Austin Rodriquez 2/18/2023

1: Linear regression takes two values, x and y, and procures a line of best fit of y as a function of x with the equation y = mx + b. 'm' is the slope and 'b' is the intercept.

Strength: Its very simple to set up and interpret. It works well on liner data. It hasa low variance.

Weaknesses: Its simplicity and bias makes it keen to under fit data that doesn't follow a perfectly liner data set.

2: The data used will be "Video Game Sales with Ratings" by Rush Kirubi which can be accessed with the url: https://www.kaggle.com/datasets/rush4ratio/video-game-sales-with-ratings (https://www.kaggle.com/datasets/rush4ratio/video-game-sales-with-ratings)

The data frame will be read into the program

```
df <- read.csv("Video_Games_Sales.csv")
```

We will have to remove null data since trying to attach a mean or median value to things like sales and rating don't apply well to video games.

```
df <- na.omit(df)
```

To preview the data, we will peek at it.

```
str(df)
```

```
## 'data.frame':    7017 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 4.13 3.6 0.24 2.53 ...
##  $ Other_Sales    : num  8.45 3.29 2.95 2.88 2.84 2.24 1.9 2.15 1.69 1.77 ...
##  $ Global_Sales   : num  82.5 35.5 32.8 29.8 28.9 ...
##  $ Critic_Score   : int  76 82 80 89 58 87 91 80 61 80 ...
##  $ Critic_Count   : int  51 73 73 65 41 80 64 63 45 33 ...
##  $ User_Score     : chr  "8" "8.3" "8" "8.5" ...
##  $ User_Count     : int  322 709 192 431 129 594 464 146 106 52 ...
##  $ Developer      : chr  "Nintendo" "Nintendo" "Nintendo" "Nintendo" ...
##  $ Rating         : chr  "E" "E" "E" "E" ...
##  - attr(*, "na.action")= 'omit' Named int [1:9702] 2 5 6 10 11 13 19 21 22 23 ...
##   ..- attr(*, "names")= chr [1:9702] "2" "5" "6" "10" ...
```

Some columns need become factors. They are: platform, release year, genre, publisher, rating, developer.

User score needs to be converted into an number.

```
df$Platform <- factor(df$Platform)
df$Year_of_Release <- factor(df$Year_of_Release)
df$Genre <- factor(df$Genre)
df$Publisher <- factor(df$Publisher)
df$Rating <- factor(df$Rating)
df$Developer <- factor(df$Developer)
df$User_Score <- as.numeric(df$User_Score)
```

The user rating will have to be multiplied by 10 due to the strange nature of how Meta critic has Critic scores on a scale from 0-100 but user scores 0-10

```
df$User_Score <- df$User_Score * 10
```

#The Sales will also be multiplied by 100000

```
df$NA_Sales <- df$NA_Sales * 1000000
df$EU_Sales <- df$EU_Sales * 1000000
df$JP_Sales <- df$JP_Sales * 1000000
df$Global_Sales <- df$Global_Sales * 1000000
df$Other_Sales <- df$Other_Sales * 1000000
```

a: The data will be divided into train(80) and test(20)

```
set.seed(123)
i <- sample(1:nrow(df), nrow(df) * .80, replace = FALSE)
train <- df[i,]
test <- df[-i,]
```

b: Various R functions will be used for basic data exploration.

```
names(train)
```

```
##  [1] "Name"            "Platform"        "Year_of_Release" "Genre"
##  [5] "Publisher"       "NA_Sales"        "EU_Sales"        "JP_Sales"
##  [9] "Other_Sales"     "Global_Sales"    "Critic_Score"    "Critic_Count"
## [13] "User_Score"      "User_Count"      "Developer"       "Rating"
```

```
dim(train)
```

```
## [1] 5613   16
```

```
summary(train)
```

```
##      Name              Platform    Year_of_Release          Genre
##  Length:5613        PS2    : 923   2007   : 483    Action      :1343
##  Class :character   X360   : 702   2008   : 462    Sports      : 755
##  Mode  :character   PS3    : 634   2009   : 452    Shooter     : 713
##                     PC     : 554   2005   : 437    Role-Playing: 592
##                     XB     : 459   2006   : 408    Racing      : 474
##                     Wii    : 379   2003   : 401    Platform    : 334
##                     (Other):1962   (Other):2970    (Other)     :1402
##                        Publisher       NA_Sales           EU_Sales
##  Electronic Arts            : 756   Min.   :       0   Min.   :       0
##  Ubisoft                    : 404   1st Qu.:   60000   1st Qu.:   20000
##  Activision                 : 388   Median :  150000   Median :   60000
##  Sony Computer Entertainment: 257   Mean   :  376574   Mean   :  222147
##  THQ                        : 243   3rd Qu.:  380000   3rd Qu.:  200000
##  Sega                       : 234   Max.   :41360000   Max.   :28960000
##  (Other)                    :3331
##     JP_Sales        Other_Sales       Global_Sales       Critic_Score
##  Min.   :      0   Min.   :      0   Min.   :   10000   Min.   :13.00
##  1st Qu.:      0   1st Qu.:  10000   1st Qu.:  110000   1st Qu.:62.00
##  Median :      0   Median :  20000   Median :  290000   Median :72.00
##  Mean   :  62293   Mean   :  77021   Mean   :  738229   Mean   :70.26
##  3rd Qu.:  10000   3rd Qu.:  70000   3rd Qu.:  740000   3rd Qu.:80.00
##  Max.   :5320000   Max.   :8450000   Max.   :82530000   Max.   :98.00
##
##   Critic_Count      User_Score   User_Count           Developer
##  Min.   :  3.00   Min.   : 5   Min.   :    4.0   EA Sports : 119
##  1st Qu.: 14.00   1st Qu.:65   1st Qu.:   11.0   EA Canada : 113
##  Median : 24.00   Median :75   Median :   27.0   Capcom    : 103
##  Mean   : 28.75   Mean   :72   Mean   :  173.5   Ubisoft   :  83
##  3rd Qu.: 39.00   3rd Qu.:82   3rd Qu.:   89.0   Konami    :  78
##  Max.   :113.00   Max.   :96   Max.   :10665.0   EA Tiburon:  69
##                                                  (Other)   :5048
##      Rating
##  T      :1943
##  E      :1683
##  M      :1165
##  E10+   : 766
##         :  53
##  AO     :   1
##  (Other):   2
```

```
str(train)
```

```
## 'data.frame':    5613 obs. of  16 variables:
##  $ Name           : chr  "Looney Tunes: Back in Action" "Bleach: Soul Resurreccion" "Up" "Dynasty Warriors 4" 
...
##  $ Platform       : Factor w/ 17 levels "3DS","DC","DS",..: 8 9 13 8 15 13 11 6 6 3 ...
##  $ Year_of_Release: Factor w/ 26 levels "1985","1988",..: 12 20 18 12 17 17 16 16 19 15 ...
##  $ Genre          : Factor w/ 12 levels "Action","Adventure",..: 5 3 1 1 9 3 11 9 12 1 ...
##  $ Publisher      : Factor w/ 273 levels "10TACLE Studios",..: 264 173 244 239 65 246 66 10 252 10 ...
##  $ NA_Sales       : num  250000 270000 220000 630000 80000 340000 600000 0 0 100000 ...
##  $ EU_Sales       : num  190000 100000 280000 210000 90000 20000 40000 1120000 270000 0 ...
##  $ JP_Sales       : num  0 70000 0 1130000 0 0 0 0 0 0 ...
##  $ Other_Sales    : num  60000 50000 60000 130000 20000 30000 70000 30000 50000 10000 ...
##  $ Global_Sales   : num  500000 490000 560000 2110000 190000 380000 710000 1150000 310000 100000 ...
##  $ Critic_Score   : int  51 58 62 78 52 72 75 92 79 50 ...
##  $ Critic_Count   : int  9 34 6 24 35 20 10 40 33 17 ...
##  $ User_Score     : num  75 72 80 93 49 84 68 85 52 78 ...
##  $ User_Count     : int  6 46 4 201 22 16 8 2360 213 6 ...
##  $ Developer      : Factor w/ 1313 levels "","10tacle Studios, Fusionsphere Systems",..: 1278 909 93 826 868 3
72 358 568 165 55 ...
##  $ Rating         : Factor w/ 8 levels "","AO","E","E10+",..: 3 8 4 8 6 8 3 6 4 4 ...
##  - attr(*, "na.action")= 'omit' Named int [1:9702] 2 5 6 10 11 13 19 21 22 23 ...
##   ..- attr(*, "names")= chr [1:9702] "2" "5" "6" "10" ...
```

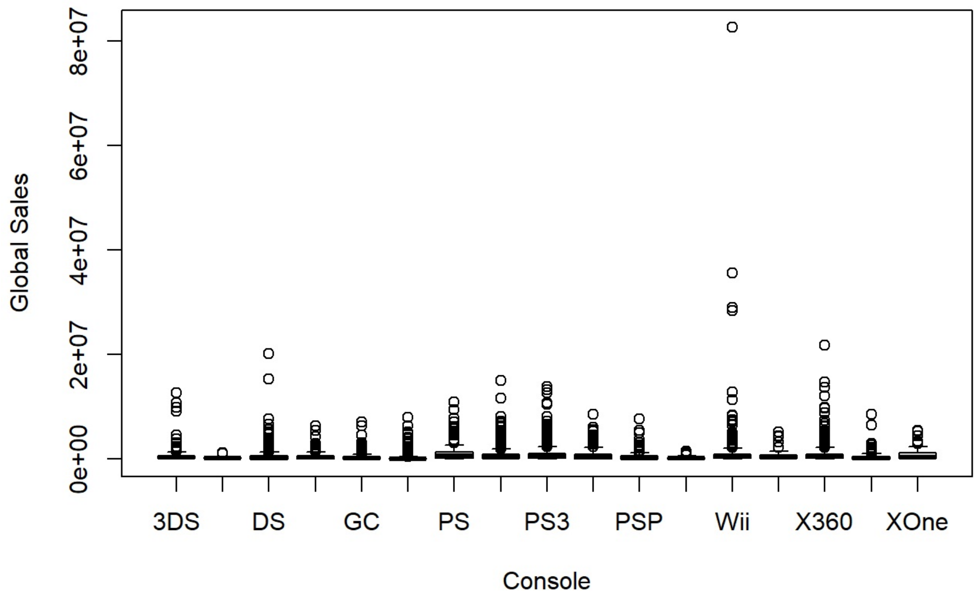
```
head(train)
```

```
##                                     Name Platform Year_of_Release    Genre
## 3940        Looney Tunes: Back in Action      PS2            2003 Platform
## 4023           Bleach: Soul Resurreccion      PS3            2011 Fighting
## 3549                                  Up      Wii            2009   Action
## 785                   Dynasty Warriors 4      PS2            2003   Action
## 7779                Conflict: Denied Ops     X360            2008  Shooter
## 4978 Naruto: Clash of Ninja Revolution 2      Wii            2008 Fighting
##                                   Publisher NA_Sales EU_Sales JP_Sales
## 3940 Warner Bros. Interactive Entertainment   250000   190000        0
## 4023                  Nippon Ichi Software   270000   100000    70000
## 3549                                   THQ   220000   280000        0
## 785                             Tecmo Koei   630000   210000  1130000
## 7779                     Eidos Interactive    80000    90000        0
## 4978                      Tomy Corporation   340000    20000        0
##      Other_Sales Global_Sales Critic_Score Critic_Count User_Score User_Count
## 3940       60000       500000           51            9         75          6
## 4023       50000       490000           58           34         72         46
## 3549       60000       560000           62            6         80          4
## 785       130000      2110000           78           24         93        201
## 7779       20000       190000           52           35         49         22
## 4978       30000       380000           72           20         84         16
##          Developer Rating
## 3940       Warthog      E
## 4023        Racjin      T
## 3549  Asobo Studio   E10+
## 785    Omega Force      T
## 7779 Pivotal Games      M
## 4978      Eighting      T
```

As is noted by the data, there are 5613 entries on video games containing information such as Title, Publisher, Release Year, Sales, ESRB ratings, and Scores.

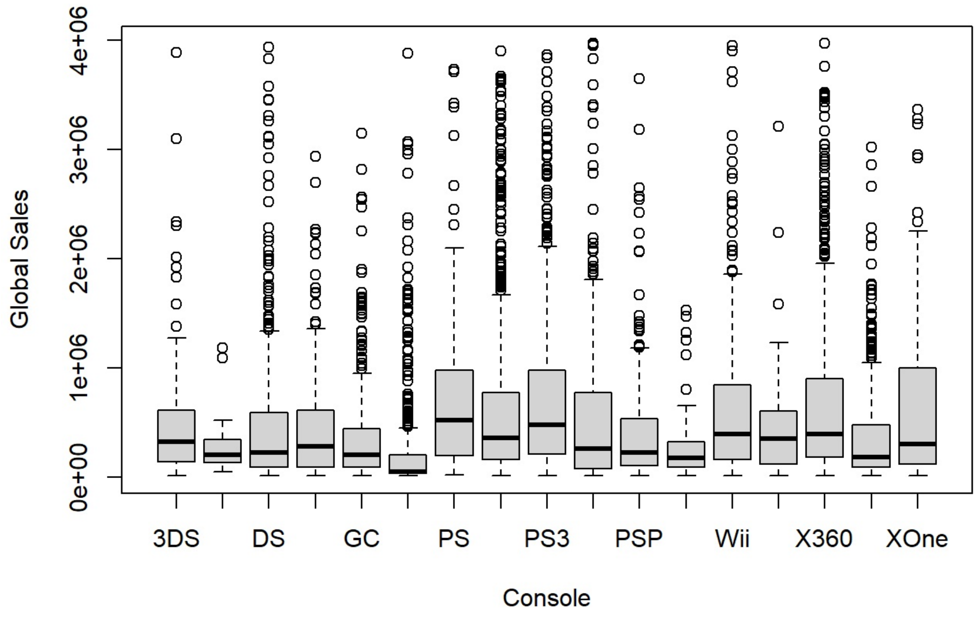
c: Next, some graphs will be used to try and make sense of the data. The first will be Global Sales as a function of the Platform the game released on. To clarify, not all games are platform exclusive and can be released on multiple different platforms.

```
plot(train$Platform, train$Global_Sales, xlab = "Console", ylab = "Global Sales")
```



It would appear there are some extreme outliers that are making the data unusable. As a guess, the cutoff point for realistic entry will be 40 million since only one game sold significantly more than that.

```
train <- subset(train, Global_Sales < 4000000)
plot(train$Platform, train$Global_Sales, xlab = "Console", ylab = "Global Sales")
```




```
levels(train$Platform)
```

```
##  [1] "3DS"  "DC"   "DS"   "GBA"  "GC"   "PC"   "PS"   "PS2"  "PS3"  "PS4"
## [11] "PSP"  "PSV"  "Wii"  "WiiU" "X360" "XB"   "XOne"
```

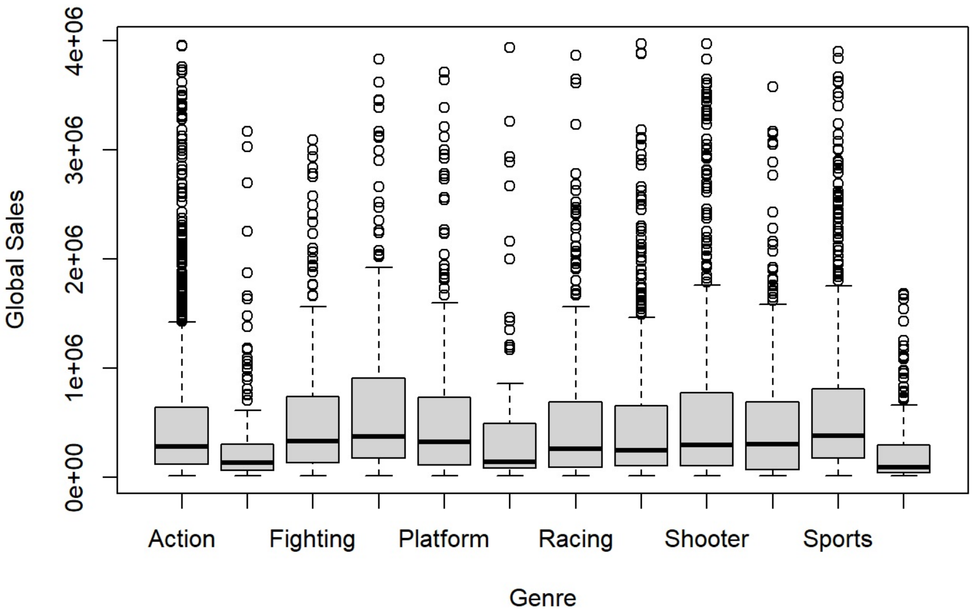
```
WiiUMedian <-(subset(train, Platform == "WiiU"))
median((WiiUMedian$Global_Sales))
```

```
## [1] 350000
```

This data is very rough, but it would appear all the consoles have a rough median of 350 thousand. Some of the larger game companies main colsoles like PlayStation, Xbox, and Nintendo have the highest overall sales. Handheld units such as the PSVita and GameBoy Advanced have less sales.

Next, sales as a function of genre will be checked.

```
plot(train$Genre, train$Global_Sales, xlab = "Genre", ylab = "Global Sales")
```




```
levels(train$Genre)
```

```
##  [1] "Action"       "Adventure"    "Fighting"     "Misc"         "Platform"
##  [6] "Puzzle"       "Racing"       "Role-Playing" "Shooter"      "Simulation"
## [11] "Sports"       "Strategy"
```

```
ActionMedian <-(subset(train, Genre == "Action"))
median((ActionMedian$Global_Sales))
```

```
## [1] 280000
```

The median is roughly consistent across genres at about 280 thousand. Genres such as Sport and surprisingly Misc sold the most. Strategy and surprisingly Adventure game sold the least.

Next, the cor functions will be used to find trends in the numerical data.

```
cor(train[6:14], use="complete")
```

```
##               NA_Sales   EU_Sales   JP_Sales Other_Sales Global_Sales
## NA_Sales     1.0000000 0.57952717 0.13927812  0.53357266    0.8949817
## EU_Sales     0.5795272 1.00000000 0.14577893  0.61162404    0.8265761
## JP_Sales     0.1392781 0.14577893 1.00000000  0.13938471    0.3566772
## Other_Sales  0.5335727 0.61162404 0.13938471  1.00000000    0.7160424
## Global_Sales 0.8949817 0.82657611 0.35667721  0.71604239    1.0000000
## Critic_Score 0.2984213 0.27508787 0.13214737  0.21343413    0.3319263
## Critic_Count 0.3081760 0.32196489 0.14806516  0.22715531    0.3601730
## User_Score   0.1140989 0.07773282 0.14893739  0.05624837    0.1324579
## User_Count   0.1397130 0.30167729 0.02363287  0.17707858    0.2241563
##              Critic_Score Critic_Count User_Score User_Count
## NA_Sales        0.2984213    0.3081760 0.11409894 0.13971300
## EU_Sales        0.2750879    0.3219649 0.07773282 0.30167729
## JP_Sales        0.1321474    0.1480652 0.14893739 0.02363287
## Other_Sales     0.2134341    0.2271553 0.05624837 0.17707858
## Global_Sales    0.3319263    0.3601730 0.13245788 0.22415635
## Critic_Score    1.0000000    0.3679886 0.58280018 0.23261475
## Critic_Count    0.3679886    1.0000000 0.18886315 0.28350284
## User_Score      0.5828002    0.1888631 1.00000000 0.02333167
## User_Count      0.2326147    0.2835028 0.02333167 1.00000000
```

NA and EU game sales have a decent cor of .57. This is most likely due to the cultures of the two being similar and thus having similar taste in video games.

Critic score and user score seem to have a decent correlation of .58. This is most likely due to critics grading without bias and knowing more about what games ought to provide. A casual reviewer will be more influenced by their lack of knowledge and just how they felt playing the game.

Unfortunately, it doesn't appear as if user and critic scores have much influence on sales. Critics scores only have a .33 cor and user only have a .13 cor.

d: Unfortunately, it doesn't seem as there are many useful relations with the data set. Even though NA sales and global sales are strongly correlated, the issue is NA sales are being recorded directly within global sales. Critic and user reviews also occur independely of each other, so its not realisitic to put either as a funciton of the ohter.

Even though the correlation is poor, sales and a function of critic reviews will be tried as its the only logical connection.

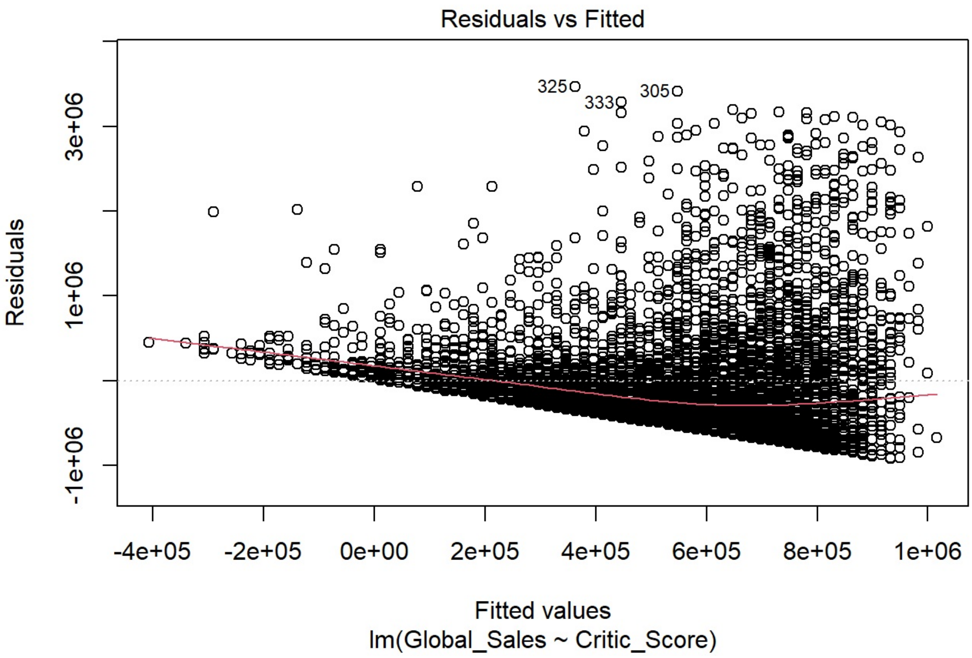
```
lm1 <- lm(Global_Sales~Critic_Score, data=train)
summary(lm1)
```

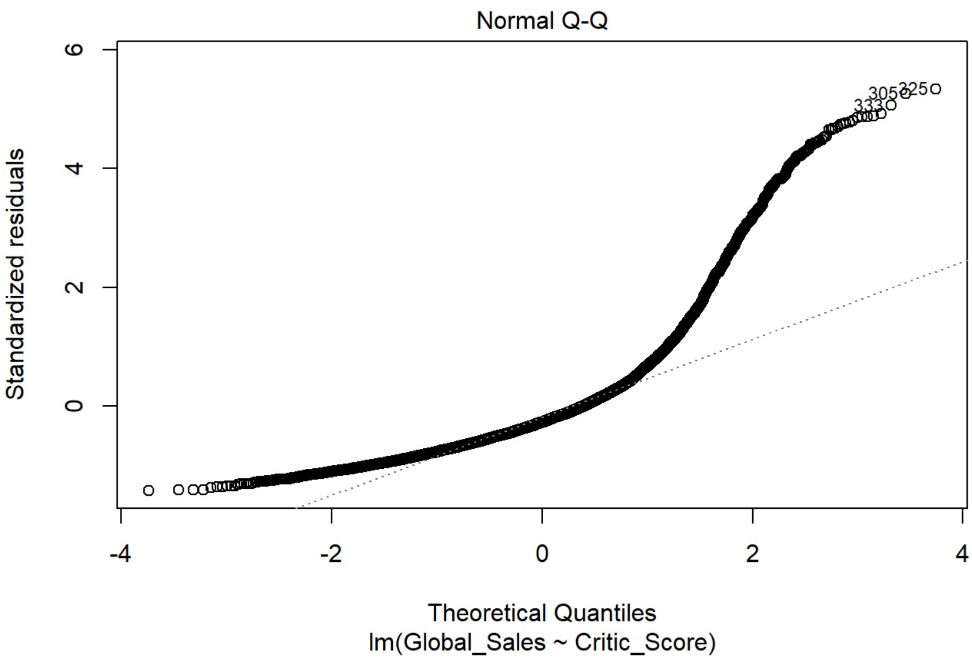
```
##
## Call:
## lm(formula = Global_Sales ~ Critic_Score, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -922200 -406753 -173745  167596 3467231
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -625359.5    45837.5  -13.64   <2e-16 ***
## Critic_Score   16747.9      644.3   25.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 649300 on 5457 degrees of freedom
## Multiple R-squared:  0.1102, Adjusted R-squared:   0.11
## F-statistic: 675.7 on 1 and 5457 DF,  p-value: < 2.2e-16
```

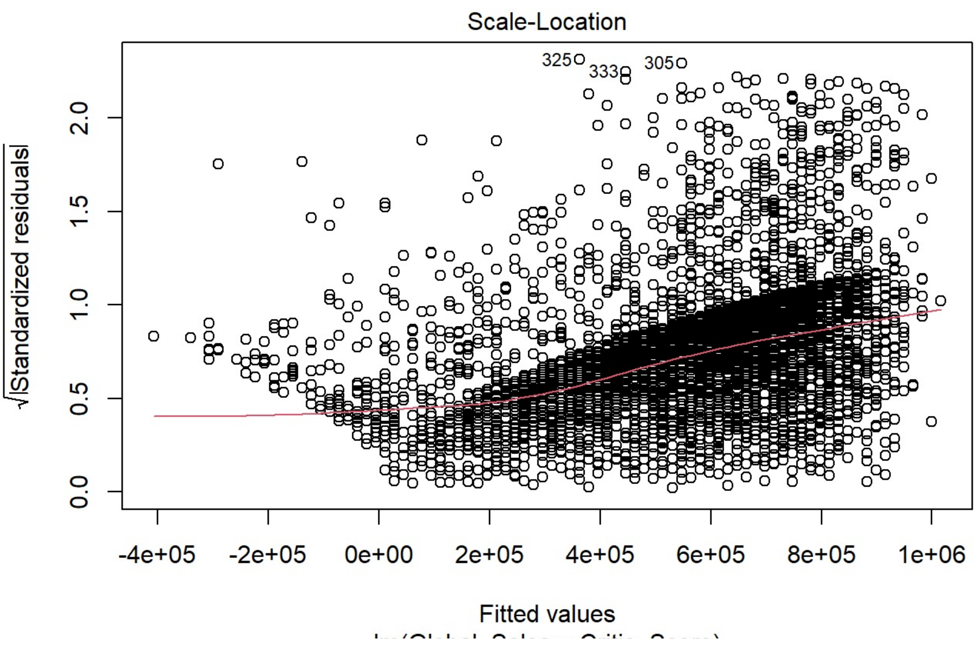
As expected, the min and max from the data vary by several million, and Q1 and Q3 vary by hundreds of thousands. The model suggests a roughly 16 thousand sales increase per critic point awarded with an error range of about half a million. The R squared accuracy is only .11, which is awful. At least the p value is low while the F statistic is high.

e: Even though we know the data is awful, a residual model will be used next.

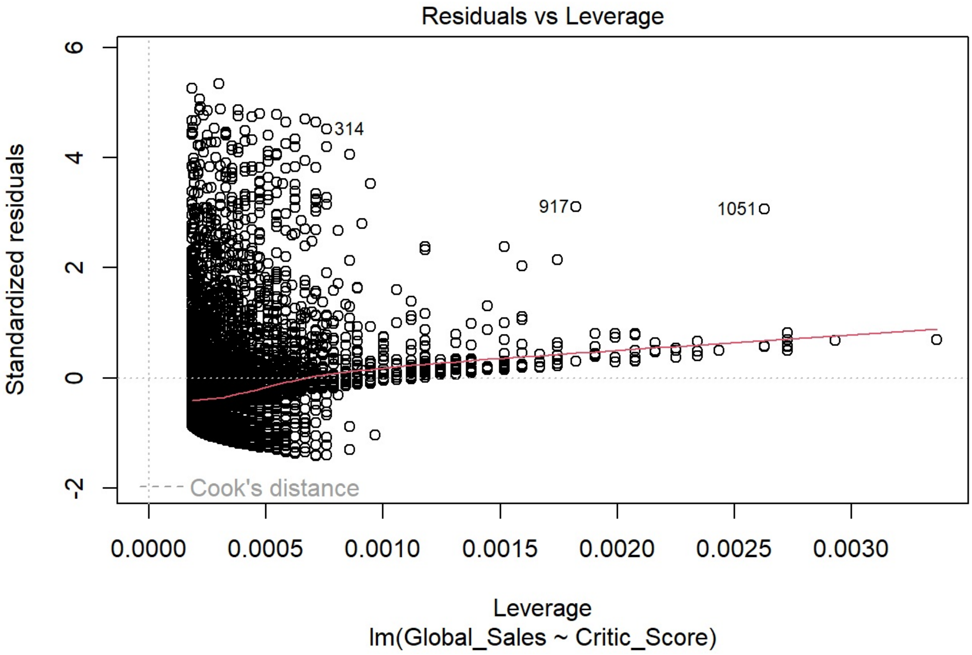
```
plot(lm1)
```


Residuals vs Fitted
Residuals
-1e+06
1e+06
3e+06
325
333
305
-4e+05
-2e+05
0e+00
2e+05
4e+05
6e+05
8e+05
1e+06
Fitted values
lm(Global_Sales ~ Critic_Score)


Normal Q-Q
Standardized residuals
0
2
4
6
305
325
333
-4
-2
0
2
4
Theoretical Quantiles
lm(Global_Sales ~ Critic_Score)


Scale-Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
325
333
305
-4e+05
-2e+05
0e+00
2e+05
4e+05
6e+05
8e+05
1e+06
Fitted values

lm(Global_Sales ~ Critic_Score)



In residual vs. fitted, it doesn't seem very useful. The red line follows no distinct pattern and The spread of residuals above and below the line aren't even.

In normal Q-Q, as expected, the residuals wildly fly off the line at the 1 mark. This means the residuals aren't normally distributed. It makes sense, considering the abundant amount of outliers in the sales above 1 million.

In scale location, the line starts off decently horizontal, but begins to grow rapidly. Also, the spread is not great. It starts off almost exclusively above the line, but then is mostly below the line.

In residuals vs. leverage, the graph has a lot of influential points since half the data cuts through one of cook's lines.

As expected, the 4 graphs denote a terrible model.

f: Next, the platform will be included to develop a multiple liner regression model.

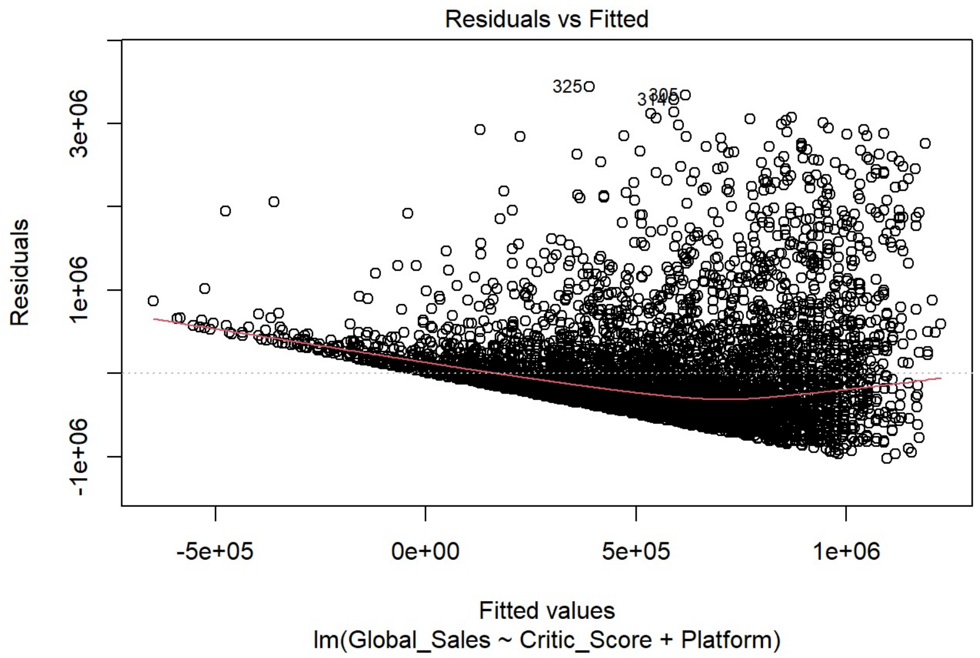
```
lm2 <- lm(Global_Sales~Critic_Score+Platform, data=train)
summary(lm2)
```

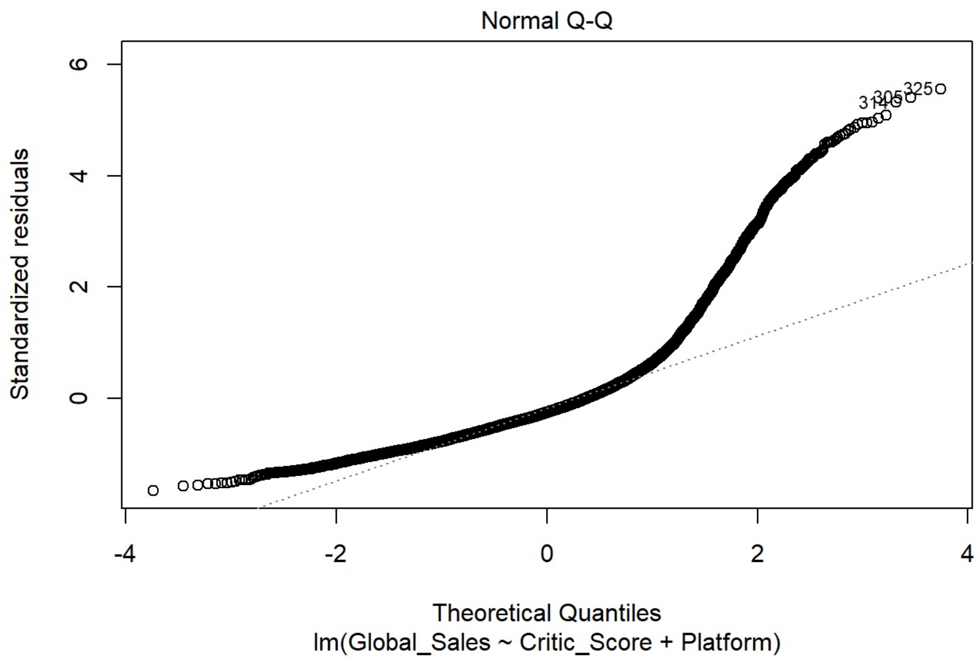
```
##
## Call:
## lm(formula = Global_Sales ~ Critic_Score + Platform, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1026450  -377974  -156053   164029  3440888
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -795845.2    70456.7 -11.296  < 2e-16 ***
## Critic_Score    19225.3      631.3  30.454  < 2e-16 ***
## PlatformDC    -532201.1   181072.7  -2.939 0.003305 **
## PlatformDS      50663.6    64538.3   0.785 0.432479
## PlatformGBA    -91686.8    71428.1  -1.284 0.199328
## PlatformGC    -183501.6    66835.7  -2.746 0.006061 **
## PlatformPC    -421095.3    62019.8  -6.790 1.24e-11 ***
## PlatformPS     155797.7    80237.4   1.942 0.052224 .
## PlatformPS2    113586.7    59569.1   1.907 0.056598 .
## PlatformPS3    181242.9    61265.4   2.958 0.003107 **
## PlatformPS4     67723.5    71372.7   0.949 0.342729
## PlatformPSP    -92447.1    65871.6  -1.403 0.160541
## PlatformPSV   -314549.6    82993.2  -3.790 0.000152 ***
## PlatformWii    195963.0    64687.9   3.029 0.002462 **
## PlatformWiiU   -86323.8    93194.4  -0.926 0.354343
## PlatformX360   174174.0    60717.2   2.869 0.004139 **
## PlatformXB    -216947.6    62969.6  -3.445 0.000575 ***
## PlatformXOne    74272.8    78427.9   0.947 0.343671
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 619500 on 5441 degrees of freedom
## Multiple R-squared:  0.1922, Adjusted R-squared:  0.1897
## F-statistic: 76.15 on 17 and 5441 DF,  p-value: < 2.2e-16
```

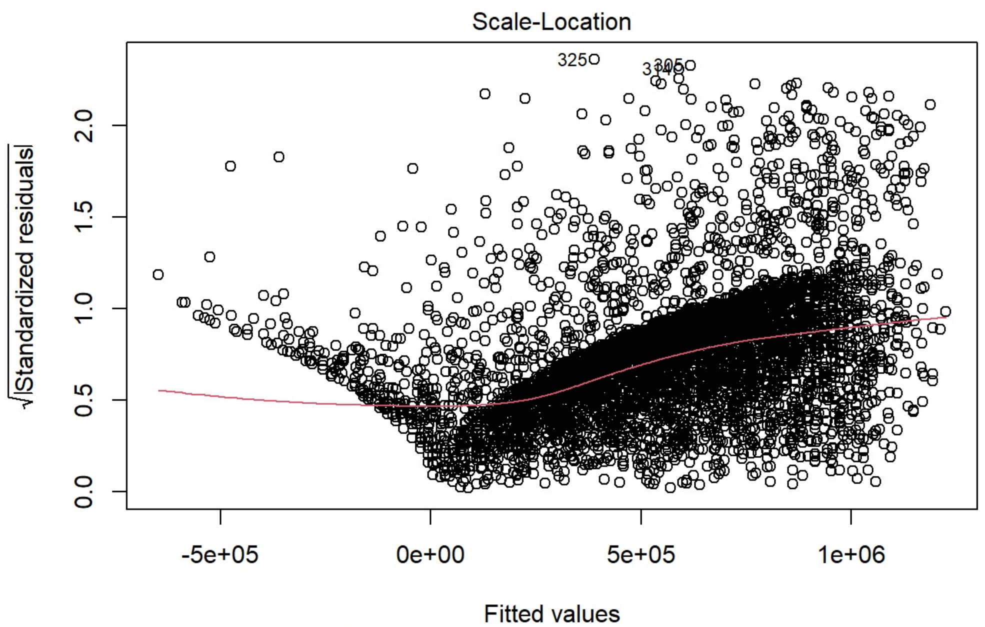
Overall, the model saw a slight improvement. Most notably, the Adjusted R-squared value increased to about .19. Its still not a good model, but it has improved. The residual error also reduced by tens of thousands. The range of min and max didn't improve much. Our F statistic went down, but the p value remained low.

Now the residual will be plotted.

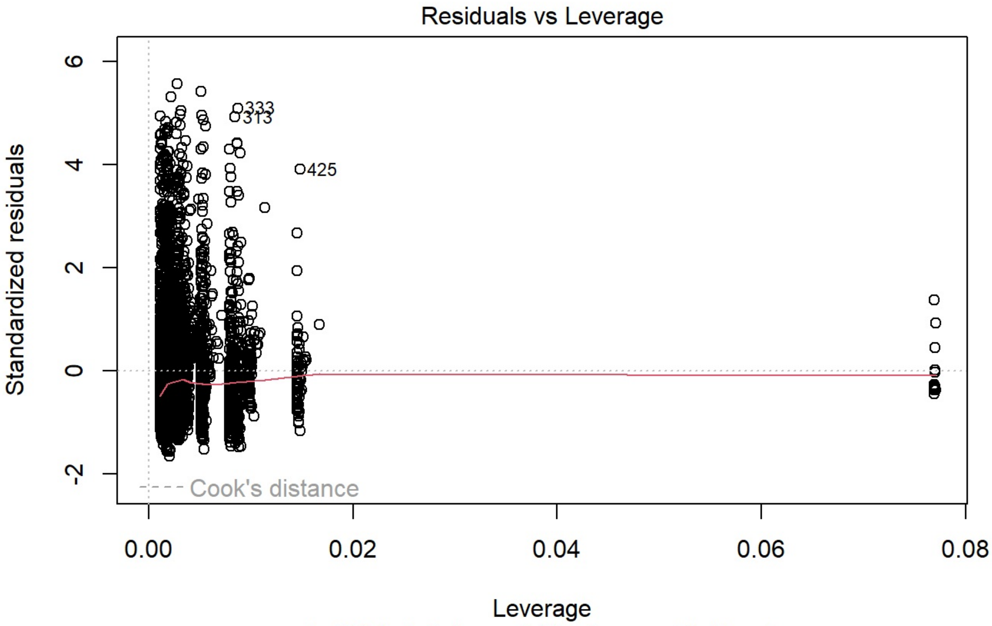
```
plot(lm2)
```


Residuals vs Fitted
Residuals
Fitted values
lm(Global_Sales ~ Critic_Score + Platform)
325
305
314


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Global_Sales ~ Critic_Score + Platform)
325
305
314


Scale-Location
√|Standardized residuals|
Fitted values
325
305
314

lm(Global_Sales ~ Critic_Score + Platform)



lm(Global_Sales ~ Critic_Score + Platform)

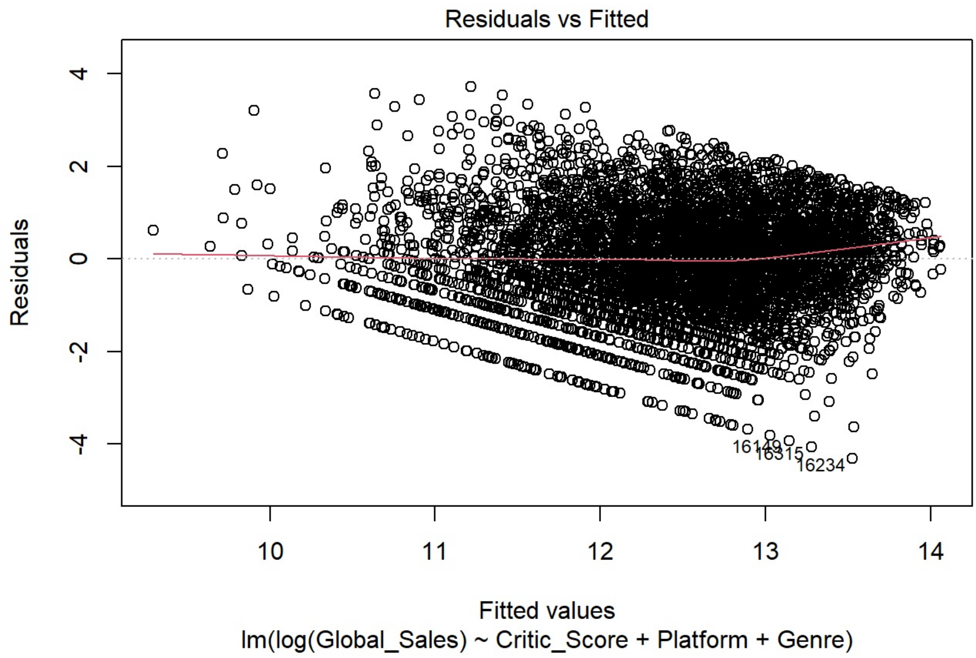
Unfortunately, all of the new graphs resemble the old graphs and have the same issues as described above. Despite the small improvement, the model is still bad, as suspected.

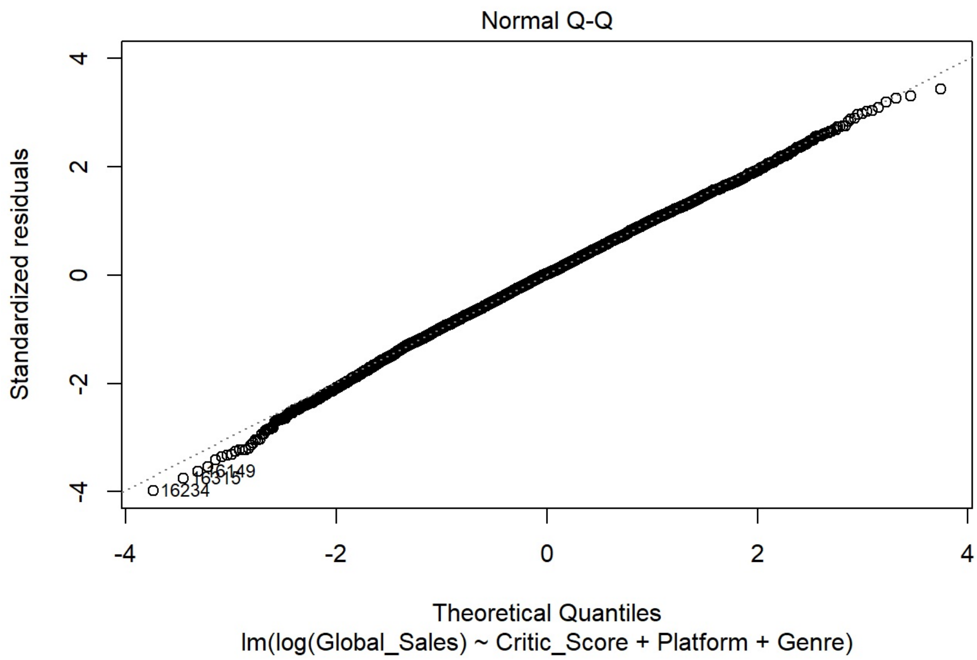
g:

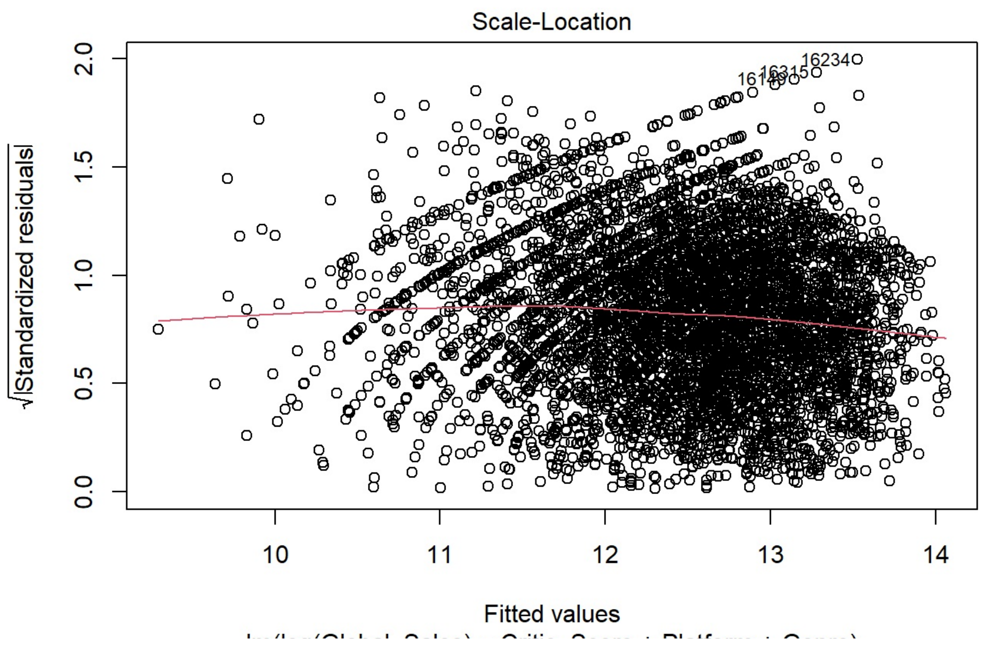
```
lm3 <- lm(log(Global_Sales)~Critic_Score+Platform+Genre, data=train)
summary(lm3)
```

```
##
## Call:
## lm(formula = log(Global_Sales) ~ Critic_Score + Platform + Genre,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3166 -0.7136  0.0200  0.7381  3.7162
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       10.058361   0.124887  80.540  < 2e-16 ***
## Critic_Score       0.038572   0.001123  34.354  < 2e-16 ***
## PlatformDC        -0.927697   0.317829  -2.919 0.003528 **
## PlatformDS        -0.057235   0.113643  -0.504 0.614538
## PlatformGBA       -0.323553   0.125631  -2.575 0.010038 *
## PlatformGC        -0.492399   0.117854  -4.178 2.99e-05 ***
## PlatformPC        -1.563879   0.110103 -14.204  < 2e-16 ***
## PlatformPS         0.212093   0.140918   1.505 0.132363
## PlatformPS2        0.144849   0.105071   1.379 0.168082
## PlatformPS3        0.313780   0.108018   2.905 0.003689 **
## PlatformPS4       -0.336518   0.125511  -2.681 0.007358 **
## PlatformPSP       -0.262939   0.115896  -2.269 0.023323 *
## PlatformPSV       -0.698278   0.145694  -4.793 1.69e-06 ***
## PlatformWii        0.322858   0.113888   2.835 0.004601 **
## PlatformWiiU      -0.268102   0.163642  -1.638 0.101408
## PlatformX360       0.239864   0.107176   2.238 0.025258 *
## PlatformXB        -0.583072   0.111546  -5.227 1.79e-07 ***
## PlatformXOne      -0.251256   0.138195  -1.818 0.069100 .
## GenreAdventure    -0.589257   0.081135  -7.263 4.33e-13 ***
## GenreFighting     -0.070778   0.069312  -1.021 0.307228
## GenreMisc          0.129744   0.069630   1.863 0.062470 .
## GenrePlatform      0.008523   0.068722   0.124 0.901304
## GenrePuzzle       -0.551773   0.117243  -4.706 2.59e-06 ***
## GenreRacing       -0.158295   0.059391  -2.665 0.007714 **
## GenreRole-Playing -0.211605   0.055544  -3.810 0.000141 ***
## GenreShooter       0.018727   0.051860   0.361 0.718027
## GenreSimulation    0.019945   0.077922   0.256 0.797994
## GenreSports       -0.046738   0.051004  -0.916 0.359514
## GenreStrategy     -0.593321   0.081109  -7.315 2.95e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.086 on 5430 degrees of freedom
## Multiple R-squared:  0.3126, Adjusted R-squared:  0.309
## F-statistic: 88.18 on 28 and 5430 DF,  p-value: < 2.2e-16
```

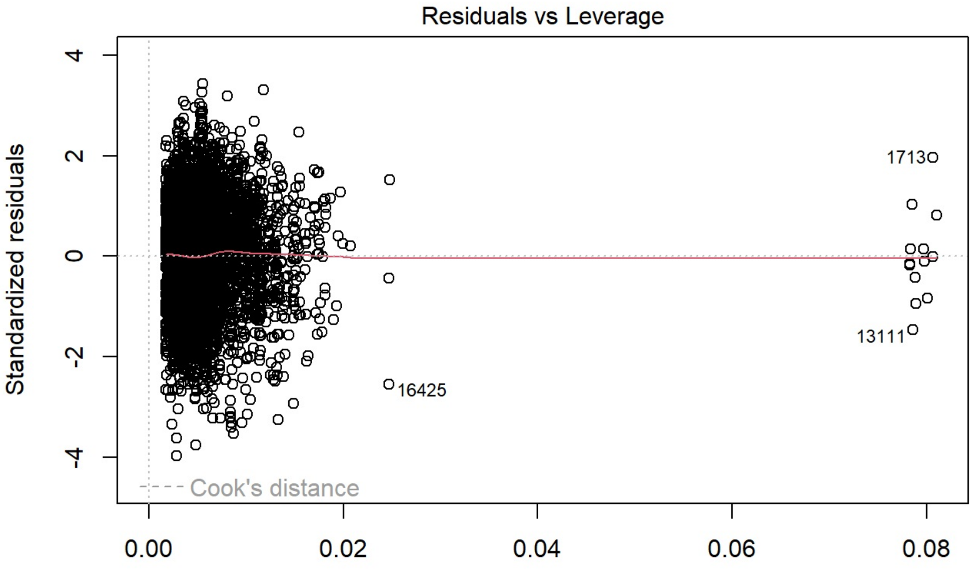
```
plot(lm3)
```


Residuals vs Fitted
Residuals
Fitted values
lm(log(Global_Sales) ~ Critic_Score + Platform + Genre)
16149
16315
16234


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(log(Global_Sales) ~ Critic_Score + Platform + Genre)
16149
16315
16234


Scale-Location
√|Standardized residuals|
Fitted values
16234
16315
16149

lm(log(Global_Sales) ~ Critic_Score + Platform + Genre)



lm(log(Global_Sales) ~ Critic_Score + Platform + Genre)

h: The third model is by far the best one. It added genre as a factor, and used the log of the sales.

The variation from the median is still large (or at least, large for a log function anyways). However, the residual error decreased a bit while the Adjusted R-squared jumped up to .30. Still not great, but its an improvement over .11 and .19 of the first two models. Its unfortunate that our F statistic keeps getting lower, but the p value remains low as well.

As far as residuals go, the third also shows great improvements.

Residuals vs. fitted: The line becomes almost perfectly horizontal (except for the end), and has a roughly symmetrical distribution. This indicates the linear relationship is a good match.

Normal Q-Q: The residuals are plotted pretty well against the line. This means they are distributed fairly normally.

Scale location: This one is only so-so. The line in the middle is decently horizontal, and distribution follows a roughly ovular distribution around the line. It’s by no means great, but a patters can sort of be recognized. This would mean the residual are spread out decently among the range of predictors.

Residuals vs Leverage: Unfortunately, all the data seems to be split by a line. This means several factors have too much sway over the data.

Overall, the third model shows many improvements over the other 2.

i. Now lm3 will be used to compare predictions with th e test data.

```
pred1 <- predict(lm1, newdata = test)
corr1 <- cor(pred1, test$Global_Sales)
mse1 <- mean((pred1-test$Global_Sales)^2)
rmse1 <- sqrt(mse1)

print("lm1")
```

```
## [1] "lm1"
```

```
print(paste("correlation: ", corr1))
```

```
## [1] "correlation:  0.270310200602469"
```

```
print(paste("mse: ", mse1))
```

```
## [1] "mse:  5198864194363.64"
```

```
print(paste("rmse: ", rmse1))
```

```
## [1] "rmse:  2280101.79473717"
```

```
pred2 <- predict(lm2, newdata = test)
corr2 <- cor(pred2, test$Global_Sales)
mse2 <- mean((pred2-test$Global_Sales)^2)
rmse2 <- sqrt(mse2)

print("lm2")
```

```
## [1] "lm2"
```

```
print(paste("correlation: ", corr2))
```

```
## [1] "correlation:  0.335563801521632"
```

```
print(paste("mse: ", mse2))
```

```
## [1] "mse:  5039282962851.88"
```

```
print(paste("rmse: ", rmse2))
```

```
## [1] "rmse:  2244834.72951839"
```

```
pred3 <- predict(lm3, newdata = test)
corr3 <- cor(pred3, test$Global_Sales)
mse3 <- mean((pred3-test$Global_Sales)^2)
rmse3 <- sqrt(mse3)

print("lm3")
```

```
## [1] "lm3"
```

```
print(paste("correlation: ", corr3))
```

```
## [1] "correlation:  0.322831823853248"
```

```
print(paste("mse: ", mse3))
```

```
## [1] "mse:  6108345435624.09"
```

```
print(paste("rmse: ", rmse3))
```

```
## [1] "rmse:  2471506.71365143"
```

The results were a bit unexpected.

The second model had the highest correlation with the test data, with the third model not too far behind it. Though .33 still isn’t great.

The third model had a significantly worse mse than the other two.

The rmse was roughly the same, but worse in the third model.

The second model must be the best mix of increasing complexity while maintaining accuracy with the randomness of the data. It had two factors, which was in between the 1 and 3 factors of the others.It also didn’t take the log of the values, which seems to have made the errors grow larger.

My guess is overall the data itself doesn’t have enough information to successfully determine video game sales. It makes sense because things such as the actual price of the game, if games of a similar type release at the same time, if sufficient advertising was done, if the game can only be sold on certain consoles, etc. will affect sales. Considering all of that, I think its reasonable to assume reviews will only influence about 30% of a games sales.