Google Play Store Apps

Catherine Lindsay, Jack Hardwick, Luca Comba

Fall 2019

Abstract:

The goal of this project was to use Google Play Store App data to create a multiple linear regression model so that we could make predictions of app ratings based on several different variables. Over our process, we first cleaned and organized our .csv file so that we could run an effective analysis based on the way that we classified our variables. We used backwards elimination to ultimately determine our variables. The model can be used to predict the rating of Google play store apps in the future.

Introduction:

The Google Play Store is Google's official store and portal for android content, commonly referred to as 'The Android Market'. This covers a wide variety of downloadable content including apps, books, games, movies, and much more other entertainment. Similar to Apple's App Store, Google offers this platform for consumers to download content and creators to upload their work for profitable gain. Our data set, however, covered only apps. We focused on predicting users' rating as our response variable. (Through analysis we were able to determine that 68.8% of apps were free and 31.2% were paid.) The biggest questions we were trying to solve initially were do paid apps have higher ratings versus free apps, and do apps with higher ratings have higher downloads? The end goal would be to create a model to give content creators the ability to predict the rating of their app in the Google Play Store based on our variables. This will give developers a chance to aim their focus on specific characteristics of the app that they are developing. In fact, this data set can potentially have multiplications to drive app-making businesses to success.

Data:

This data set was first collected from a popular data source, Kaggle.com which was based on the data from the Google Play Store website. The dataset was created on September 4th 2018. The dataset owner is Lavanya Gupta. Overall it includes 10,841 apps, ranging from a variety of different categories. For each app, there are 12 different variables. The data was taken from the Google Play Store website.

```
skim(googleplaystore)
```

Data summary

Name	googleplaystore
Number of rows	10841
Number of columns	13
_______________________	
Column type frequency:	
character	11
numeric	2
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
App	0	1	1	194	0	9660	0
Category	0	1	3	19	0	34	0
Size	0	1	3	18	0	462	0
Installs	0	1	1	14	0	22	0
Type	0	1	1	4	0	4	0
Price	0	1	1	8	0	93	0
Content Rating	1	1	4	15	0	6	0
Genres	0	1	4	37	0	120	0
Last Updated	0	1	6	18	0	1378	0
Current Ver	1	1	1	50	0	2833	0
Android Ver	1	1	3	18	0	34	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Rating	1474	0.86	4.19	0.54	1	4	4.3	4.5	19	
Reviews	1	1.00	444152.90	2927760.60	0	38	2094.0	54775.5	78158306	

The variables included categorical and numerical variables, consisting of "*Category*" which tells in which category app belongs to (stored as a character type), "*Rating*" which is the overall user rating of the app (stored as a value out of 5 with one decimal, which is numeric type), "*Reviews*" which is the number of user reviews for the app (as a numeric type), "*Size*" which is the app's size (in Megabyte or Kilobyte as a character type), "*Installs*" which is the number of user downloads/installs for the app (number amount as a character type), "*Type*" (free or paid as a character type), "*Price*" which is the price of the app (in dollar amount as a character type), "*Content Rating*" which is the age group the app is targeted (Teen/Everyone/Mature as a character type), "*Genres*" (as a character type), "*Last Updated*" the date when the app was last updated on Play Store (Date as a character type), "*Current Version*" of the app available on Play Store (as a character type), and "*Android Version*" which is the minimum required Android version (as a character type). The population of the data set are the apps. We were forced to manipulate our data to have a clean set of data that was easy to work with and free of missing values. One problem with the original dataset is that many of the applications had missing values represented by "NaN" in some of their cells, so we eliminated the rows which resulted in a total of 9,112 applications to analyze. Eliminating these values can be problematic because eliminating apps with no ratings can skew the data and may underrepresented apps with fewer users.

```
app = read_csv("googleplaystore.csv", n_max = 10472, na = c("", NA, "NaN"))
```

```
skim(app_cleaned)
```

Data summary

Name	app_cleaned
Number of rows	9111
Number of columns	20
_______________________	
Column type frequency:	
character	12
Date	2
numeric	6
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
App	0	1	1	194	0	7950	0
Category	0	1	4	19	0	33	0
Size	0	1	3	18	0	401	0
Installs	0	1	2	14	0	19	0
Type	0	1	4	4	0	2	0
Price	0	1	1	7	0	72	0
ContentRating	0	1	4	15	0	6	0
Genres	0	1	4	37	0	114	0
CurrentVersion	0	1	1	50	0	2592	0
AndroidVersion	0	1	9	18	0	31	0
isMB	0	1	1	1	0	3	0
NumberOfInstalls	0	1	13	20	0	4	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique

skim_variable	n_missing	complete_rate	min	max	median	n_unique
LastUpdated	0	1	2010-05-21	2018-08-08	2018-06-05	1274
Today	0	1	2019-12-09	2019-12-09	2019-12-09	1

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Rating	0	1.00	4.19	0.51	1	4.0	4.3	4.5	5	▁▁▁▇▇
Reviews	0	1.00	527336.17	3186618.55	1	203.0	6404.0	85346.0	78158306	▇▁▁▁▁
days_since_last_update	0	1.00	735.81	393.47	488	503.0	552.0	787.5	3489	▇▁▁▁▁
Price2	0	1.00	0.98	16.03	0	0.0	0.0	0.0	400	▇▁▁▁▁
Size2	1622	0.82	37.09	93.02	1	6.1	16.0	37.0	994	▇▁▁▁▁
Size3	1622	0.82	14069.71	93513.07	1	6.1	16.0	37.0	994000	▇▁▁▁▁

We determined that the "*Last Updated*" column was not relevant, so we used parsing to subtract the "*Last Updated*" column from the current date to depict the frequency the application is updated. This new column was named "*days_since_last_update*" and outputs a numeric value for us to use in our analysis rather than a date.

```
library(lubridate)
app_cleaned <- app_cleaned %>%
  mutate(`Last Updated` = mdy(app_cleaned$`Last Updated`))
app_cleaned <- app_cleaned %>%
  mutate("Today"=mdy("12/9/2019"))
app_cleaned <- app_cleaned %>%
  mutate(`days_since_last_update`= difftime(Today,`Last Updated`))
```

We had to parse some variable since Price was character data type and we wanted it to be numerical, so we saved it to be the numerical variable *"Price2"*

```
app_cleaned <- app_cleaned %>%
  mutate(Price2 = parse_number(Price))
```

We grouped the variables by Size to eliminate both Megabyte and Kilobyte and transform the size to a neautral platform to run an efficient analysis on *Size* as a variable.

```
app_cleaned <- app_cleaned %>%
  mutate(isMB = str_sub(Size, start = -1, end = -1)) %>%
    ungroup() %>%
      mutate(Size2 = str_sub(Size, start = 1, end = -2)) %>%
        mutate(Size2 = parse_number(Size2))

app_cleaned <- app_cleaned %>%
  mutate(Size3 = case_when(isMB == "M" ~ Size2, isMB == "k" ~  Size2*1000, isMB == "e" ~ Size2))
```

We parsed the Installs category to group the *Number of Installs* into fewer bins and we changed the predictor to a numerical value.

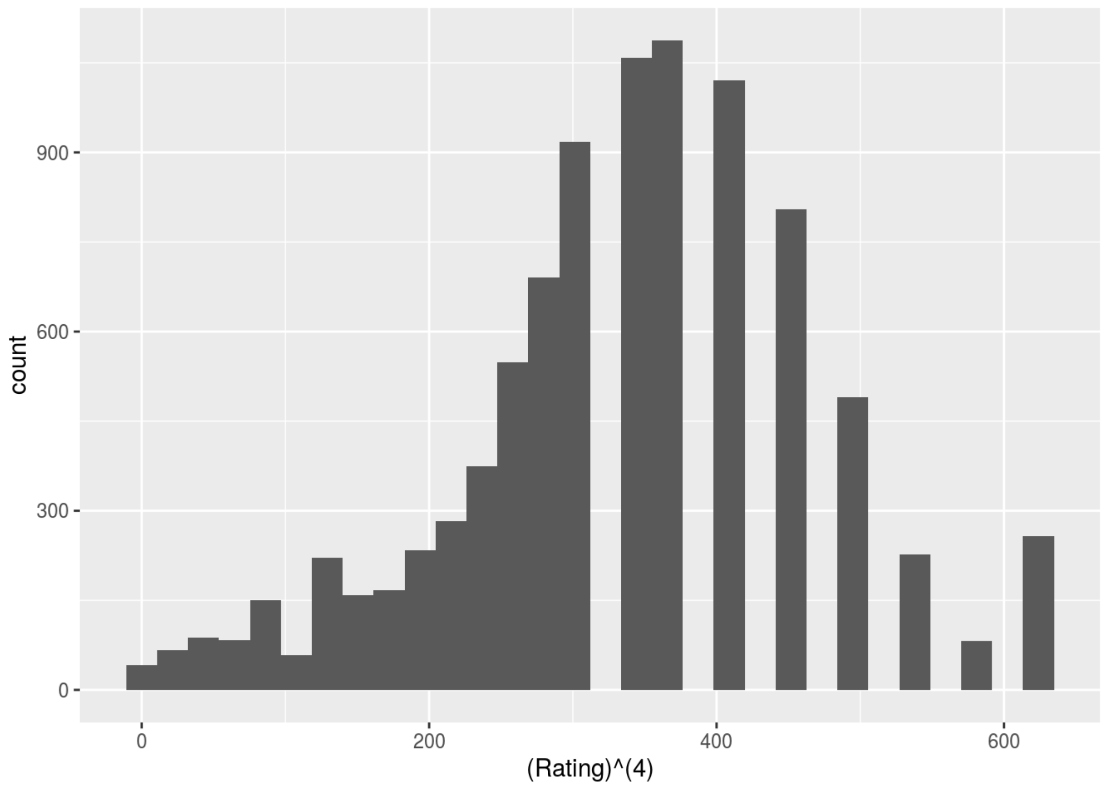
```
app_cleaned <- app_cleaned %>%
  mutate(NumberOfInstalls = parse_number(Installs)) %>%
  mutate(NumberOfInstalls = case_when(NumberOfInstalls<10000~"LessThan10000",
                                      NumberOfInstalls>=10000 & NumberOfInstalls<500000~"Between10000&50
0000",
                                      NumberOfInstalls>=500000 & NumberOfInstalls<5000000~"Between500000
&500000",
                                      NumberOfInstalls>=5000000~"GreaterThan5000000")) %>%
  mutate(NumberOfInstalls=fct_relevel(NumberOfInstalls,"LessThan10000"))
```

The initial model used "*Rating*" (quantitative) as the response variable, and "*Reviews*" (quantitative), "*Type*" (categorical), "*Size2*" (quantitative), "*days_since_last_update*" (quantitative), "*Price2*" (quantitative), "*Genres*" (categorical), "*ContentRating*" (quantitative), "*NumberOfInstalls*" (quantitative) , and "*Category*" (categorical) as predictors.

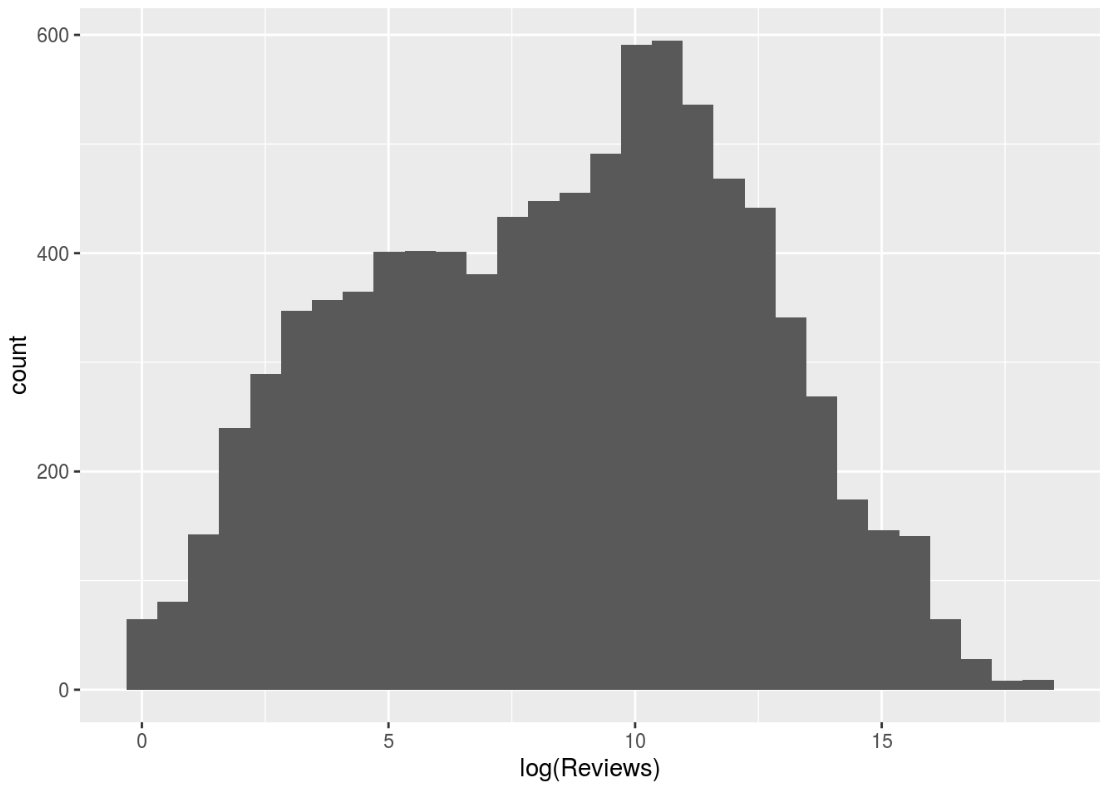
```
m1 = lm(Rating ~ Reviews + Type + Size2 + days_since_last_update + Price2 + Genres + ContentRating + Num
berOfInstalls + Category, data=app_cleaned)
summary(m1)
```

After fitting the original model, transformations were made to meet the conditions for linear regression. The conditions for multiple linear regression are not met based on the fan shape of the residual vs fitted plots, and an abnormal qq-plot.

```
ggplot(data=app_cleaned)+geom_histogram(aes(x=(Rating)^(4)))
```



```
ggplot(data=app_cleaned)+geom_histogram(aes(x=log(Reviews)))
```



After looking at histograms of the variables, the transformations were determined. “*Rating*” was raised to the fourth power and the log was taken for “*Reviews*”. Using these transformations, the plots met the conditions for multiple linear regression. Backwards Elimination was used to find the final model with the highest R^2 value and significant p-values for the variables (based on the A$ = .05 level). The non-statistically significant varaibles that were dropped were *Genres*, *Category*, *Content Rating*, and *Size2*. The variables that were originally not included in the model were *AndroidVersion* and *CurrentVersion*. These variables were not practical to include in the model because they contain too many categories and would not have high explanatory power.

Results:

SUMMARY OF FINAL MODEL

The final model contains the transformed response variable of *Rating^4*. The predictor variables use log(*Reviews*),*Type*, *days_since_last_update*, *Price2* , and *NumberOfInstalls*.

```
m2 = lm((Rating)^4 ~ log(Reviews) + days_since_last_update + Type + Price2 + NumberOfInstalls, data=app_
cleaned)
summary(m2)
```

```
##
## Call:
## lm(formula = (Rating)^4 ~ log(Reviews) + days_since_last_update +
##     Type + Price2 + NumberOfInstalls, data = app_cleaned)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.03  -65.56    1.42   71.17  387.73
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         3.420e+02  4.793e+00  71.352  < 2e-16 ***
## log(Reviews)                        1.460e+01  8.505e-01  17.172  < 2e-16 ***
## days_since_last_update             -4.089e-02  3.339e-03 -12.247  < 2e-16 ***
## TypePaid                            2.940e+01  5.377e+00   5.467 4.69e-08 ***
## Price2                             -3.663e-01  7.970e-02  -4.597 4.35e-06 ***
## NumberOfInstallsBetween10000&500000  -1.017e+02  4.873e+00 -20.868  < 2e-16 ***
## NumberOfInstallsBetween500000&500000 -1.271e+02  7.118e+00 -17.853  < 2e-16 ***
## NumberOfInstallsGreaterThan5000000   -1.492e+02  9.334e+00 -15.984  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 118.8 on 9103 degrees of freedom
## Multiple R-squared:  0.08573,    Adjusted R-squared:  0.08503
## F-statistic: 121.9 on 7 and 9103 DF,  p-value: < 2.2e-16
```

The full model has significant p-values at the 0.05 alpha level for all variables. A noteworthy coefficient is *log(Reviews)* in which a 1-unit review increase, it is expected to be 14.60 increase in *Rating^4*, holding all else constant. There is a positive correlation between *log(Reviews)* and *Rating^4*.

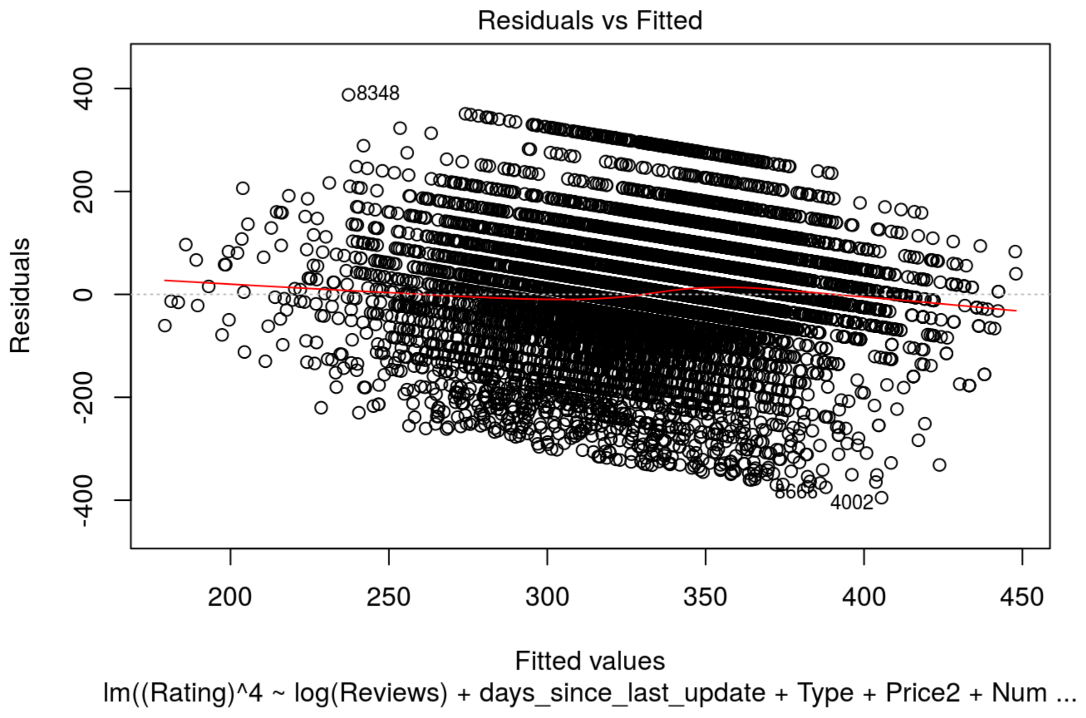
The coefficient for *days_since_last_update* is -.04089, so for a one-unit increase in *days_since_last_update* we would expect the *Rating^4* to decrease by .04089, holding all else constant. There is a negative correlation between *days_since_last_update* and *Rating^4*.

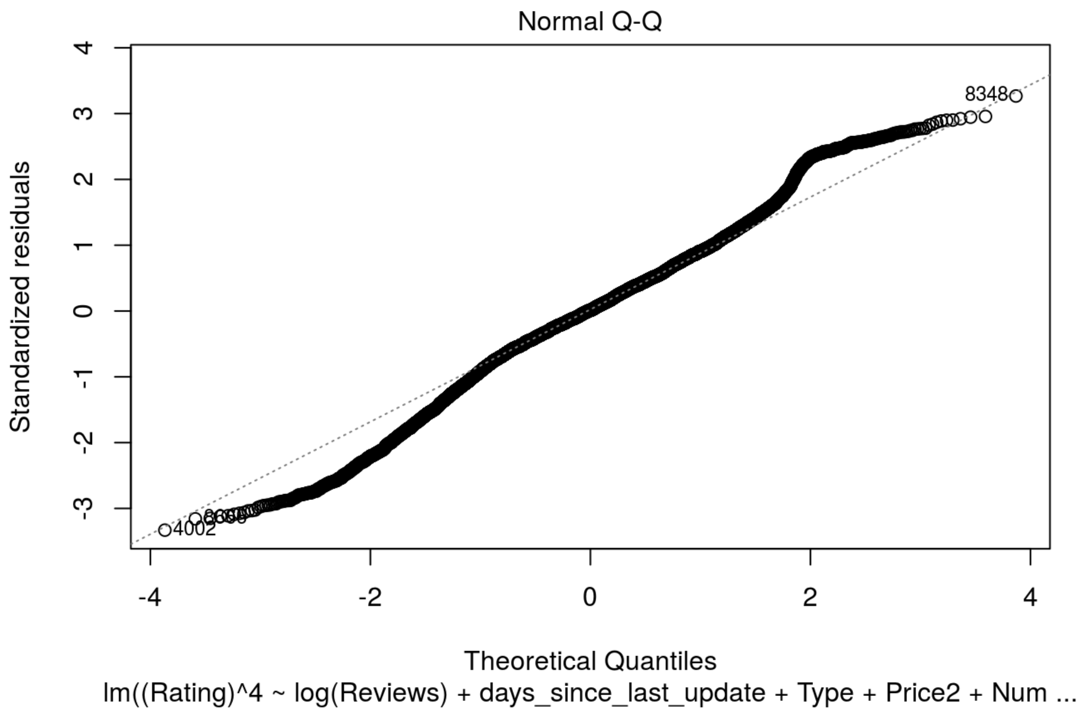
The coefficient for *TypePaid* is 29.40, so compared to free apps, we would expect paid apps to have a *Rating^4* that is 29.40 higher. There is a positive correlation between *TypePaid* and *Rating^4*.

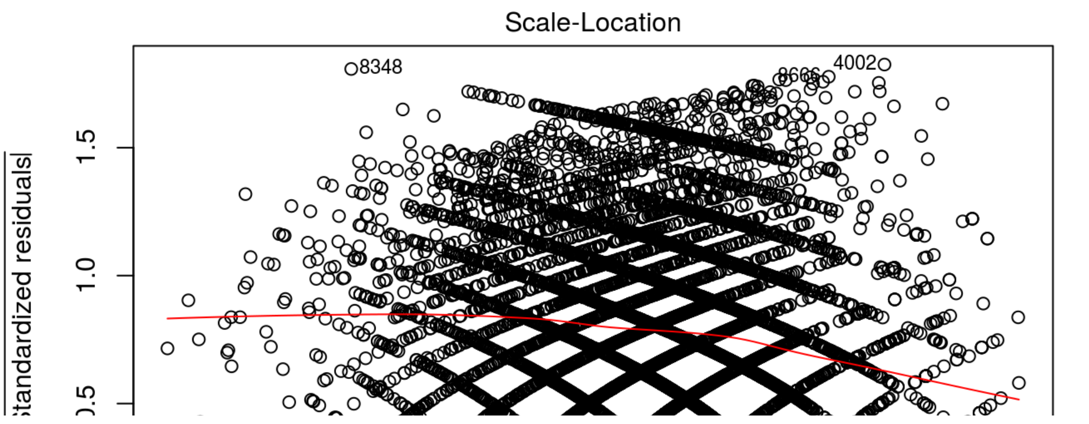
There is a negative correlation between *Rating^4* and *Price2*. The coefficient for *Price2* is -.3663, so for a 1-unit increase in the *Price2*, we would expect *Rating^4* to decrease by .3663, holding all else constant.

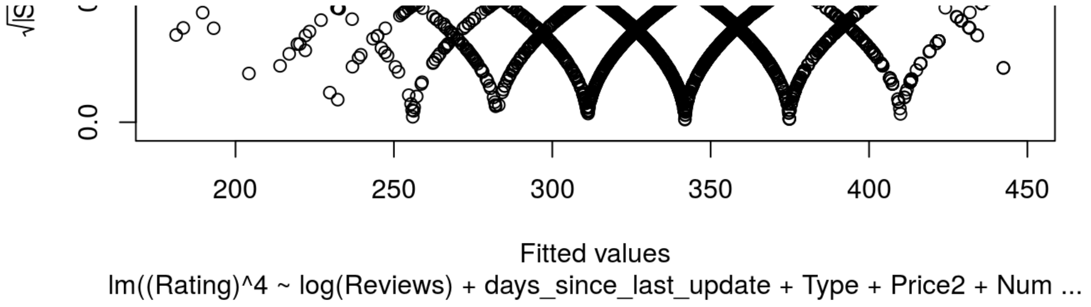
There is a negative correlation between *Rating^4* and *NumberOfInstalls*. The coefficients for *NumberOfInstalls* are very large, so the number of Installs can greatly impact the *Rating*$^{4.}$ *The reference group is “LessThan10,000”. Compared to the reference group for a 1-unit increase in NumberOfInstallsBetween10000&500000, we would expect to see a 101.7 decrease in the Rating*$_{4}$.Compared to the reference group for a 1-unit increase in NumberOfInstallsBetween500000&5000000, we would expect to see a 127.1 decrease in the *Rating*$^{4.}$*Compared to the reference group for a 1-unit increase in NumberOfInstallsGreaterThan5000000, we would expect to see a 149.2 decrease in the Rating*$_{4}$.

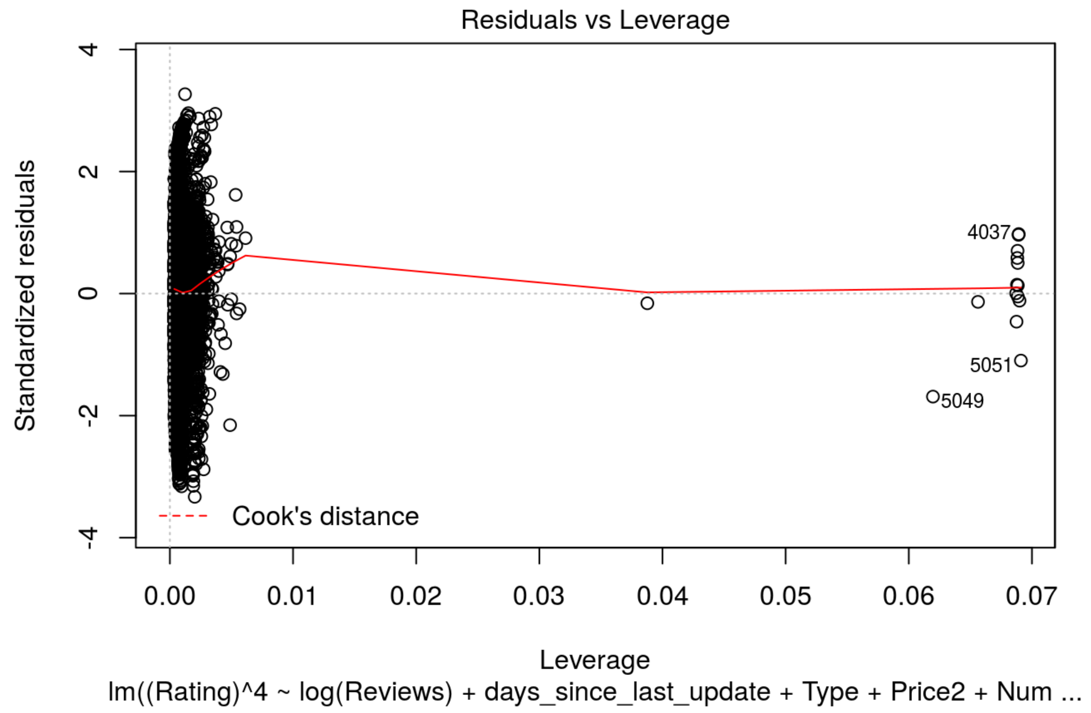
```
plot(m2)
```


Residuals vs Fitted
Residuals
Fitted values
lm((Rating)^4 ~ log(Reviews) + days_since_last_update + Type + Price2 + Num ...
8348
4002
8666
400
200
0
-200
-400
200
250
300
350
400
450


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm((Rating)^4 ~ log(Reviews) + days_since_last_update + Type + Price2 + Num ...
8348
4002
4
3
2
1
0
-1
-2
-3
-4
-2
0
2
4


Scale-Location
√|Standardized residuals|
8348
8666
4002
1.5
1.0
0.5





Based on the residual plots for the full model, the conditions for multiple linear regression are met. The normality condition is met based on a straight QQ plot. The equality of variance condition is met based on the residual vs fitted plot which appears fairly normal, besides a slight fan shape. The independence condition is met because there is independence between the variables (the apps are independent of one another).

The model was then checked for multicollinearity by looking at the VIF values of our full model.VIF’s close to 5 suggest multicollinearity between two or more variables.

```
vif(m2)
```

```
##                            GVIF Df GVIF^(1/(2*Df))
## log(Reviews)           7.042174  1        2.653709
## days_since_last_update 1.114723  1        1.055804
## Type                   1.202294  1        1.096492
## Price2                 1.054868  1        1.027068
## NumberOfInstalls       7.431136  3        1.396934
```

The VIF for the variables are all less than 5, which suggests that there is no collinear values.

Conclusion:

The purpose of this project was to predict the rating of an app based on various factors. This model can be used to predict the success of future applications. This can describe the relationship between ratings, reviews, frequency of updates, price of app, and number of installs.

This model is limited because we eliminated 1,730 of the data entries from the overall data set due to missing values. Eliminating these variables can under-represent the population, and remove statistically significant information from the results. Due to many categorical variables, it was difficult to produce a model with a high R^2 value, as categorical predictors have low explanatory power. Our full model can only explain 8.503% of the variabilty in the model using *log(Reviews)*, *Type days_since_last_update* , *Price2*, and *NumberOfInstalls* as predictors.

This model can be used for inference because it meets the conditions for inference, as the apps are independent of one another, the residual vs. fitted plots show equality of variance, and the QQ-plot shows normality. Therefore predictions from this model can be applied to a larger population of other apps not represented in google apps, and can even be applied to apps that have yet to be created.