ML Portfolio 2: Linear Regression

Bushra Rahman

How Linear Regression Works

Linear regression is a quantitative regression algorithm that models data on the equation *y* = *wx* + *b*, where *x* represents the predictor value(s) and *y* represents the target value(s). The linear regression algorithm calculates the line that best models *y* as a function of *x*, the line of best fit. Various evaluation metrics, such as correlation and RMSE, can then be used to analyze how accurately the linear model fits the data. Linear regression works well on data that follows a linear trend, but because it is a high-bias low-variance algorithm, it is prone to underfitting on data that isn't linear.

Linear Regression on VGSales

The CSV file **"vgsales"** is a dataset of video games with sales greater than 100,000 copies, scraped from the VGChartz Network and uploaded to Kaggle (URL https://www.kaggle.com/datasets/gregorut/videogamesales (https://www.kaggle.com/datasets/gregorut/videogamesales)). This dataset contains 16,598 rows of video game titles and 11 columns of variables. Of those 11 variables, 5 are sales in the millions in different regions, making them quantitative variables and thus suitable for regression. Two quantitative variables in particular seem interesting: **North American Sales** and **Global Sales**. It feels accurate to claim that video games that sell well in North America will sell well globally, thus implying a linear relationship. Using linear regression, this claim can be further analyzed.

Train and Test Sets

After importing the CSV file into a data frame, it needs to be divided into a train set and a test set. By setting a seed and randomly sampling the row indices into a vector, a train set can be created that contains 80% of the rows. The remaining 20% are then put into the test set.

```
#import data
setwd("C:/Users/Bushra/CS4375/Portfolio2")
vgsales <- read.csv("vgsales.csv")
set.seed(1234)
i <- sample(1:nrow(vgsales), nrow(vgsales)*0.8, replace=FALSE)
train <- vgsales[i,]
test <- vgsales[-i,]
```

Data Exploration

Now that the training data has been procured, it can be explored using R's built-in data exploration functions. The 5 functions used below are dim(), str(), summary(), head(), and tail().

The **dim()** function returns the row and column dimensions of the data frame, which are 13278 and 11 respectively. This is because 13278 is 80% of 16598, the size of the entire vgsales dataset.

```
dim(train)
```

```
## [1] 13278    11
```

The **str()** function displays information on the data frame’s structure, like the column names, their data type, and the first few entries in each column. This output shows that Rank is an integer vector, the 5 Sales columns are numerical vectors, and the rest of the columns are character string vectors.

```
str(train)
```

```
## 'data.frame':    13278 obs. of  11 variables:
##  $ Rank        : int  7453 8017 7163 8087 9197 623 15243 10886 935 12689 ...
##  $ Name        : chr  "Mortal Kombat: Special Forces" "Reel Fishing II" "Dark Souls II" "Batt
le Commander: Hachibushu Shura no Heihou" ...
##  $ Platform    : chr  "PS" "PS" "XOne" "SNES" ...
##  $ Year        : chr  "2000" "2000" "2015" "1991" ...
##  $ Genre       : chr  "Fighting" "Sports" "Role-Playing" "Strategy" ...
##  $ Publisher   : chr  "Midway Games" "Victor Interactive" "Namco Bandai Games" "Banpresto"
...
##  $ NA_Sales    : num  0.12 0.1 0.13 0 0.11 1.15 0 0.09 0.85 0.04 ...
##  $ EU_Sales    : num  0.08 0.07 0.07 0 0.03 1.14 0 0 0.71 0.01 ...
##  $ JP_Sales    : num  0 0 0 0.18 0 0.06 0.02 0 0.13 0 ...
##  $ Other_Sales : num  0.01 0.01 0.02 0 0 0.13 0 0.01 0.16 0 ...
##  $ Global_Sales: num  0.21 0.18 0.22 0.18 0.14 2.48 0.02 0.09 1.86 0.06 ...
```

The **summary()** function returns statistical information on each column, like its mean and its count of NA entries. At a glance, this function lists the means (in millions) of each sale: 0.2627 for **NA_Sales**, 0.1456 for **EU_Sales**, 0.07825 for **JP_Sales**, 0.0483 for **Other_Sales**, and 0.5352 for **Global_Sales**. Because of the seed, the training data and its means will remain the same on different runs of the program. The summary() function also shows no NA entries, which provides reassurance in the reliability of the data.

```
summary(train)
```

```
##       Rank           Name             Platform           Year          
##  Min.   :    1   Length:13278       Length:13278       Length:13278      
##  1st Qu.: 4156   Class :character   Class :character   Class :character  
##  Median : 8324   Mode  :character   Mode  :character   Mode  :character  
##  Mean   : 8314                                                           
##  3rd Qu.:12444                                                           
##  Max.   :16600                                                           
##     Genre             Publisher            NA_Sales          EU_Sales      
##  Length:13278       Length:13278       Min.   : 0.0000   Min.   : 0.0000  
##  Class :character   Class :character   1st Qu.: 0.0000   1st Qu.: 0.0000  
##  Mode  :character   Mode  :character   Median : 0.0800   Median : 0.0200  
##                                        Mean   : 0.2627   Mean   : 0.1456  
##                                        3rd Qu.: 0.2400   3rd Qu.: 0.1100  
##                                        Max.   :41.4900   Max.   :29.0200  
##     JP_Sales         Other_Sales        Global_Sales    
##  Min.   : 0.00000   Min.   : 0.0000   Min.   : 0.0100  
##  1st Qu.: 0.00000   1st Qu.: 0.0000   1st Qu.: 0.0600  
##  Median : 0.00000   Median : 0.0100   Median : 0.1700  
##  Mean   : 0.07825   Mean   : 0.0483   Mean   : 0.5352  
##  3rd Qu.: 0.04000   3rd Qu.: 0.0400   3rd Qu.: 0.4700  
##  Max.   :10.22000   Max.   :10.5700   Max.   :82.7400  
```

Finally, the **head()** and **tail()** functions display the first and last few rows of the training data. This output provides a glimpse at how the actual data is organized in table format.

```
head(train)
```

	R…	Name	Platform	Y…	Genre
	<int>	<chr>	<chr>	<chr>	<chr>
7452	7453	Mortal Kombat: Special Forces	PS	2000	Fighting
8016	8017	Reel Fishing II	PS	2000	Sports
7162	7163	Dark Souls II	XOne	2015	Role-Playing
8086	8087	Battle Commander: Hachibushu Shura no Heihou	SNES	1991	Strategy
9196	9197	NFL Blitz 20-03	GC	2002	Sports
623	623	Tomb Raider: The Last Revelation	PS	1998	Action

6 rows | 1-6 of 12 columns

```
tail(train)
```

	Rank	Name	Platform	Y…	Genre
	<int>	<chr>	<chr>	<chr>	<chr>
6242	6243	Tak and the Guardians of Gross	PS2	2008	Action

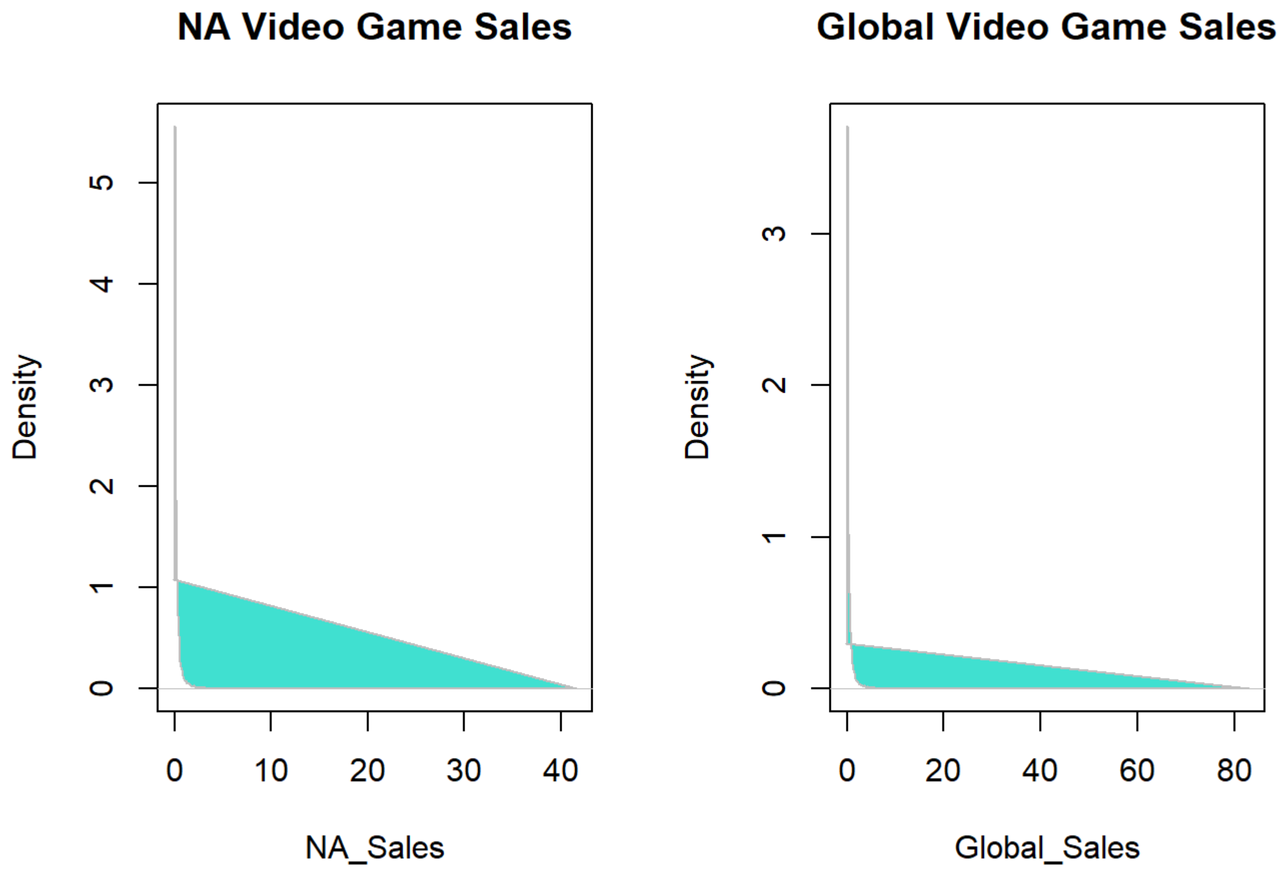
	Rank <int>	Name <chr>	Platform <chr>	Y… <chr>	Genre <chr>
15562	15564	Lovely x Cation 1 & 2	PSV	2015	Action
15244	15246	GunParade Orchestra: Midori no Shou	PS2	2006	Adventure
4609	4610	Vanquish	X360	2010	Shooter
1744	1745	Final Fantasy Tactics: The War of the Lions	PSP	2007	Role-Playing
15463	15465	Crysis: Warhead	PC	2008	Shooter

6 rows | 1-6 of 12 columns

Informative Graphs

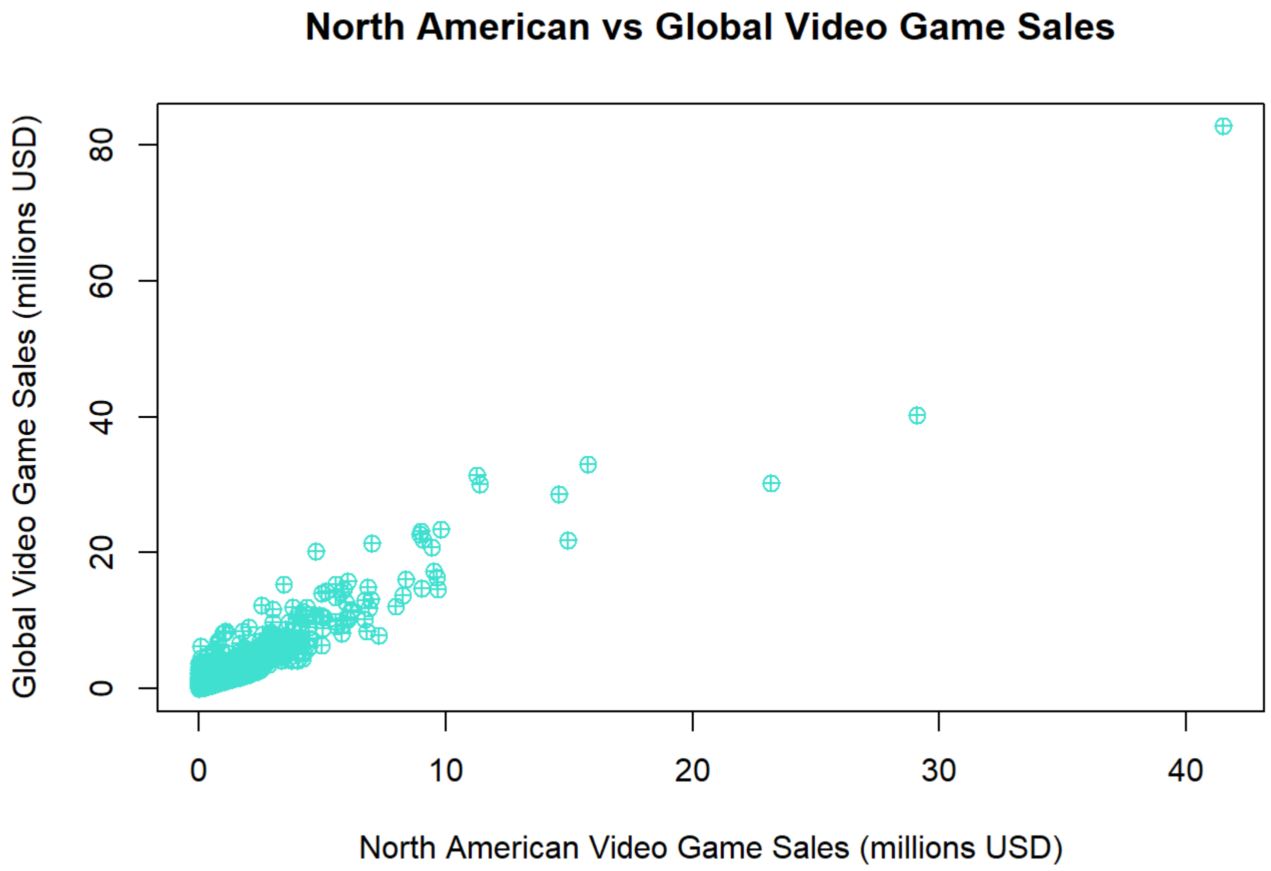
Graphs can be used to visually represent the data in different columns. Below are 2 **kernel density plots** for each target column, and a **scatter plot** for them both. Kernel density plots depict the distribution of the data over a continuous interval, with smoothing applied. The kernel density plots for NA_Sales and Global_Sales both seem to follow a similar distribution, with a far denser aggregation of lower sales and steadily sparser aggregations of higher sales.

```
par(mfrow=c(1,2)) # Change the panel layout to 1x2

dens <- density(train$NA_Sales)
plot(dens, main="NA Video Game Sales", xlab="NA_Sales")
polygon(dens, col="turquoise", border="gray")

dens <- density(train$Global_Sales)
plot(dens, main="Global Video Game Sales", xlab="Global_Sales")
polygon(dens, col="turquoise", border="gray")
```



Scatter plots are commonly used to plot 2 quantitative vectors against each other. At a glance, there appears to be a linear relationship between the two variables. A few leverage points can be noticed as well, like the single entry that made 40mil in North America and 80mil globally, but they also seem to align with the linear distribution.

```
par(mfrow=c(1,1)) # Change the panel layout to 1x1
plot(train$NA_Sales, train$Global_Sales, pch=10, cex=1.2, col="turquoise", main="North American
vs Global Video Game Sales", xlab="North American Video Game Sales (millions USD)", ylab="Global
Video Game Sales (millions USD)")
```



Simple Linear Regression

Simple linear regression is done using one predictor variable on the target. In this case, the predictor is NA_Sales and the target is Global_Sales, so the function call lm() should model Global_Sales as a function of NA_Sales. The model is stored in lm1, which is then displayed. The display shows the estimated coefficients: **b = 0.05231** and **w = 1.83781**. The intercept parameter *b* is just used for fitting the data, but the slope parameter *w* holds more information: it means that for every 1 unit change in *x*, there is an estimated 1.84 unit change in *y*. Since unit is millions for this dataset, that means that for every 1mil earned in North American sales, 1.84mil was earned in global sales.

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

```
##
## Call:
## lm(formula = Global_Sales ~ NA_Sales, data = train)
##
## Coefficients:
## (Intercept)     NA_Sales
##     0.05231      1.83781
```

This information can be used in the equation **w ∗ NA_Sales + b = Global_Sales** to check on some example data points. For example, going back to the output of head(), the first entry *Mortal Kombat: Special Forces* is shown with 0.12 in NA_Sales and 0.21 in Global_Sales. These values can be substituted into the equation the check its accuracy: **1.84** ∗ **0.12 + 0.05 = 0.2708**, which is not too far off from 0.21. Naturally, however, there is some residual.

Model Summary

The overloaded summary() function can be used to output the summary of a linear model's metrics. This information is used to evaluate the accuracy of the model.

```
summary(lm1)
```

```
##
## Call:
## lm(formula = Global_Sales ~ NA_Sales, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2558  -0.1026  -0.0423   0.0277  11.4381
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.052315   0.004636   11.29   <2e-16 ***
## NA_Sales    1.837811   0.005475  335.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5078 on 13276 degrees of freedom
## Multiple R-squared:  0.8946, Adjusted R-squared:  0.8946
## F-statistic: 1.127e+05 on 1 and 13276 DF,  p-value: < 2.2e-16
```

The first part of the summary is the **Residuals**. These are the errors: the distance from each data point to the regression line, given by the equation **residual = predicted - actual**. The residuals range from -13.2558 to 11.4381, which aren't too large when considering that there are over 10k actual data points.

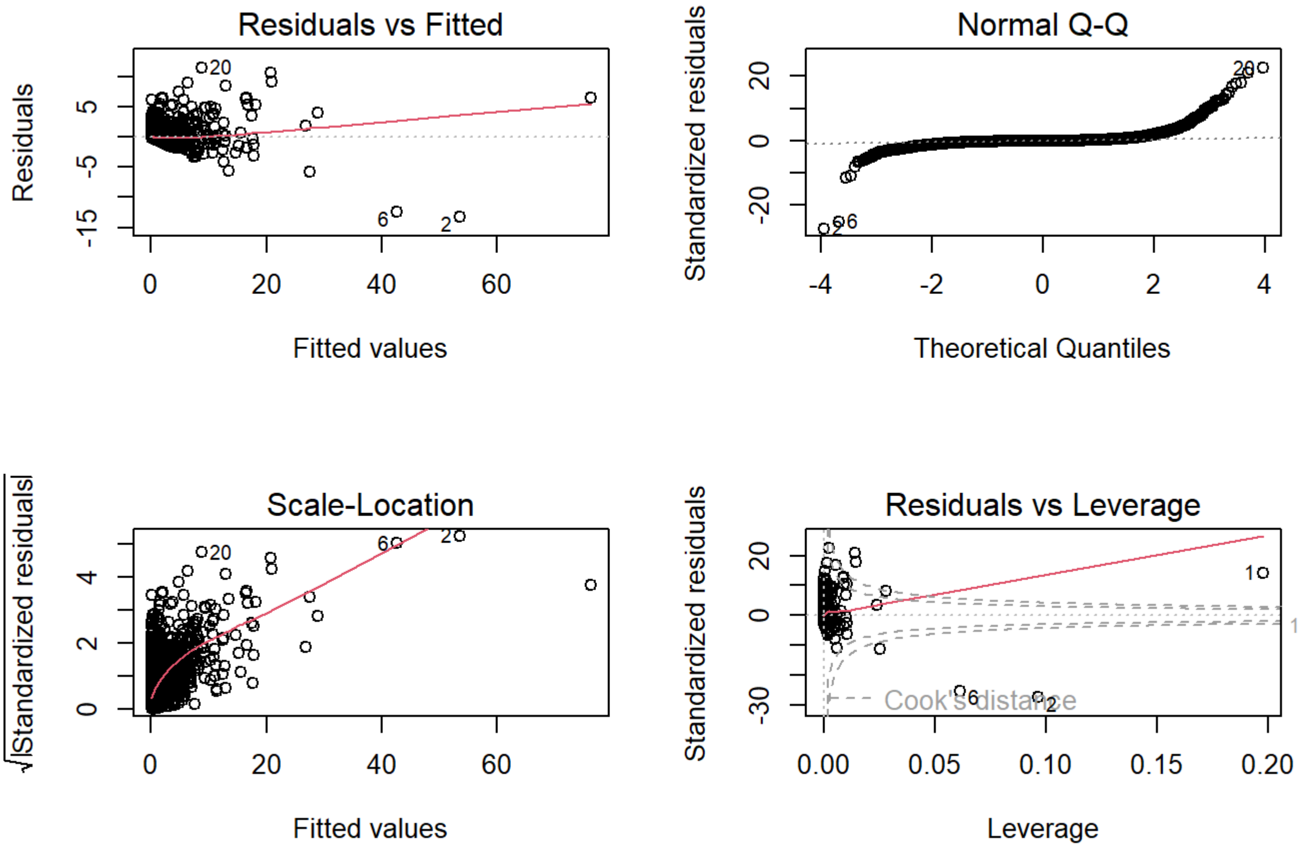
The second part of the summary is the **Coefficients**. The first column, the **Estimates**, matches the coefficients outputed by lm1 from earlier. The second column, the **Standard Error**, is the estimate of variation and is used to predict confidence intervals. The third column, the **t-value**, is the number of standard deviations between the estimated coefficient and 0. A t-value of 0 implies no relationship between the variables — so since the t-values in the summary aren't 0, a relationship does exist. The last column, the **p-value**, is the probability of seeing a larger t-value under the assumption that the null hypothesis is true. The p-value needs to be small enough, typically less than 0.05, in order for the null hypothesis to be rejectable. The significant codes act as visual aids for this: if the p-values are accompanied by 1, 2, or 3 asterisks, then that means they are between 0 and 0.05, and thus are small enough. The p-values in the summary are accompanied by 3 asterisks, which is ideal.

Finally, the third part of the summary is the **statistical information** on how well the model fit the data. The **RSE**, or residual standard error, is calculated from the RSS, or residual sum of squares. The RSE measures the lack of fit of the model in units of y. So the given RSE of 0.5078 means that the model was off from the data by 0.5078 million dollars. Next, **R-squared** is a value between 0 and 1 that represents how well the variance in the model is explained by the predictors. In simple linear regression, this is equal to the correlation squared. Because the R-squared in the summary is 0.8946, very close to 1, it means that nearly all variation in Global_Sales can be predicted by NA_Sales. Finally, the **F-statistic** quantifies how likely it is that the predictor variable isn't a good predictor on the target. An F-statistic greater than 1 and a small associated p-value implies that the predictor *is* good. Since the F-statistic in the summary is 1.127∗10^5, and the associated p-value is 2.2∗10^(-16), they meet those requirements.

Plotting the Residuals

There are 4 built-in diagnostic plots for linear regression analysis: Residuals vs Fitted, Normal Q-Q, Scale-Location, and Residuals vs Leverage.

```
par(mfrow=c(2,2)) # Change the panel layout to 2x2
plot(lm1)
```



The first plot, **Residuals vs Fitted**, shows any non-linear relationship between the the predictor and the target, if there is a non-linear relationship (eg. a parabola). Because the actual plot shows a patternless spread of residuals around a horizontal line, there is no non-linear relationship in the data.

The second plot, **Normal Q-Q**, shows how normal the distribution of residuals is. The plot shows that the middle of the distribution is aligned with the dashed line, but with significant deviation towards the left and right ends, which may be problematic.

The third plot, **Scale-Location** or Spread-Location, shows the how equally residuals are spread along the range of the predictor. Instead of a horizontal line with points distributed patternlessly along it, the actual plot shows a steep incline with a lot of clustering on the left side. This implies that the spread of the residuals is not equal, but rather, widens and widens along the range of the predictor. This is problematic.

The fourth plot, **Residuals vs Leverage**, shows if any outliers are influential in determining the regression line. For this plot, instead of a patternless equal spread, the key is to note any outlying points on the far right side. In the actual plot, there is one such point, labeled 1, and a couple other outliers, labeled 6 and 2. Furthermore, the lines for Cook’s distance are very noticable, and the outliers are far away from those lines. This means that these outliers are significantly influential in determining the regression line.

Multiple Linear Regression

Multiple linear regression is done using multiple predictor variables on the target. Presumably, the sales for Europe and Japan in addition to North America will better predict the global sales. So, those 2 variables will now also be added to the linear regression model as predictors.

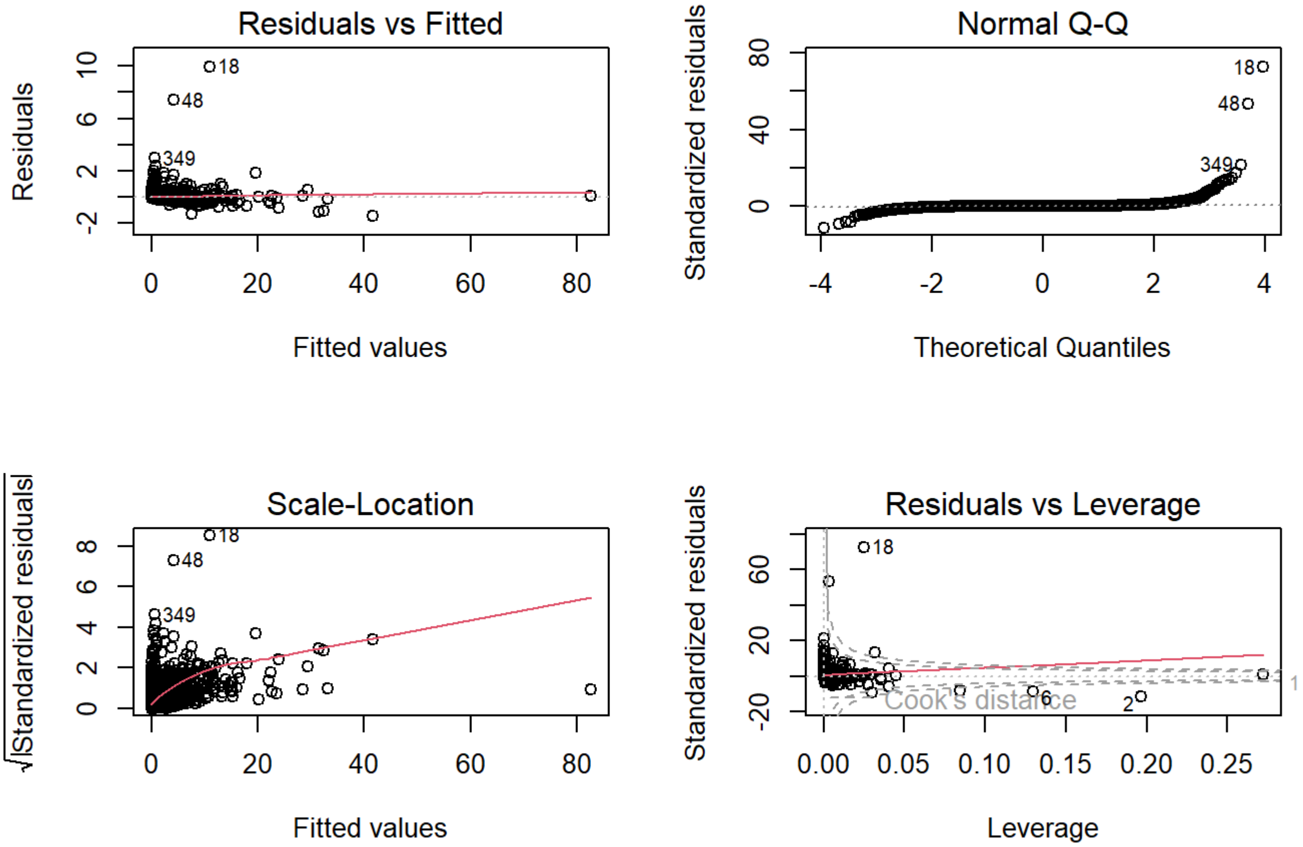
```
lm2 <- lm(Global_Sales~NA_Sales+EU_Sales+JP_Sales, data=train)
summary(lm2)
```

```
## 
## Call:
## lm(formula = Global_Sales ~ NA_Sales + EU_Sales + JP_Sales, data = train)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -1.4420 -0.0119 -0.0046  0.0020  9.9266 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 0.005956   0.001270    4.69 2.76e-06 ***
## NA_Sales    1.060753   0.002483  427.12  < 2e-16 ***
## EU_Sales    1.206663   0.003851  313.31  < 2e-16 ***
## JP_Sales    0.955877   0.004430  215.77  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.1383 on 13274 degrees of freedom
## Multiple R-squared:  0.9922, Adjusted R-squared:  0.9922 
## F-statistic: 5.621e+05 on 3 and 13274 DF,  p-value: < 2.2e-16
```

The **summary** for this model shows that it is a better fit than the previous model. The p-values for all 3 predictors bear the significant code closest to 0, which is good. The value for R-squared is 0.9922, even closer to 1 than the previous model, meaning that almost 100% of the variation in Global_Sales can be predicted by the 3 sales predictors. Finally, the F-statistic is very large and its associated p-value is very small, which is also a good sign that all 3 predictors are closely linked to the target.

Next, the **residual plots** for the new model show some changes in comparison to the previous model.

```
par(mfrow=c(2,2)) # Change the panel layout to 2x2
plot(lm2)
```



The **Residuals vs Fitted** plot shows an even straighter horizontal line than the previous model did, which is good. The **Normal Q-Q** plot has less deviation than the previous model did, which is also good, although some significant deviation can still be noticed towards the right end. The **Scale-Location** plot still does not show an equal and random spread, however, which is not such a good sign. Finally, the **Residuals vs Leverage** plot shows the outliers are closer to the lines for Cook's Distance, which is an improvement from the previous model.

Improving the Model Using Polynomial Regression

Linear regression does not always entail a straight line and a degree-1 model. Polynomial regression can be used to create a model of a higher degree, like a parabola. Higher degree models may be able to capture more complex trends better than a linear model can. Below, polynomial models of degrees 2, 3, and 4 are created on multiple predictors. The models are then compared using **anova()**, which shows that the models gradually have lower RSS values for higher degrees. A smaller RSS indicates a better model fit, so this is good. Until the degree 4 model, the RSS seems to drop steadily; after that, there only seems to be diminishing returns. Furthermore, the p-value of the degree-4 model is significantly smaller than that of the degree-5 model. So, out of all the polynomial models, the degree-4 model seems to be the best fit.

```
deg2 <- lm(Global_Sales~poly(NA_Sales+EU_Sales+JP_Sales, 2), data=train)
deg3 <- lm(Global_Sales~poly(NA_Sales+EU_Sales+JP_Sales, 3), data=train)
deg4 <- lm(Global_Sales~poly(NA_Sales+EU_Sales+JP_Sales, 4), data=train)
deg5 <- lm(Global_Sales~poly(NA_Sales+EU_Sales+JP_Sales, 5), data=train)
anova(deg2,deg3,deg4,deg5)
```

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	13275	284.4783	NA	NA	NA	NA
2	13274	273.5872	1	10.8911119	539.467103	5.226006e-117
3	13273	268.0368	1	5.5504257	274.928046	3.937968e-61
4	13272	267.9437	1	0.0930104	4.607064	3.185857e-02

4 rows

The summary of the degree-4 model shows low p-values, an low RSE near 0, a high R-squared near 1, and a high F-statistic with a low associated p-value. These are all good signs of the model's fit.

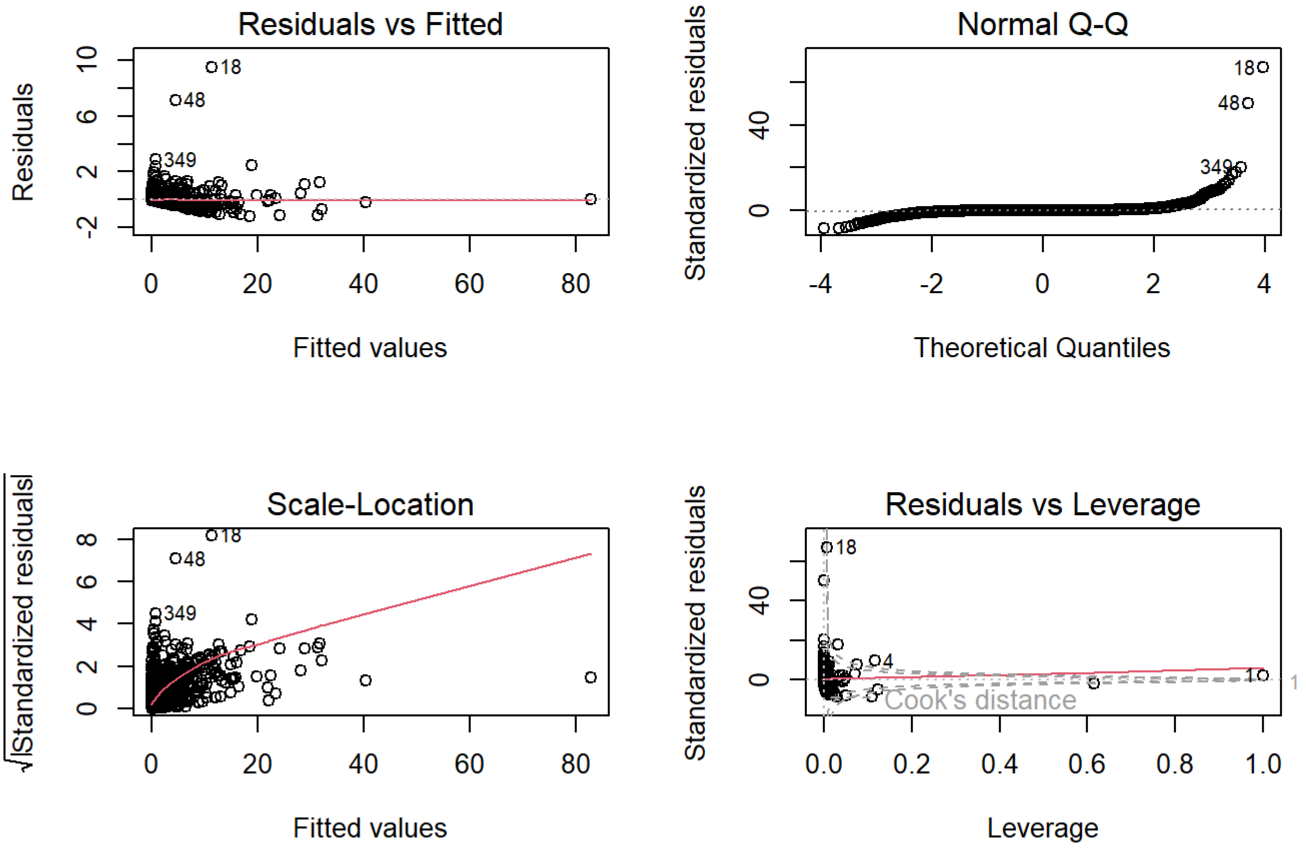
```
summary(deg4)
```

```
##
## Call:
## lm(formula = Global_Sales ~ poly(NA_Sales + EU_Sales + JP_Sales,
##     4), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1905 -0.0115 -0.0041  0.0041  9.4697
##
## Coefficients:
##                                           Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                              5.352e-01  1.233e-03  433.971   <2e-16
## poly(NA_Sales + EU_Sales + JP_Sales, 4)1 1.794e+02  1.421e-01 1262.764   <2e-16
## poly(NA_Sales + EU_Sales + JP_Sales, 4)2 2.334e-01  1.421e-01    1.642    0.101
## poly(NA_Sales + EU_Sales + JP_Sales, 4)3 3.300e+00  1.421e-01   23.223   <2e-16
## poly(NA_Sales + EU_Sales + JP_Sales, 4)4 2.356e+00  1.421e-01   16.579   <2e-16
##
## (Intercept)                              ***
## poly(NA_Sales + EU_Sales + JP_Sales, 4)1 ***
## poly(NA_Sales + EU_Sales + JP_Sales, 4)2
## poly(NA_Sales + EU_Sales + JP_Sales, 4)3 ***
## poly(NA_Sales + EU_Sales + JP_Sales, 4)4 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1421 on 13273 degrees of freedom
## Multiple R-squared:  0.9917, Adjusted R-squared:  0.9917
## F-statistic: 3.988e+05 on 4 and 13273 DF,  p-value: < 2.2e-16
```

The residual plots show that not much has changed, however. The Normal Q-Q plot still doesn't uniformly follow a straight line, the Scale-Location plot is still not evenly distributed, and the Residuals vs Leverage plot still shows influential outliers.

```
par(mfrow=c(2,2)) # Change the panel layout to 2x2
plot(deg4)
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```



Comparing the Results

Three potential models have now been created to model the training data. Using **anova()**, the best model can be determined.

```
anova(lm1,lm2,deg4)
```

	Res.Df <dbl>	RSS <dbl>	Df <dbl>	Sum of Sq <dbl>	F <dbl>	Pr(>F) <dbl>
1	13276	3423.8335	NA	NA	NA	NA
2	13274	253.7085	2	3170.12501	78491.23	0
3	13273	268.0368	1	-14.32822	NA	NA

3 rows

Since lm2 has the lowest RSS and the lowest p-value, it is the best model of the 3. So, multiple linear regression produced the best fitting model. This makes sense because the data visually seems linear as opposed to quadratic or quartic. Multiple predictors produced a better fit than one predictor because global sales are the sum of regional sales, thus implying that any video game with a low global sale will have a low regional sale for at least one region. So, the more regions are used as predictors, the better.

Predicting on the Test Data

Predictions can now be made on the test data using the **predict()** function, which creates a vector of predicted values for each model. These predictions are then evaluated using correlation and RMSE, or root mean squared error. Correlation shows how closely the predicted values match the actual Global_Sales values in the test data, so 1 is the most ideal correlation to see. RMSE is the square root of MSE, the mean squared error, which is the errors squared and then averaged out. RMSE indicates how many y-units the predicted values were off from the test data, so a low RMSE is ideal.

First, calculate the predictions and the MSEs:

```
pred1 <- predict(lm1, newdata=test)
pred2 <- predict(lm2, newdata=test)
pred3 <- predict(deg4, newdata=test)

mse1 <- mean((pred1 - test$Global_Sales)^2)
mse2 <- mean((pred2 - test$Global_Sales)^2)
mse3 <- mean((pred3 - test$Global_Sales)^2)
```

Next, the correlations:

```
cor(pred1, test$Global_Sales)
```

```
## [1] 0.9262168
```

```
cor(pred2, test$Global_Sales)
```

```
## [1] 0.9989872
```

```
cor(pred3, test$Global_Sales)
```

```
## [1] 0.9980838
```

Finally, the RMSEs:

```
sqrt(mse1)
```

```
## [1] 0.5992114
```

```
sqrt(mse2)
```

```
## [1] 0.06915136
```

```
sqrt(mse3)
```

```
## [1] 0.09396109
```

Of the 3 predictions, pred2 has the highest correlation at about 0.998, and the lowest RMSE at about 0.069. This means that its correlation to the actual test values is nearly 100%, and it is only off from the actual test values by about 0.069 million dollars. Thus, these results corroborate the claim that lm2, the multiple linear regression model, is the most accurate model of the data.