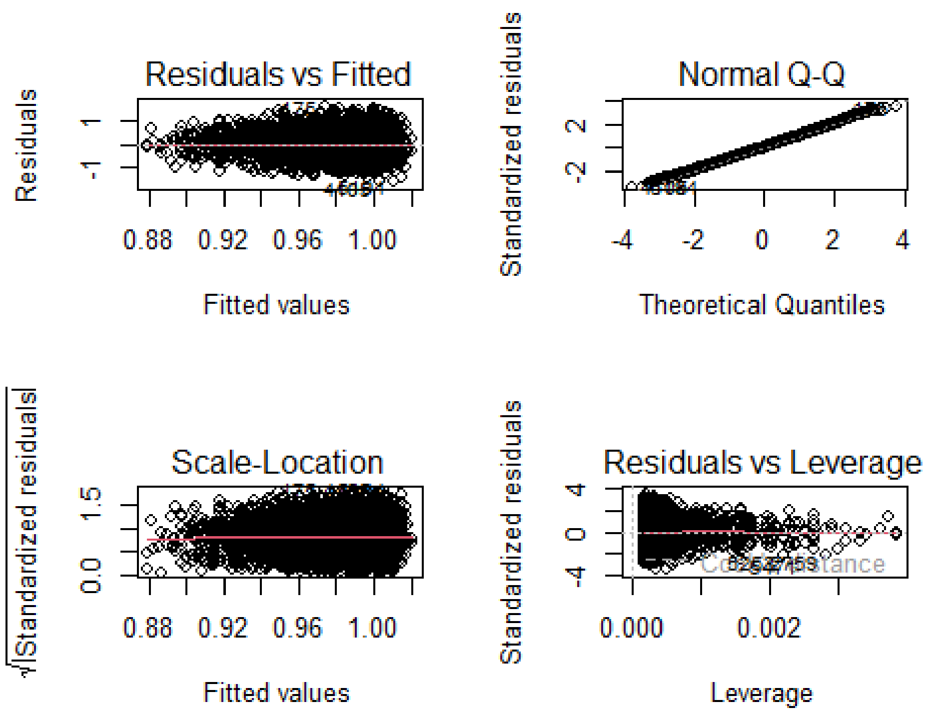
Simple Linear Regression - User Score

```
# Fit a simple linear regression model
lm.fit <- lm(Global_Sales ~ User_Score, data = Video.Game)

# Get summary of the model
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Global_Sales ~ User_Score, data = Video.Game)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5780 -0.3167 -0.0091  0.3170  1.5938
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.871663   0.031109  28.020  < 2e-16 ***
## User_Score  0.015481   0.004301   3.599 0.000322 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4687 on 5508 degrees of freedom
## Multiple R-squared:  0.002346,   Adjusted R-squared:  0.002165
## F-statistic: 12.95 on 1 and 5508 DF,  p-value: 0.0003223

# Create a sequence of User_Score values for prediction
x_seq <- seq(from = min(Video.Game$User_Score), to =
max(Video.Game$User_Score), by = 1)

# Get the predicted values from the model
y_hat <- predict(lm.fit, newdata = list(User_Score = x_seq))

# Calculate the residuals
residuals <- Video.Game$Global_Sales - y_hat

# Calculate the RMSE
RMSE <- sqrt(mean(residuals^2))

# Print the RMSE
cat("The RMSE of User Score for the simple regression model is:", RMSE, "\n")
## The RMSE of User Score for the simple regression model is: 0.4720381

# We will now plot Global_Sales and User_Score along with the least squares
regression line
plot(User_Score, Global_Sales, pch = 20, col = "blue", main = "Video Game
Sales vs. User Score")

# Plot the simple regression line
lines(x_seq, y_hat, col = "red", lwd = 2)
```



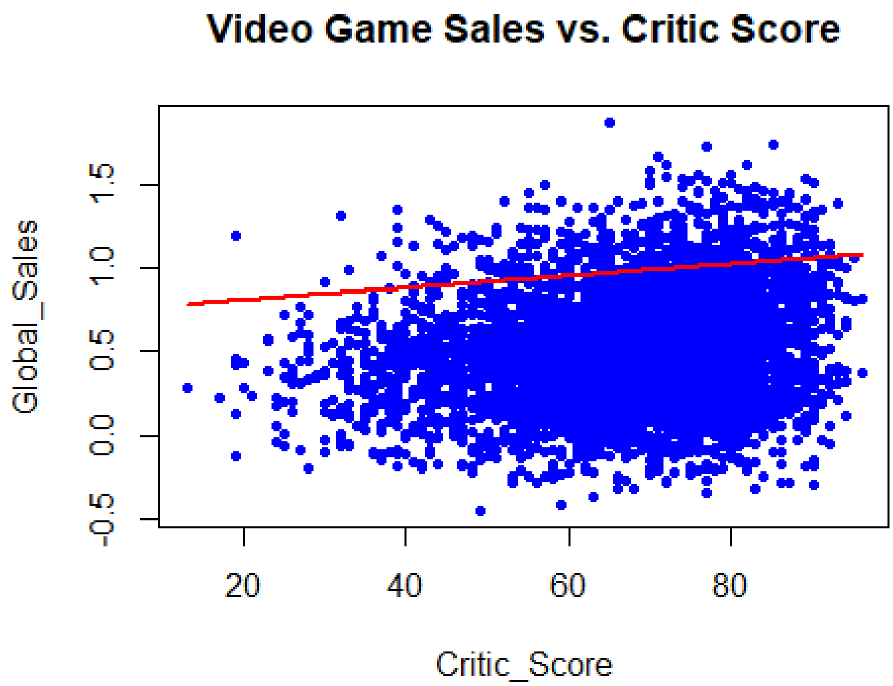

```
# Produce four diagnostic plots
par(mfrow = c(2,2))
plot(lm.fit)
```



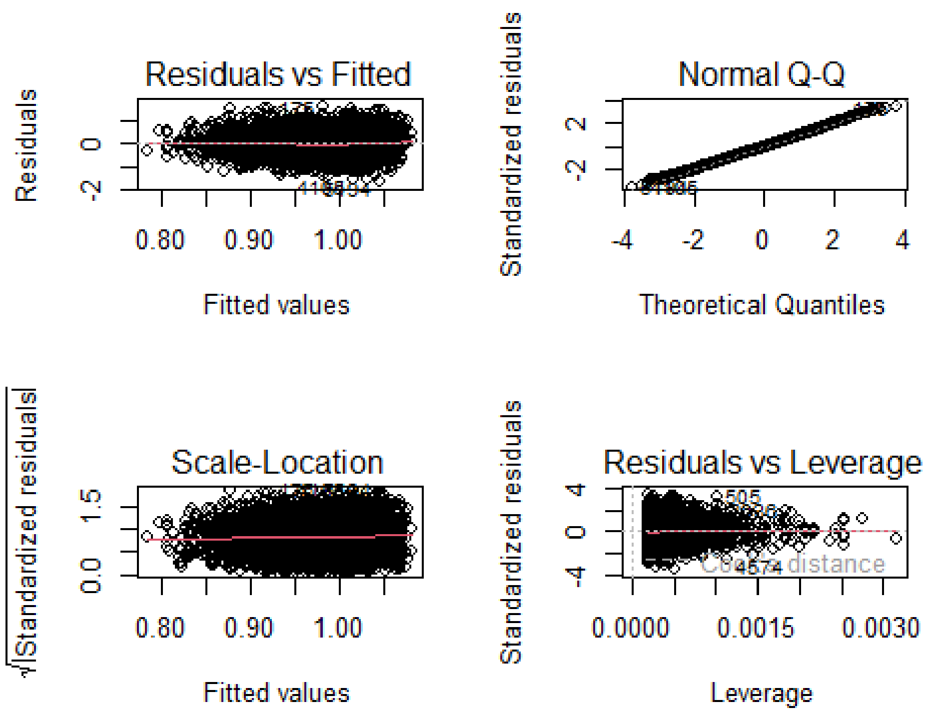
Simple Linear Regression - Critic Score

```
# Fit a simple linear regression model
lm.fit <- lm(Global_Sales ~ Critic_Score, data = Video.Game)

# Get summary of the model
summary(lm.fit)
##
## Call:
## lm(formula = Global_Sales ~ Critic_Score, data = Video.Game)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.61045 -0.31863 -0.00187  0.31152  1.58623
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.7390682  0.0321127  23.015  < 2e-16 ***
## Critic_Score 0.0035641  0.0004634   7.692 1.71e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4667 on 5508 degrees of freedom
## Multiple R-squared:  0.01063,    Adjusted R-squared:  0.01045
## F-statistic: 59.16 on 1 and 5508 DF,  p-value: 1.708e-14
```

```r
# Create a sequence of Critic Score values for prediction
x_seq <- seq(from = min(Video.Game$Critic_Score), to =
max(Video.Game$Critic_Score), by = 1)

# Get the predicted values from the model
y_hat <- predict(lm.fit, newdata = list(Critic_Score = x_seq))

# Calculate the residuals
residuals <- Video.Game$Global_Sales - y_hat
## Warning in Video.Game$Global_Sales - y_hat: longer object length is not a
multiple of shorter object length

# Calculate the RMSE
RMSE <- sqrt(mean(residuals^2))

# Print the RMSE
cat("The RMSE of Critic Score for the simple regression model is:", RMSE,
"\n")
## The RMSE of Critic Score for the simple regression model is: 0.4779891

# We will now plot Global_Sales and Critic_Score along with the least squares
regression line
plot(Critic_Score, Global_Sales, pch = 20, col = "blue", main = "Video Game
Sales vs. Critic Score")

# Plot the simple regression line
lines(x_seq, y_hat, col = "red", lwd = 2)
```




```
# Produce four diagnostic plots
par(mfrow = c(2,2))
plot(lm.fit)
```



Simple Linear Regression - Confidence Intervals and Prediction Intervals - Critic_Score

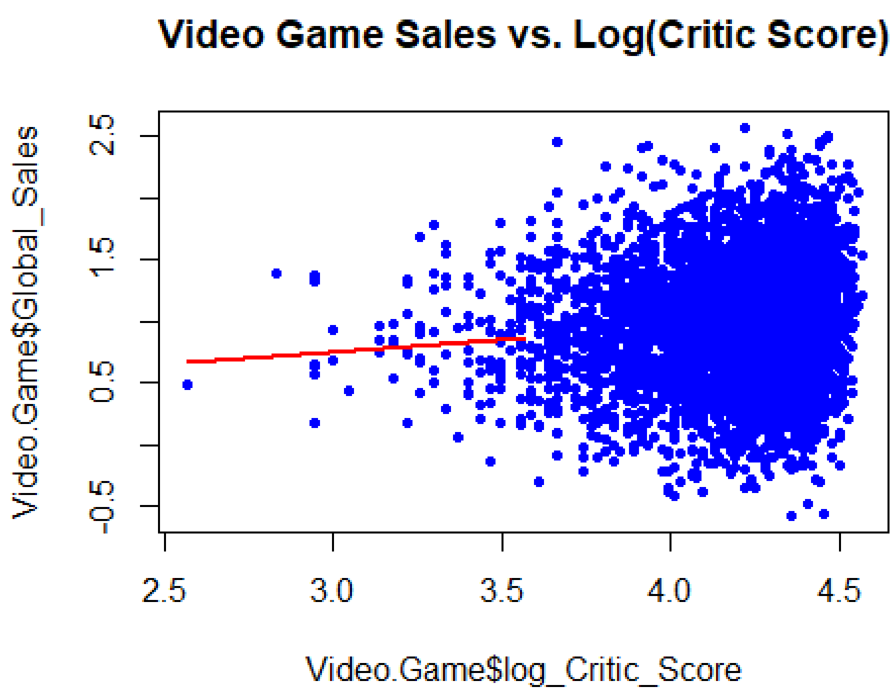
```
# Indicating the dataset for the linear regression
lm.fit <- lm(Global_Sales ~ Critic_Score, data = Video.Game)

# Create a sequence of Critic Score values that are present in the dataset
critic_score_seq <- seq(from = min(Video.Game$Critic_Score), to =
max(Video.Game$Critic_Score), by = 1)

# Predict the Global Sales for each value in the sequence with 95% confidence
interval
confid <- predict(lm.fit, newdata = data.frame(Critic_Score =
critic_score_seq), interval = "confidence", level = 0.95)

# View the Confidence Interval for Global Sales
print(head(confid))
##         fit       lwr       upr
## 1 0.7854013 0.7339759 0.8368266
## 2 0.7889654 0.7384214 0.8395093
## 3 0.7925294 0.7428659 0.8421929
## 4 0.7960935 0.7473094 0.8448776
## 5 0.7996576 0.7517519 0.8475633
## 6 0.8032217 0.7561932 0.8502502
```

```
# Predict the Global Sales for each value in the sequence with prediction
interval
pred <- predict(lm.fit, newdata = data.frame(Critic_Score =
critic_score_seq), interval = "prediction")

# View the Prediction Interval for Global Sales
print(head(pred))
##         fit        lwr      upr
## 1 0.7854013 -0.1310078 1.701810
## 2 0.7889654 -0.1273947 1.705325
## 3 0.7925294 -0.1237825 1.708841
## 4 0.7960935 -0.1201711 1.712358
## 5 0.7996576 -0.1165607 1.715876
## 6 0.8032217 -0.1129512 1.719395
```

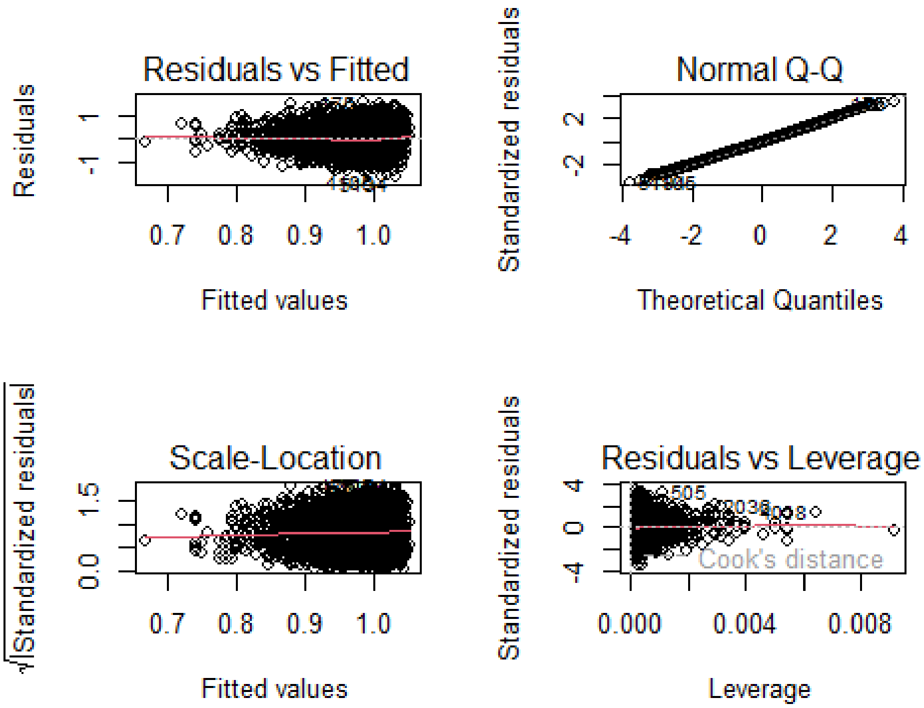
Simple Linear Regression - Log transformation - Critic Score

```
# Log transformation
Video.Game$log_Critic_Score <- log(Video.Game$Critic_Score)

# Fit a simple linear regression model with log-transformed Critic Score
lm.fit_log <- lm(Global_Sales ~ log_Critic_Score, data = Video.Game)

# Get summary of the model
summary(lm.fit_log)
##
## Call:
## lm(formula = Global_Sales ~ log_Critic_Score, data = Video.Game)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.59616 -0.31848 -0.00317  0.31241  1.58163
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.17281    0.11413   1.514     0.13
## log_Critic_Score  0.19273    0.02716   7.095 1.46e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4671 on 5508 degrees of freedom
## Multiple R-squared:  0.009056,   Adjusted R-squared:  0.008876
## F-statistic: 50.34 on 1 and 5508 DF,  p-value: 1.459e-12

# Create a sequence of Critic Score values for prediction
x_seq <- seq(from = min(Video.Game$log_Critic_Score), to =
max(Video.Game$log_Critic_Score), by = 1)

# Predict Global Sales for each value in the sequence with the log model
```

```r
y_hat_log <- predict(lm.fit_log, newdata = data.frame(log_Critic_Score =
x_seq))

# Calculate the residuals
residuals_log <- Video.Game$Global_Sales - predict(lm.fit_log)

# Calculate the RMSE
RMSE_log <- sqrt(mean(residuals_log^2))

# Print the RMSE
cat("The RMSE of the log model is:", RMSE_log)
## The RMSE of the log model is: 0.4670106

# We will now plot Global_Sales and log(Critic_Score) along with the least
squares regression line
plot(Video.Game$log_Critic_Score, Video.Game$Global_Sales, pch = 20, col =
"blue", main = "Video Game Sales vs. Log(Critic Score)")

# Plot the log regression line
lines(x_seq, y_hat_log, col = "red", lwd = 2)
```



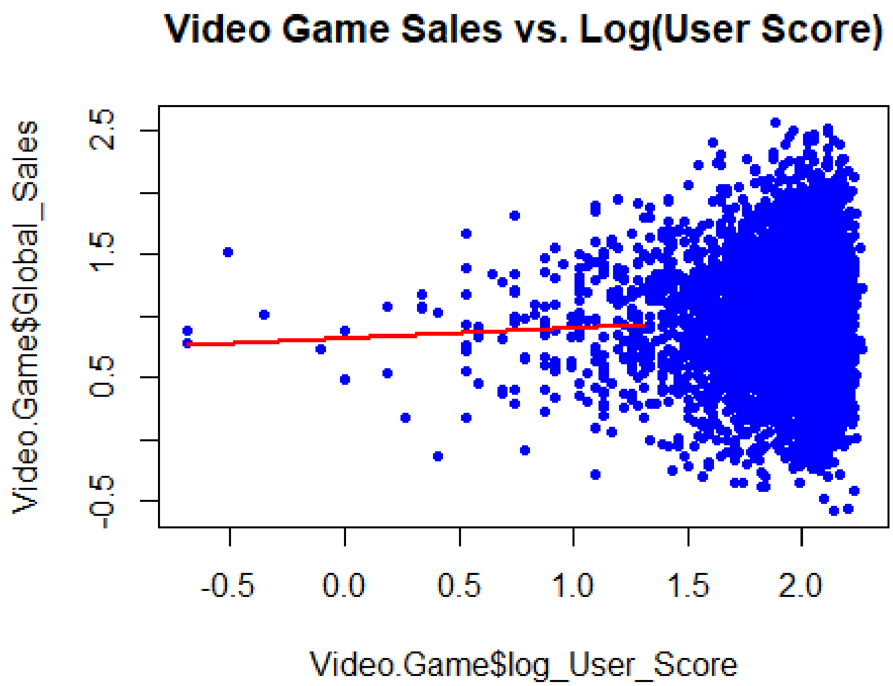

```r
# Produce four diagnostic plots
par(mfrow = c(2,2))
plot(lm.fit_log)
```



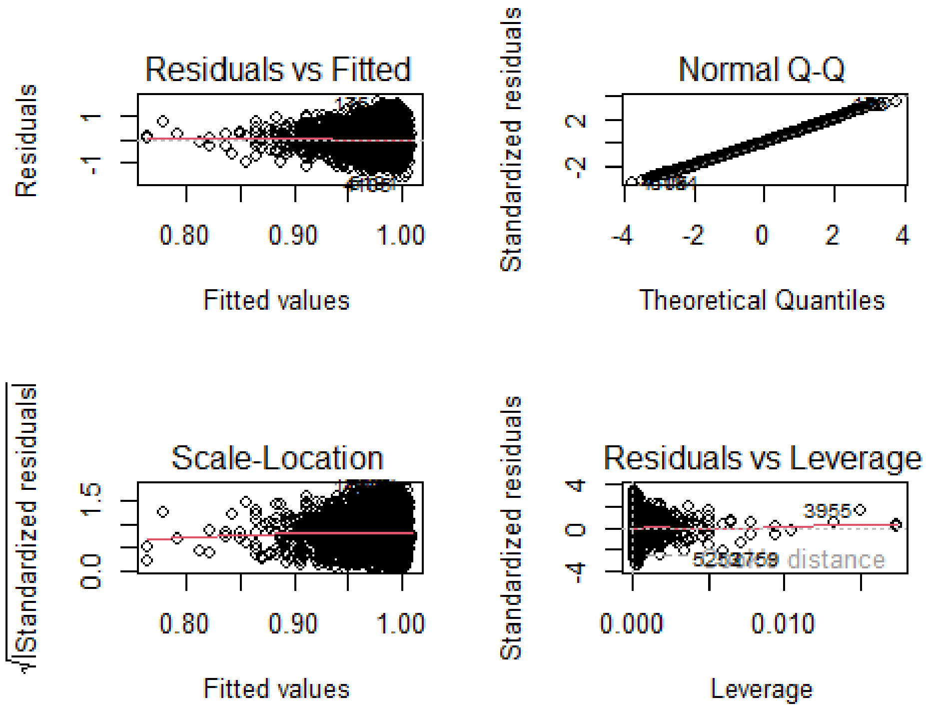
Simple Linear Regression - Log transformation - User Score

```
# Log transformation
Video.Game$log_User_Score <- log(Video.Game$User_Score)

# Fit a simple linear regression model with log-transformed Critic Score
lm.fit_log <- lm(Global_Sales ~ log_User_Score, data = Video.Game)

# Get summary of the model
summary(lm.fit_log)
##
## Call:
## lm(formula = Global_Sales ~ log_User_Score, data = Video.Game)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.57363 -0.31682 -0.00997  0.31728  1.58975
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.82175    0.04568  17.989  < 2e-16 ***
## log_User_Score  0.08275    0.02347   3.526 0.000425 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4687 on 5508 degrees of freedom
```

```
## Multiple R-squared:  0.002253,   Adjusted R-squared:  0.002071
## F-statistic: 12.44 on 1 and 5508 DF,  p-value: 0.0004247

# Create a sequence of Critic Score values for prediction
x_seq <- seq(from = min(Video.Game$log_User_Score), to =
max(Video.Game$log_User_Score), by = 1)

# Predict Global Sales for each value in the sequence with the log model
y_hat_log <- predict(lm.fit_log, newdata = data.frame(log_User_Score =
x_seq))

# Calculate the residuals
residuals_log <- Video.Game$Global_Sales - predict(lm.fit_log)

# Calculate the RMSE
RMSE_log <- sqrt(mean(residuals_log^2))

# Print the RMSE
cat("The RMSE of the log model is:", RMSE_log)
## The RMSE of the log model is: 0.468611

# We will now plot Global_Sales and log(Critic_Score) along with the least
squares regression line
plot(Video.Game$log_User_Score, Video.Game$Global_Sales, pch = 20, col =
"blue", main = "Video Game Sales vs. Log(User Score)")

# Plot the log regression line
lines(x_seq, y_hat_log, col = "red", lwd = 2)
```



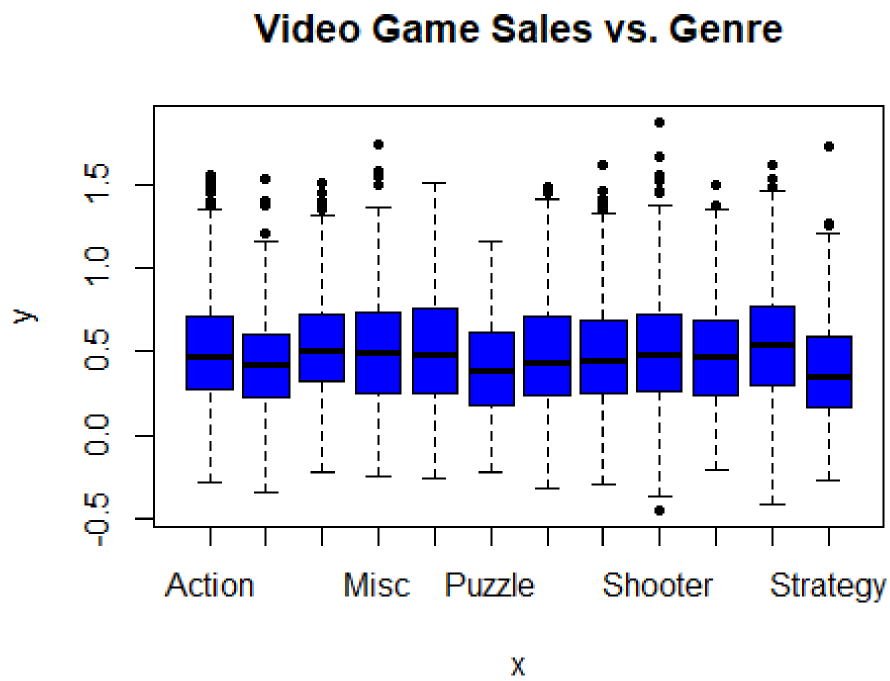

```
# Produce four diagnostic plots
par(mfrow = c(2,2))
plot(lm.fit_log)
```



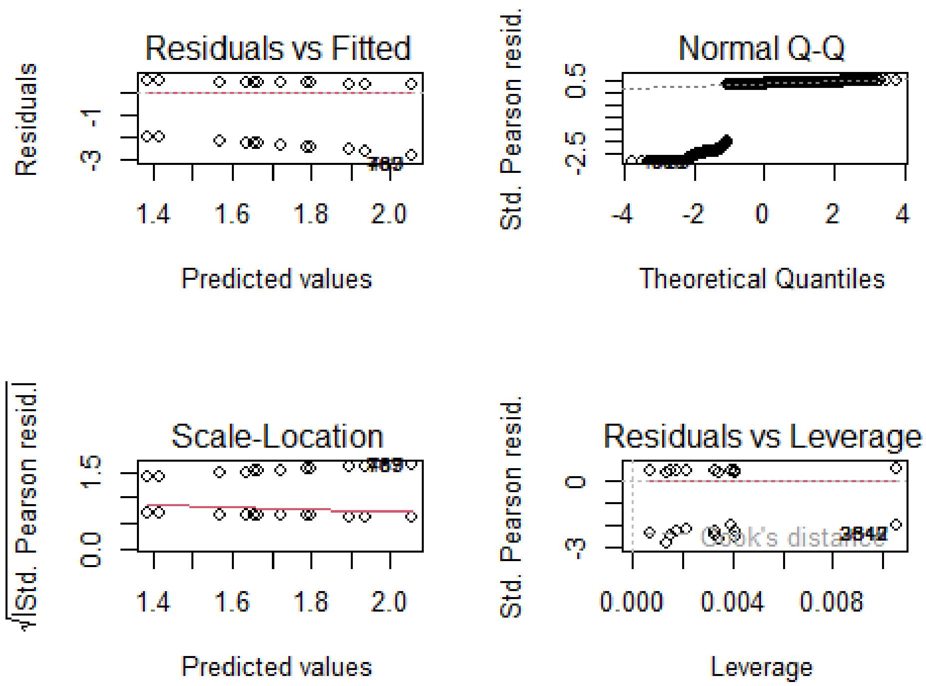
Logistic Regression - Genre

```
set.seed(123)

# Create binary variable based on Global_Sales threshold of
revised_global_sales_mean
Video.Game$hit_median <- ifelse(Video.Game$Global_Sales >=
revised_global_sales_mean, 1, 0)

# fit the logistic regression model
glm.fits <- glm(hit_median ~ Genre, family = binomial, data = Video.Game)

# Get summary of the model
summary(glm.fits)
##
## Call:
## glm(formula = hit_median ~ Genre, family = binomial, data = Video.Game)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0849   0.4915   0.5562   0.5888   0.6681
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.72347    0.07638  22.564   <2e-16 ***
## GenreAdventure       0.17208    0.20440   0.842   0.3998
```

```
## GenreFighting     -0.08984    0.17386  -0.517   0.6053
## GenreMisc          0.21447    0.19394   1.106   0.2688
## GenrePlatform      0.06449    0.17990   0.358   0.7200
## GenrePuzzle       -0.33718    0.26763  -1.260   0.2077
## GenreRacing       -0.15486    0.14470  -1.070   0.2845
## GenreRole-Playing -0.05905    0.13697  -0.431   0.6664
## GenreShooter       0.07543    0.13552   0.557   0.5778
## GenreSimulation   -0.06892    0.19073  -0.361   0.7179
## GenreSports        0.32914    0.13823   2.381   0.0173 *
## GenreStrategy     -0.30762    0.17518  -1.756   0.0791 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4639.1  on 5509  degrees of freedom
## Residual deviance: 4620.0  on 5498  degrees of freedom
## AIC: 4644
##
## Number of Fisher Scoring iterations: 4

# Create a sequence of Genre values for prediction
x_seq <- levels(Video.Game$Genre)

# Get the predicted values from the model
y_hat <- predict(glm.fits, newdata = list(Genre = x_seq), type = "response")

# Calculate the residuals
residuals <- Video.Game$Global_Sales - y_hat
## Warning in Video.Game$Global_Sales - y_hat: longer object length is not a
multiple of shorter object length

# Calculate the RMSE
RMSE <- sqrt(mean(residuals^2))

# Print the RMSE
cat("The RMSE of Genre for the logistic regression model is:", RMSE, "\n")
## The RMSE of Genre for the logistic regression model is: 0.488807

# We will now plot Global_Sales and Genre along with the logistic regression
curve
plot(Genre, Global_Sales, pch = 20, col = "blue", main = "Video Game Sales
vs. Genre")

# Plot the logistic regression curve
lines(x_seq, y_hat, col = "red", lwd = 2)
## Warning in xy.coords(x, y): NAs introduced by coercion
```




```r
# Produce four diagnostic plots
par(mfrow = c(2,2))
plot(glm.fits)
```

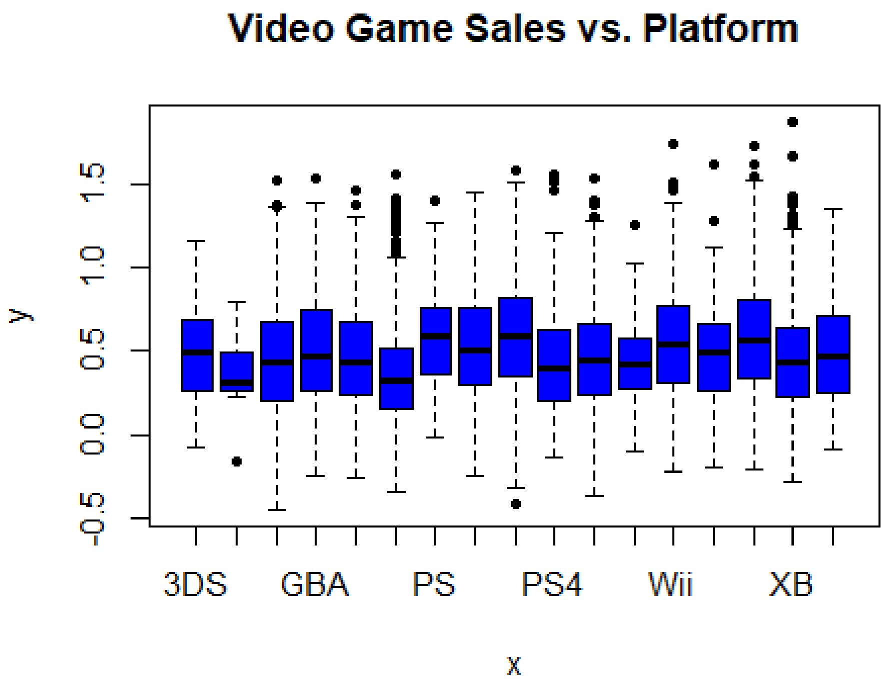


Logistic Regression - Platform

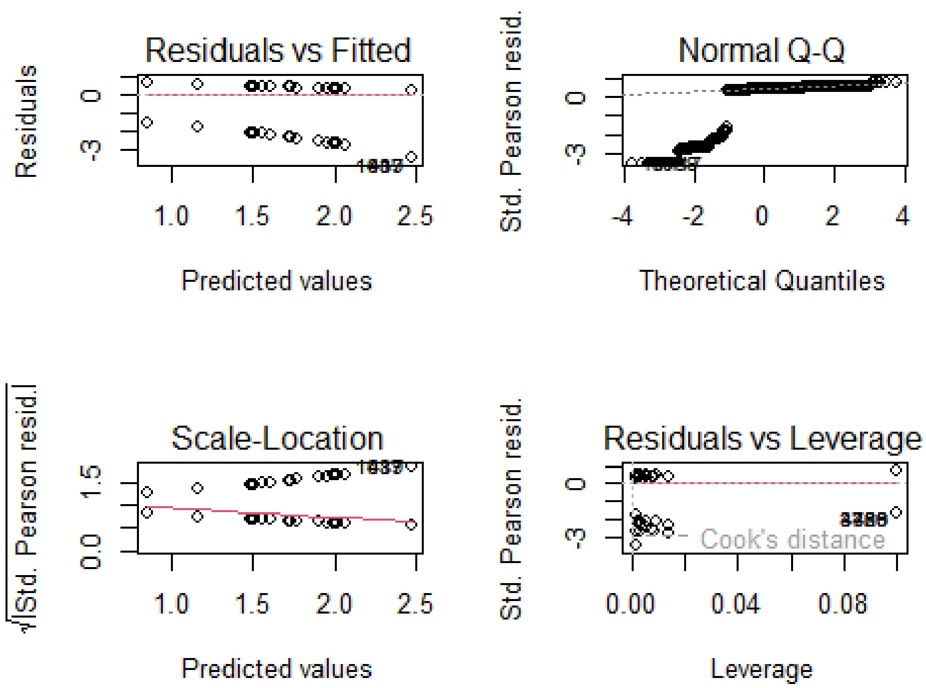
```
set.seed(123)

# Create binary variable based on Global_Sales threshold of
revised_global_sales_mean
Video.Game$hit_median <- ifelse(Video.Game$Global_Sales >=
revised_global_sales_mean, 1, 0)

# fit the logistic regression model
glm.fits <- glm(hit_median ~ Platform, family = binomial, data = Video.Game)

# Get summary of the model
summary(glm.fits)
##
## Call:
## glm(formula = hit_median ~ Platform, family = binomial, data = Video.Game)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2618   0.4890   0.5152   0.6191   0.8446
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.91172    0.25984   7.357 1.88e-13 ***
## PlatformDC    -1.06442    0.73737  -1.444  0.14887
```

```
## PlatformDS   -0.30541    0.29373  -1.040  0.29845
## PlatformGBA  -0.14322    0.32846  -0.436  0.66282
## PlatformGC   -0.17968    0.30513  -0.589  0.55596
## PlatformPC   -0.75521    0.27609  -2.735  0.00623 **
## PlatformPS    0.15197    0.45650   0.333  0.73920
## PlatformPS2   0.04072    0.27928   0.146  0.88407
## PlatformPS3   0.56541    0.30451   1.857  0.06335 .
## PlatformPS4  -0.41779    0.32346  -1.292  0.19648
## PlatformPSP  -0.40406    0.29533  -1.368  0.17126
## PlatformPSV  -0.43445    0.35498  -1.224  0.22099
## PlatformWii   0.09852    0.30642   0.322  0.74781
## PlatformWiiU -0.19874    0.41812  -0.475  0.63456
## PlatformX360  0.07776    0.28685   0.271  0.78634
## PlatformXB   -0.35709    0.28457  -1.255  0.20953
## PlatformXOne  0.10318    0.38532   0.268  0.78886
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4639.1  on 5509  degrees of freedom
## Residual deviance: 4553.0  on 5493  degrees of freedom
## AIC: 4587
##
## Number of Fisher Scoring iterations: 5

# Create a sequence of Platform values for prediction
x_seq <- levels(Video.Game$Platform)

# Get the predicted values from the model
y_hat <- predict(glm.fits, newdata = list(Platform = x_seq), type =
"response")

# Calculate the residuals
residuals <- Video.Game$Global_Sales - y_hat
## Warning in Video.Game$Global_Sales - y_hat: longer object length is not a
multiple of shorter object length

# Calculate the RMSE
RMSE <- sqrt(mean(residuals^2))

# Print the RMSE
cat("The RMSE of Platform for the logistic regression model is:", RMSE, "\n")
## The RMSE of Platform for the logistic regression model is: 0.491246

# We will now plot Global_Sales and Platform along with the logistic
regression curve
plot(Platform, Global_Sales, pch = 20, col = "blue", main = "Video Game Sales
vs. Platform")
```

```r
# Plot the logistic regression curve
lines(x_seq, y_hat, col = "red", lwd = 2)
## Warning in xy.coords(x, y): NAs introduced by coercion
```




```r
# Produce four diagnostic plots
par(mfrow = c(2,2))
plot(glm.fits)
```

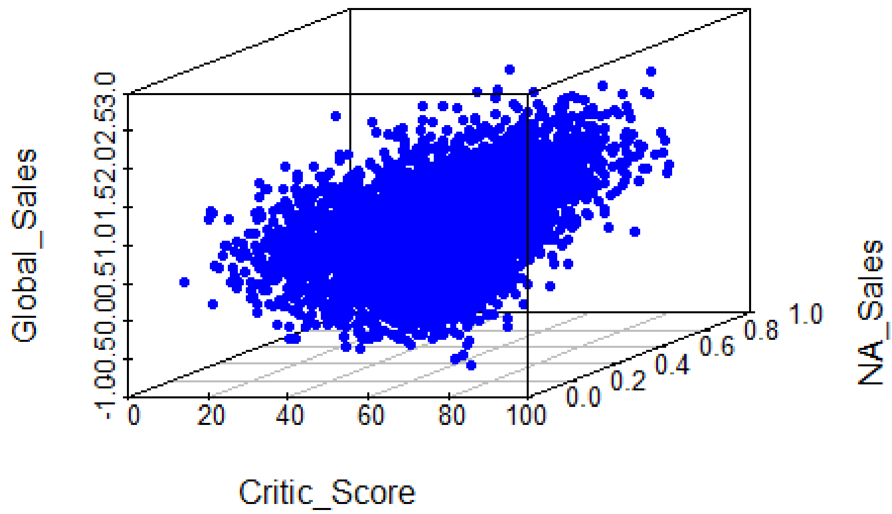


Multiple Regression - NA_Sales, Genre, Critic_Score, & Critic_Count

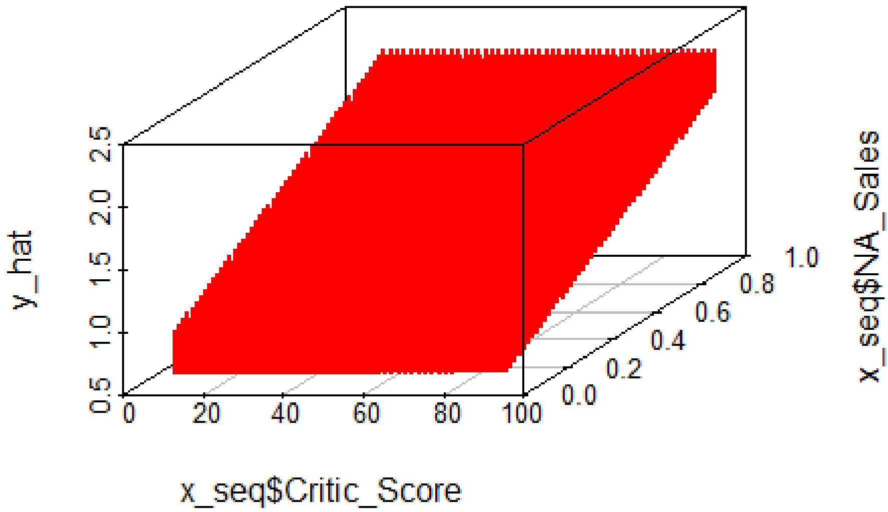
```
set.seed(123)

# fit the linear regression model
lm.fits <- lm(Global_Sales ~ NA_Sales + Genre + Critic_Score + Critic_Count,
data = Video.Game)

# Get summary of the model
summary(lm.fits)
##
## Call:
## lm(formula = Global_Sales ~ NA_Sales + Genre + Critic_Score +
##     Critic_Count, data = Video.Game)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.46876 -0.28017  0.00244  0.28316  1.49236
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.7226853  0.0300009  24.089  < 2e-16 ***
## NA_Sales            1.2739096  0.0389434  32.712  < 2e-16 ***
## GenreAdventure      0.0468104  0.0293363   1.596   0.1106
## GenreFighting      -0.0598050  0.0269160  -2.222   0.0263 *
## GenreMisc          -0.0427569  0.0275468  -1.552   0.1207
```

```
## GenrePlatform     -0.0285683  0.0267512  -1.068   0.2856    
## GenrePuzzle       -0.0561119  0.0447225  -1.255   0.2097    
## GenreRacing       -0.0211052  0.0227701  -0.927   0.3540    
## GenreRole-Playing  0.0086815  0.0211274   0.411   0.6812    
## GenreShooter      -0.0049118  0.0202135  -0.243   0.8080    
## GenreSimulation   -0.0295905  0.0294779  -1.004   0.3155    
## GenreSports       -0.0155668  0.0198832  -0.783   0.4337    
## GenreStrategy     -0.0280031  0.0290330  -0.965   0.3348    
## Critic_Score       0.0001778  0.0004554   0.391   0.6962    
## Critic_Count       0.0023434  0.0003919   5.980 2.37e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.4206 on 5495 degrees of freedom
## Multiple R-squared:  0.1984, Adjusted R-squared:  0.1964 
## F-statistic: 97.17 on 14 and 5495 DF,  p-value: < 2.2e-16

# Create a sequence of Genre, Critic_Score, Critic_Count and NA_Sales values for prediction
subset_data <- na.omit(Video.Game[c("Genre", "Critic_Score", "Critic_Count", "NA_Sales", "Global_Sales")])

x_seq <- expand.grid(
  Genre = unique(subset_data$Genre),
  Critic_Score = seq(min(subset_data$Critic_Score), max(subset_data$Critic_Score), length.out = 50),
  Critic_Count = seq(min(subset_data$Critic_Count), max(subset_data$Critic_Count), length.out = 50),
  NA_Sales = seq(min(subset_data$NA_Sales), max(subset_data$NA_Sales), length.out = 50)
)

# Get the predicted values from the model
y_hat <- predict(lm.fits, newdata = x_seq)

# Calculate the residuals
residuals <- subset_data$Global_Sales - y_hat
## Warning in subset_data$Global_Sales - y_hat: longer object length is not a multiple of shorter object length

# Calculate the RMSE
RMSE <- sqrt(mean(residuals^2))

# Print the RMSE
cat("The RMSE of Genre, Critic_Score, Critic_Count, and NA_Sales for the linear regression model is:", RMSE, "\n")
## The RMSE of Genre, Critic_Score, Critic_Count, and NA_Sales for the linear regression model is: 0.7486262
```

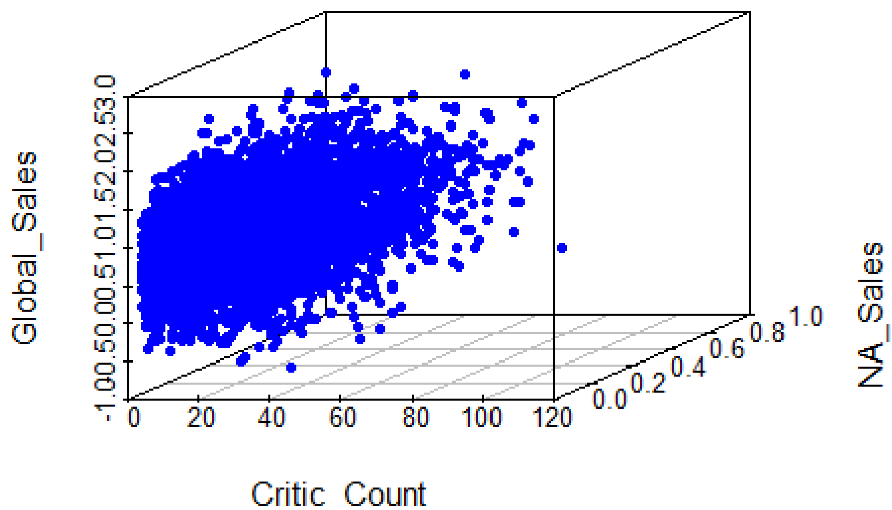
```r
# Create a 3D scatter plot with Critic_Score, NA_Sales, and Global_Sales
scatterplot3d(subset_data$Critic_Score, subset_data$NA_Sales, 
subset_data$Global_Sales,
              color = "blue", pch = 20, xlab = "Critic_Score", ylab = 
"NA_Sales", zlab = "Global_Sales")
```



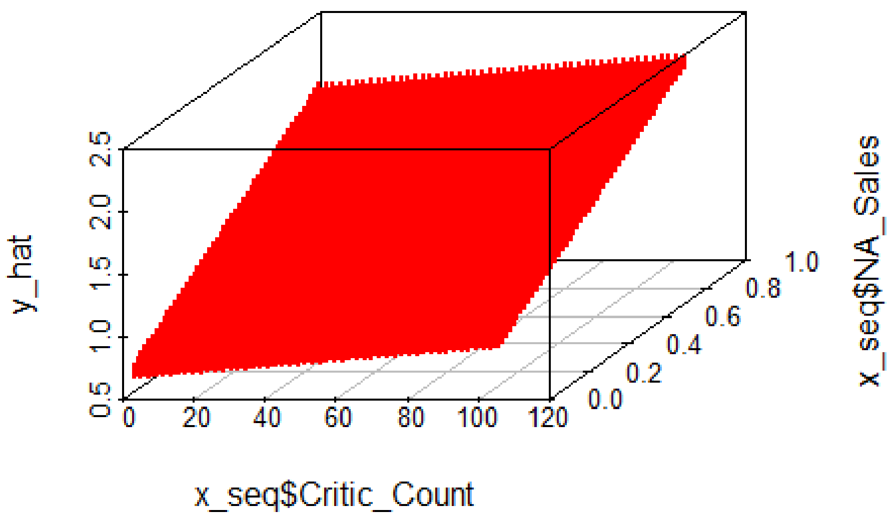

```r
# Add linear regression curve to the 3D scatter plot
scatterplot3d(x_seq$Critic_Score, x_seq$NA_Sales, y_hat,
              color = "red", add = TRUE, type = "l", lwd = 2)
## Warning in title(main, sub, ...): "add" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "add" is not a 
graphical parameter
```



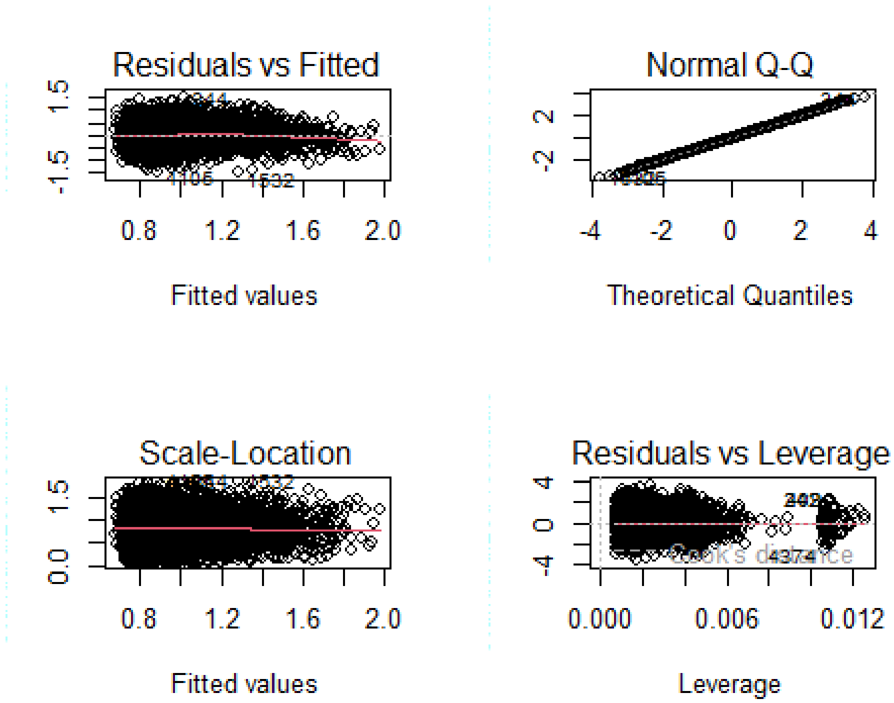

```
# Create a 3D scatter plot with Critic_Count, NA_Sales, and Global_Sales
scatterplot3d(subset_data$Critic_Count, subset_data$NA_Sales,
subset_data$Global_Sales,
              color = "blue", pch = 20, xlab = "Critic_Count", ylab =
"NA_Sales", zlab = "Global_Sales")
```




```
# Add linear regression curve to the 3D scatter plot
scatterplot3d(x_seq$Critic_Count, x_seq$NA_Sales, y_hat,
              color = "red", add = TRUE, type = "l", lwd = 2)
## Warning in title(main, sub, ...): "add" is not a graphical parameter

## Warning in title(main, sub, ...): "add" is not a graphical parameter
```




```
# Produce diagnostic plots
par(mfrow = c(2,2))
plot(lm.fits)
```

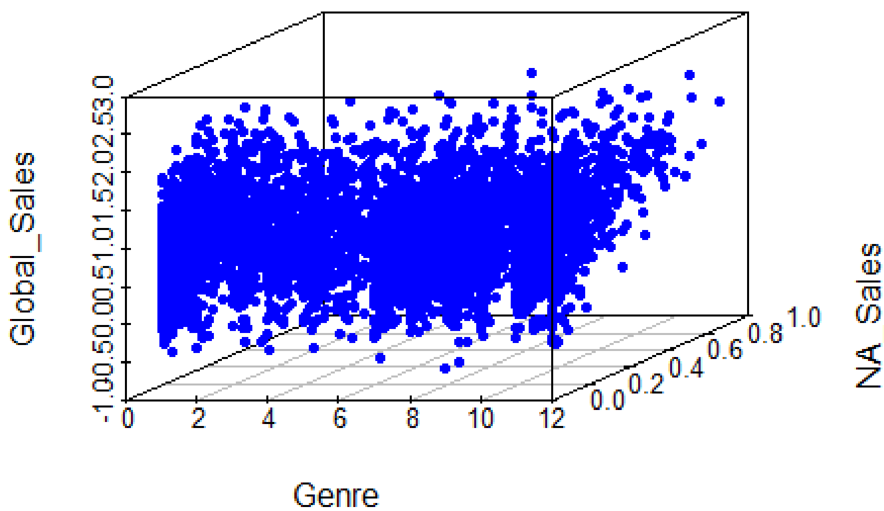


Multiple Regression - Regional_Sales, Genre, Critic_Score

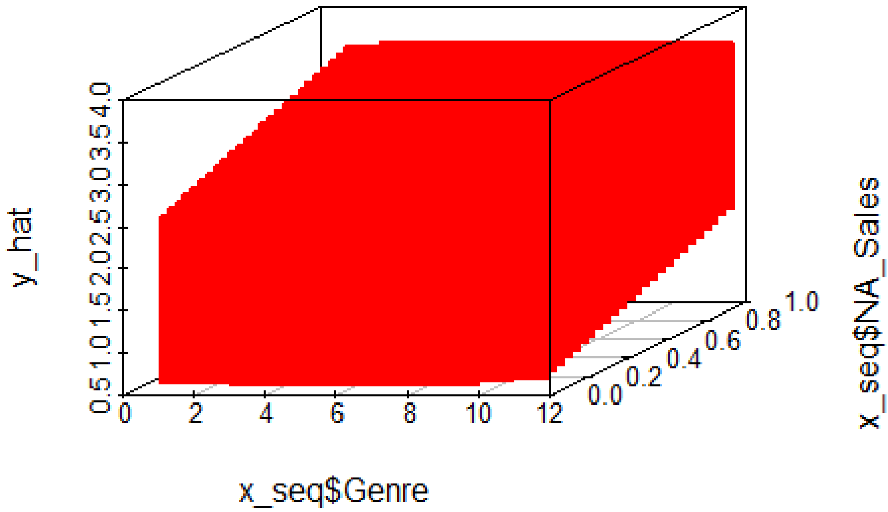
```
set.seed(123)

# Create weights based on each region's contribution to sales (excluding
Other_Region_Sales)
weights <- (Video.Game$NA_Sales + Video.Game$EU_Sales + Video.Game$JP_Sales)
/ sum(Video.Game$NA_Sales + Video.Game$EU_Sales + Video.Game$JP_Sales)

# Replace missing or negative weights with 0
weights[is.na(weights) | weights < 0] <- 0

# Fit the linear regression model with weights
lm.fits <- lm(Global_Sales ~ Critic_Score + Genre + NA_Sales + EU_Sales +
JP_Sales, data = Video.Game, weights = weights)

# Get summary of the model
summary(lm.fits)
##
## Call:
## lm(formula = Global_Sales ~ Critic_Score + Genre + NA_Sales +
##     EU_Sales + JP_Sales, data = Video.Game, weights = weights)
##
## Weighted Residuals:
##         Min          1Q      Median          3Q         Max
```

```
## -0.0258025 -0.0028484 -0.0000271  0.0028017  0.0240309
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.7902813  0.0312229  25.311  < 2e-16 ***
## Critic_Score       -0.0013355  0.0004462  -2.993  0.00277 **
## GenreAdventure      0.0471782  0.0330955   1.426  0.15406
## GenreFighting      -0.0371957  0.0243018  -1.531  0.12593
## GenreMisc           0.0147476  0.0247751   0.595  0.55169
## GenrePlatform      -0.0124587  0.0246529  -0.505  0.61333
## GenrePuzzle         0.0460494  0.0488207   0.943  0.34560
## GenreRacing        -0.0030512  0.0217440  -0.140  0.88841
## GenreRole-Playing   0.0298905  0.0204237   1.464  0.14338
## GenreShooter        0.0330287  0.0190219   1.736  0.08256 .
## GenreSimulation    -0.0451897  0.0274588  -1.646  0.09988 .
## GenreSports         0.0394924  0.0176845   2.233  0.02558 *
## GenreStrategy       0.0573543  0.0339483   1.689  0.09119 .
## NA_Sales            1.0322691  0.0300013  34.407  < 2e-16 ***
## EU_Sales            1.1885530  0.0418598  28.394  < 2e-16 ***
## JP_Sales            0.9964170  0.0569562  17.494  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.005324 on 5494 degrees of freedom
## Multiple R-squared:  0.2966, Adjusted R-squared:  0.2946
## F-statistic: 154.4 on 15 and 5494 DF,  p-value: < 2.2e-16

# Create a sequence of Genre, Critic_Score, NA_Sales, EU_Sales, JP_Sales
values for prediction
subset_data <- na.omit(Video.Game[c("Genre", "Critic_Score","NA_Sales",
"EU_Sales", "JP_Sales", "Global_Sales")])
x_seq <- expand.grid(
  Genre = unique(subset_data$Genre),
  Critic_Score = seq(min(subset_data$Critic_Score),
max(subset_data$Critic_Score), length.out = 25),
  NA_Sales = seq(min(subset_data$NA_Sales), max(subset_data$NA_Sales),
length.out = 25),
  EU_Sales = seq(min(subset_data$EU_Sales), max(subset_data$EU_Sales),
length.out = 25),
  JP_Sales = seq(min(subset_data$JP_Sales), max(subset_data$JP_Sales),
length.out = 25)
)

# Get the predicted values from the model
y_hat <- predict(lm.fits, newdata = x_seq)

# Calculate the residuals
residuals <- subset_data$Global_Sales - y_hat
## Warning in subset_data$Global_Sales - y_hat: longer object length is not a
multiple of shorter object length
```

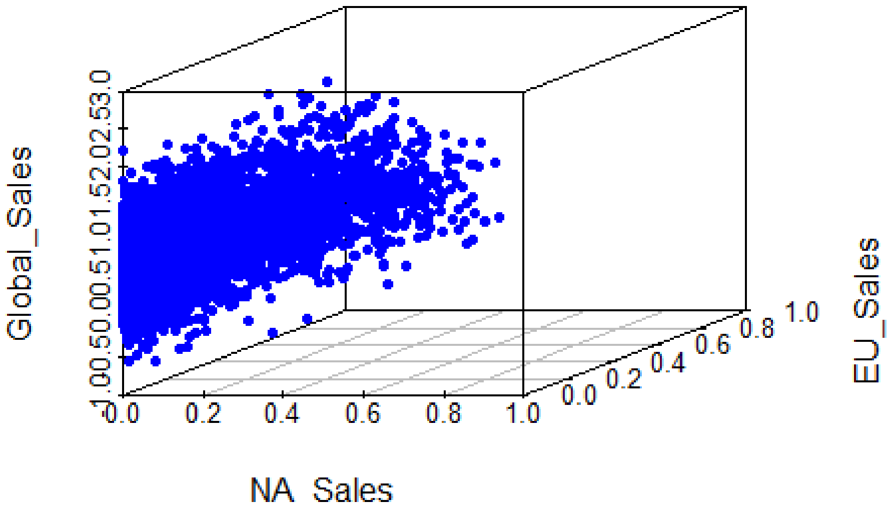
```r
# Calculate the RMSE
RMSE <- sqrt(mean(residuals^2))

# Print the RMSE
cat("The RMSE of Genre, Critic_Score, NA_Sales, EU_Sales, JP_Sales for the
linear regression model is:", RMSE, "\n")
## The RMSE of Genre, Critic_Score, NA_Sales, EU_Sales, JP_Sales for the
linear regression model is: 1.343167

# Create a 3D scatter plot with Genre, NA_Sales, and Global_Sales
scatterplot3d(subset_data$Genre, subset_data$NA_Sales,
subset_data$Global_Sales,
              color = "blue", pch = 20, xlab = "Genre", ylab = "NA_Sales",
zlab = "Global_Sales")
```



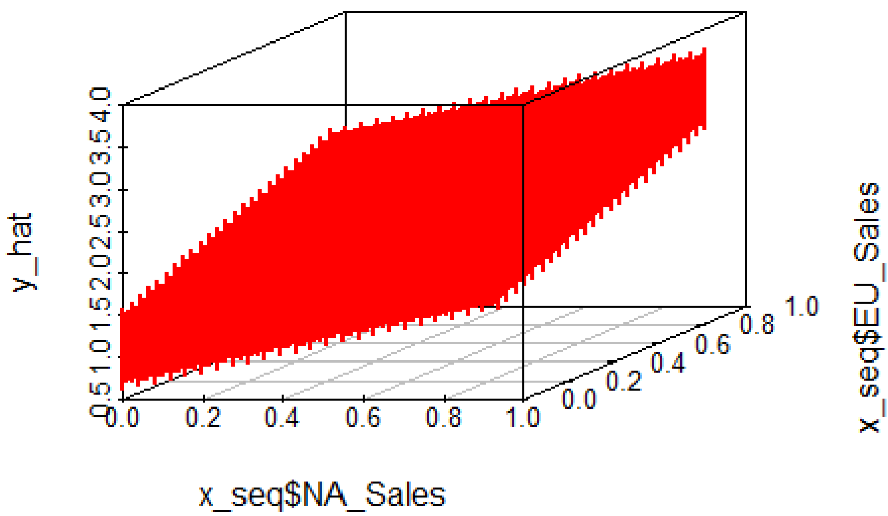

```r
# Add linear regression curve to the 3D scatter plot
scatterplot3d(x_seq$Genre, x_seq$NA_Sales, y_hat,
              color = "red", add = TRUE, type = "l", lwd = 2)
## Warning in title(main, sub, ...): "add" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "add" is not a
graphical parameter
```



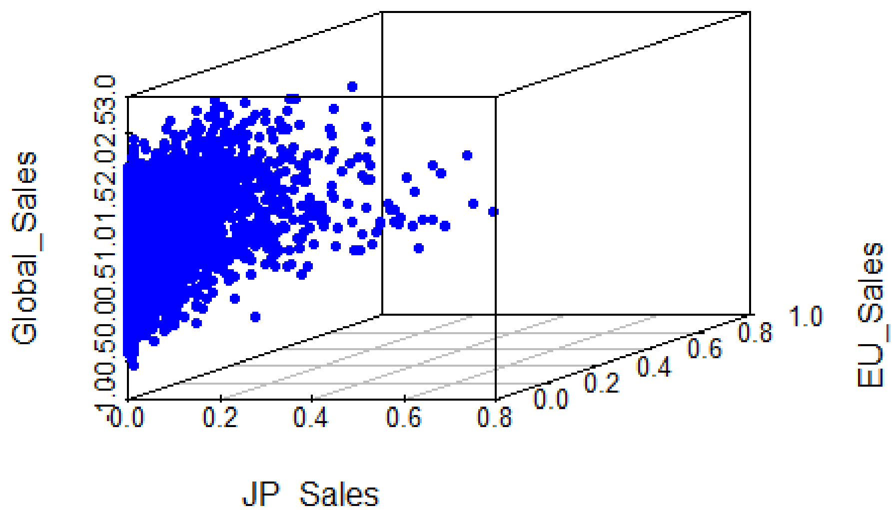

```
# Create a 3D scatter plot with NA_Sales, EU_Sales, and Global_Sales
scatterplot3d(subset_data$NA_Sales, subset_data$EU_Sales,
subset_data$Global_Sales,
              color = "blue", pch = 20, xlab = "NA_Sales", ylab = "EU_Sales",
zlab = "Global_Sales")
```



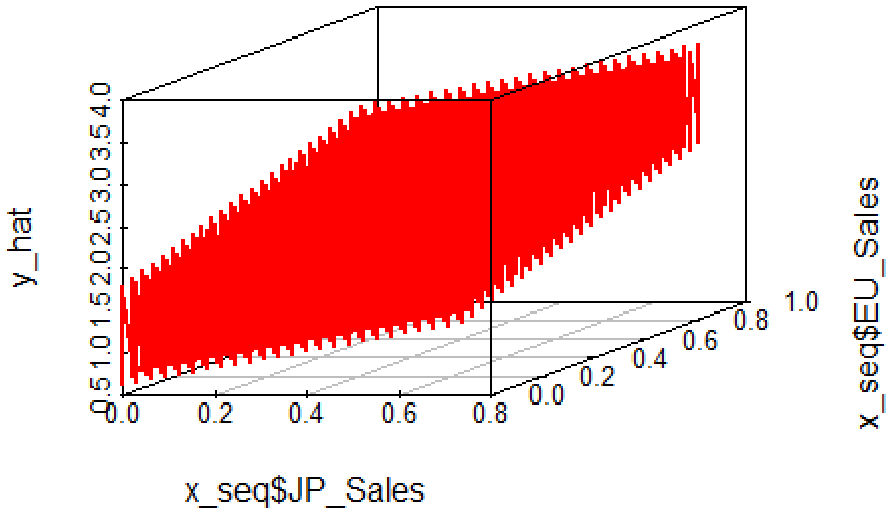

```
# Add linear regression curve to the 3D scatter plot
scatterplot3d(x_seq$NA_Sales, x_seq$EU_Sales, y_hat,
              color = "red", add = TRUE, type = "l", lwd = 2)
## Warning in title(main, sub, ...): "add" is not a graphical parameter

## Warning in title(main, sub, ...): "add" is not a graphical parameter
```



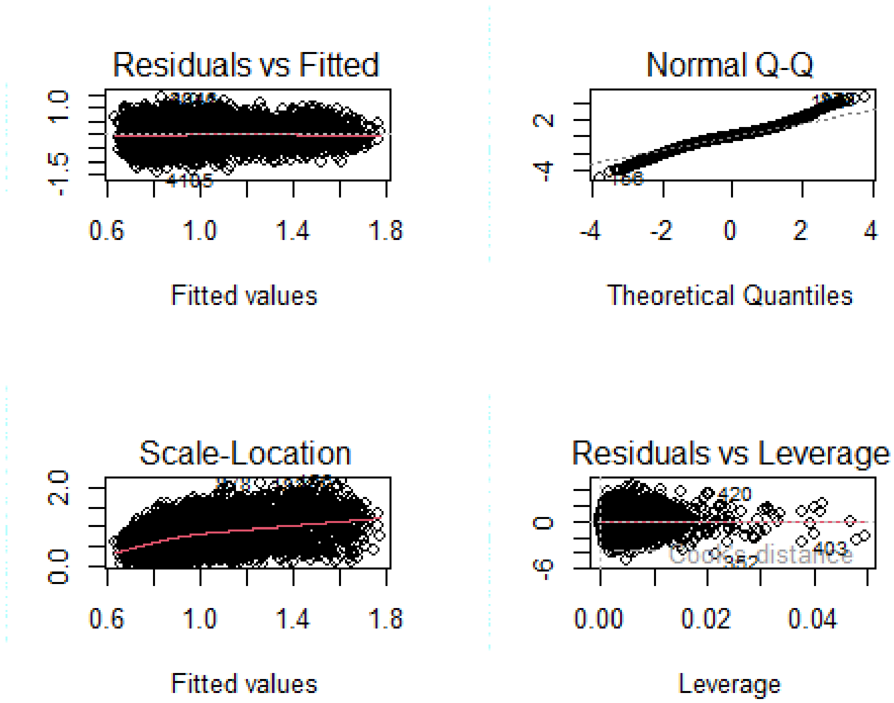

```
# Create a 3D scatter plot with JP_Sales, EU_Sales, and Global_Sales
scatterplot3d(subset_data$JP_Sales, subset_data$EU_Sales,
subset_data$Global_Sales,
              color = "blue", pch = 20, xlab = "JP_Sales", ylab = "EU_Sales",
zlab = "Global_Sales")
```




```
# Add linear regression curve to the 3D scatter plot
scatterplot3d(x_seq$JP_Sales, x_seq$EU_Sales, y_hat,
              color = "red", add = TRUE, type = "l", lwd = 2)
## Warning in title(main, sub, ...): "add" is not a graphical parameter

## Warning in title(main, sub, ...): "add" is not a graphical parameter
```




```r
# Produce diagnostic plots
par(mfrow = c(2,2))
plot(lm.fits)
```


Residuals vs Fitted
1.0
-1.5
0.6
1.0
1.4
1.8
Fitted values
Normal Q-Q
2
-4
-4
-2
0
2
4
Theoretical Quantiles
Scale-Location
2.0
0.0
0.6
1.0
1.4
1.8
Fitted values
Residuals vs Leverage
420
0
-6
403
0.00
0.02
0.04
Leverage