An Analysis of Google Play Apps

Introduction

https://www.kaggle.com/lava18/google-play-store-apps

This Kaggle Dataset is a compilation of all the Google Play apps at the time the data was scraped. Most likely, the data was scraped in late 2018. There are about 14,000 apps. There are 13 variables: Name, Category, Rating, Reviews, Size, Installs, Type, Price, Content Rating, Genres, Last Updated, Current Version, and Android Version. In our research, we are performing a hypothesis test on rating and building regression models with rating as the response variable.

Please note that we edited the dataset because there was an inconsistent line. Thus, we took that off. While transferring, all the letters of the variables became lowercase.

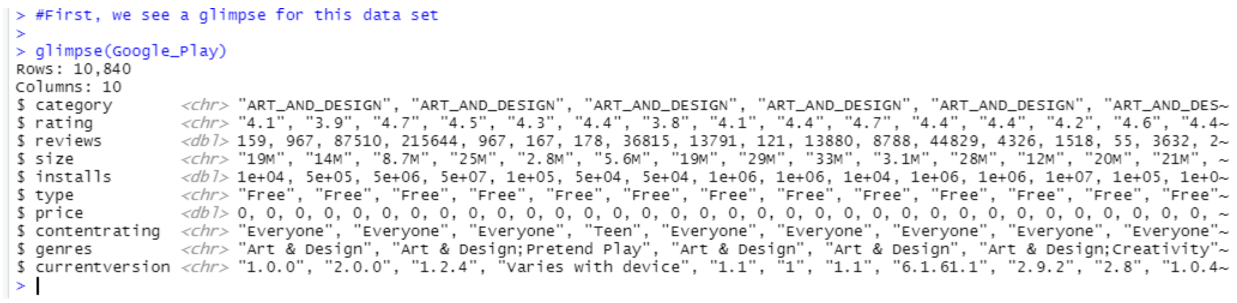
Questions of Interest

1. Is the mean population rating equal to 3, or is it not?
2. What is the 99% Confidence Interval for Mean Population Rating?
3. What is the effect of category and reviews on rating?
4. What is the effect of category and installs on rating?

Analysis

Exploratory Data Analysis

Here, we will conduct Exploratory Data Analysis.

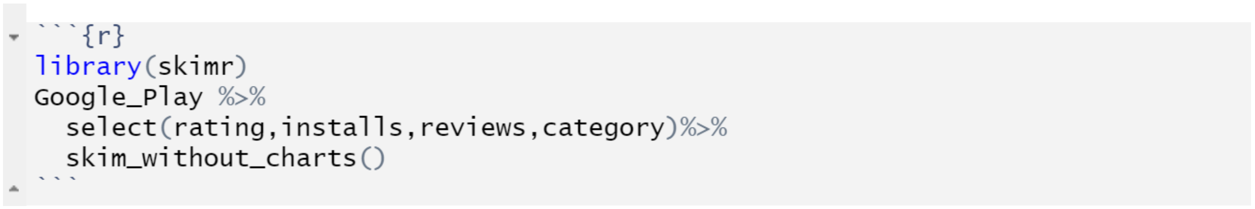


```
> #First, we see a glimpse for this data set
>
> glimpse(Google_Play)
Rows: 10,840
Columns: 10
$ category       <chr> "ART_AND_DESIGN", "ART_AND_DESIGN", "ART_AND_DESIGN", "ART_AND_DESIGN", "ART_AND_DESIGN", "ART_AND_DES~
$ rating         <chr> "4.1", "3.9", "4.7", "4.5", "4.3", "4.4", "3.8", "4.1", "4.4", "4.7", "4.4", "4.4", "4.2", "4.6", "4.4~
$ reviews        <dbl> 159, 967, 87510, 215644, 967, 167, 178, 36815, 13791, 121, 13880, 8788, 44829, 4326, 1518, 55, 3632, 2~
$ size           <chr> "19M", "14M", "8.7M", "25M", "2.8M", "5.6M", "19M", "29M", "33M", "3.1M", "28M", "12M", "20M", "21M", ~
$ installs       <dbl> 1e+04, 5e+05, 5e+06, 5e+07, 1e+05, 5e+04, 5e+04, 1e+06, 1e+06, 1e+04, 1e+06, 1e+06, 1e+07, 1e+05, 1e+0~
$ type           <chr> "Free", "Free", "Free", "Free", "Free", "Free", "Free", "Free", "Free", "Free", "Free", "Free", "Free"~
$ price          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ contentrating  <chr> "Everyone", "Everyone", "Everyone", "Teen", "Everyone", "Everyone", "Everyone", "Everyone", "Everyone"~
$ genres         <chr> "Art & Design", "Art & Design;Pretend Play", "Art & Design", "Art & Design", "Art & Design;Creativity"~
$ currentversion <chr> "1.0.0", "2.0.0", "1.2.4", "Varies with device", "1.1", "1", "1.1", "6.1.61.1", "2.9.2", "2.8", "1.0.4~
> |
```

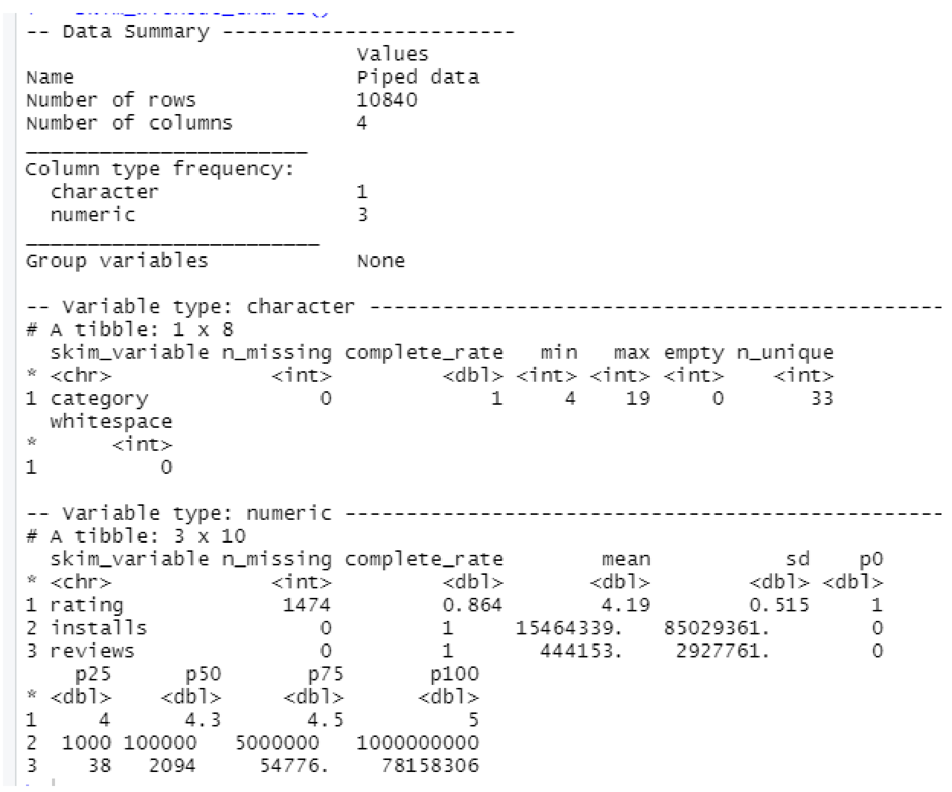
We notice that many of the measurement variables are listed as chr. We must convert them to the fct form. We will convert rating, installs, and reviews, from character to numeric.

```
Google_Play$rating <- as.numeric(Google_Play$rating)

## Warning: NAs introduced by coercion

Google_Play$reviews<-as.numeric(Google_Play$reviews)
Google_Play$installs<-as.numeric(Google_Play$installs)

class(Google_Play$rating)

## [1] "numeric"

class(Google_Play$reviews)

## [1] "numeric"

class(Google_Play$installs)

## [1] "numeric"
```

Next, we are going to compute common summary statistics using the skim() function.



~~~
```{r}
library(skimr)
Google_Play %>%
  select(rating,installs,reviews,category)%>%
  skim_without_charts()
```
~~~



```
-- Data Summary ------------------------
                           Values
Name                       Piped data
Number of rows             10840
Number of columns          4
_______________________
Column type frequency:
  character                1
  numeric                  3
________________________
Group variables            None

-- Variable type: character ------------------------------------------------
# A tibble: 1 x 8
  skim_variable n_missing complete_rate   min   max empty n_unique
* <chr>             <int>         <dbl> <int> <int> <int>    <int>
1 category              0             1     4    19     0       33
  whitespace
*      <int>
1          0

-- Variable type: numeric --------------------------------------------------
# A tibble: 3 x 10
  skim_variable n_missing complete_rate        mean           sd    p0
* <chr>             <int>         <dbl>       <dbl>        <dbl> <dbl>
1 rating             1474         0.864        4.19        0.515     1
2 installs              0         1     15464339.    85029361.       0
3 reviews               0         1       444153.     2927761.       0
    p25       p50       p75       p100
* <dbl>     <dbl>     <dbl>      <dbl>
1     4       4.3       4.5          5
2  1000 100000    5000000   1000000000
3    38    2094     54776.    78158306
```

Is the mean population equal to four?

We want to test the hypothesis that the mean rating is 4.

Null Hypothesis: The mean rating is 4.

$\mu=4$

Alternative Hypothesis: The mean rating is not equal to 4.

$\mu\neq4$

To do this, we are going to take a sample from this data and apply the technique of bootstrapping.

Here, we take a random sample with 100 observations. We set seed to be equal to one. We use the filter function to filter out all values that are NaN.

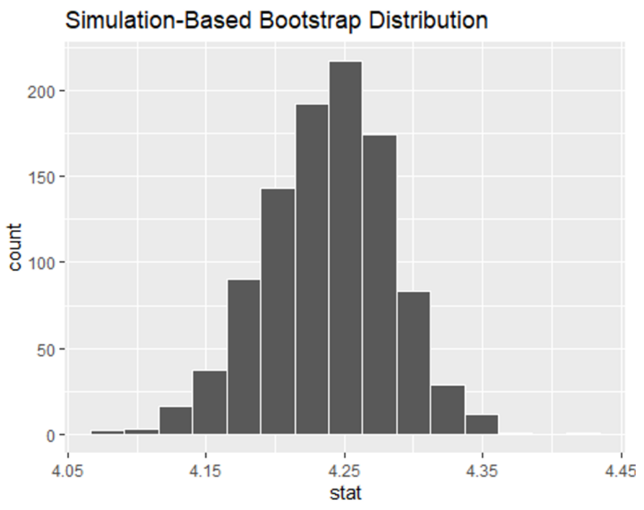
```
library(infer)
set.seed(1)
AppsSample <- Google_Play %>%
  filter(!(rating=="NaN"))%>%
  select(rating)%>%
rep_sample_n(size=100)
AppsSample

## # A tibble: 100 x 2
## # Groups:   replicate [1]
##    replicate rating
##        <int>  <dbl>
##  1         1    4
##  2         1    4
##  3         1    4.1
##  4         1    4.2
##  5         1    3
##  6         1    4.6
##  7         1    4.4
```

Now, we use this sample as our means of bootstrapping.

We create a bootstrap distribution with the three steps: specify, generate, and calculate. We have 1000 reps, and we are calculating the mean rating.

```
bootstrap_distribution_Rating <- AppsSample %>%
  specify(response=rating)%>%
  generate(reps=1000,type="bootstrap")%>%
  calculate(stat="mean")
bootstrap_distribution_Rating

## Response: rating (numeric)
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1  4.25
##  2         2  4.3
##  3         3  4.28
##  4         4  4.24
##  5         5  4.28
##  6         6  4.21
##  7         7  4.28
##  8         8  4.24
##  9         9  4.26
## 10        10  4.19
## # ... with 990 more rows
```

Now we visualize the results



We notice that the distribution is symmetric. We will use the Shapiro Test to further clarify this.

```
shapiro.test(bootstrap_distribution_Rating$stat)

##
##  Shapiro-Wilk normality test
##
## data:  bootstrap_distribution_Rating$stat
## W = 0.99747, p-value = 0.1227
```

The p value is 0.1227, so the distribution is normal.

In our case, both CI Methods are acceptable. However, I have chosen to calculate the Percentile CI.

```
percentile_ci <- bootstrap_distribution_Rating %>%
get_confidence_interval(level = 0.95, type = "percentile")
percentile_ci

## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     4.14     4.32
```

In this CI, we conclude that we are 95% confident that the true mean Rating is between 4.14 and 4.32.

We also use a Standard Error CI

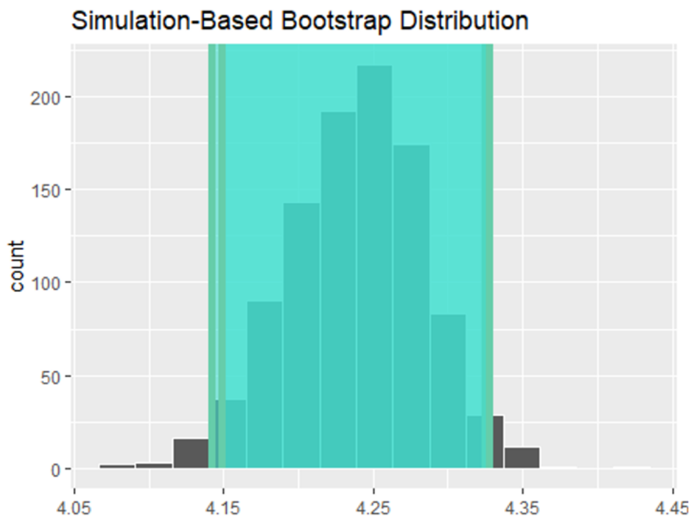
```
x_bar <- AppsSample %>%
  summarize(Avg=mean(rating))%>%
  select(Avg)
x_bar

## # A tibble: 1 x 1
##     Avg
##   <dbl>
## 1  4.24

standard_error_ci <- bootstrap_distribution_Rating %>%
  get_confidence_interval(level=0.95,type="se",point_estimate=x_bar)
standard_error_ci

## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     4.15     4.33
```

We derive x_bar, then we use it in the get_confidence_interval function.

In this CI, we conclude that we are 95% confident that the true mean Rating is between 4.15 and 4.33.

Now, we plot both CIs onto the graph.

```
visualize(bootstrap_distribution_Rating) +
  shade_confidence_interval(endpoints=percentile_ci)+
  shade_confidence_interval(endpoints=standard_error_ci)
```



We see that 4 is not included in any of these confidence intervals. Therefore, we reject the null at the alpha=0.05 level. We can conclude that the mean rating of Google Play apps is not equal to four. We can also conclude that the mean rating is rather greater than four.

What is the 99% Confidence Interval for Mean Number of Reviews?

At the 99% Level, we want to see the range of values that contain the true mean number of reviews. Thus, we construct another bootstrap confidence interval.

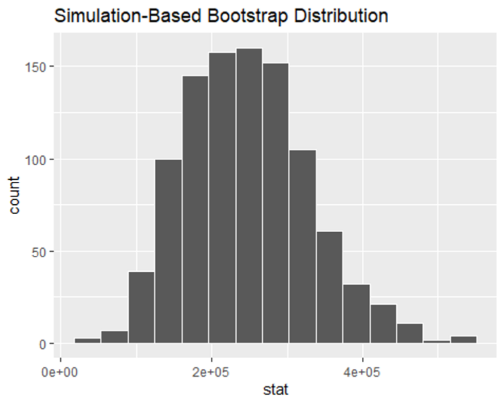
Setting seed=2, we take another sample.

```
set.seed(2)
AppsSample2 <- Google_Play %>%
  select(reviews)%>%
rep_sample_n(size=100)
AppsSample2

## # A tibble: 100 x 2
## # Groups:   replicate [1]
##    replicate reviews
##        <int>   <dbl>
##  1         1  130549
##  2         1    5997
##  3         1      55
##  4         1     315
##  5         1 3016297
##  6         1     478
##  7         1       0
##  8         1      26
##  9         1      48
## 10         1      41
## # ... with 90 more rows
```

```
bootstrap_distribution_Reviews <- AppsSample2 %>%
  specify(response=reviews)%>%
  generate(reps=1000,type="bootstrap")%>%
  calculate(stat="mean")
bootstrap_distribution_Reviews
```

```
## Response: reviews (numeric)
## # A tibble: 1,000 x 2
##    replicate    stat
##        <int>   <dbl>
##  1         1 345800.
##  2         2 155790.
##  3         3 124465.
##  4         4 157902.
##  5         5 174272.
##  6         6 291752.
##  7         7 239361.
##  8         8  67012.
##  9         9 133285.
## 10        10 231194.
## # ... with 990 more rows
```

We take another bootstrap distribution.



We visualize it and see that it is skewed. So we compute the percentile CI.

We see that the lower CI is 67,712 and the Upper CI is 490,971. Thus, we are 99% confident that the true mean number of reviews is between 67,712 and 490,971.

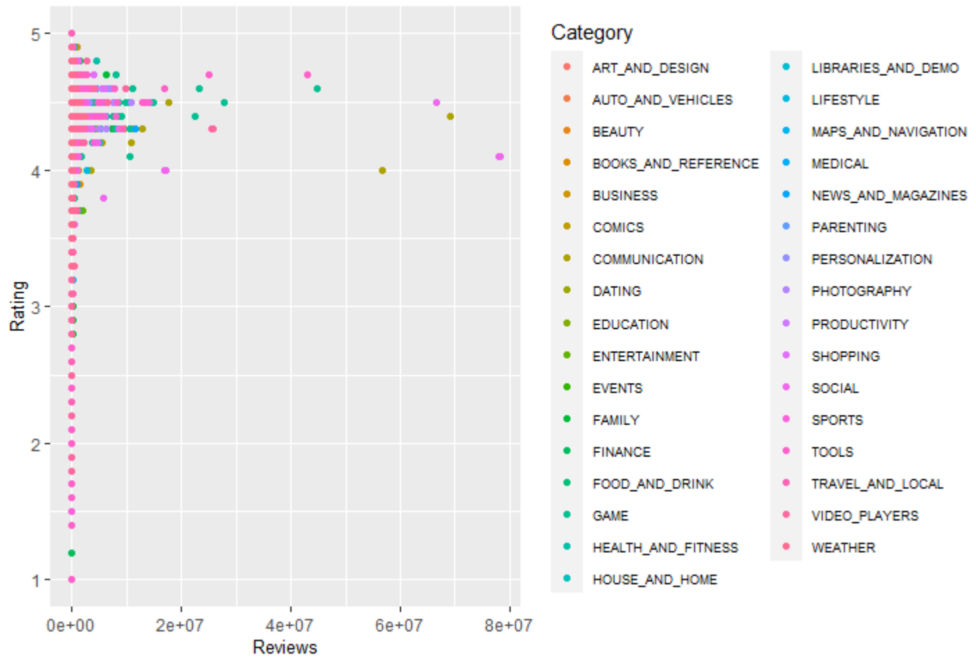
Regression: 1 Categorical, 1 Numeric on 1 Numeric

In this data set we will be using the predictors category, rating, reviews, and installs in our data set. Therefore, we use a subset to isolate these predictors.

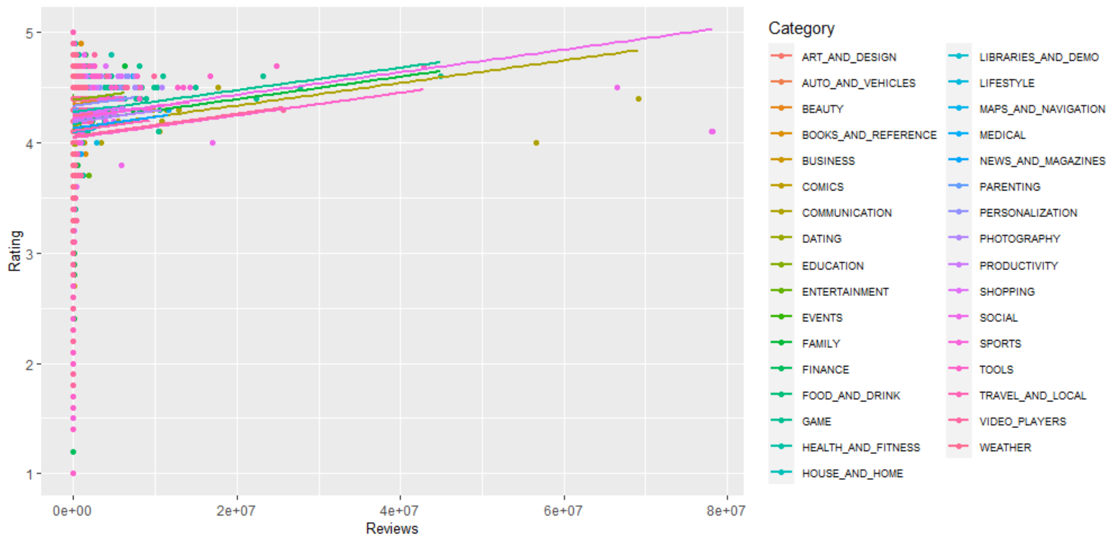
a.) **What is the effect of Category and Reviews on Rating?**

```r
library(ggplot2)
library(moderndive)

# Category, Reviews on Rating
ggplot(Google_Play,
       aes(x = reviews, y = rating, color = category)) +
  geom_point() +
  labs(x = "Reviews", y = "Rating", color = "Category") +
  theme(axis.title = element_text(size=10), legend.text = element_text(size=7)) +
  geom_parallel_slopes(se = FALSE)

google_model_parallel_slopes <- lm(rating ~ reviews + category, data = Google_Play)
# Get regression table:
get_regression_table(google_model_parallel_slopes)
summary(google_model_parallel_slopes)
```

The scatter plot below represents the relationship between the category, reviews, and rating of our data set. The categories are listed on the right side, and they are color coordinated for the scatter plot.

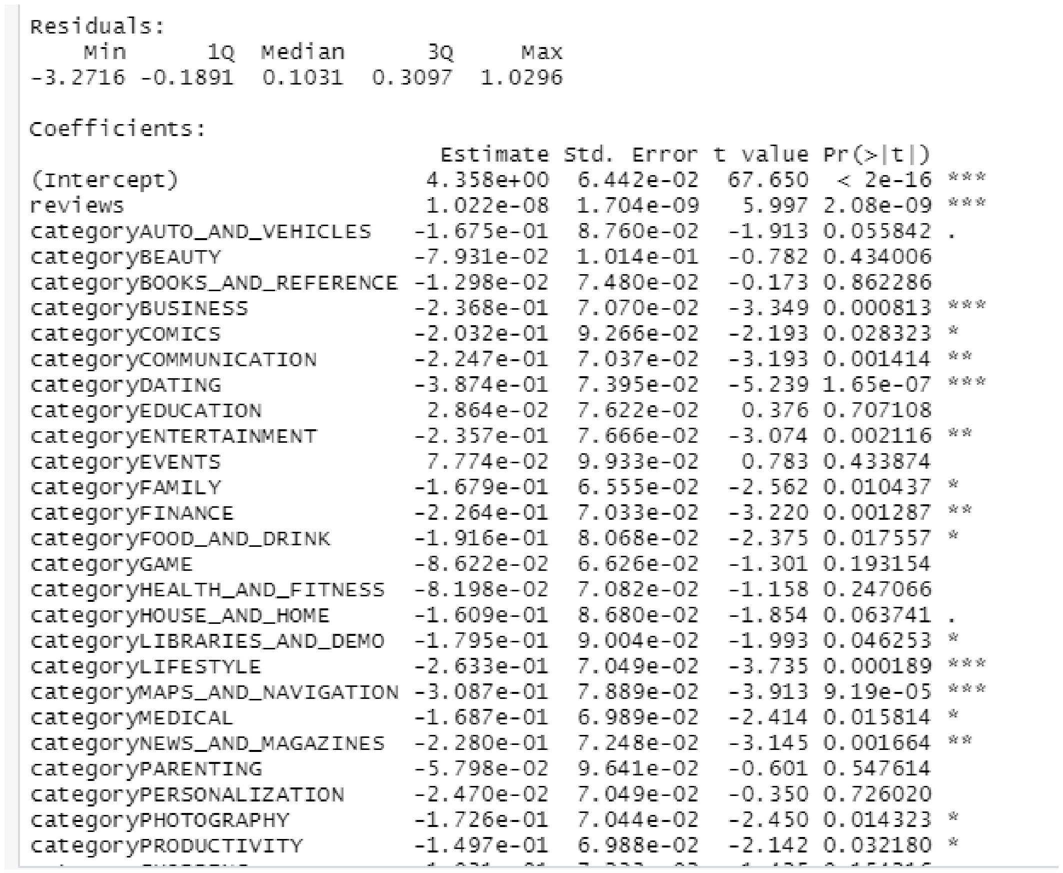


The lines on the scatterplots below are used to demonstrate in which direction everything is heading.

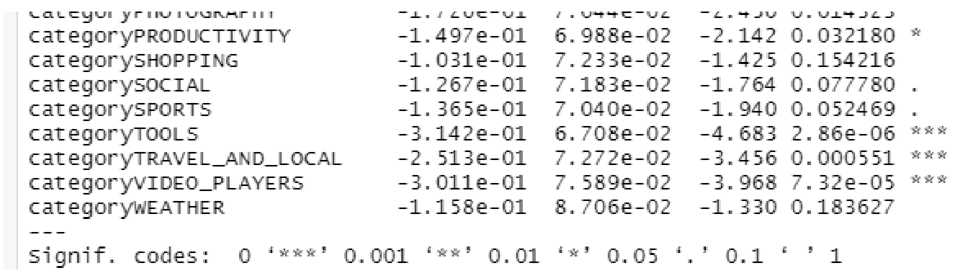


From the scatter plot above, we notice that the categories of the applications and their ratings are in sync. This means that people who rate applications high are more likely to also leave a review. Reviews seem to have a positive slope after ratings above 4.

Regression Model:



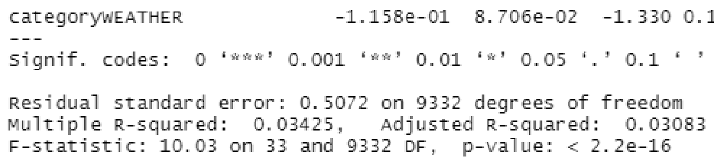
```
Residuals:
    Min      1Q  Median      3Q     Max 
-3.2716 -0.1891  0.1031  0.3097  1.0296 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                  4.358e+00  6.442e-02  67.650  < 2e-16 ***
reviews                      1.022e-08  1.704e-09   5.997 2.08e-09 ***
categoryAUTO_AND_VEHICLES   -1.675e-01  8.760e-02  -1.913 0.055842 .  
categoryBEAUTY              -7.931e-02  1.014e-01  -0.782 0.434006    
categoryBOOKS_AND_REFERENCE -1.298e-02  7.480e-02  -0.173 0.862286    
categoryBUSINESS            -2.368e-01  7.070e-02  -3.349 0.000813 ***
categoryCOMICS              -2.032e-01  9.266e-02  -2.193 0.028323 *  
categoryCOMMUNICATION       -2.247e-01  7.037e-02  -3.193 0.001414 ** 
categoryDATING              -3.874e-01  7.395e-02  -5.239 1.65e-07 ***
categoryEDUCATION            2.864e-02  7.622e-02   0.376 0.707108    
categoryENTERTAINMENT       -2.357e-01  7.666e-02  -3.074 0.002116 ** 
categoryEVENTS               7.774e-02  9.933e-02   0.783 0.433874    
categoryFAMILY              -1.679e-01  6.555e-02  -2.562 0.010437 *  
categoryFINANCE             -2.264e-01  7.033e-02  -3.220 0.001287 ** 
categoryFOOD_AND_DRINK      -1.916e-01  8.068e-02  -2.375 0.017557 *  
categoryGAME                -8.622e-02  6.626e-02  -1.301 0.193154    
categoryHEALTH_AND_FITNESS  -8.198e-02  7.082e-02  -1.158 0.247066    
categoryHOUSE_AND_HOME      -1.609e-01  8.680e-02  -1.854 0.063741 .  
categoryLIBRARIES_AND_DEMO  -1.795e-01  9.004e-02  -1.993 0.046253 *  
categoryLIFESTYLE           -2.633e-01  7.049e-02  -3.735 0.000189 ***
categoryMAPS_AND_NAVIGATION -3.087e-01  7.889e-02  -3.913 9.19e-05 ***
categoryMEDICAL             -1.687e-01  6.989e-02  -2.414 0.015814 *  
categoryNEWS_AND_MAGAZINES  -2.280e-01  7.248e-02  -3.145 0.001664 ** 
categoryPARENTING           -5.798e-02  9.641e-02  -0.601 0.547614    
categoryPERSONALIZATION     -2.470e-02  7.049e-02  -0.350 0.726020    
categoryPHOTOGRAPHY         -1.726e-01  7.044e-02  -2.450 0.014323 *  
categoryPRODUCTIVITY        -1.497e-01  6.988e-02  -2.142 0.032180 *  
```



```
categoryPHOTOGRAPHY          -1.726e-01  7.044e-02  -2.450 0.014325
categoryPRODUCTIVITY         -1.497e-01  6.988e-02  -2.142 0.032180 *
categorySHOPPING             -1.031e-01  7.233e-02  -1.425 0.154216
categorySOCIAL               -1.267e-01  7.183e-02  -1.764 0.077780 .
categorySPORTS               -1.365e-01  7.040e-02  -1.940 0.052469 .
categoryTOOLS                -3.142e-01  6.708e-02  -4.683 2.86e-06 ***
categoryTRAVEL_AND_LOCAL     -2.513e-01  7.272e-02  -3.456 0.000551 ***
categoryVIDEO_PLAYERS        -3.011e-01  7.589e-02  -3.968 7.32e-05 ***
categoryWEATHER              -1.158e-01  8.706e-02  -1.330 0.183627
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

There are a few predictors in the model above. From the short data set, we see that business, comics, communication, dating, entertainment, family, finance, food and drink, libraries and demo, lifestyle, maps and navigation, medical, news and magazine, photography, productivity, tools, travel and local, and video players apps are predictors of this model because the p-value is lower than 0.05.

R-Squared:



```
categoryWEATHER              -1.158e-01  8.706e-02  -1.330 0.1
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 0.5072 on 9332 degrees of freedom
Multiple R-squared:  0.03425,	Adjusted R-squared:  0.03083
F-statistic: 10.03 on 33 and 9332 DF,  p-value: < 2.2e-16
```

Here is the r squared, indicating a weak correlation of 0.03425 and adjusted at 0.03083. Therefore, the model only explains 3.083% of the variability.

Interpretation:

After further examining the regression model we notice that for every increase in ratings the reviews also increase by 1.022e-08. We also noticed this in the scatterplot. There is an upward slope of positive 0.00000001022. The intercept is the mean rating for apps in the arts and design category.

The Best Fitted Model:

Y(hat) = b0 + b1(x) + b2(x) + b3(x) +………b33(x).

= 4.358 + (-1.675e-01)x + (-7.931e-02)x + (-)x + (-1.298e-02)x + (-2.368e-01)x + (-2.032e-01)x + (-2.247e-01)x + (-3.874e-01)x + (2.864e-02)x + ….. (the 24 more rows)

The intercept (b0) is the mean rating of apps whose category is Art and Design. The mean rating for this category is 4.358.

Category: Auto and Vehicles (b1 mean rating)

4.358 + (1.675e-01) = 4.1905

Category: Beauty (b2 mean rating)

4.358 + (-7.931e-02) = 4.27869

Category: Books and Reference (b3 mean rating)

4.358 + (-1.298e-02) = 4.34502

Category: Business (b4 mean rating)

4.358 + (-2.368e-01) = 4.1212

Category: Comics (b5 mean rating)

4.358 + (-2.032e-01) = 4.1548

Category: Communication (b6 mean rating)

4.358 + (-2.247e-01) = 4.133

Category: Dating (b7 mean rating)

4.358 + (-3.874e-01) = 3.9706

Category: Education (b8 mean rating)

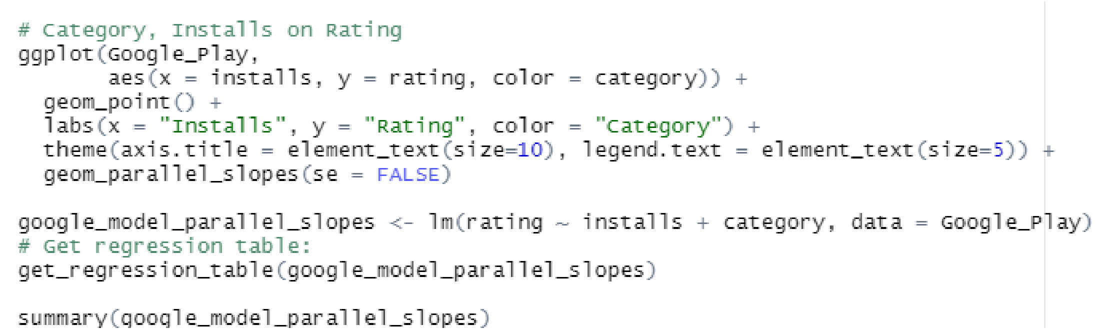
4.358 + (2.864e-02) = 4.38664

(and so on).

Conclusion:

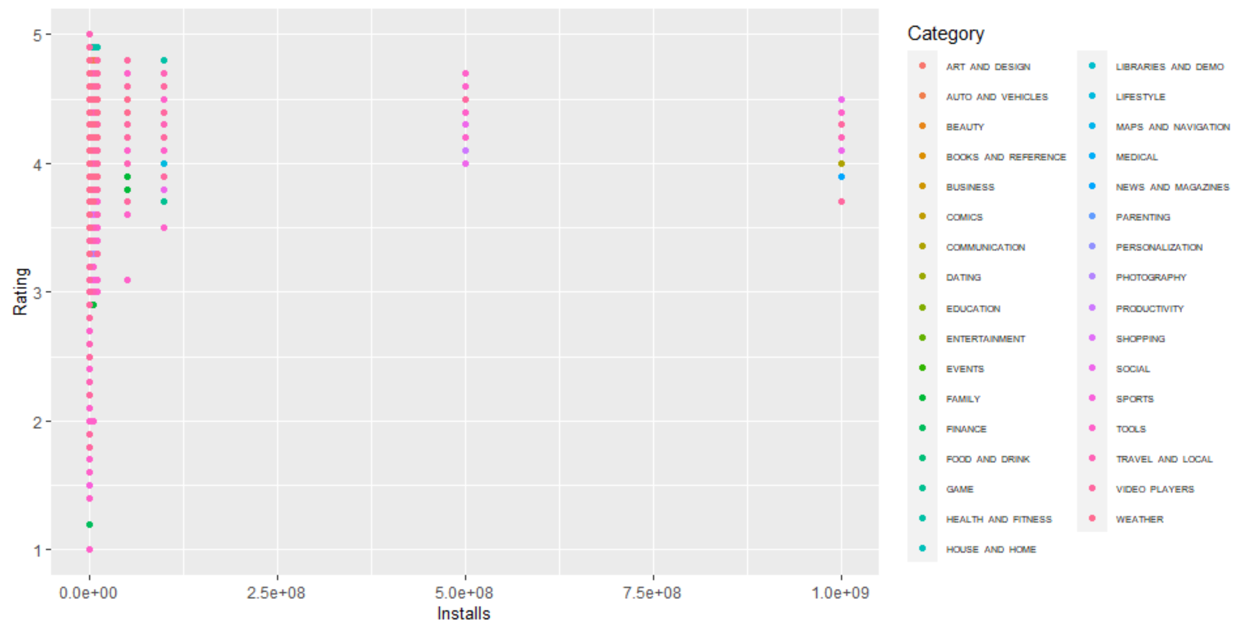
The average mean rating when the category is art and design on reviews is 4.358. By examining the scatter plots we see that it is accurate. The scatter plot shows many category colors in the y direction up to 5 in the ratings. This demonstrates that as the ratings get higher the reviews are sometimes there and other times not really. We can assume that people are more likely to rate high from 4-5 than to leave reviews on whether they enjoy the application or not, this may be because leaving a review is a bit more time consuming. We see that there are lower ratings but few to no reviews.

b.) What is the effect of Category and Installs on Rating?

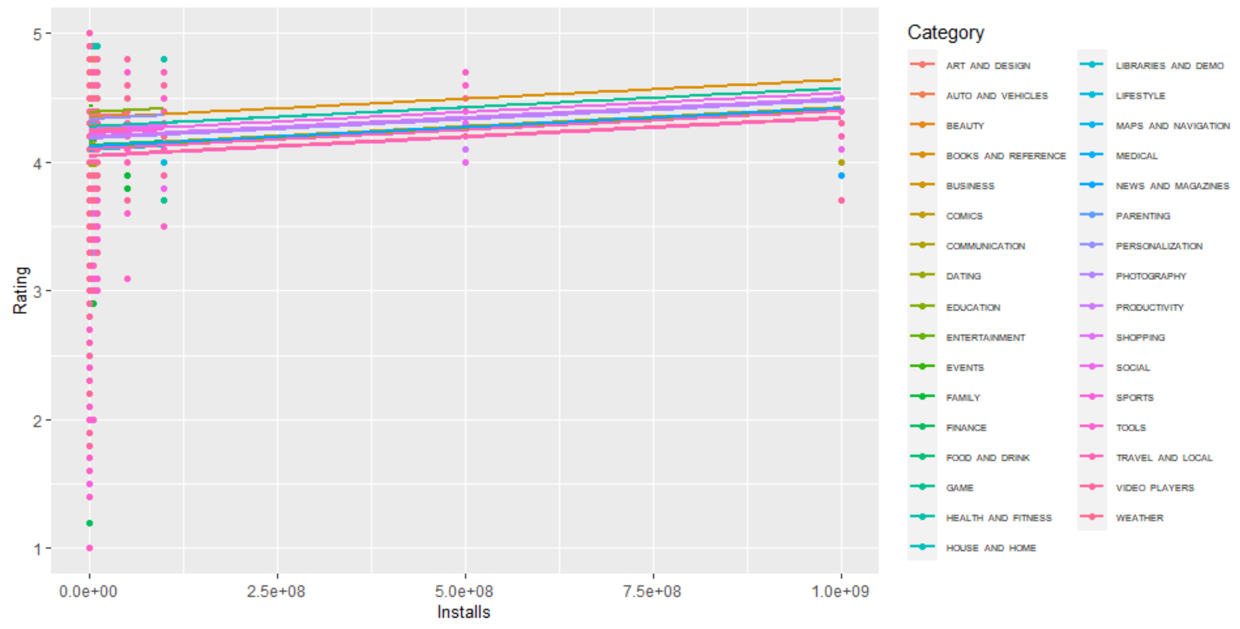


```
# Category, Installs on Rating
ggplot(Google_Play,
       aes(x = installs, y = rating, color = category)) +
  geom_point() +
  labs(x = "Installs", y = "Rating", color = "Category") +
  theme(axis.title = element_text(size=10), legend.text = element_text(size=5)) +
  geom_parallel_slopes(se = FALSE)

google_model_parallel_slopes <- lm(rating ~ installs + category, data = Google_Play)
# Get regression table:
get_regression_table(google_model_parallel_slopes)

summary(google_model_parallel_slopes)
```

The scatter plot below represents the relationship between the category, installs, and rating of our data set. The categories are listed on the right side, and they are color coordinated for the scatter plot.

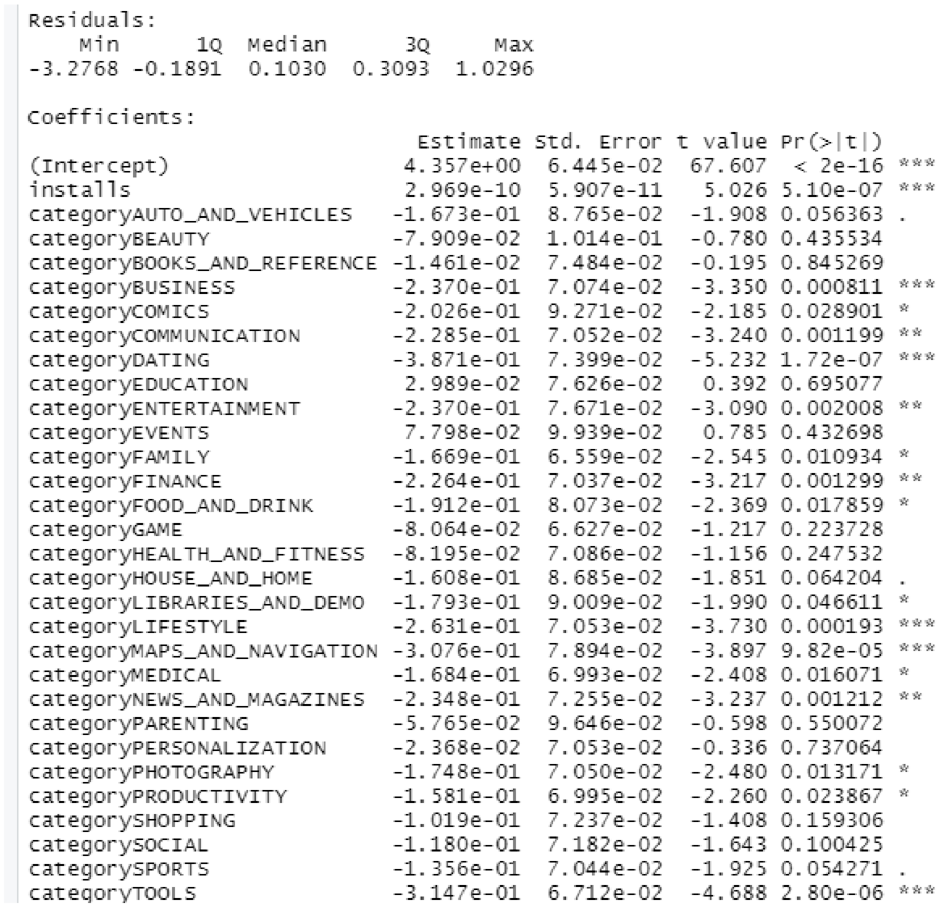


The lines on the scatterplots below are used to demonstrate in which direction everything is heading.

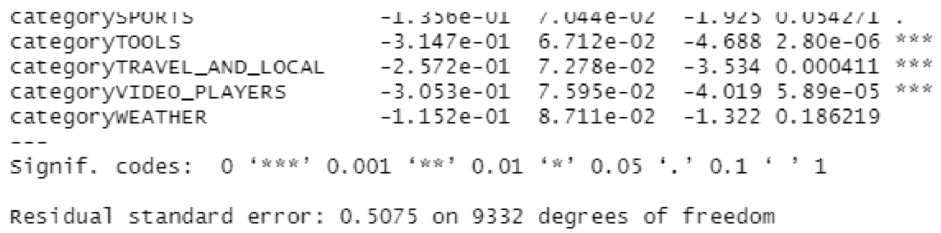


From the scatter plot above we notice that there are a ton of installs heading in a positive slope direction due to the category of the application. Although this is true, the ratings stay at a constant high of about 4. This we will see later that the number we average in our data findings from our regression model.

Regression Model:



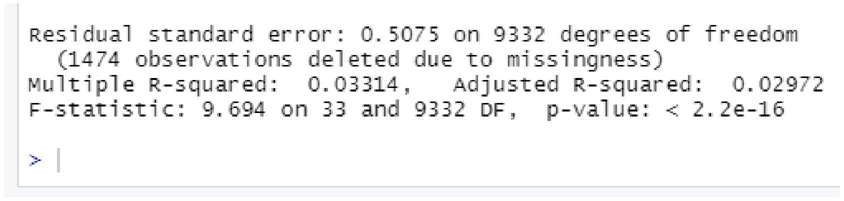
```
Residuals:
    Min      1Q  Median      3Q     Max
-3.2768 -0.1891  0.1030  0.3093  1.0296

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  4.357e+00  6.445e-02  67.607  < 2e-16 ***
installs                     2.969e-10  5.907e-11   5.026 5.10e-07 ***
categoryAUTO_AND_VEHICLES   -1.673e-01  8.765e-02  -1.908 0.056363 .
categoryBEAUTY              -7.909e-02  1.014e-01  -0.780 0.435534
categoryBOOKS_AND_REFERENCE -1.461e-02  7.484e-02  -0.195 0.845269
categoryBUSINESS            -2.370e-01  7.074e-02  -3.350 0.000811 ***
categoryCOMICS              -2.026e-01  9.271e-02  -2.185 0.028901 *
categoryCOMMUNICATION       -2.285e-01  7.052e-02  -3.240 0.001199 **
categoryDATING              -3.871e-01  7.399e-02  -5.232 1.72e-07 ***
categoryEDUCATION            2.989e-02  7.626e-02   0.392 0.695077
categoryENTERTAINMENT       -2.370e-01  7.671e-02  -3.090 0.002008 **
categoryEVENTS               7.798e-02  9.939e-02   0.785 0.432698
categoryFAMILY              -1.669e-01  6.559e-02  -2.545 0.010934 *
categoryFINANCE             -2.264e-01  7.037e-02  -3.217 0.001299 **
categoryFOOD_AND_DRINK      -1.912e-01  8.073e-02  -2.369 0.017859 *
categoryGAME                -8.064e-02  6.627e-02  -1.217 0.223728
categoryHEALTH_AND_FITNESS  -8.195e-02  7.086e-02  -1.156 0.247532
categoryHOUSE_AND_HOME      -1.608e-01  8.685e-02  -1.851 0.064204 .
categoryLIBRARIES_AND_DEMO  -1.793e-01  9.009e-02  -1.990 0.046611 *
categoryLIFESTYLE           -2.631e-01  7.053e-02  -3.730 0.000193 ***
categoryMAPS_AND_NAVIGATION -3.076e-01  7.894e-02  -3.897 9.82e-05 ***
categoryMEDICAL             -1.684e-01  6.993e-02  -2.408 0.016071 *
categoryNEWS_AND_MAGAZINES  -2.348e-01  7.255e-02  -3.237 0.001212 **
categoryPARENTING           -5.765e-02  9.646e-02  -0.598 0.550072
categoryPERSONALIZATION     -2.368e-02  7.053e-02  -0.336 0.737064
categoryPHOTOGRAPHY         -1.748e-01  7.050e-02  -2.480 0.013171 *
categoryPRODUCTIVITY        -1.581e-01  6.995e-02  -2.260 0.023867 *
categorySHOPPING            -1.019e-01  7.237e-02  -1.408 0.159306
categorySOCIAL              -1.180e-01  7.182e-02  -1.643 0.100425
categorySPORTS              -1.356e-01  7.044e-02  -1.925 0.054271 .
categoryTOOLS               -3.147e-01  6.712e-02  -4.688 2.80e-06 ***
```



```
categorySPORTS              -1.356e-01  7.044e-02  -1.925 0.054271 .
categoryTOOLS               -3.147e-01  6.712e-02  -4.688 2.80e-06 ***
categoryTRAVEL_AND_LOCAL    -2.572e-01  7.278e-02  -3.534 0.000411 ***
categoryVIDEO_PLAYERS       -3.053e-01  7.595e-02  -4.019 5.89e-05 ***
categoryWEATHER             -1.152e-01  8.711e-02  -1.322 0.186219
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5075 on 9332 degrees of freedom
```

There are a lot of predictors in the model above. From the condensed data set we see that business, comics, communication, dating, entertainment, family, finance, food and drink, libraries and demo, lifestyle, maps and navigation, medical, news and magazine, photography, productivity, tools, travel and local, and video players apps are predictors of this model. These are all at the 5% significance level, meaning the p-value is below 0.05.

R-Squared:



```
Residual standard error: 0.5075 on 9332 degrees of freedom
  (1474 observations deleted due to missingness)
Multiple R-squared:  0.03314,	Adjusted R-squared:  0.02972 
F-statistic: 9.694 on 33 and 9332 DF,  p-value: < 2.2e-16

> |
```

Here is the regression error metric, or r squared, indicating a weak correlation of 0.03314 and adjusted at 0.02972. Therefore, the model explains only 2.972% of the variability. Comparing it to the variability for category and reviews on rating, this is not a better fit for the model.

Interpretation:

After examining the regression model, we see that as the ratings increase, the installs increase by 2.969e-10. This can also show that the slope is increasing in the scatter-plot lines by 0.0000000002969. The mean of the ratings when the category is art and design is 4.357.

The Best Fitted Model:

Y(hat) = b0 + b1(x) + b2(x) + b3(x) +………b33(x).

= 4.357 + (-1.673e-01)x + (-7.909e-02)x + (-1.461e-02)x + (-2.370e-01)x + (-2.026e-01)x + (-2.285e-01)x + (-3.871e-01)x + (-2.989e-02)x + ….. (the 25 more rows)

The intercept (b0) is the mean rating of apps, whose category of these apps is Art and Design. The mean of the ratings on this category is 4.357.

Category: Auto and Vehicles (b1 mean rating)

4.357 + (-1.673e-01) = 4.1897

Category: Beauty (b2 mean rating)

4.357 + (-7.909e-02) = 4.27791

Category: Books and Reference (b3 mean rating)

4.357 + (-1.461e-02) = 4.34239

Category: Business (b4 mean rating)

4.357 + (-2.370e-01 = 4.120

Category: Comics (b5 mean rating)

4.357 + (-2.026e-01) = 4.1544

Category: Communication (b6 mean rating)

4.357 + (-2.285e-01) = 4.1285

Category: Dating (b7 mean rating)

4.357 + (-3.871e-01) = 3.9699

Category: Education (b8 mean rating)

4.357 + (-2.989e-02) = 4.32711

(more not shown).

Conclusion:

First, we see in our hypothesis test that the population mean rating is not equal to four. We notice in our confidence interval that since four is not in the confidence interval, that we reject our null hypothesis. The actual mean rating of 4.19 is in fact within the confidence interval. Second, our 99% confidence interval for the mean number of reviews is very wide. The population mean of 444,153 is in fact in the interval.

After further examining our data, we see that the average mean for ratings due to category art and design for installs is 4.358. This means that there are ratings about 4.358 when comparing both installs and categories. The increase we see on the installs due to ratings is 2.969e-10 which is why in the scatterplots we notice a slight upward slope when we get the slope lines. This makes sense because as the ratings get higher the installs increase. This may mean that Google Play users might actually take into consideration the ratings to decide whether an application might be worth the download. I can attest to having read ratings and formulated whether or not to download an application based on the ratings.

In conclusion, through these regression models we see in the first model that as ratings increase, the reviews also increase by 1.022e-08. This means that reviews are slightly dependent on ratings. Similarly, in the second model we see that installs are slightly dependent on ratings. This means that as ratings increase, installs also increase by 2.969e-10. Due to the models and their variability, we see that the model for category and reviews on ratings had a higher variability than the model for category and installs on rating. This means category and reviews on rating is a better fit for the model. These models have helped us understand the bonds between categories, installs, reviews, and ratings on applications. It has also allowed us to construct our own thoughts into what other outside factors might affect these results. This is insightful for anyone trying to understand what a computer software engineer has to study when creating an application.

Appendix

Exploratory Data Analysis

Import Data Set

```
library(readr)
#We import a new data set using the read.csv function
Google_Play <- read.csv(("C:\\Users\\colle\\Downloads\\GooglePlay495.csv")

library(tidyverse)
#First, we glimpse through the data set
glimpse(Google_Play)

#We change Rating, Price, Reviews to numeric
Google_Play$rating <- as.numeric(Google_Play$rating)
Google_Play$reviews <- as.numeric (Google_Play$reviews)
Google_Play$installs <- as.numeric(Google_Play$installs)

#We Check the Class
class(Google_Play$rating)
class(Google_Play$installs)
class(Google_Play$reviews)

#We Skim without charts
library(skimr)
Google_Play %>%
  select(rating,installs,reviews,category)%>%
  skim_without_charts()
```

Bootstrap Hypothesis Test for MU=4

```
#We take a sample
library(infer)
set.seed(1)
AppsSample <- Google_Play %>%
  filter(!(rating=="NaN"))%>%
  select(rating)%>%
rep_sample_n(size=100)
AppsSample

#We take a bootstrap distribution
bootstrap_distribution_Rating <- AppsSample %>%
  specify(response=rating)%>%
  generate(reps=1000,type="bootstrap")%>%
  calculate(stat="mean")
bootstrap_distribution_Rating

visualize(bootstrap_distribution_Rating)

#We perform a Shapiro Test
shapiro.test(bootstrap_distribution_Rating$stat)

#We Derive a Percentile CI
percentile_ci <- bootstrap_distribution_Rating %>%
get_confidence_interval(level = 0.95, type = "percentile")
percentile_ci

#We Derive X Bar
x_bar <- AppsSample %>%
  summarize(Avg=mean(rating))%>%
  select(Avg)
x_bar

#We Derive the Standard Error CI
standard_error_ci <- bootstrap_distribution_Rating %>%
  get_confidence_interval(level=0.95,type="se",point_estimate=x_bar)
standard_error_ci

#We Visualize the bootstrap distribution with the 2 CIs
visualize(bootstrap_distribution_Rating) +
  shade_confidence_interval(endpoints=percentile_ci)+
  shade_confidence_interval(endpoints=standard_error_ci)
```

99% CI for Mean Number of Reviews

```
#We take another sample setting seed as 2.
set.seed(2)
AppsSample2 <- Google_Play %>%
  select(reviews)%>%
rep_sample_n(size=100)
AppsSample2
```

```
#We take a bootstrap distribution for reviews.
bootstrap_distribution_Reviews <- AppsSample2 %>%
  specify(response=reviews)%>%
  generate(reps=1000,type="bootstrap")%>%
  calculate(stat="mean")
bootstrap_distribution_Reviews

visualize(bootstrap_distribution_Reviews)

#We derive another percentile CI
percentile_ci2 <- bootstrap_distribution_Reviews %>%
get_confidence_interval(level = 0.99, type = "percentile")
percentile_ci2

#We add it onto our graph
visualize(bootstrap_distribution_Reviews)+
  shade_confidence_interval(endpoints=percentile_ci2)
```

Linear Regression

```
library(ggplot2)
library(moderndive)

ggplot(Google_Play,
       aes(x = reviews, y = rating, color = category)) +
  geom_point() +
  labs(x = "Reviews", y = "Rating", color = "Category") +
  theme(axis.title = element_text(size=10), legend.text = element_text(size=7)) +
  geom_parallel_slopes(se = FALSE)

google_model_parallel_slopes <- lm(rating ~ reviews + category, data = Google_Play)
# Get regression table:
get_regression_table(google_model_parallel_slopes)

summary(google_model_parallel_slopes)


# Category, Installs on Rating
ggplot(Google_Play,
       aes(x = installs, y = rating, color = category)) +
  geom_point() +
  labs(x = "Installs", y = "Rating", color = "Category") +
  theme(axis.title = element_text(size=10), legend.text = element_text(size=5)) +
  geom_parallel_slopes(se = FALSE)

google_model_parallel_slopes <- lm(rating ~ installs + category, data = Google_Play)
# Get regression table:
get_regression_table(google_model_parallel_slopes)

summary(google_model_parallel_slopes)
```