**Google Play Store: India**

**App Analysis**



Debasis Chatterjee

Rodney Koch

James Robertson

**Table of Contents**

**Introduction 3**

Business Questions 3

**The Data Set 4**

Data Load 4

Data Cleansing 5

**Data Analysis Methods 7**

[**Descriptive Statistics**](#_2mwcj12y2ohi) **7**

Figure 1 7

Figure 2 8

**Hypothesis Tests 11**

Figure 3 11

Figure 4 11

**Machine Learning 13**

Figure 5 14

Figure 6 15

Figure 7 16

Figure 8 16

Figure 9 17

**Linear Modeling and Regression 18**

Figures 10-15 18

Figures 16-21 19

Figures 22-24 20

**Conclusions 23**

Answers to Business Questions 23

Figure 25 24

**Appendix A: Unexecuted R Code** 25

**Introduction**

We are a hypothetical data science consulting firm and we have been approached by an app developer with a set of data regarding the India Google Play Store. They would like us to analyze the data and answer questions for them. The data contains thirteen variables from one point in time and some of the data is cumulative.

## **Business Questions**

The development group is looking to create a hit app in India and has brought the following questions to the table:

**Which apps are installed the most?   
 Which categories?  
 Which age groups?  
 Type (Paid vs. Free)**Which are most popular in the last year?  
Which are growing in popularity?  
**How can I make a “best selling”/highly installed app?   
Which categories have the highest ratings?**

**Which are liked the most?**   
How can I get more people to install an app?  
What motivates people to write reviews?  
**Are ratings or reviews more correlated with installs?**

**Does size or installs have any effect on price?**

**Does price have any effect on size or installs?**

**Can we predict high total worth of an app with category, rating, reviews, size, pay type, and/or content rating?**

**Can we predict installs for a given category?**

After reviewing the data provided we believe we can find an answer the **bolded** questions. We did not have the installation date so we could not answer “which are most popular by year” and “which apps are growing in popularity”. We also didn’t have review data (only review counts and ratings) so we were not able to determine what motivates people to write reviews or how to get more people to install an app. We did find some interesting anecdotal answers.

*Debasis Chatterjee cleaned the data, made histograms, and ran hypothesis testing*

*Rodney Koch ran linear regression*

*James Robertson ran the ksvm machine learning algorithm*

**The Dataset**

# **Description of the data set**

The data set consists of data regarding apps which have been downloaded from the GooglePlayStore in India.

# **Data load**

Figure 1- Survey Data Files

The individual .csv files were downloaded and saved locally from<https://www.kaggle.com/lava18/google-play-store-apps>. A detailed review of each field was conducted and 13 variables were identified as required in order to conduct our analysis and modeling.

|  |  |
| --- | --- |
| **Column Name** | **Definition** |
| App | Application name |
| Category | Category the app belongs to |
| Rating | Overall user rating of the app (as when scraped) |
| Reviews | Number of user reviews for the app (as when scraped) |
| Size | Size of the app (as when scraped) |
| Installs | Number of user downloads/installs for the app (as when scraped) |
| Type | Paid or Free |
| Price | Price of the app (as when scraped) |
| Content Rating | Age group the app is targeted at - Children / Mature 21+ / Adult |
| Genre | An app can belong to multiple genres (apart from its main category). For eg, a musical family game will belong to family and music |
| Last Updated | Date when the app was last updated on Play Store (as when scraped) |
| Current Ver | Current version of the app available on Play Store (as when scraped) |
| Android Ver | Minimum required Android version (as when scraped) |

The selected fields were then read into a dataframe (googleSalesData) in R using “*read.csv”* within the readGoogleSalesData function. This function was created to read the csv file and cleanse the data by calling the dataCleanUpProcess function.

#function readGoogleSalesData to read data from CSV

#@param vector for file path and file name

readGoogleSalesData <- function(csvFile)

{

csvFile

setwd(csvFile[1])

googleSalesData <- read.csv(csvFile[2])

googleSalesDataCleaned <- dataCleanUpProcess(googleSalesData)

return(googleSalesDataCleaned)

}

# **Data Cleansing**

The dataCleanUpProcess function was written to provide the data cleansing needed to create a usable dataset for analysis purposes. The following procedures were performed:

1. Rating - The maximum Rating is 5; where it was greater than 5, it is replaced with the mean rating
2. Size - The Size column was in an alphanumeric format of a number followed by a letter (e.g. 100K or 4M) to indicate the size in bytes. The function converts Size to a numeric field in bytes and removed values which couldn’t be converted (e.g. NULLs, NAs, NaNs, and “varies with device”).
3. Reviews - Converted to numerics.
4. Installs – Removes non-numeric characters (i.e. commas and plus signs) and converts the field to a numeric
5. Price - Removes non-numeric characters (i.e. commas and dollar signs) and converts the field to a numeric
6. Date – Converted format from January 07, 2018 to ‘2018-01-07’ (YYYY-MM-DD).
7. Other fields - converted to factor or character type depending on usage.

#function dataCleanUpProcess to clean up data

#@param verctor for file path and file name

dataCleanUpProcess <- function(salesdfCleaneUp){

columnNames <- colnames(salesdfCleaneUp)

for (i in c(1:length(columnNames))) {

print(paste(columnNames[i]))

if(columnNames[i] == "Rating"){

salesdfCleaneUp <- salesdfCleaneUp[salesdfCleaneUp$Rating<=5,] #remove data if rating is more than 5

salesdfCleaneUp$Rating[is.nan(salesdfCleaneUp$Rating)] <- mean(salesdfCleaneUp$Rating, nan.rm=TRUE) #replace NaN with mean of Rating

}

else if(columnNames[i] == "Reviews"){

salesdfCleaneUp$Reviews <- as.numeric(salesdfCleaneUp$Reviews) #Convert Reviews column to as numeric

}

else if(columnNames[i] == "Size"){

salesdfCleaneUp <- salesdfCleaneUp[complete.cases(salesdfCleaneUp), ] #Remove all Null/Na/NaN from the data set as it's creating problem for data manipulation.

salesdfCleaneUp$Size <- gsub("Varies with device","0",salesdfCleaneUp$Size) # Replace "Varies with device" with "0"

options(scipen=999) #

salesdfCleaneUp$Size <- as.double(sub('\\D$', '', salesdfCleaneUp$Size))\*c(1e9, 1e6, 1e3)[match( sub('\\d\*\\.\*\\d\*', '', salesdfCleaneUp$Size), c('G', 'M', 'k'))] #Change download size 1.2M or 1.8k to double format.

}

else if(columnNames[i] == "Installs"){

salesdfCleaneUp$Installs <- gsub("\\D|\\s","",salesdfCleaneUp$Installs) #Remove all characters and space from Installs

salesdfCleaneUp$Installs <- as.numeric(salesdfCleaneUp$Installs) # convert factor to numeric data type

}

else if(columnNames[i] == "Price"){

salesdfCleaneUp$Price <- as.numeric(gsub("[\\$,]", "", salesdfCleaneUp$Price)) #Remove '#' and ',' from Price and convert to numeric from Factor data type

}

else if(columnNames[i] == "Last.Updated"){

#unlist(strsplit(salesdfCleaneUp$Last.Updated, ",|\\s"))

salesdfCleaneUp$Last.Updated <- mdy(salesdfCleaneUp$Last.Updated) #Convert date format from 'January 07, 2018' to '2018-01-07' as Date type

# salesdfCleaneUp$Last.Updated <- as.character.Date(salesdfCleaneUp$Last.Updated,"%m-%d-%Y")

}

#salesdfCleaneUp <- data.frame(lapply(salesdfCleaneUp, as.character), stringsAsFactors=FALSE)

salesdfCleaneUp %>% map\_if(is.factor, as.character) %>% as\_data\_frame -> salesdfCleaneUp #Convert all Factors to Character type

salesdfCleaneUp[, c("Category", "Type", "Content.Rating", "Genres", "Current.Ver", "Android.Ver")] <- lapply(salesdfCleaneUp[, c("Category", "Type", "Content.Rating", "Genres", "Current.Ver", "Android.Ver")], factor) #COnverted some specific column back to Factor from Character type

}

return(salesdfCleaneUp)

}

After data cleansing, the resulting googlePlayStoreData data frame consists of 9366 observations of 13 variables:

Classes ‘tbl\_df’, ‘tbl’ and 'data.frame': 9366 obs. of 13 variables:  
 $ App : chr "Photo Editor & Candy Camera & Grid & ScrapBook" "Coloring book moana" "U Launcher Lite â\200“ FREE Live Cool Themes, Hide Apps" "Sketch - Draw & Paint" ...  
 $ Category : Factor w/ 33 levels "ART\_AND\_DESIGN",..: 1 1 1 1 1 1 1 1 1 1 ...  
 $ Rating : num 4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...  
 $ Reviews : num 159 967 87510 215644 967 ...  
 $ Size : num 19000000 14000000 8700000 25000000 2800000 5600000 19000000 29000000 33000000 3100000 ...  
 $ Installs : num 10000 500000 5000000 50000000 100000 50000 50000 1000000 1000000 10000 ...  
 $ Type : Factor w/ 2 levels "Free","Paid": 1 1 1 1 1 1 1 1 1 1 ...  
 $ Price : num 0 0 0 0 0 0 0 0 0 0 ...  
 $ Content.Rating: Factor w/ 6 levels "Adults only 18+",..: 2 2 2 5 2 2 2 2 2 2 ...  
 $ Genres : Factor w/ 115 levels "Action","Action;Action & Adventure",..: 10 12 10 10 11 10 10 10 10 11 ...  
 $ Last.Updated : Date, format: NA NA NA NA ...  
 $ Current.Ver : Factor w/ 2596 levels "","0.0.0.2","0.0.1",..: 106 934 421 2589 248 101 248 2199 1343 1318 ...  
 $ Android.Ver : Factor w/ 32 levels "1.0 and up","1.5 and up",..: 15 15 15 18 20 8 15 18 10 15 ...

**Data Analysis Methods**

# **Descriptive Statistics**

We used some traditional statistic tools on the data set to get some superficial answers. We first checked the proportion of free apps to paid apps. We also checked the top ten categories of apps, top five by content rating, categories with the highest average ratings, and apps with the highest total value. Value was calculated using the price of the app multiplied by the number of installs. This is where we find some interesting results, we see four apps with a title of “I am rich” or some variation thereof, four games, and two other paid apps.

General Summary Statistics

summary(googlePlayStoreData)

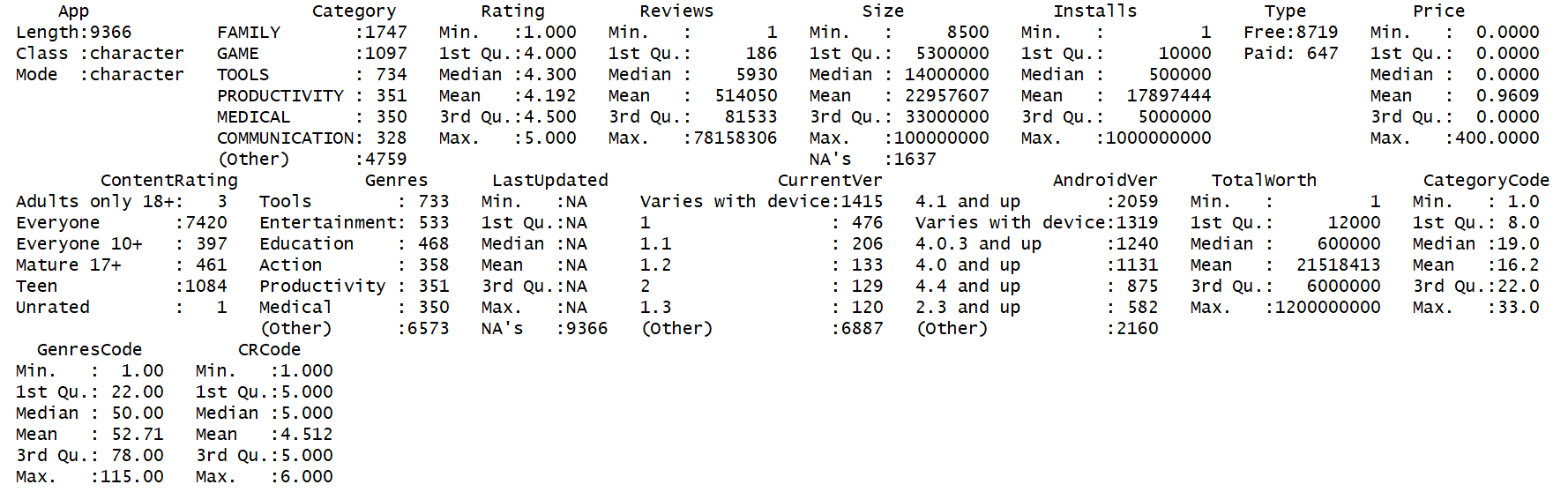


Figure 1

Plot a Pie Chart of Free vs. Paid Apps

#Plot Paid vs. Free Apps

#Install the Scales Package

install.packages("scales")

library(scales)

dfGroup <- data.frame(

group = c("Free", "Paid"),

value = c(1,2)

)

dfGroup[1,2] <- sum(appFreeInstalledByCategory$Category\_Ct)

dfGroup[2,2] <- sum(appPaidInstalledByCategory$Category\_Ct)

dfGroup$pctvalue <- dfGroup$value\*100/sum(dfGroup$value)

bpBud<- ggplot(dfGroup, aes(x="", y=value, fill=group))+

geom\_bar(width = 1, stat = "identity")

bpBud

blank\_theme <- theme\_minimal()+

theme(

axis.title.x = element\_blank(),

axis.title.y = element\_blank(),

panel.border = element\_blank(),

panel.grid=element\_blank(),

axis.ticks = element\_blank(),

plot.title=element\_text(size=14, face="bold")

)

# Graph the Free vs. Paid Pie Chart

pieBud <- bpBud + coord\_polar("y", start=0) +

blank\_theme +

theme(axis.text.x=element\_blank()) +

# Note that the vector below positions each percentage in the correct slice of the pie

geom\_text(aes(y = c(2500,9370), label = percent(pctvalue/100)), size=4) +

# Add the Title and Center it (Left Justification is the default since ggplot2 2.2)

ggtitle("Free vs. Paid App Downloads") + theme(plot.title = element\_text(hjust = 0.5)) +

# Add the Legend Title

guides(fill=guide\_legend(title="Group"))

pieBud

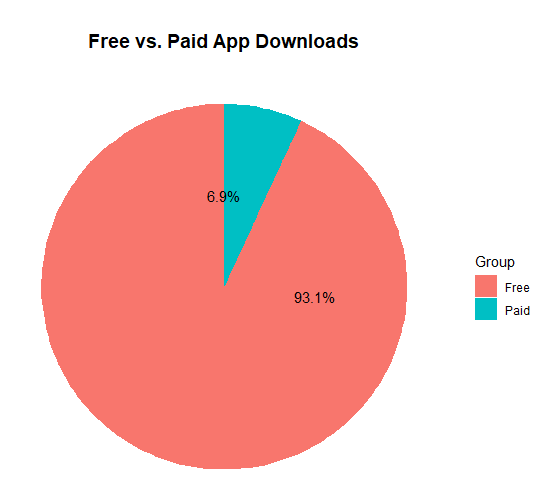


Figure 2

The top 10 Categories of all apps downloaded and installed, top 5 categories of free and paid apps:

#Top 10 overall categories of installed apps for all years

appInstalledByCategory <- sqldf("select Category, count(Category) Category\_Ct from D group by Category order by Category\_Ct DESC")

appInstalledByCategory[1:10,]

Category Category\_Ct  
FAMILY 1747  
GAME 1097

TOOLS 734  
PRODUCTIVITY 351  
MEDICAL 350  
COMMUNICATION 328  
FINANCE 323  
SPORTS 319  
PHOTOGRAPHY 317

LIFESTYLE 314

Top 5 Free App Categories

# Top overall downloaded free apps by category

appFreeInstalledByCategory <- sqldf("select Category, count(Category) Category\_Ct from D WHERE Price = 0 group by Category order by Category\_Ct DESC")

appFreeInstalledByCategory[1:5,]

Category Category\_Ct

FAMILY 1585

GAME 1020

TOOLS 671

PRODUCTIVITY 333

FINANCE 310

Top 5 Paid App Categories

# Top overall downloaded paid apps Category Type

appPaidInstalledByCategory <- sqldf("select Category, count(Category) Category\_Ct from D WHERE Price > 0 group by Category order by Category\_Ct DESC")

appPaidInstalledByCategory[1:5,]

Category Category\_Ct

FAMILY 162

MEDICAL 88

GAME 77

PERSONALIZATION 67

TOOLS 63

Most downloaded apps by Content Rating

# Top overall downloaded apps by Age Group (Content Rating)

appInstalledByCR <- sqldf("select Distinct(ContentRating), count(ContentRating) ContentRating\_Ct from D group by ContentRating order by ContentRating\_Ct DESC")

appInstalledByCR[1:6,]

ContentRating ContentRating\_Ct  
Everyone 7420  
Teen 1084  
Mature 17+ 461

Everyone 10+ 397

Adults only 18+ 3

Unrated 1

Categories with the highest Average Ratings

# Top overall Average Rating by Category

appAvgRatingByCategory <- sqldf("select Distinct(Category), avg(Rating) Avg\_Rating from D group by Category order by Avg\_Rating DESC")

appAvgRatingByCategory[1:10,]

Category Avg\_Rating  
EVENTS 4.435556  
EDUCATION 4.389032  
ART\_AND\_DESIGN 4.358065  
BOOKS\_AND\_REFERENCE 4.346067  
PERSONALIZATION 4.335987  
PARENTING 4.300000  
GAME 4.286326  
BEAUTY 4.278571  
HEALTH\_AND\_FITNESS 4.277104  
SHOPPING 4.259664

Paid Apps with the highest Total Worth

# Top overall Total Worth by Apps

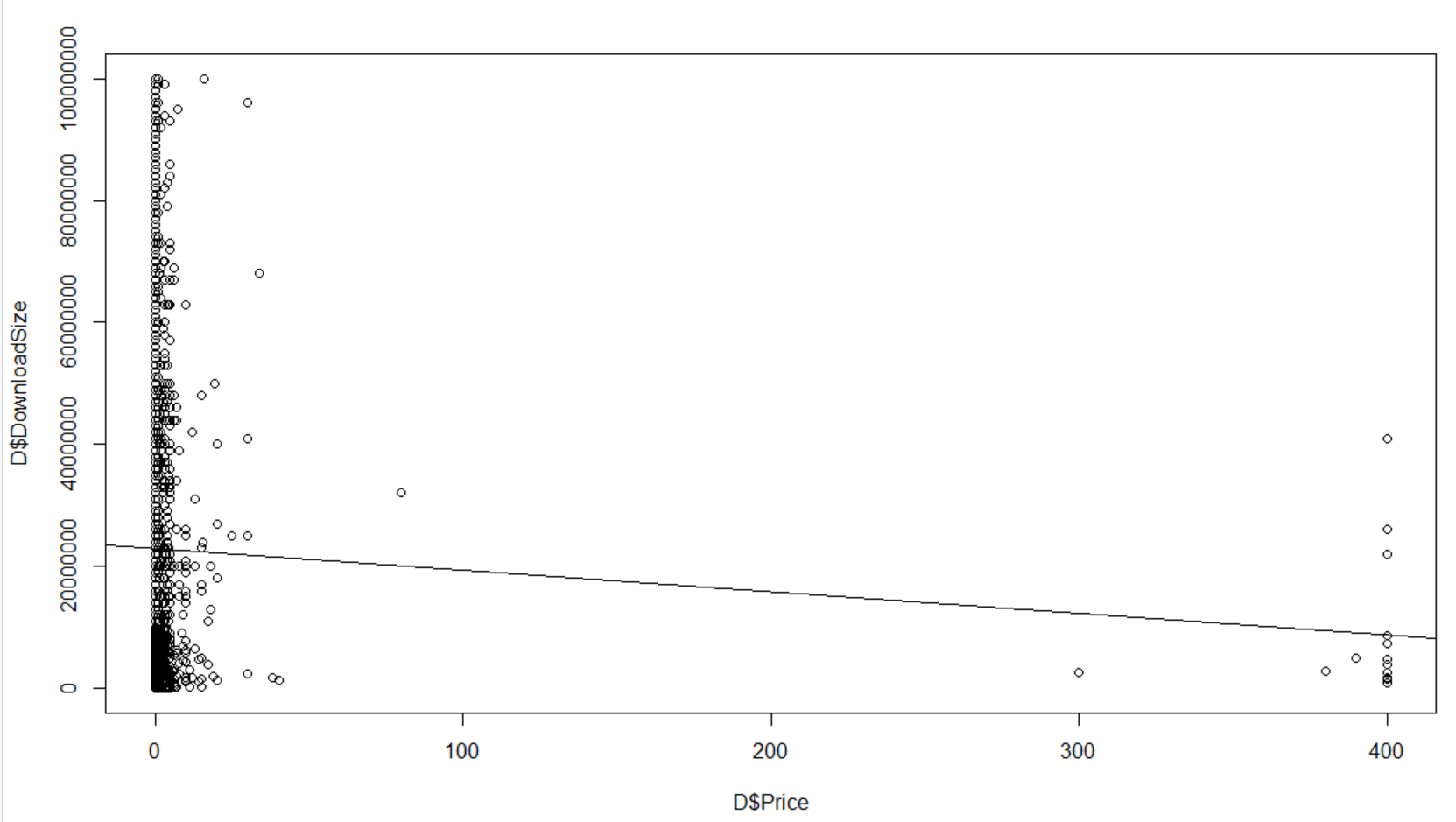
appTotalWorthByPaidApp <- sqldf("select App, (Price\*Installs) Total\_Worth from D group by App order by Total\_Worth DESC")

appTotalWorthByPaidApp[1:10,]

App Total\_Worth  
Minecraft 69900000  
I am rich 39999000  
I Am Rich Premium 19999500

Hitman Sniper 9900000  
Grand Theft Auto: San Andreas 6990000  
Facetune - For Free 5990000  
Sleep as Android Unlock 5990000  
DraStic DS Emulator 4990000  
I'm Rich - Trump Edition 4000000  
I am Rich Plus 3999900

**Hypothesis Tests**

****

**Figure 3**

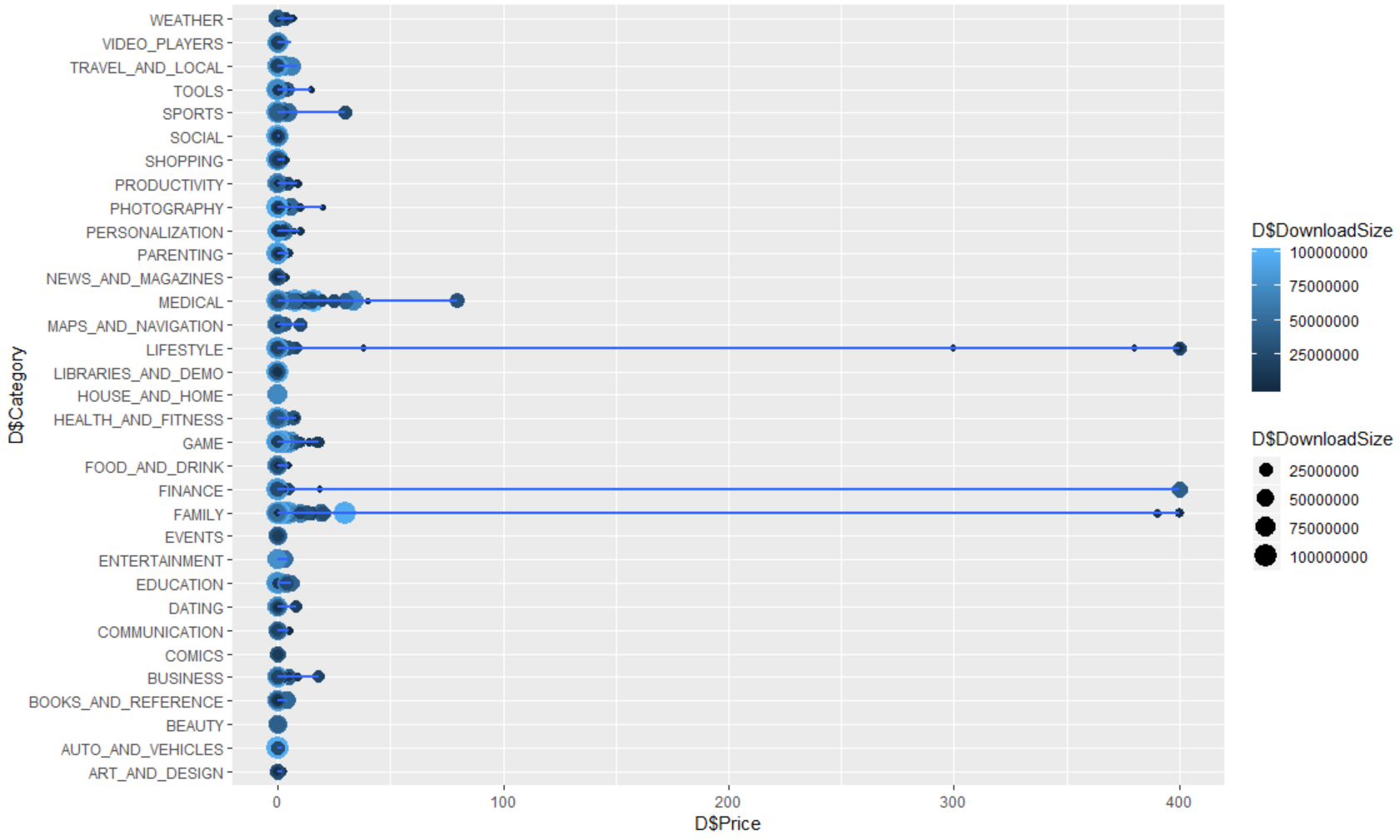
****

Figure 4

Call:

lm(formula = D$DownloadSize ~ D$Price, data = D)

Residuals:

Min 1Q Median 3Q Max

-22988972 -17697472 -8997472 10002528 77567832

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 22997472 267169 86.078 <0.0000000000000002 \*\*\*

D$Price -35354 15322 -2.307 0.0211 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 23440000 on 7727 degrees of freedom

(1637 observations deleted due to missingness)

Multiple R-squared: 0.0006885, Adjusted R-squared: 0.0005592

F-statistic: 5.324 on 1 and 7727 DF, p-value: 0.02106

The result shows that model 1 and 2 does not provide a significantly better fit to the data compared to model. It means both Download Size and Installation do not have any significance on price. Hypothesis test if Price has any effect on download and install

**Machine Learning**

**Model Code**

The code used to put together our ksvm model:

GoogleData<- googlePlayStoreData[,2:10]

colnames(GoogleData)[colSums(is.na(GoogleData)) > 0]

GoogleData$Size[is.na(GoogleData$Size)] <- mean(GoogleData$Size, na.rm=TRUE)

# find the NAs in column "Ozone"

#and replace them by the mean value of this column

GoogleData$Installs<-as.factor(GoogleData$Installs)

#create a random index for test data

randIndex <- sample(1:dim(GoogleData)[1])

#create a cutpoint for 2/3 test data

cutpoint2\_3<- floor(2\*dim(GoogleData)[1]/3)

#create a dataframe for the training data

trainData <- GoogleData[randIndex[1:cutpoint2\_3],]

dim(trainData)

head(trainData)

#create a dataframe for the test data

testData<-GoogleData[randIndex[(cutpoint2\_3+1):dim(GoogleData)[1]],]

svmOutput <- ksvm(Installs~., #Set Installs as the target predicting variables

data = trainData,

kernel = "rbfdot",

C = 10,

cross =10,

prob.model = TRUE)

svmPred <- predict(svmOutput, testData)

We began by creating training data to train our model and test data to determine the accuracy of our predictions. We note that we are not using app names, Last.Updated, Current.Ver, or Android.Ver in our variables. They are not directly tied to app creation and can be and often are updated post app creation. It is important to note that when this code was run with installs as a numerical value, the results has 0 accuracy and the error percentage was very high. We converted the installs to factors and found the model was able to predict installs accurately.

**Overall accuracy of the model**

compTable <- data.frame(testData[,5],svmPred)

colnames(compTable) <- c("test", "prediction")

predacc <- length(which(compTable$test==compTable$prediction))/dim(compTable)[1]

predacc

**The accuracy of our model was tested a few times with a few different conditions and the best accuracy was 36%**. See the following table

|  |
| --- |
| C, cross Accuracy%  10, 10 35%  5, 5 33%  2, 2 31%  100, 100 36%  100, 10 36%  10, 100 33%  20, 20 34% |

**Searching for localized accuracy**

It is difficult to see any useful trends in our data from a point plot, so after careful consideration we decided to investigate using heat maps. We defined the error by converting the download categories to numerical values and found the numerical error in the estimation then divided the numerical error by the larger of the prediction and the test data. The fill will represent the percent error.

OutputHeatMapCategory <- ggplot(data = testData, aes(x = Installs, y = Category)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapCategory

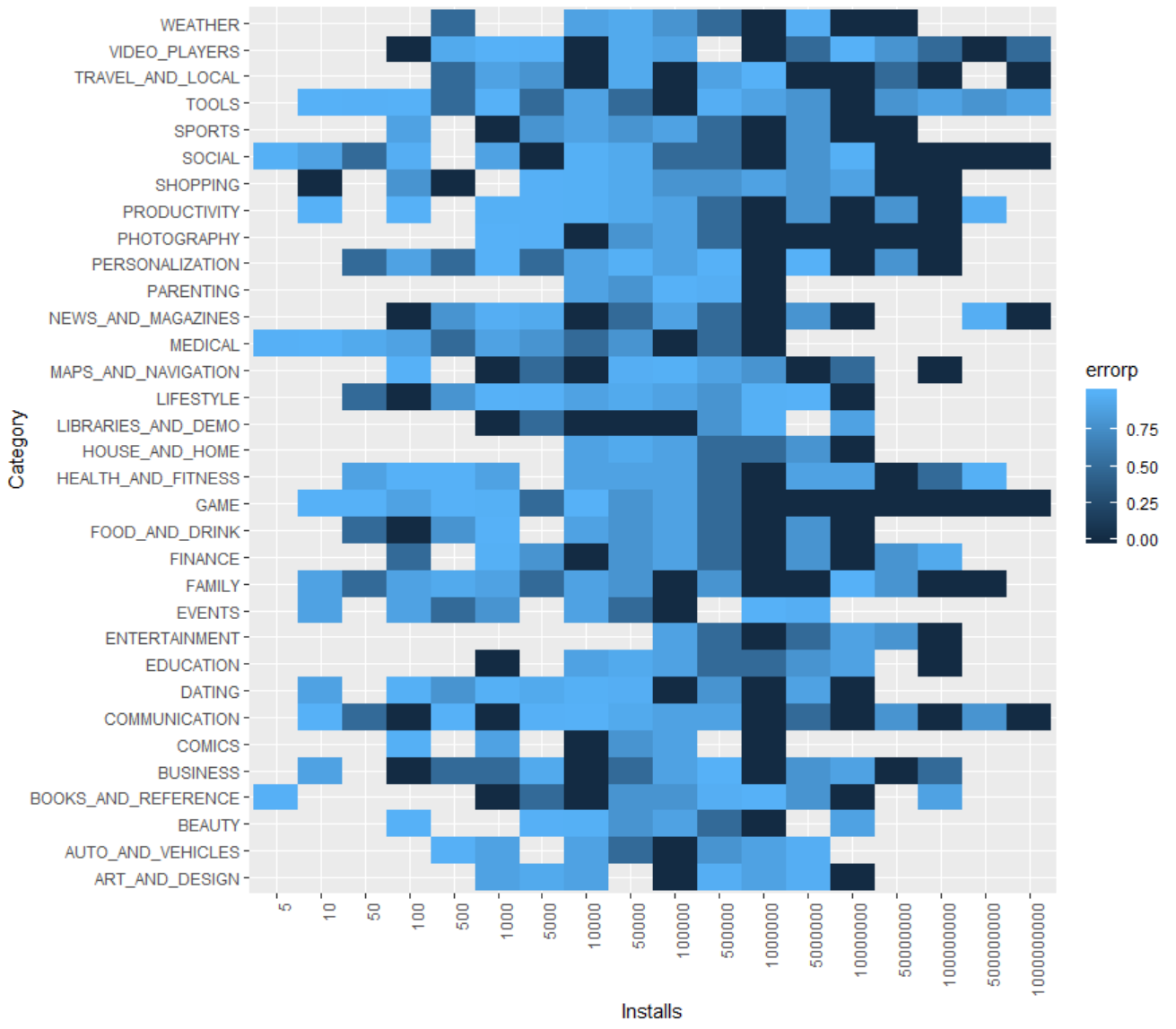


Figure 5

We can see that while the overall model is 36% accurate overall the accuracy prediction increases to ~66% for 1,000,000 installs. We can also see that the model has less error for 10,000,000 installs on this iteration. We saw 1,000 and 10,000 having less error as well during different iterations but all iterations were best at predicting 1,000,000. We note that we are unable to accurately predict any row for all install levels. If we were able to predict for any row we would be able to predict how many downloads an app would have given analyzed variables.

We looked at other heatmaps and found similar results.

Ratings to Installs:

OutputHeatMapRating <- ggplot(data = testData, aes(x = Installs, y = Rating)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapRating

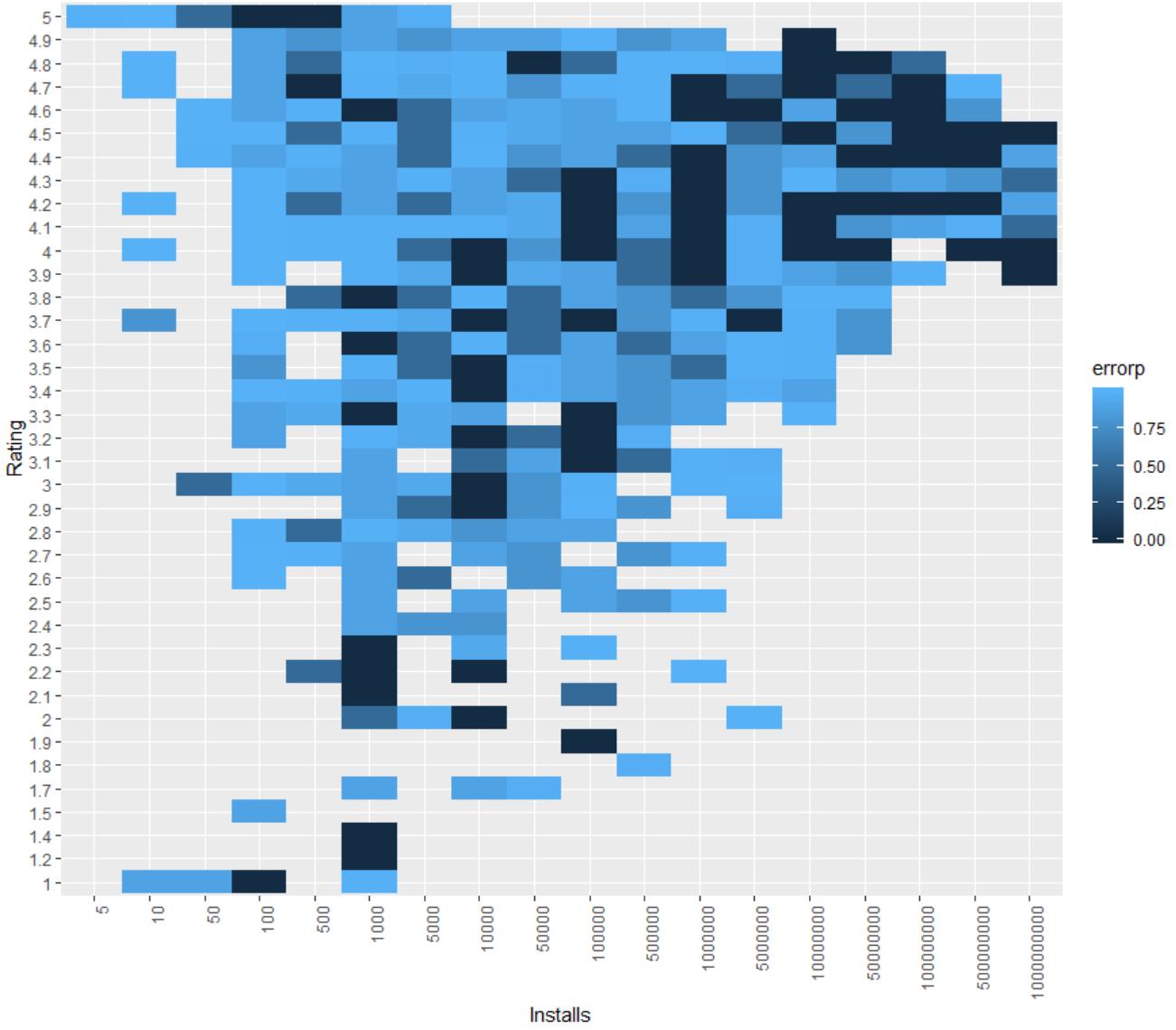


Figure 6

Content Rating to Installs

OutputHeatMapContent <- ggplot(data = testData, aes(x = Installs, y = Content.Rating)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapContent

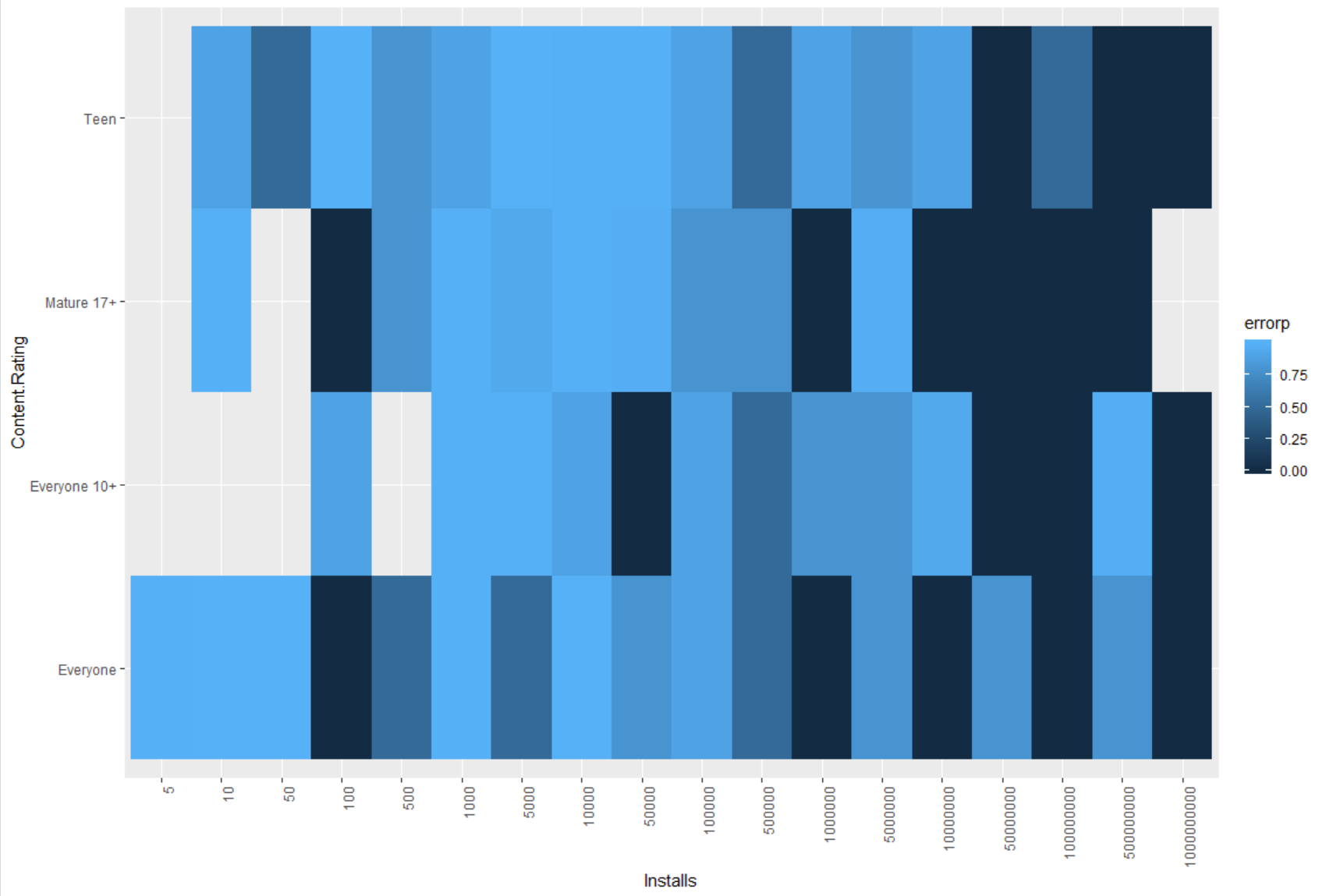


Figure 7

Paid or Free to Installs

OutputHeatMapType <- ggplot(data = testData, aes(x = Installs, y = Type)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapType

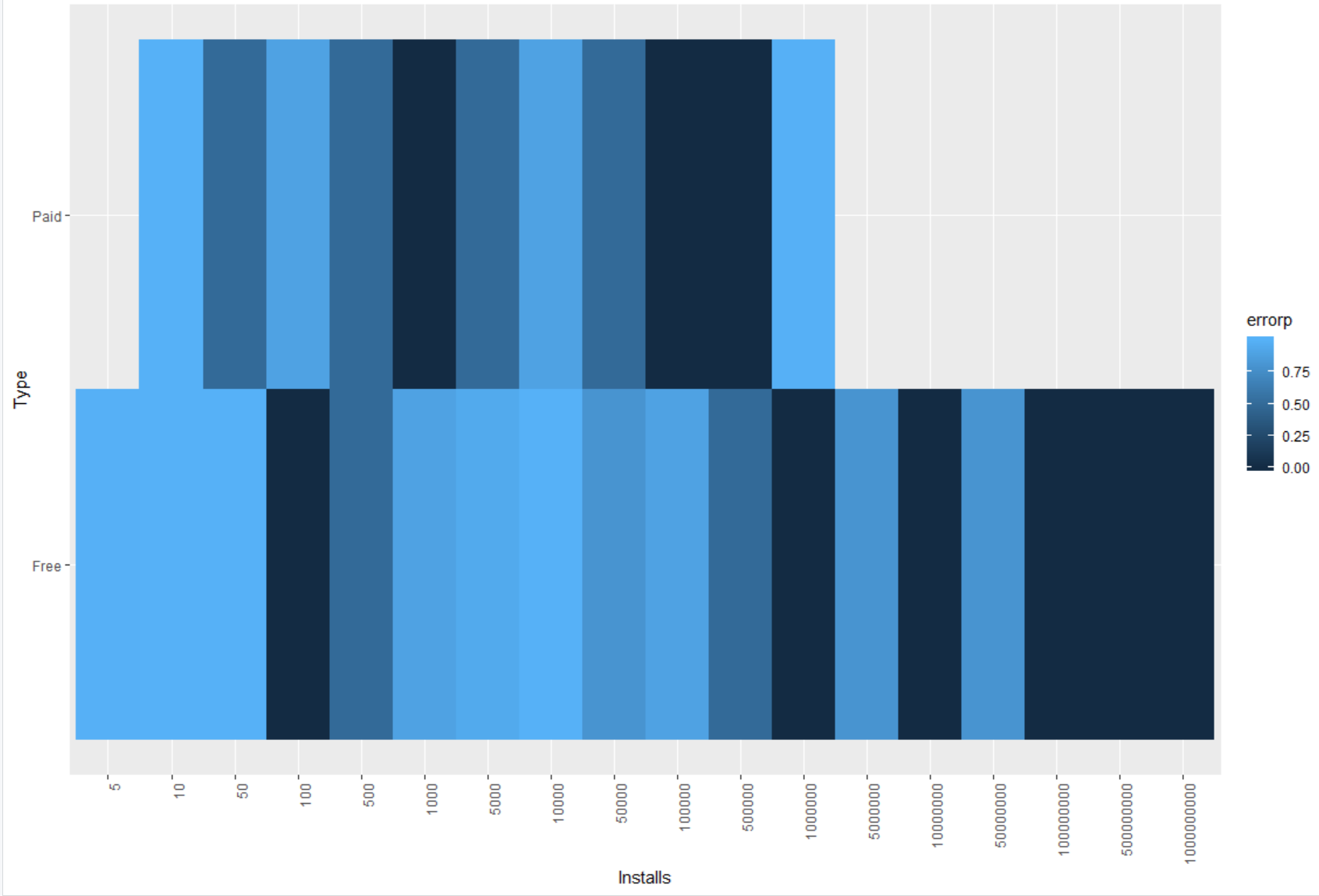


Figure 8

Genres to Installs

OutputHeatMapGenres <- ggplot(data = testData, aes(x = Installs, y = Genres)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapGenres

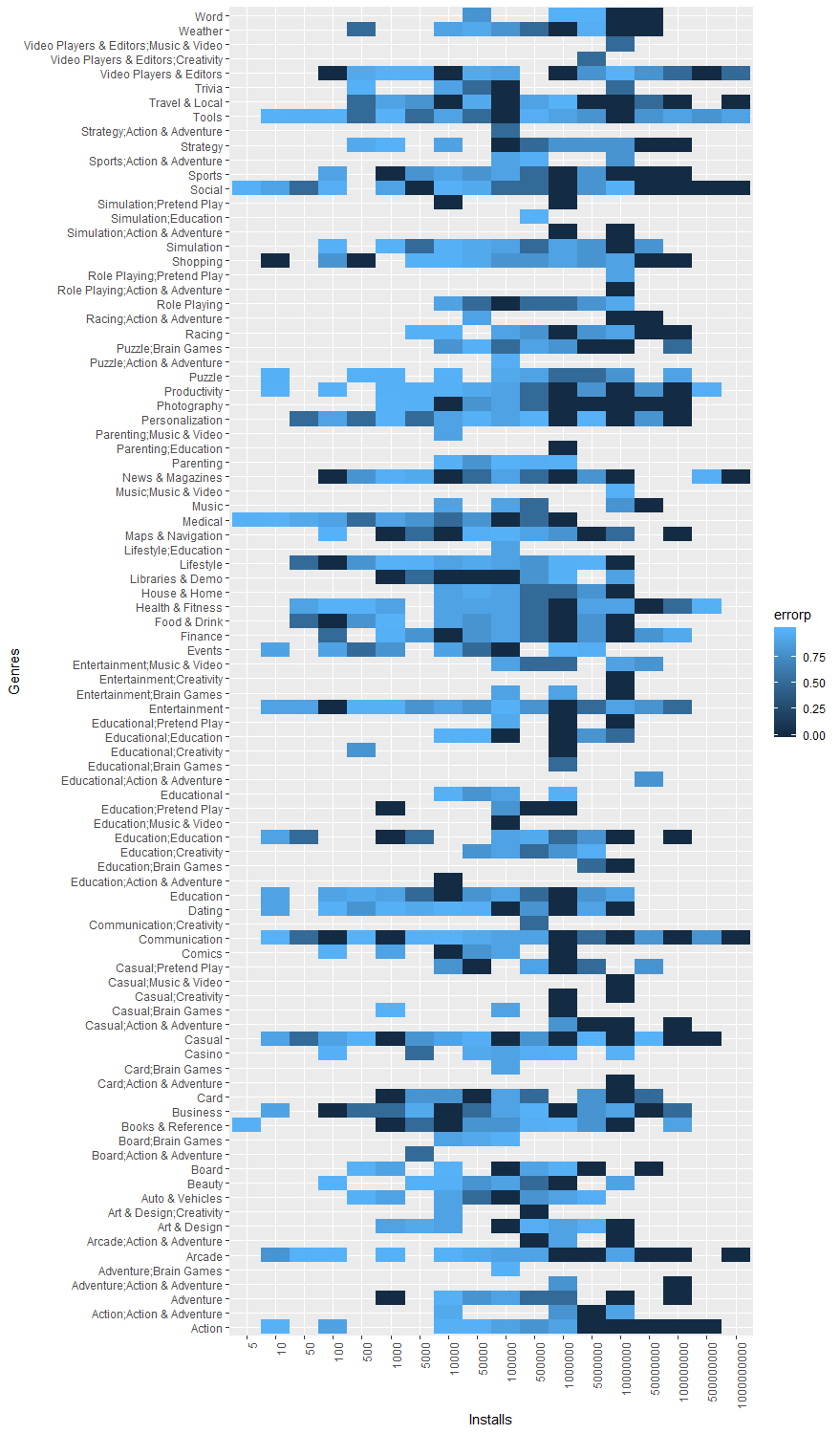


Figure 9

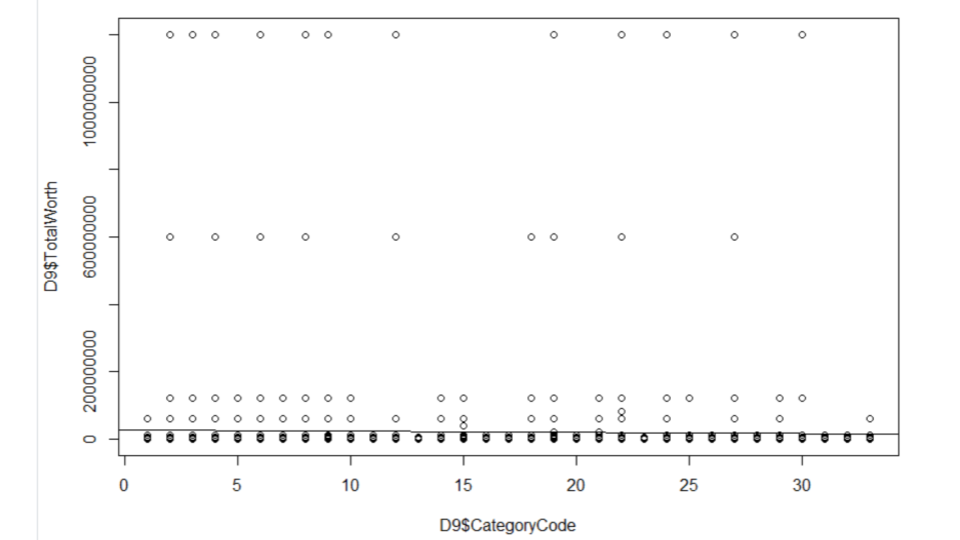
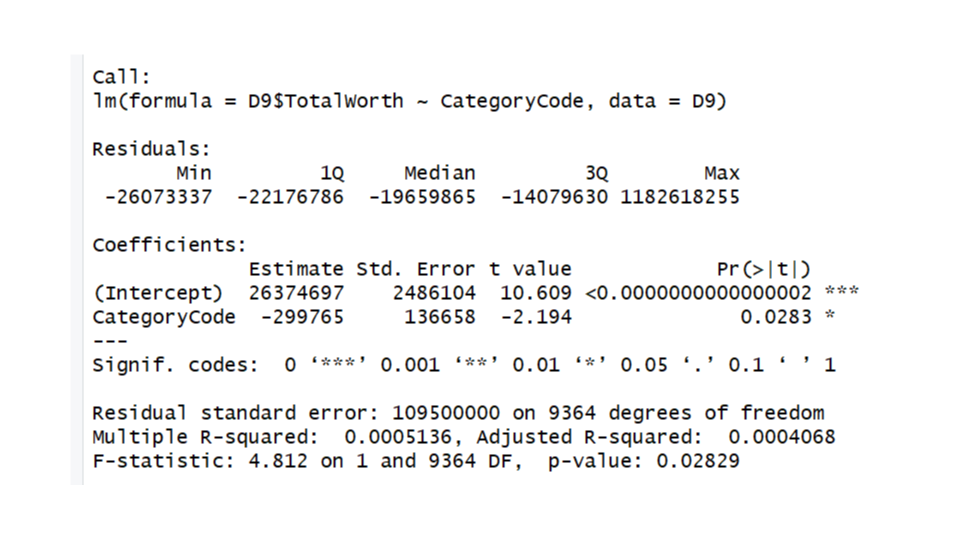
**Linear Modeling and Regression**

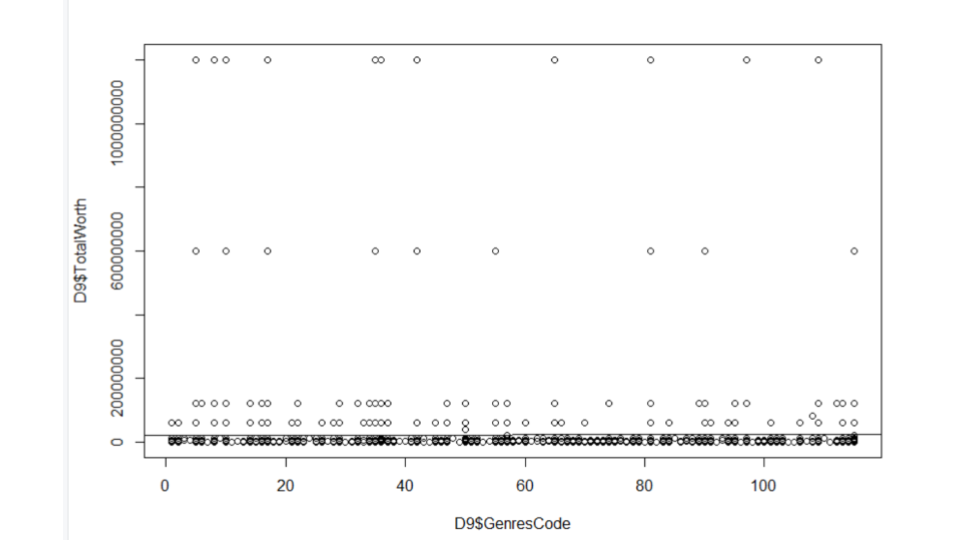
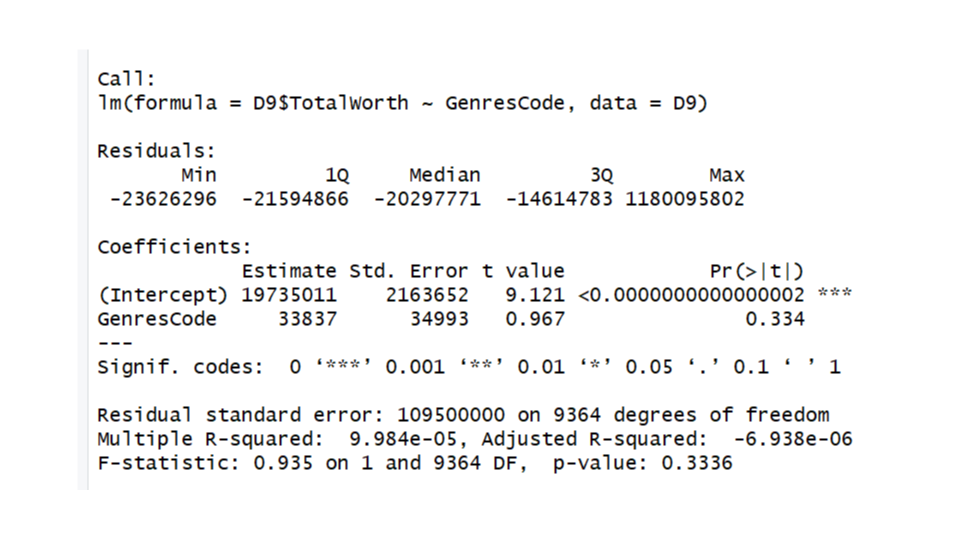
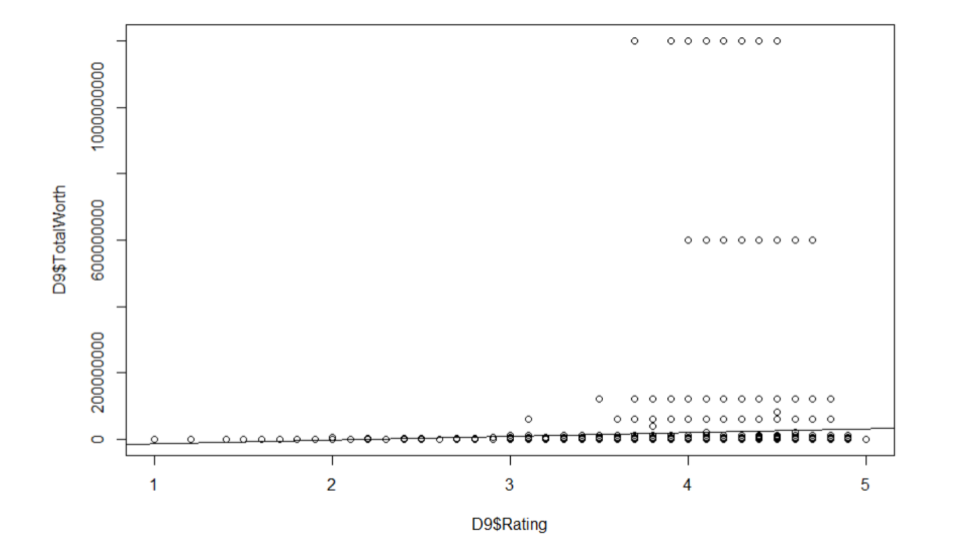
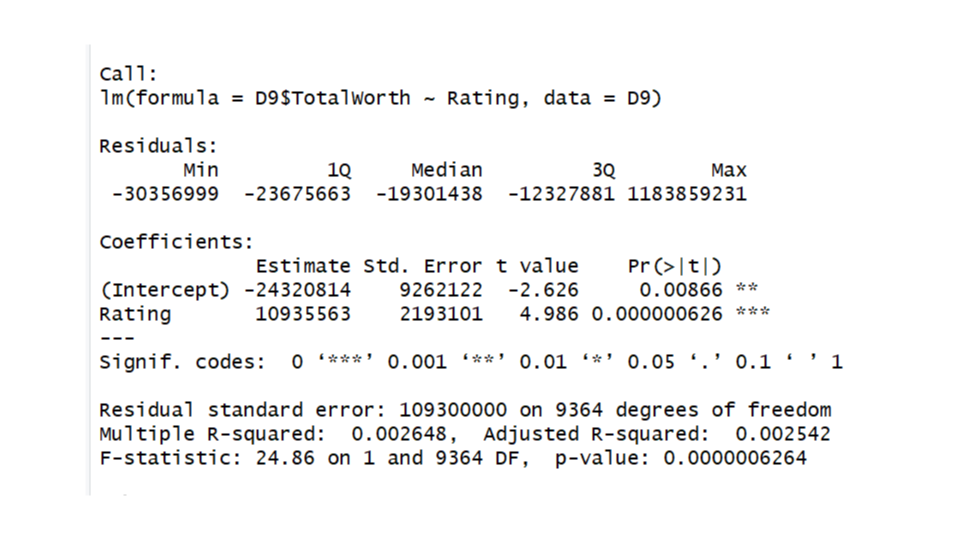
As we went through the linear regression we were checking for associations with total worth. We defined total worth as:

TotalWorth = (Installs \* Price) + (1.20 \* Installs)

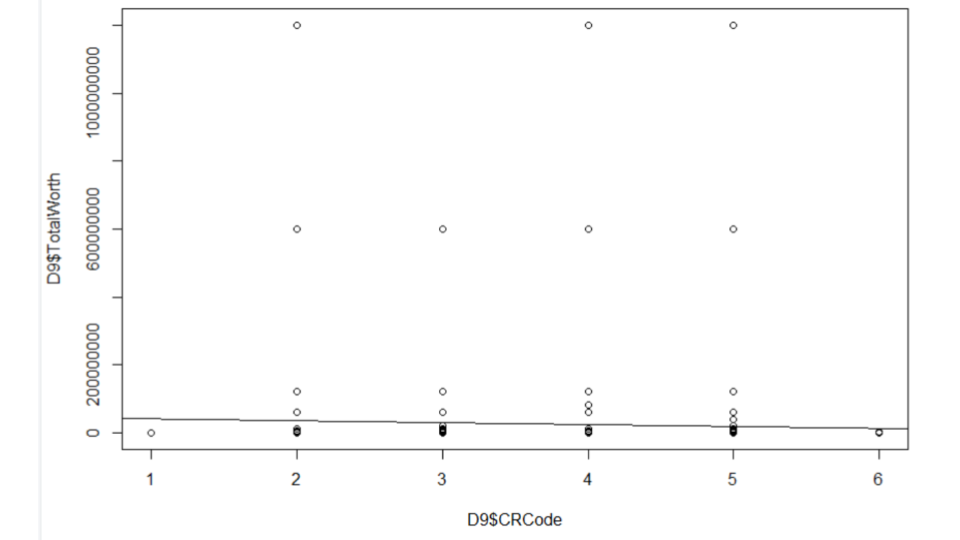
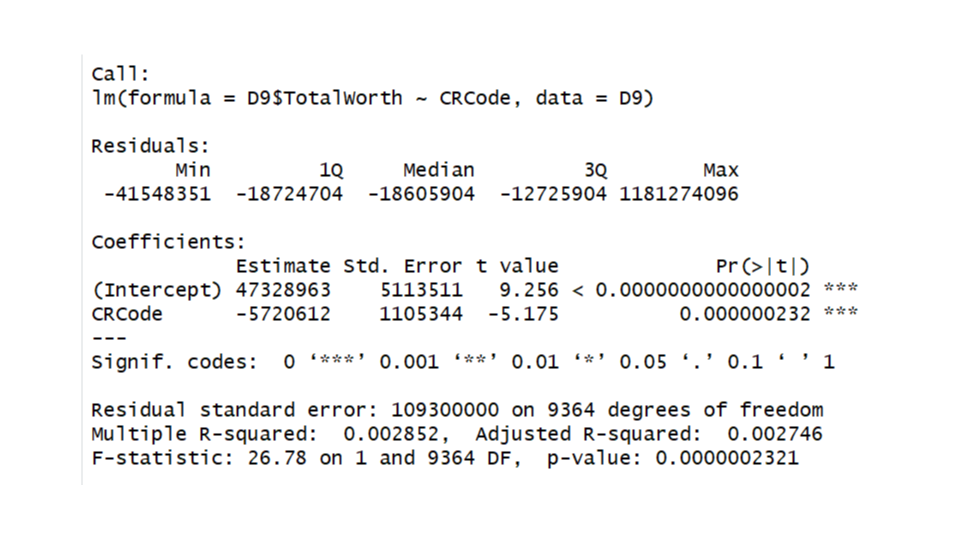
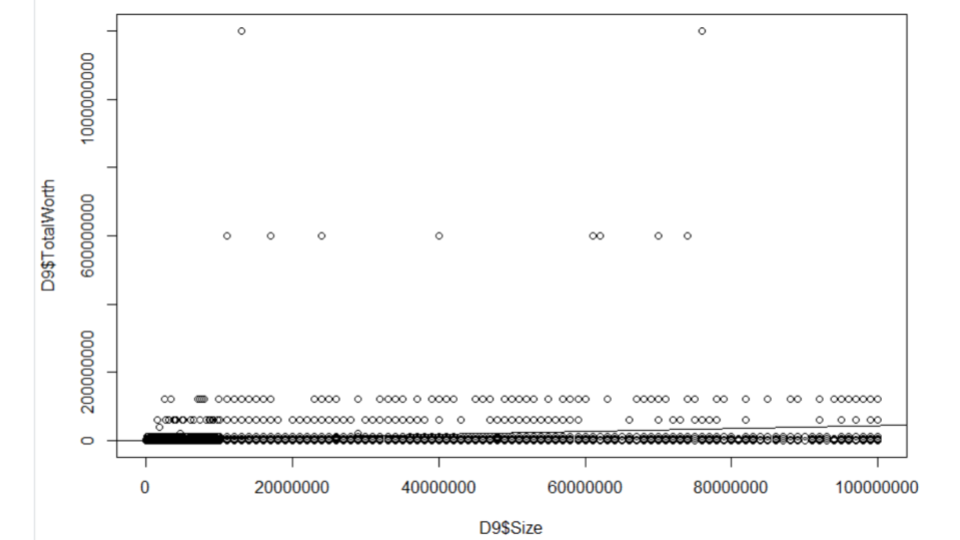
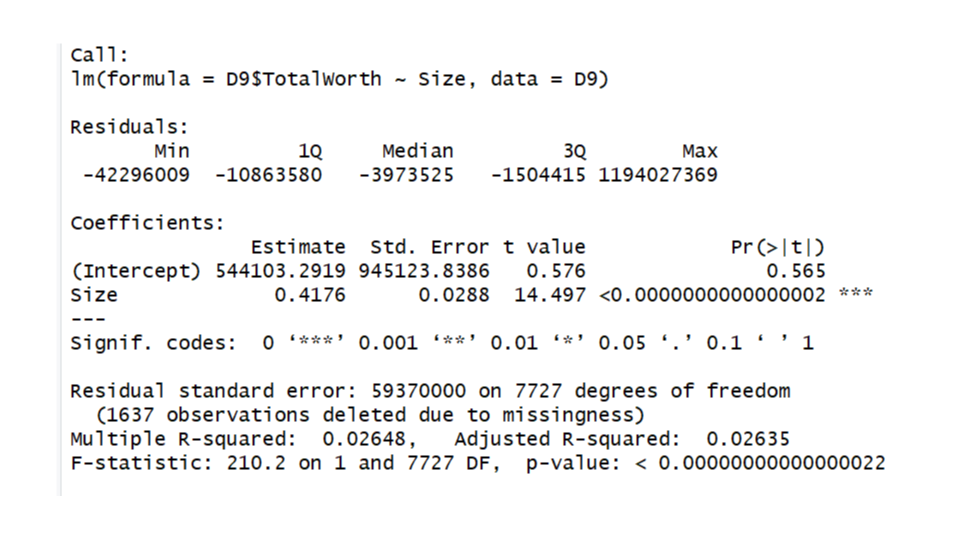
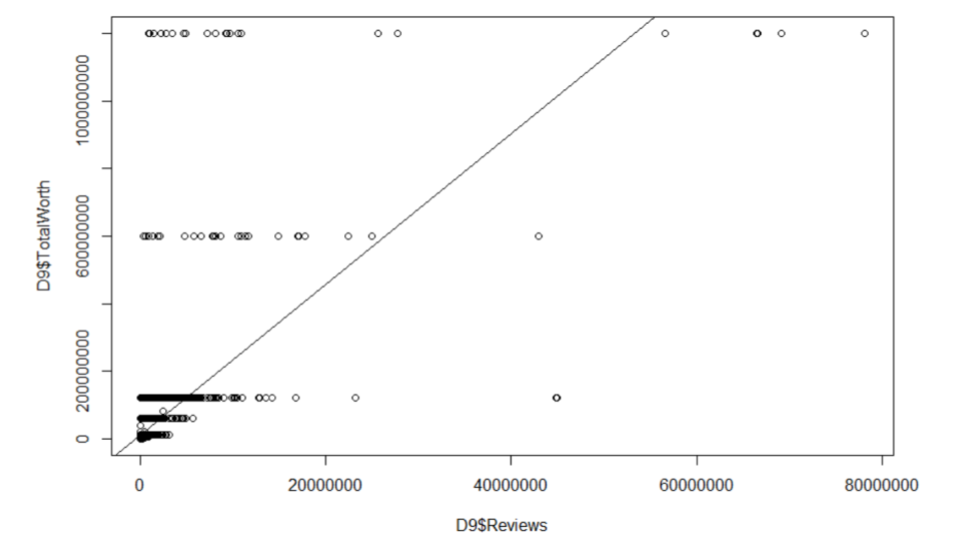
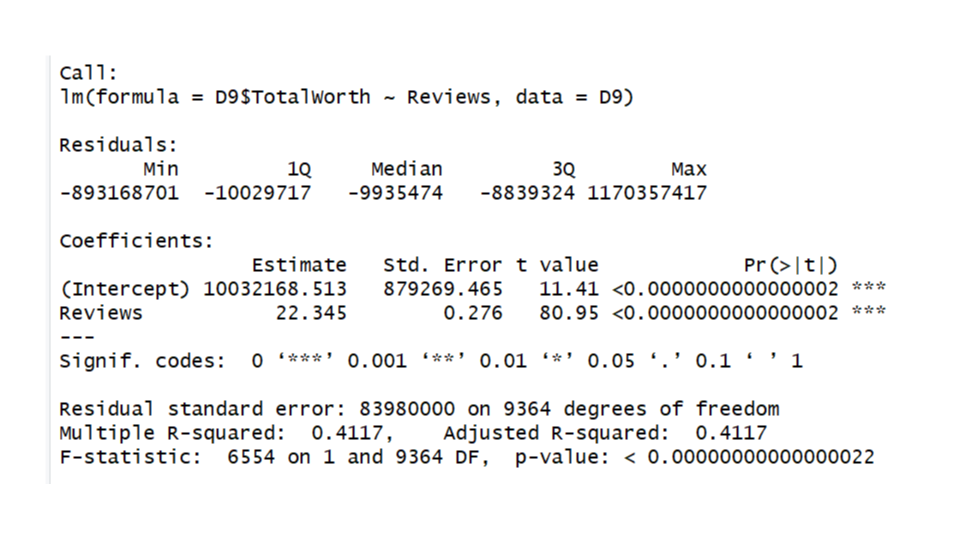
The $1.20 is an estimation of the value of each install based on <https://www.quora.com/How-much-is-an-app-user-worth>.

In our regression models where we were looking for correlation with any of our other variables we found very significant p values but our r values were always less than .5 and usually close to zero. We found large residuals in our models which indicates they are not particularly good. We did not analyze the correlation of installs and total worth because they would not be independent.

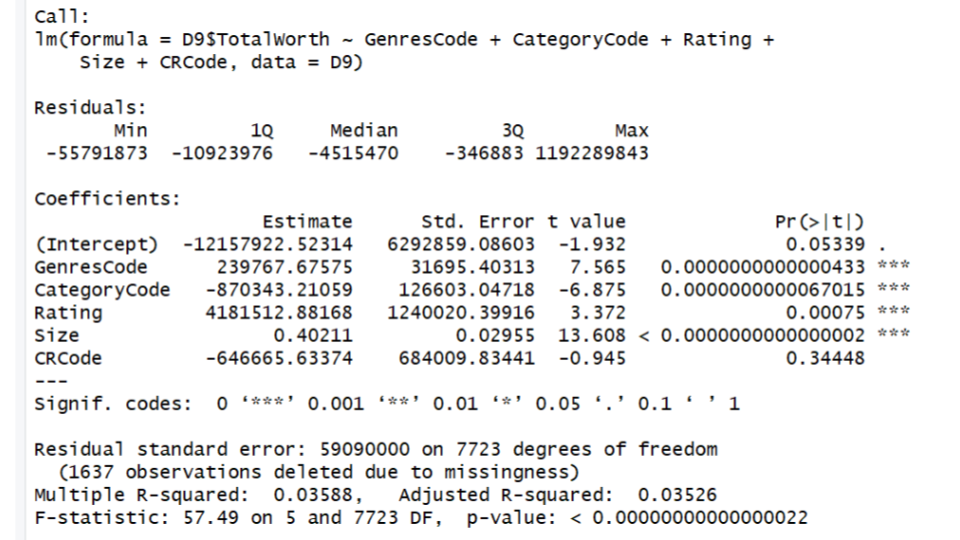
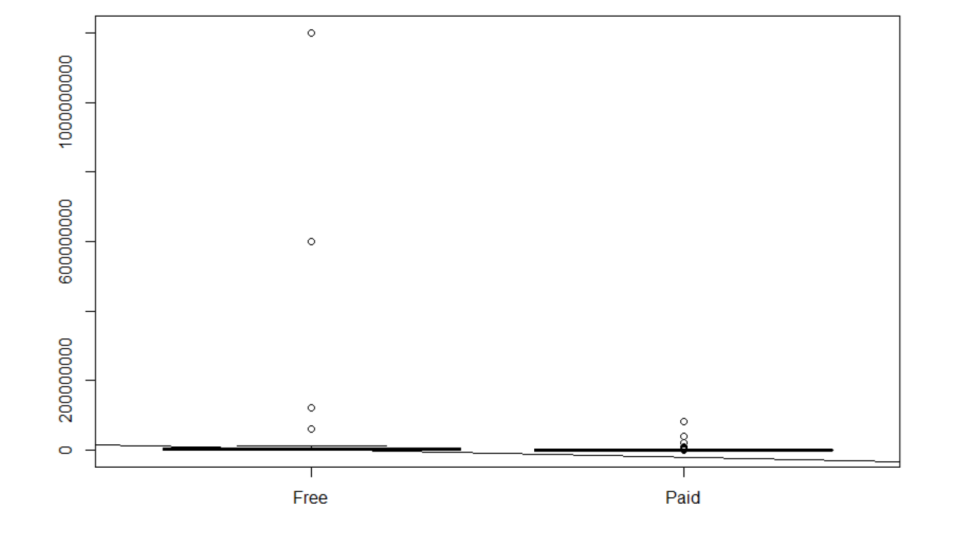
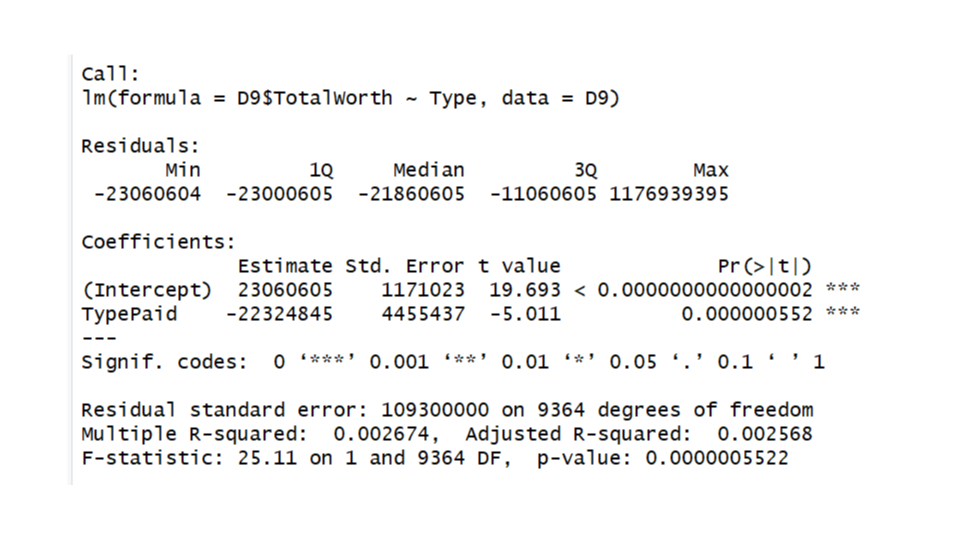




Figures 10-15



Figures 16-21



Figures 22-24

Relationship of Reviews and Ratings to Installs

# Model 9: Predict Installs from Reviews

m9 <- lm(formula = Installs ~ Reviews, data=D9)

# Display the regression analysis summary for Model 9:

summary(m9)

#

# Model 10: Predict Installs from Rating

m10 <- lm(formula = Installs ~ Rating, data=D9)

# Display the regression analysis summary for Model 10:

summary(m10)

#

# Model 11: Predict NetWorth from Installs

m11 <- lm(formula = TotalWorth ~ Installs, data=D9)

# Display the regression analysis summary for Model 11:

summary(m11)

**Model 9: Predict Installs from Reviews**

Call:  
lm(formula = Installs ~ Reviews, data = D9)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-744214668 -8325487 -8255133 -7376406 975332890   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 8326245.37 732763.70 11.36 <0.0000000000000002 \*\*\*  
Reviews 18.62 0.23 80.94 <0.0000000000000002 \*\*\*  
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
Residual standard error: 69990000 on 9364 degrees of freedom  
Multiple R-squared: 0.4117, Adjusted R-squared: 0.4116   
F-statistic: 6552 on 1 and 9364 DF, p-value: < 0.00000000000000022

**Model 10: Predict Installs from Rating:**

Call:  
lm(formula = Installs ~ Rating, data = D9)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-25247756 -19691242 -16053565 -10244146 986574693   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) -20223202 7718496 -2.620 0.0088 \*\*   
Rating 9094192 1827599 4.976 0.000000661 \*\*\*  
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
Residual standard error: 91120000 on 9364 degrees of freedom  
Multiple R-squared: 0.002637, Adjusted R-squared: 0.002531   
F-statistic: 24.76 on 1 and 9364 DF, p-value: 0.0000006606

**Model 11: Predict Total Worth from Installs:**

Call:  
lm(formula = TotalWorth ~ Installs, data = D9)  
  
Residuals:  
 Min 1Q Median 3Q Max   
 -42713 -42709 -42644 -42024 69857976   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 42712.7179778 12075.0903805 3.537 0.000406 \*\*\*  
Installs 1.1999311 0.0001299 9238.880 < 0.0000000000000002 \*\*\*  
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
Residual standard error: 1147000 on 9364 degrees of freedom  
Multiple R-squared: 0.9999, Adjusted R-squared: 0.9999   
F-statistic: 8.536e+07 on 1 and 9364 DF, p-value: < 0.00000000000000022

**Linear Regression Conclusion:**

Our goal was to predict the Total Worth of an app based on the data provided. Since the Total Worth was calculated based on the Installs variable (see Model 11), the R-squared for Installs vs. Total Worth was .9999 (i.e. 1) meaning it was perfectly collinear and logically couldn’t be used to predict Total Worth. Using models 1 through 7, we see that while all of the remaining variables were significant (p-value < .05), none of the independent variables had an adjusted R-squared of greater than 3% except for the number of Reviews which had an R-squared value of .4117 and was much more predictive that the other variables. However, logically the number of reviews is highly correlated to the number of installs (i.e. the more installs the more likely someone is to write a review). In Model 8, when we used all of the other variables in an attempt to predict Total Worth, we again had a significant result (p-value < .05); however, the total R-squared was only .03526, meaning that all of the variables together only explained 3.5% of the Total Worth. Consequently, we had to conclude that we could not successfully predict the Total Worth of an app based on the data which we had.

A separate business question asked at the beginning was the relationship of Installs to Reviews and Ratings. Based on Models 9 and 10, Ratings had little relationship to installs (low R-squared). Reviews had a higher correlation to Installs (.4117).

**Conclusions**

With the analysis we did we were able to provide the following answers.

**Which apps are downloaded the most?   
 Which categories?**

FAMILY 1747  
 GAME 1097

TOOLS 734  
 PRODUCTIVITY 351  
 MEDICAL 350  
 **Which age groups?**

Everyone 7420  
 Teen 1084  
 Mature 17+ 461

Everyone 10+ 397

Adults only 18+ 3

**Type (Paid vs. Free)**

FREE 8719

PAID 647  
**How can I make a “best selling”/highly installed app?   
Which categories have the highest ratings?**

EVENTS 4.435556  
 EDUCATION 4.389032  
 ART\_AND\_DESIGN 4.358065  
 BOOKS\_AND\_REFERENCE 4.346067  
 PERSONALIZATION 4.335987

**Are ratings or reviews more correlated with installs?**

Reviews are more correlated than ratings with installs; however a larger number of installs makes a larger number of reviews more likely. Ratings were not highly correlated with installs (low R-squared).

**Does size or installs have any effect on price?**

No, they do not.

**Does price have any effect on size or installs?**

Yes, both price and size have an effect on size and installs.

**Can we predict high total worth of an app with category, rating, reviews, size, pay type, and/or content rating?**

We are unable to do so using linear regression.

**Can we predict installs for a given category?**

We can predict better than randomly guessing, but we still only have a 36% accuracy at best.

During our analysis we did take a look at the 15 most expensive apps and we noticed two things about them. First, they all had some variation of the phrase “I am rich”. Second, all of them had over 100 downloads. Surprisingly, what we would deem unethical on first glance, this seems to be the best way to improve the likelihood of launching a successful app. While this finding is interesting and potentially useful, we do not recommend this approach.

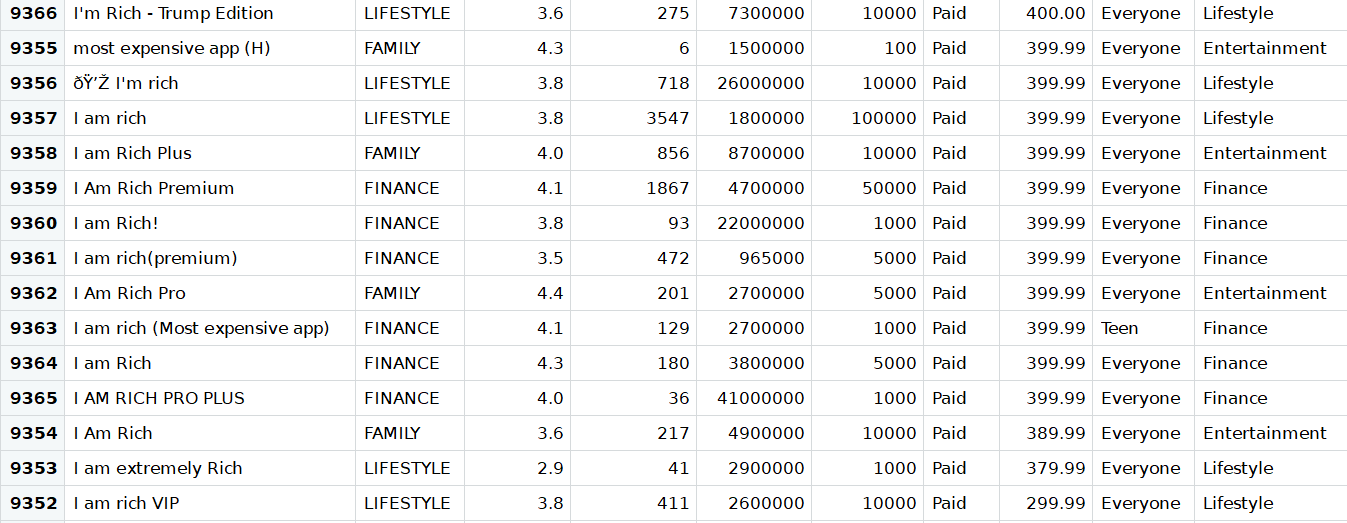
****

Figure 25

We determined that we cannot accurately predict an apps popularity/profitability with the analysis run and with the data provided. We would hypothesize that with text mining or natural language processing and review data, we might be able to better predict attributes of successful apps.

**Appendix A: Unexecuted R Code**

###################################Supporting Library install and load#######################################

install.packages("package\_name") #The function install.packages() is used to install a package from CRAN

install.packages("devtools")#Install before data.table

remove.packages("data.table")

install.packages("data.table")

#remove.packages("read.table")

#install.packages("read.table")

install.packages("Rcpp")

install.packages("readr")

install.packages("sqldf")

install.packages("moments")

install.packages("reshape2")

install.packages("ggplot2")

install.packages("openintro") # states data

install.packages("ggmap")

install.packages("readxl")

install.packages("gdata") # to reformat some data sets, such as using cbindX function

install.packages("zipcode")

install.packages("gsubfn")

install.packages("e1071", dep = TRUE)

install.packages("gridExtra")

install.packages("RH2")#need to extract year from date field

install.packages("Metrics")

library(devtools)

library(data.table)

#library(read.table)

library(Rcpp)

library(readr)

library(sqldf)

library(moments)

library(reshape2)

library(ggplot2)

library(openintro)

library(ggmap)

library(readxl)

library(gdata)

library(zipcode)

library(gsubfn)

library(e1071)

library(gridExtra)

library(RH2)

library(Metrics)

packages=c("arulesViz", "kernlab","e1071","gridExtra","ggplot2", "caret", "arules")

#use this function to check if each package is on the local machine

#if a package is installed, it will be loaded

#if any are not, the missing package(s) will be installed and loaded

package.check <- lapply(packages, FUN = function(x) {

if (!require(x, character.only = TRUE)) {

install.packages(x, dependencies = TRUE)

library(x, character.only = TRUE)

}

})

#mapping factor to character

library(purrr)

library(dplyr)

#Date format

install.packages("tidyverse")

install.packages("lubridate")

install.packages("nycflights13")

library(tidyverse)

library(lubridate)

library(nycflights13)

#verify they are loaded (library)

search()

###################################FUnction declearation#######################################

#function readGoogleSalesData to read data from CSV

#@param verctor for file path and file name

readGoogleSalesData <- function(csvFile)

{

csvFile

setwd(csvFile[1])

googleSalesData <- read.csv(csvFile[2])

googleSalesDataCleaned <- dataCleanUpProcess(googleSalesData)

return(googleSalesDataCleaned)

}

#function dataCleanUpProcess to clean up data

#@param verctor for file path and file name

dataCleanUpProcess <- function(salesdfCleaneUp){

columnNames <- colnames(salesdfCleaneUp)

for (i in c(1:length(columnNames))) {

print(paste(columnNames[i]))

if(columnNames[i] == "Rating"){

salesdfCleaneUp <- salesdfCleaneUp[salesdfCleaneUp$Rating<=5,] #remove data if rating is more than 5

salesdfCleaneUp$Rating[is.nan(salesdfCleaneUp$Rating)] <- mean(salesdfCleaneUp$Rating, nan.rm=TRUE) #replace NaN with mean of Rating

}

else if(columnNames[i] == "Reviews"){

salesdfCleaneUp$Reviews <- as.numeric(salesdfCleaneUp$Reviews) #Convert Reviews column to as numeric

}

else if(columnNames[i] == "Size"){

salesdfCleaneUp <- salesdfCleaneUp[complete.cases(salesdfCleaneUp), ] #Remove all Null/Na/NaN from the data set as it's creating forblem for data manipulation.

salesdfCleaneUp$Size <- gsub("Varies with device","0",salesdfCleaneUp$Size) # Replace "Varies with device" with "0"

options(scipen=999) #

salesdfCleaneUp$Size <- as.double(sub('\\D$', '', salesdfCleaneUp$Size))\*c(1e9, 1e6, 1e3)[match( sub('\\d\*\\.\*\\d\*', '', salesdfCleaneUp$Size), c('B', 'M', 'k'))] #Change download size 1.2M or 1.8k to double format.

}

else if(columnNames[i] == "Installs"){

salesdfCleaneUp$Installs <- gsub("\\D|\\s","",salesdfCleaneUp$Installs) #Remove all characters and space from Installs

salesdfCleaneUp$Installs <- as.numeric(salesdfCleaneUp$Installs) # convert factor to numeric data type

}

else if(columnNames[i] == "Price"){

salesdfCleaneUp$Price <- as.numeric(gsub("[\\$,]", "", salesdfCleaneUp$Price)) #Remove '#' and ',' from Price and convert to numeric from Factor data type

}

else if(columnNames[i] == "Last.Updated"){

#unlist(strsplit(salesdfCleaneUp$Last.Updated, ",|\\s"))

salesdfCleaneUp$Last.Updated <- mdy(salesdfCleaneUp$Last.Updated) #Convert date format from 'January 07, 2018' to '2018-01-07' as Date type

# salesdfCleaneUp$Last.Updated <- as.character.Date(salesdfCleaneUp$Last.Updated,"%m-%d-%Y")

}

#salesdfCleaneUp <- data.frame(lapply(salesdfCleaneUp, as.character), stringsAsFactors=FALSE)

salesdfCleaneUp %>% map\_if(is.factor, as.character) %>% as\_data\_frame -> salesdfCleaneUp #Convert all Factors to Character type

salesdfCleaneUp[, c("Category", "Type", "Content.Rating", "Genres", "Current.Ver", "Android.Ver")] <- lapply(salesdfCleaneUp[, c("Category", "Type", "Content.Rating", "Genres", "Current.Ver", "Android.Ver")], factor) #COnverted some specific column back to Factor from Character type

}

return(salesdfCleaneUp)

}

# create a function to do sampling

printVecInfo <- function(v, x){

samp <- sample(v,x,replace=TRUE)

samp <- samp[!is.na(samp)]

meanNum <- sum(samp)/length(samp)

return(meanNum)

}

###################################Main Scope of the execution#######################################

csvFile <- c("C:\\Project\\", "googleplaystore.csv") #Prepare vector to read csv file

googlePlayStoreData<-readGoogleSalesData(csvFile) #Call function readGoogleSalesData to read csv file and load data into data frame.

rownames(googlePlayStoreData) <- NULL #Reset the row index for dataset googlePlayStoreData

str(googlePlayStoreData) #display the structure of the dataset googlePlayStoreData

summary(googlePlayStoreData) #display the summery of the dataset googlePlayStoreData

D <- googlePlayStoreData

#@histogram---------------------------------Start------------------------------

hist(D$Installs)

plot(D$Last.Updated,D$Installs, type ="h")

plot(D$Last.Updated, D$Rating, type ="b")

quantileInstallData <- quantile(D$Installs, c(0.90, 0.99))

quantileInstallData

skewnessPriceData <- skewness(D$Price)

skewnessPriceData

#@histogram---------------------------------END--------------------------------

#@Sampling---------------------------------Start-------------------------------

Price <- replicate(20,mean(replicate(10,printVecInfo(D$Price, 10))))

hist(Price)

Size <- replicate(20,mean(replicate(10,printVecInfo(D$Size, 10))))

hist(Size)

Installs <- replicate(20,mean(replicate(10,printVecInfo(D$Installs, 10))))

hist(Installs)

Reviews <- replicate(20,mean(replicate(10,printVecInfo(D$Reviews, 10))))

hist(Reviews)

Rating <- replicate(20,mean(replicate(10,printVecInfo(D$Rating, 10))))

hist(Rating)

#@Sampling---------------------------------END---------------------------------

#@Plot and GGPlot---------------------------------Start------------------------

namesOfColumns <- c("App","Category","Rating","Reviews","DownloadSize","Installs","Type","Price","AgeGroup","Genres","LastUpdated","CurrentVer","AndroidVer")

colnames(D) <- namesOfColumns

#Extracting year from Last.update and adding to the data set

#D <- D[,-14]#Used it during denug and clean drop the column

UpdateYear <- sqldf("select year(LastUpdated) UpdateYear from D")

D <- cbind(D, UpdateYear)

#download based on type, Category, Year

appDownloadByTypeCategory <- sqldf("select sum(DownloadSize) TotalDownload, Type, Category, UpdateYear from D group by Type, Category, UpdateYear")

#install based on type, Category, Year

appInstalledByTypeCategory <- sqldf("select sum(Installs) SumInstalled, Type, Category, UpdateYear from D group by Type, Category, UpdateYear")

#App(x) download(BLOCk PLOT) based on Price(Y)

appDownloadByPrice <- sqldf("select App, sum(DownloadSize) TotalDownload, Price from D group by App, Price")

#App(x) installed(BLOCk PLOT) based on Price(Y)

appInstalledByPrice <- sqldf("select App, sum(Installs) SumInstalled, Price from D group by App, Price")

#App(x) download(BLOCk PLOT) based on Category(Y)

appDownloadByCategory <- sqldf("select App, sum(DownloadSize) TotalDownload, Category from D group by App, Category")

#App(x) installed(BLOCk PLOT) based on Category(Y)

appInstalledByCategory <- sqldf("select App, sum(Installs) SumInstalled, Category from D group by App, Category")

#Top rating five application download rate over the year.

GoogleData<- googlePlayStoreData[,2:9]

colnames(GoogleData)[colSums(is.na(GoogleData)) > 0]

GoogleData$Size[is.na(GoogleData$Size)] <- mean(GoogleData$Size, na.rm=TRUE) # find the NAs in column "Ozone" and replace them by the mean value of this column

GoogleData$Installs<-as.factor(GoogleData$Installs)

#create a random index for test data

randIndex <- sample(1:dim(GoogleData)[1])

#create a cutpoint for 2/3 test data

cutpoint2\_3<- floor(2\*dim(GoogleData)[1]/3)

#create a dataframe for the training data

trainData <- GoogleData[randIndex[1:cutpoint2\_3],]

dim(trainData)

head(trainData)

#create a dataframe for the test data

testData<-GoogleData[randIndex[(cutpoint2\_3+1):dim(GoogleData)[1]],]

#Set Installs as the target predicting variables

svmOutput <- ksvm(Installs~., data = trainData,kernel = "rbfdot", C = 100,cross =10,prob.model = TRUE)

svmPred <- predict(svmOutput,testData)

compTable <- data.frame(testData[,5],svmPred)

colnames(compTable) <- c("test", "prediction")

predacc <- length(which(compTable$test==compTable$prediction))/dim(compTable)[1]

predacc

#converting from factors to numeric, have to use character otherwise converts to assigned factor number

compTable$test <-as.character(compTable$test)

compTable$test <-as.numeric(compTable$test)

compTable$prediction <-as.character(compTable$prediction)

compTable$prediction <-as.numeric(compTable$prediction)

testData$Installs <-as.character(testData$Installs)

testData$Installs <-as.numeric(testData$Installs)

testData$prediction <- compTable$prediction

#inserting the error into the testdata

testData$errors<- abs(compTable$test-compTable$prediction)

testData$errora<-with(testData, pmax(Installs, prediction))

testData$errorp<-testData$errors/testData$errora

#plotting error

OrdTest<-testData[order(-testData$errorp),]

svmPlot <- ggplot(OrdTest) + geom\_point(aes(x=Category, y=Installs, color=errorp, size=errorp))

svmPlot

#converting installs back to factors for plotting the heat map

testData$Installs <-as.factor(testData$Installs)

OutputHeatMapCategory <- ggplot(data = testData, aes(x = Installs, y = Category)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapCategory

# While this model is overall only about 35% accuratye, we do notice that it is very good at predicting apps that will hit one million downloads.

# we also notice that it is pretty good at predicting sports, social, shopping, games, communications, and books and reference categories.

testData$Rating<-as.factor(testData$Rating)

OutputHeatMapRating <- ggplot(data = testData, aes(x = Installs, y = Rating)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapRating

OutputHeatMapContent <- ggplot(data = testData, aes(x = Installs, y = Content.Rating)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapContent

OutputHeatMapType <- ggplot(data = testData, aes(x = Installs, y = Type)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapType

OutputHeatMapCType <- ggplot(data = testData, aes(x = Category, y = Type)) + geom\_tile(aes(fill = errorp)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))

OutputHeatMapCType

# From these heatmaps we can start to have an idea of catebories to focus our efforts on in order to hit 1000000 downloads

#@Plot and GGPlot--------------------------------ENd--------------------------------

#@Hypothesis Test---------------------------------Start-----------------------------

#@Test 1 (Check if DownloadSize or Installation does have any effect on Price)

#Priced based on DownloadSize and Installation

hypothesisD <- sqldf("select sum(Price) Price,DownloadSize, Installs from D group by DownloadSize, Installs")

hypothesisD <- hypothesisD[complete.cases(hypothesisD), ]

#salesdfCleaneUp <- salesdfCleaneUp[complete.cases(salesdfCleaneUp), ]

modelDownload <- lm(formula = hypothesisD$Price ~ hypothesisD$DownloadSize, data = hypothesisD)

#D <- sqldf("select App, Installs, Category, Price from D group by Category, Price")

modelInstalled <- lm(formula = hypothesisD$Price ~ hypothesisD$Installs, data = hypothesisD)

#Hypothesis test if Price has any effect on download and

anova(modelInstalled,modelDownload)

#@Conclude :

#The result shows that model 1 and 2 does not provide a significantly better fit to the data compared to model.

#It means both of them Download Size and Installation does not have any significance on price.

#@Test 2

#@Test 1 (Check if Price has any effect on TotalDownloa or Installation)

modelDownload <- lm(D$DownloadSize ~ D$Price, data = D)

plot(D$DownloadSize ~ D$Price, D)

abline(modelDownload)

g <- ggplot(D, aes(x = D$Price, y = D$Category)) + geom\_point(aes(size = D$DownloadSize, color = D$DownloadSize))

g <- g + stat\_smooth(method = "lm")

g

summary(modelDownload)

#abline(coef = coef(modelDownload))//Same as above line

sum.model <- summary(modelDownload)

sum.model$adj.r.squared

modelInstalled <- lm(formula = D$Installs ~ D$Price, D)

plot(D$Installs ~ D$Price, D)

abline(modelInstalled)

g <- ggplot(D, aes(x = D$Price, y = D$Category)) + geom\_point(aes(size = D$Installs, color = D$Installs))

g <- g + stat\_smooth(method = "lm")

g

summary(modelInstalled)

sum.model <- summary(modelInstalled)

sum.model$adj.r.squared

#@Conclude :

# In first test P is 0.0211 (P < Alphs(0.05)). P is low Home must go. It means Price does not have any effect on Total download.

# In second test P is 0.2499 (P > Alphs(0.05)). In this case we could not reject Home. It means Price has effect on Installation.

#@Hypothesis Test---------------------------------ENd-------------------------------

#@Linear regression module---------------------------------Start--------------------

#Find which category of application people like to download most (top five) and show the comparison between paid and free.

#Predict best selling app in Linear model for 2019 of total worth.

#Find which category of application people like to download most (top five) and show the comparison between paid and free.

#Top 5 overall categories of installed apps for all years

appInstalledByCategory <- sqldf("select Category, count(Category) Category\_Ct from D group by Category order by Category\_Ct DESC")

appInstalledByCategory[1:5,]

# Top overall downloaded free apps by category

appFreeInstalledByCategory <- sqldf("select Category, count(Category) Category\_Ct from D WHERE Price = 0 group by Category order by Category\_Ct DESC")

appFreeInstalledByCategory[1:5,]

# Top overall downloaded paid apps Category Type

appFreeInstalledByCategory <- sqldf("select Category, count(Category) Category\_Ct from D WHERE Price > 0 group by Category order by Category\_Ct DESC")

appFreeInstalledByCategory[1:5,]

#

# Copy the data frame to allow analysis and changes without impacting other code

D9 <- D

#

colnames(D9) <- c("App","Category","Rating", "Reviews", "Size", "Installs", "Type", "Price", "ContentRating", "Genres", "LastUpdated", "CurrentVer", "AndroidVer","UpdateYear")

#Determine the distinct categories

DistinctCategory <- sqldf("select Distinct(Category), 0 int from D9 order by Category DESC")

colnames(DistinctCategory) <- c("Category","CategoryCode")

DistinctCategory

DistinctCategory$CategoryCode <- as.integer(rownames(DistinctCategory))

DistinctCategory

#Add a ColumnCategory Code to D which is the numeric equivalent of the Category

D9$CategoryCode <- match(D9$Category,DistinctCategory$Category,nomatch=0)

#

#Determine the distinct Genres

DistinctGenres <- sqldf("select Distinct(Genres) from D9 order by Genres DESC")

#Assign the Row number as the numeric equivalent of the Genre

DistinctGenres$GenresCode <- as.integer(rownames(DistinctGenres))

DistinctGenres

#Add a Column Category Genres to D which is the numeric equivalent of the Genres

D9$GenresCode <- match(D9$Genres,DistinctGenres$Genres,nomatch=0)

#

#Determine the distinct Content.Rating

DistinctCR <- sqldf("select Distinct(ContentRating) from D9 order by ContentRating DESC")

#Assign the Row number as the numeric equivalent of the Content Rating

DistinctCR$CRCode <- as.integer(rownames(DistinctCR))

DistinctCR

#Add a Column Category Genres to D which is the numeric equivalent of the Genres

D9$CRCode <- match(D9$ContentRating,DistinctCR$ContentRating,nomatch=0)

#Predict best selling app in Linear model for 2019 of total worth.

# Use estimated value of $1.20/user as per Quora

D9$TotalWorth = (D9$Installs \* D9$Price) + 1.20 \* D9$Installs

#D9$TotalWorth = (D9$Installs \* D9$Price)

# Use ggplot to plot Fawn Count vs. Adult Count

#g <- ggplot(df, aes(x = AdultCt, y = FawnCt)) + geom\_point() + stat\_smooth(method = "lm", col="red")

#g

# Model 1: Predict Total Worth for 2019 Using Category

m1 <- lm(formula = D9$TotalWorth ~ CategoryCode, data=D9)

# Display the regression analysis summary for Model 1:

summary(m1)

# Plot the regression line

plot(D9$CategoryCode,D9$TotalWorth)

abline(m1)

# Model 2: Predict Total Worth for 2019 Using Rating

m2 <- lm(formula = D9$TotalWorth ~ Rating, data=D9)

# Display the regression analysis summary for Model 2:

summary(m2)

# Plot the regression line

plot(D9$Rating,D9$TotalWorth)

abline(m2)

# Model 3: Predict Total Worth for 2019 Using Genres

m3 <- lm(formula = D9$TotalWorth ~ GenresCode, data=D9)

# Display the regression analysis summary for Model 3:

summary(m3)

# Plot the regression line

plot(D9$GenresCode,D9$TotalWorth)

abline(m3)

#

# Model 4: Predict Total Worth for 2019 Using Reviews

m4 <- lm(formula = D9$TotalWorth ~ Reviews, data=D9)

# Display the regression analysis summary for Model 4:

summary(m4)

# Plot the regression line

plot(D9$Reviews,D9$TotalWorth)

abline(m4)

# Model 5: Predict Total Worth for 2019 Using Size

m5 <- lm(formula = D9$TotalWorth ~ Size, data=D9)

# Display the regression analysis summary for Model 5:

summary(m5)

# Plot the regression line

plot(D9$Size,D9$TotalWorth)

abline(m5)

#

# Model 6: Predict Total Worth for 2019 Using ContentRating

m6 <- lm(formula = D9$TotalWorth ~ CRCode, data=D9)

# Display the regression analysis summary for Model 6:

summary(m6)

# Plot the regression line

plot(D9$CRCode,D9$TotalWorth)

abline(m6)

#

# Model 7: Predict Total Worth for 2019 Using Type

m7 <- lm(formula = D9$TotalWorth ~ Type, data=D9)

# Display the regression analysis summary for Model 7:

summary(m7)

# Plot the regression line

plot(D9$Type,D9$TotalWorth)

abline(m7)

#

# Model 8: Predict Total Worth for 2019 Using Genres, Category, Rating, Size

m8 <- lm(formula = D9$TotalWorth ~ GenresCode + CategoryCode + Rating + Size + CRCode, data=D9)

# Display the regression analysis summary for Model 8:

summary(m8)

#

# Model 9: Predict Installs from Reviews

m9 <- lm(formula = Installs ~ Reviews, data=D9)

# Display the regression analysis summary for Model 9:

summary(m9)

#

# Model 10: Predict Installs from Rating

m10 <- lm(formula = Installs ~ Rating, data=D9)

# Display the regression analysis summary for Model 10:

summary(m10)

#

# Model 11: Predict NetWorth from Installs

m11 <- lm(formula = TotalWorth ~ Installs, data=D9)

# Display the regression analysis summary for Model 11:

summary(m11)

#

#@Linear regression module---------------------------------ENd---------------------- -