Research Topic: Google Play Store Application Rating Data

Research Questions:

What is the relationship between the app rating and the threshold of 3.5?

Is there a relationship between the App Rating and factors such as Rating Count, In-App Purchases, and In-App Advertisements?

Sampling Technique Used: Stratified Sampling

A random sample of 60% of the applications from the Google-PlayStore.xlsx dataset is selected for analysis. This sample is a representative of the entire dataset and will help in making conclusions about the population of apps on the Google Play Store.

```
set.seed(7)
stratified_sample <- data.frame() strata
<- unique(df[,c("Ad Supported")])
for (s in strata) { stratum_data <- df[df$`Ad Supported`
        %in% s, ] stratum_sample <- stratum_data
        %>%
        sample_frac(0.6)
        stratified_sample <- rbind(stratified_sample, stratum_sample)
}
```

Rating <dbl>	Rating Count <dbl>	Free <dbl>	Maximum Inst... <dbl>	Ad Supported <dbl>
4.3	26	1	2486	1
4.8	6	1	529	1
3.3	257	1	126143	1
4.8	26	1	270	1
4.3	22	1	2992	1
4.5	8	1	992	1
4.1	628	1	684612	1
4.5	249	1	24297	0
4.0	25	1	20519	1
4.5	67	1	2683	0

1-10 of 334,904 rows | 2-6 of 8 columns Previous 1 2 3 4 5 6 ... 100 Next

Explanatory Data Analysis:

To answer research questions for this project, we have selected following variables to work with -

App Rating: A continuous variable which ranges values from 1 to 5.

App Rating Count: A continuous variable which indicates the total number of reviews received by the mobile application.

In-App Ads: A binary variable, has value 1 if the app supports ads and 0 otherwise.

In-App Purchases: A binary variable, has value 1 if the app has in-app purchases and 0 otherwise.

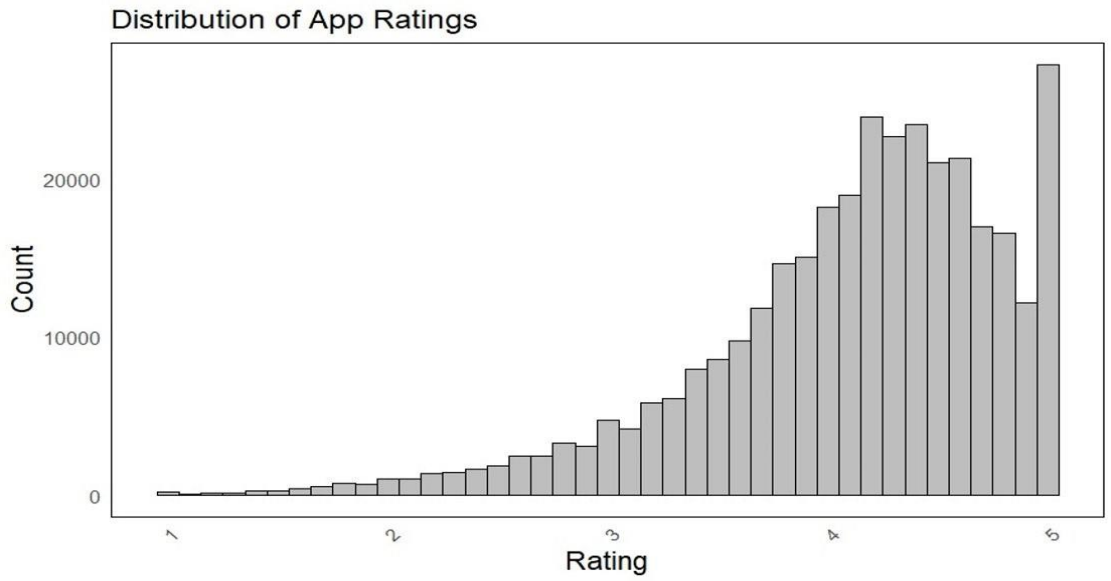
Descriptive statistics:

Following are the summary statistics for the variables

	Rating	Rating Count	In App Purchases	Ad Supported
Min	1	5	0	0
Max	5	138557570	1	1
Mean	2.2	5516.997	0.1257077	0.5747737
Standard Deviation	2.11	350677.3	0.3315201	0.494378

#Distribution of App Ratings

```
ggplot(stratified_sample, aes(x = Rating)) +
  geom_histogram(binwidth = 0.1, fill = "grey", color = "black") +
theme_minimal() +
  labs(title = "Distribution of App Ratings",
x = "Rating",       y = "Count") +
  theme(text = element_text(size = 12),        axis.title
= element_text(size = 14),        panel.grid.major =
element_blank(),        panel.grid.minor =
element_blank(),        panel.background =
element_rect(fill = "white"),        axis.text.x =
element_text(angle = 45, hjust = 1))
```



The code produces a histogram that illustrates how often each app rating occurs within the dataset, the ratings are primarily clustered towards the higher end, with the most frequent rating being 5.

#Average Rating by App category ggplot(mean_summary, aes(x =

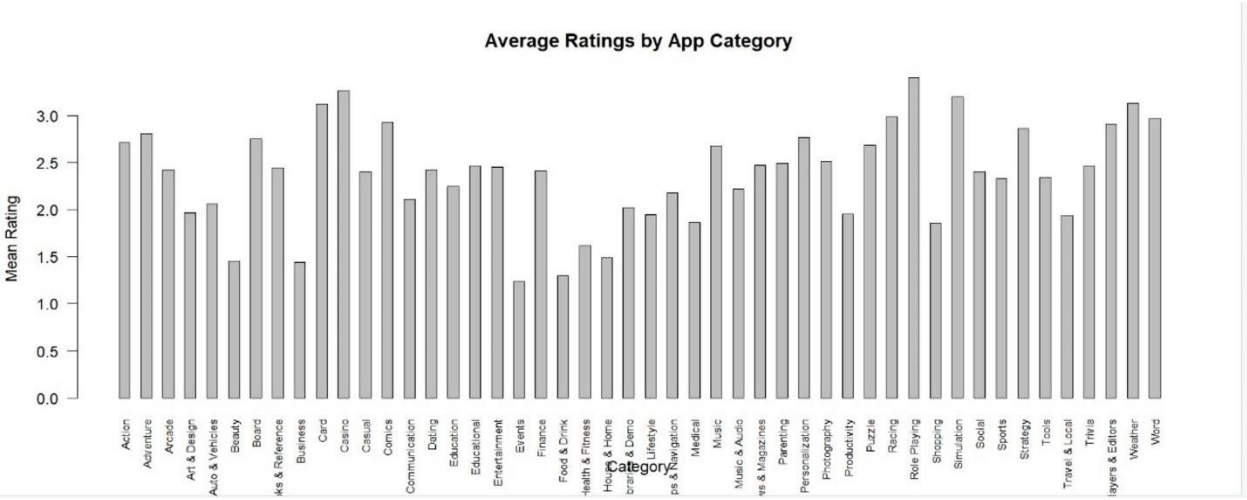
Category, y = Rating, fill = Category)) + geom_bar(stat = "identity") +

theme_minimal() +

labs(title = "Average Ratings by App Category",

x = "Category", y = "Average Rating") +

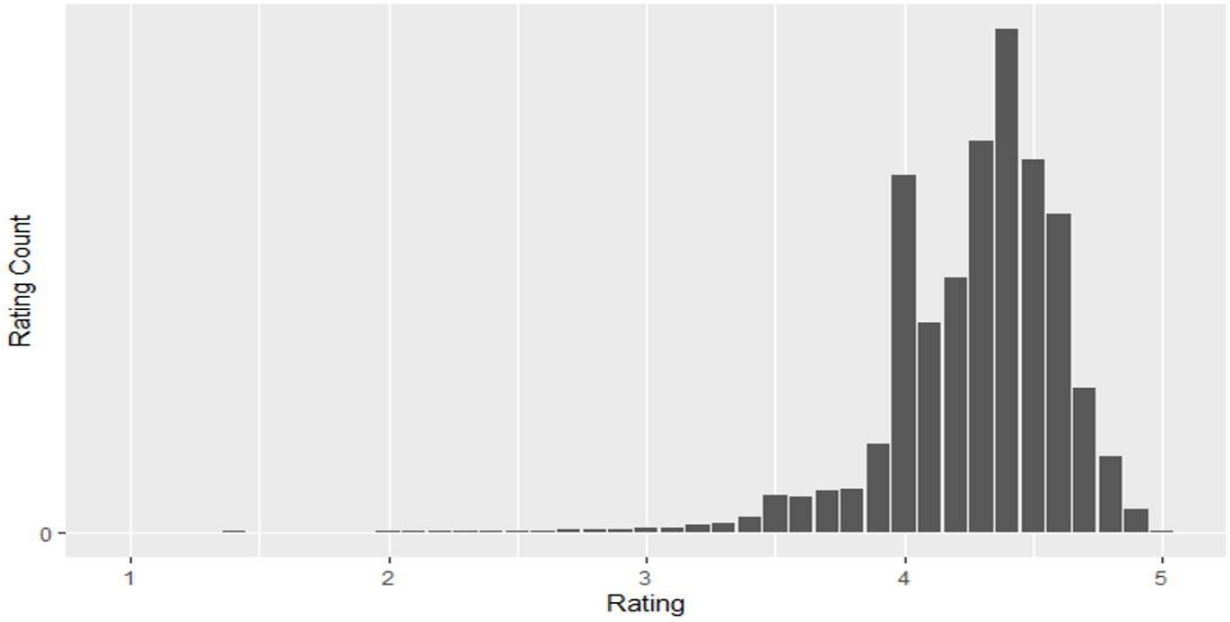
theme(axis.text.x = element_text(angle = 45, hjust = 1))



The code creates a bar chart showcasing the average rating for different app categories, with each category's average rating directly corresponding to the height of its bar.

RATING DISTRIBUTION WITH RATING COUNT

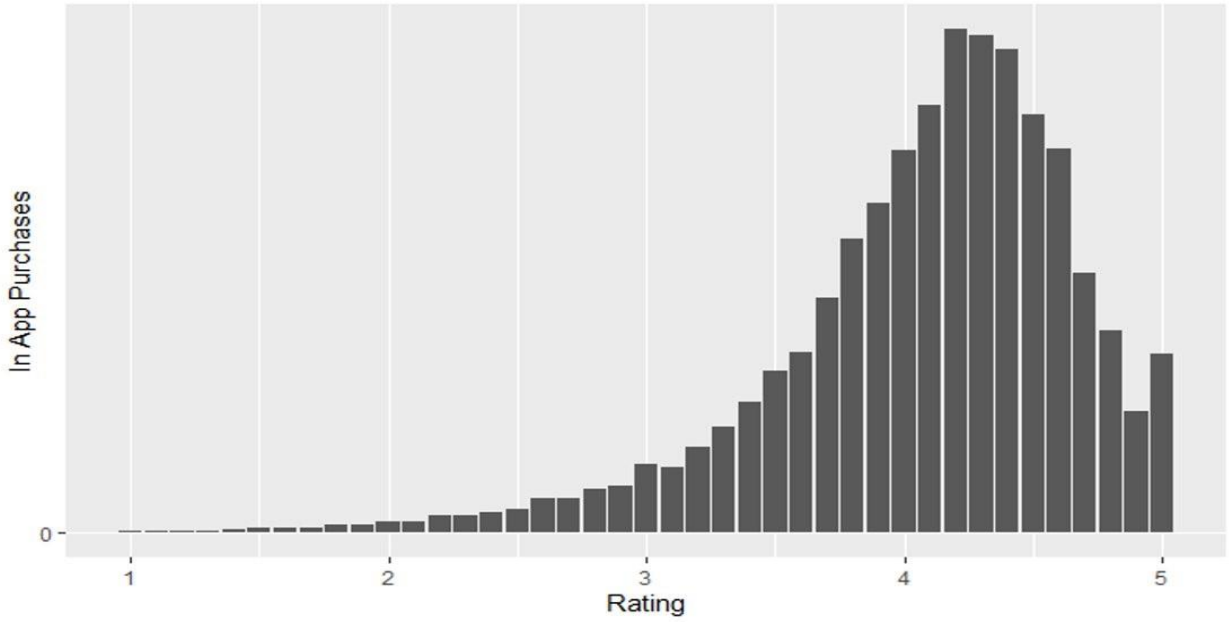
```
stratified_sample %>%

  ggplot(aes(x = Rating, y = `Rating Count`)) + geom_bar(stat = "identity") +
scale_y_continuous(breaks = seq(0, max(stratified_sample$`Rating`), by = 100))
```



The code produces a bar chart that displays the number of times each app rating occurs, with the majority of ratings concentrated around the higher end of the scale.

RATING DISTRIBUTION WITH IN-APP PURCHASES

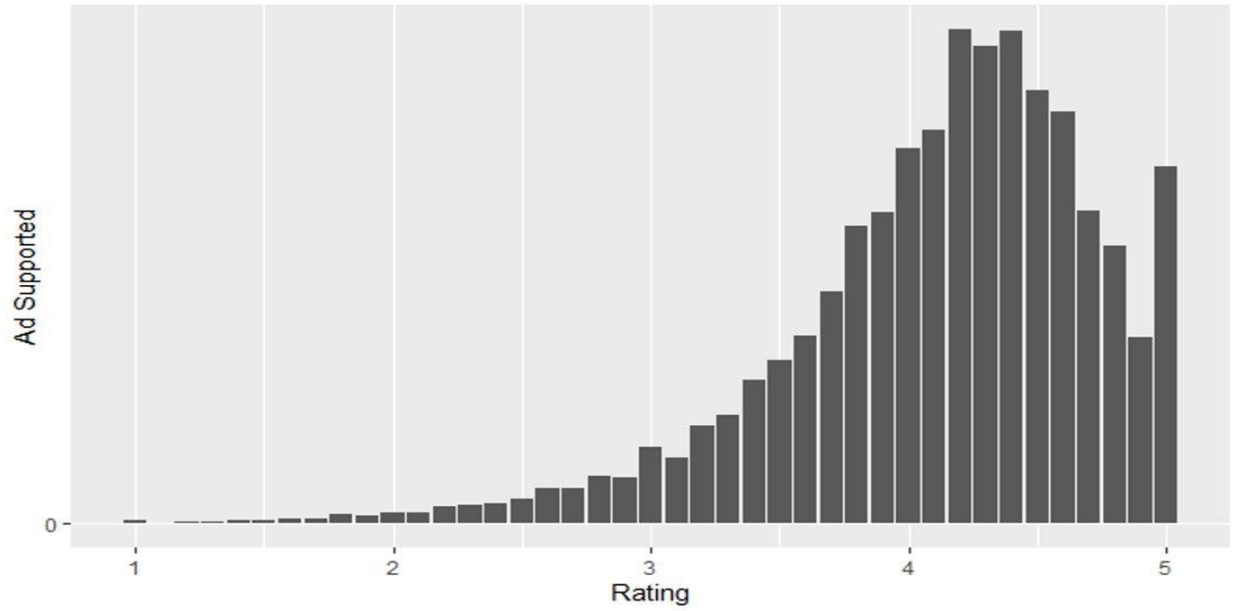
```
stratified_sample %>%
  ggplot(aes(x = Rating, y = `In App Purchases`)) + geom_bar(stat = "identity") +
scale_y_continuous(breaks = seq(0, max(stratified_sample$Rating), by = 100))
```



The bar chart plotting the relationship between app ratings and the number of in-app purchases, revealing that apps with higher ratings tend to have more in-app purchases. The visualization opts for a simple and uncluttered design, facilitating easy interpretation of the data.

RATING DISTRIBUTION WITH IN-APP ADS

```
stratified_sample %>%
  ggplot(aes(x = Rating, y = `Ad Supported`)) + geom_bar(stat = "identity") +
scale_y_continuous(breaks = seq(0, max(stratified_sample$Rating), by = 100))
```



This histogram shows the distribution of apps that support ads across different rating categories. Similar to the in-app purchases distribution, there is a skew towards higher ratings. This could imply that higher-rated apps are more likely to be ad-supported, although this doesn't necessarily indicate a causal relationship.

 # The Bayesian Estimation:

The Bayesian estimation of the average rating can be represented using the formula for the posterior distribution:

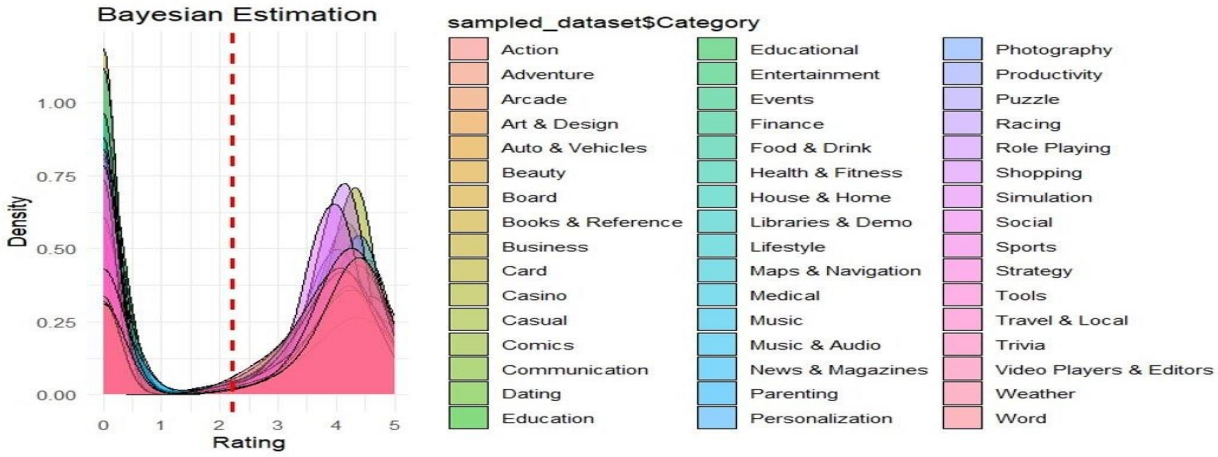
$$\text{Posterior}(\mu|\text{data}) \propto \text{Likelihood}(\text{data}|\mu) \times \text{Prior}(\mu)$$

In this case, assuming a normal distribution for the prior:

$$\text{Prior}(\mu) \sim N(\text{mean}(\text{data}), \text{sd}(\text{data}))$$

Code:

```
library(ggplot2)

ggplot(stratified_sample, aes(x = Rating, fill = Category)) +

geom_density(alpha = 0.5, color = "black", size = 0.5) +

 geom_vline(aes(xintercept = average_rating_bayesian), linetype = "dashed", color = "red", size = 1) +

 labs(title = "Bayesian Estimation",

x = "Rating",       y = "Density") +

theme_minimal()
```



The code snippet and the resulting graph depict a Bayesian estimation approach, visualizing the distribution of app ratings for different categories with a density plot. Each category's distribution is represented by a distinct colored curve, and the Bayesian estimated average rating across all categories is marked by a dashed red vertical line, demonstrating a nuanced look at the spread and central tendency of ratings in the dataset.

Inferential Statistics:

Confidence Intervals:

Confidence intervals give us a range where we can expect the true average rating to fall. The R code `calculate_ci` calculates this range based on our data, adjusting for any missing values. It uses the sample's average rating, variability, and sample size to compute the 95% confidence interval. This means we're 95% sure that the real average rating is between the lower and upper limits calculated by the function. When we apply this to our rating data, we get an interval that tells us where the true mean likely lies with high confidence.

```
calculate_ci <- function(data) {
mean_rating <- mean(data, na.rm = TRUE)
  se <- sd(data, na.rm = TRUE) / sqrt(length(na.omit(data))) # Corrected for NA values
margin_of_error <- qt(0.975, df = length(na.omit(data)) - 1) * se   lower_ci <-
mean_rating - margin_of_error   upper_ci <- mean_rating + margin_of_error
return(c(mean_rating, lower_ci, upper_ci))
}
overall_ci <- calculate_ci(stratified_sample$Rating)
names(overall_ci) <- c("Mean Rating", "Lower CI", "Upper CI")
print(overall_ci)
```

A confidence interval indicates where the Rating is likely to reside.

95% confidence interval for mean:

Lower: 2.199204

Upper: 2.209668

95% probability that mean of income lies between 2.199204 and 2.209668

Hypothesis Testing:

H0: βj = 0 (Null hypothesis for parameter βj)

H1: βj ≠ 0 (Alternative hypothesis for parameter βj)

Here, H0 represents the null hypothesis, stating that there is no effect (βj = 0), and H1 is the alternative hypothesis, suggesting that there is an effect (βj ≠ 0).

In Bayesian hypothesis testing, we can use the posterior samples of each parameter to assess the probability that the parameter is equal to or different from zero. The code checks if the mean of the posterior samples times the null hypothesis is less than or equal to zero to determine whether the effect is significant.

t_test_result <- t.test(stratified_sample$Rating, mu = 3.5, alternative = "greater")

print(t_test_result) p_value <-

t_test_result$p.value t_statistic <-

t_test_result$statistic

print(paste("p-value:", p_value))

print(paste("t-statistic:", t_statistic))

Extracting and printing the 95% confidence interval

conf_interval <- t_test_result$conf.int

print(paste("95% Confidence Interval: [", conf_interval[1], ", ", conf_interval[2], "]", sep=""))

Null Hypothesis (H0): App rating is equal to 3.5

Alternative Hypothesis (H1): App rating is less than 3.5

App rating is greater than 3.5

App rating is less than 3.5

App rating is not equal to 3.5

After testing, we can observe that the rating is significantly greater than 3.5

Regression Analysis:

SIMPLE LINEAR REGRESSION ANALYSIS:

Simple Linear Regression analysis allows us to examine the relation between two variables.

Model 1:

Rating ~ Rating Count

Dependent variable: App Rating

Independent variable: App Rating Count

```
Call:
lm(formula = stratified_sample$Rating ~ stratified_sample$`Rating Count`)

Residuals:
     Min       1Q   Median       3Q      Max
-3.10183 -0.30185  0.09817  0.49817  0.89817

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                    4.102e+00  1.101e-03 3725.84  < 2e-16 ***
stratified_sample$`Rating Count` 8.989e-09  3.317e-09    2.71  0.00673 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6881 on 390719 degrees of freedom
Multiple R-squared:  1.88e-05,  Adjusted R-squared:  1.624e-05
F-statistic: 7.344 on 1 and 390719 DF,  p-value: 0.006729
```

The residuals have a range from -3.10183 to 0.89817. The residual standard error is 0.6881, which represents the standard deviation of the residuals. The R-squared is 1.88e-05, indicating that only a very small proportion of the variability in the ratings is explained by the Rating Count.

So, as the R square is very small, we have included additional relevant independent variables into the next multiple regression model that might contribute to explaining the variability in the dependent variable (App Rating).

MULTI LINEAR REGRESSION ANALYSIS:

Multi Linear Regression analysis allows us to examine the relation between more than two variables.

Model 2:

Rating ~ Rating Count + In App Purchases

Dependent variable: In-App Rating

Independent variable: App Rating Count and In-App Purchases

```
Call:
lm(formula = stratified_sample$Rating ~ stratified_sample$`Rating Count` +
    stratified_sample$"In App Purchases")

Residuals:
     Min       1Q   Median       3Q      Max
-3.10545 -0.30545  0.09456  0.49455  0.92327

Coefficients:
                                  Estimate Std. Error  t value Pr(>|t|)
(Intercept)                      4.105e+00  1.177e-03 3486.781   <2e-16
stratified_sample$`Rating Count`  9.630e-09  3.318e-09    2.903   0.0037
stratified_sample$"In App Purchases" -2.872e-02  3.316e-03   -8.659   <2e-16

(Intercept)                      ***
stratified_sample$`Rating Count`  **
stratified_sample$"In App Purchases" ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.688 on 390718 degrees of freedom
Multiple R-squared:  0.0002107, Adjusted R-squared:  0.0002055
F-statistic: 41.16 on 2 and 390718 DF,  p-value: < 2.2e-16
```

In the model 2 as well we can see the adjusted R square has increased a bit after taking the “In App Purchases” as another independent variable to the first model but still this is very low to explain the variability in the ratings.

Model 3:

Rating ~ Rating Count + In App Purchases + Ad Supported

Dependent variable: In-App Rating

Independent variable: App Rating Count , In-App Purchases and In-App Ads

```
Call:
lm(formula = stratified_sample$Rating ~ stratified_sample$`Rating Count` +
    stratified_sample$"In App Purchases" + stratified_sample$`Ad Supported`)

Residuals:
    Min      1Q  Median      3Q     Max
-3.1298 -0.3298  0.1243  0.4702  0.9638

Coefficients:
                                  Estimate Std. Error  t value Pr(>|t|)
(Intercept)                      4.076e+00  1.704e-03 2391.674  < 2e-16 ***
stratified_sample$`Rating Count`  9.556e-09  3.315e-09    2.883  0.00395 **
stratified_sample$"In App Purchases" -3.955e-02  3.344e-03  -11.826  < 2e-16 ***
stratified_sample$`Ad Supported`  5.409e-02  2.245e-03   24.093  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6875 on 390717 degrees of freedom
Multiple R-squared:  0.001694,  Adjusted R-squared:  0.001686
F-statistic:   221 on 3 and 390717 DF,  p-value: < 2.2e-16
```

Here we added the “Ads Supported” as independent variable and verified if there is any difference in the Adjusted R square. But these independent variables did not add much explanatory power. And, as we add the independent variable to the model our f-statistic values are also getting increased, it may be due to relevance of additional variables, overfitting or multicollinearity.

The model has statistical significance (p-values) but explains only a very small proportion of the variance in the rating.

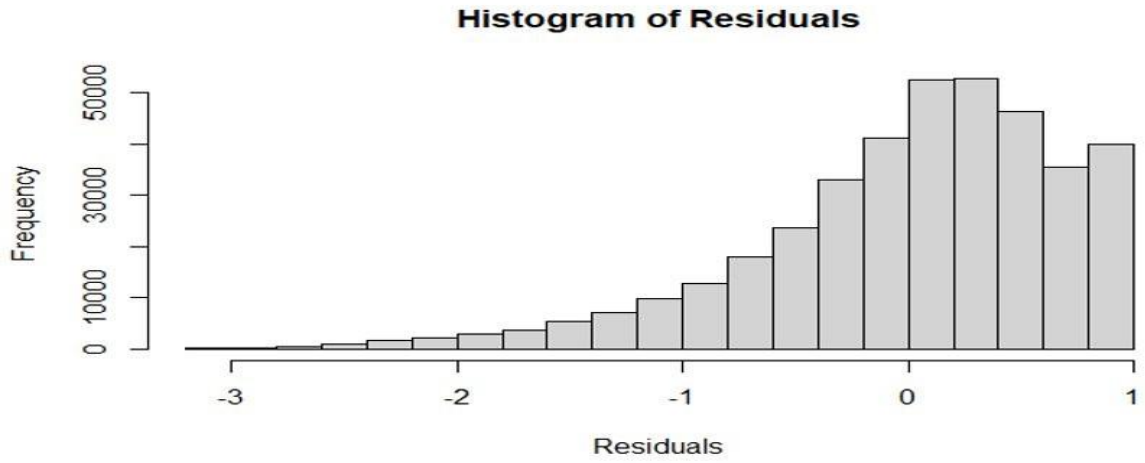
The low R-squared values indicate that the model may not be a good fit for predicting the rating based on these variables alone. So, we tried even changing the proportion of our sample size, but I did not make much impact.

RESIDUAL DIAGNOSTICS

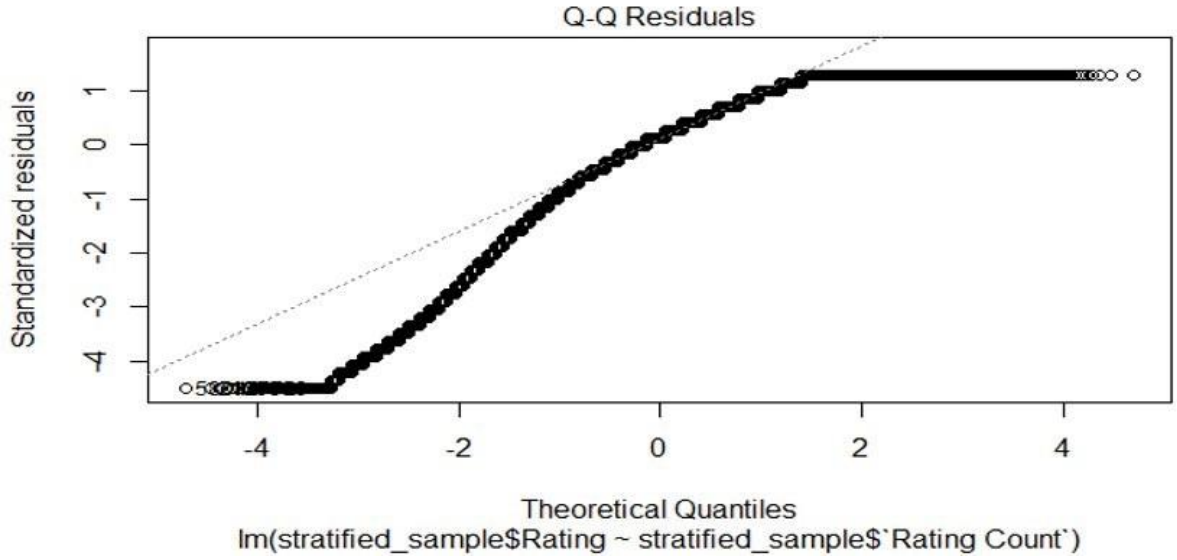
Residual diagnostics serve to check the fulfillment of regression assumptions.

hist(model$residuals, main = "Histogram of Residuals", xlab = "Residuals")

To assess the normality of residuals:



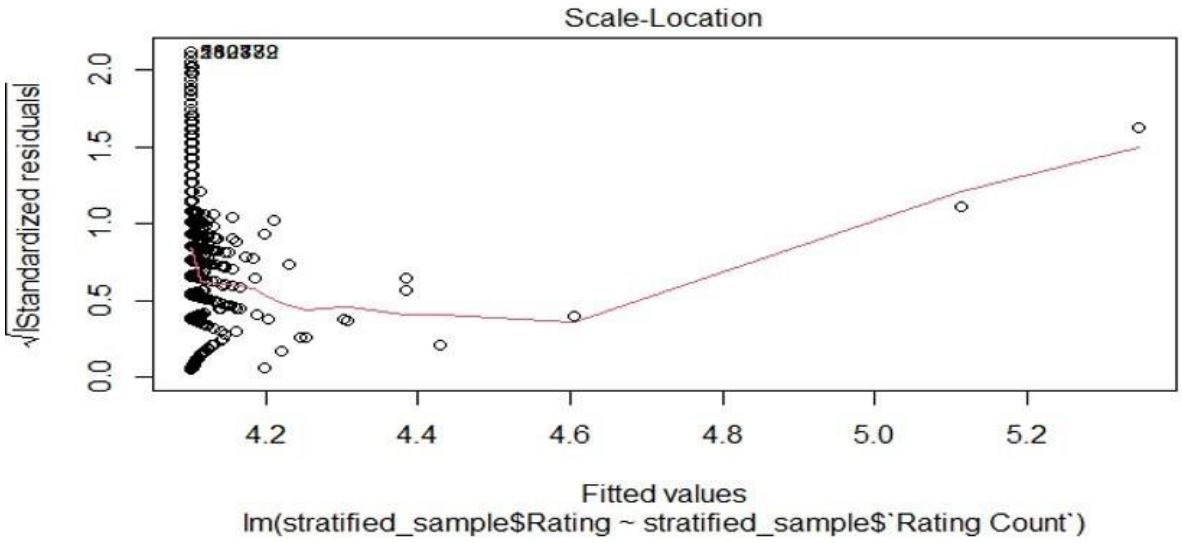
The histogram shows that the residuals are not symmetrically distributed about zero, which is a sign that the residuals may not be normally distributed. The skewness towards the left suggests the presence of outliers or a long tail in that direction.



plot(model, which = 2)

The Q-Q plot compares the distribution of the residuals to a normal distribution. In a perfect situation, the points should lie on the diagonal line. The deviation of points from the line, especially in the tails, indicates that the residuals are not normally distributed, or this could indicate the presence of outliers.

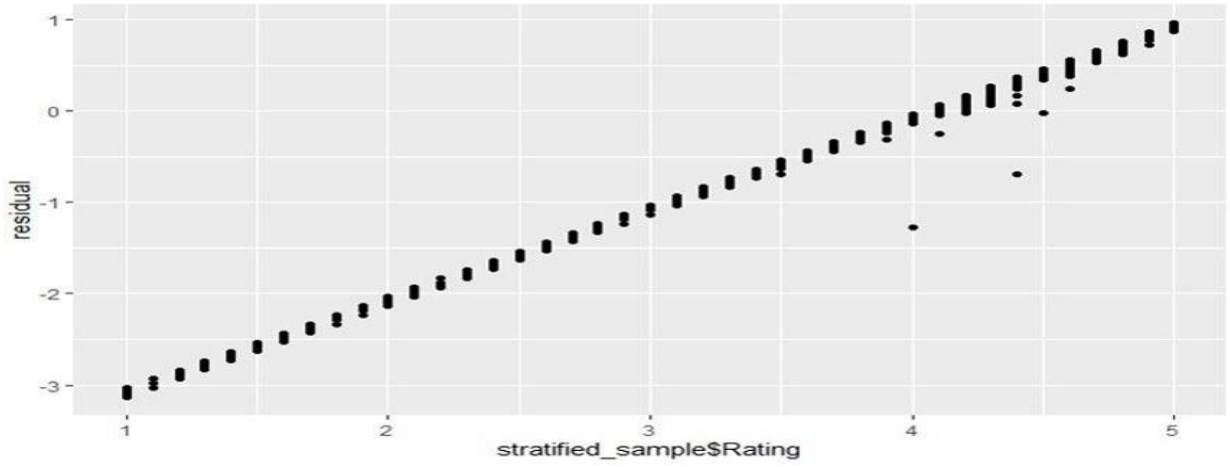
For Homoscedasticity:



plot(model, which = 3)

The 'U' shape of the red line indicates that the variance is not constant and suggests the presence of heteroscedasticity. The plot displays a pattern where the spread of the residuals increases with the fitted values, which is indicated by the upward trend of the red line. This suggests that the variance of the residuals is not constant. This means the assumption of constant variance of the residuals (homoscedasticity) is violated.

For Heteroscedasticity:



```
resdf=data.frame(cbind(residual=residuals(model), Salary=stratified_sample$`Rating`))
ggplot(data = resdf, mapping = aes(x = stratified_sample$`Rating`, y = residual)) +
geom_point()
```

The plot shows a clear pattern where the residuals increase with the fitted values, which suggests non-linearity in the relationship between the predictors and the outcome. The increasing spread of residuals as the fitted values increase (forming a funnel shape) also indicates heteroscedasticity. Given the pattern observed, it might be necessary consider a nonlinear model to better fit the data.

To assess the Autocorrelation in residuals:

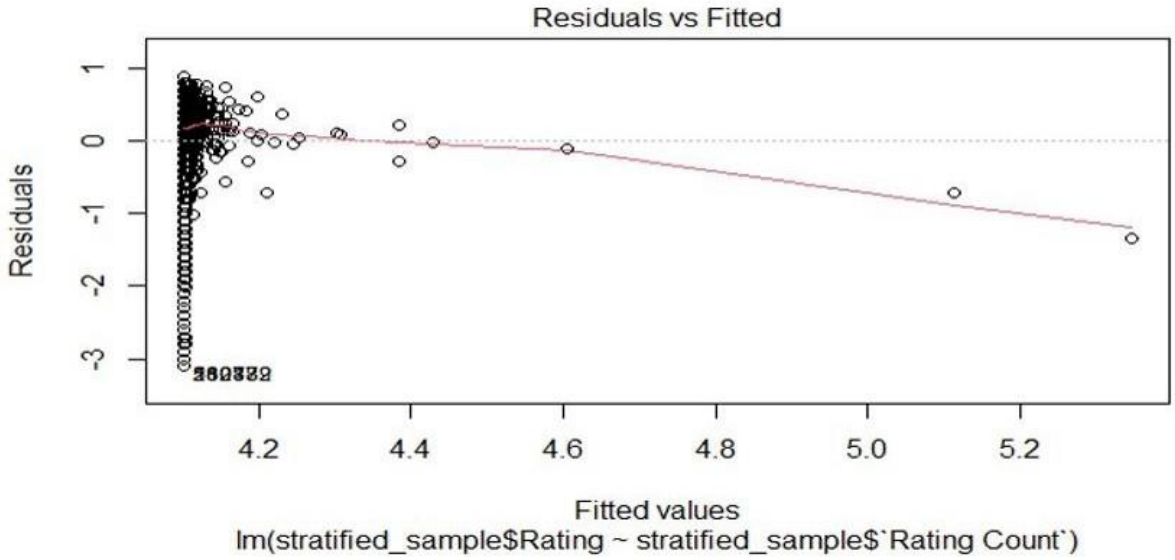
```
#library(car)
durbinWatsonTest(model)
```

```
durbinWatsonTest(model)
```

```
 lag Autocorrelation D-W Statistic p-value
   1    0.0003252982      1.999349   0.858
 Alternative hypothesis: rho != 0
```

The P value is greater than 0.5 and Durbin-Watson statistic is approx. 2, which is indicating that there is no auto correlation, and the residuals are independent to each other.

For Linearity and Identify potential patterns:



plot(model, which = 1)

This plot should ideally show no discernible pattern if the linear model is appropriate. However, the plot reveals a clear pattern, with residuals decreasing as the fitted values increase, suggesting that the linear model may not be the best fit for this data. the variance of the residuals is non-constant.

Conclusion:

Associations Not Causes: The data shows links between app features and ratings, but we can't say these features cause higher ratings.

Need for Better Models: Adding more data to the models helped a little, but they still do not explain app ratings well, hinting that other important factors are missing.

Beyond Linear Regression: The patterns in the plots hint that app ratings might be influenced by complex, non-linear relationships that our models are not capturing.

Model Issues: The residual plots show our model does not meet key assumptions - the residuals aren't evenly spread and don't follow a normal distribution, suggesting the model might not be the best fit.

Future Research:

Here we added the “Ads Supported” as independent variable and saw if there is any difference in the Adjusted R square.

But these independent variables did not add much explanatory power.

And also, as we add the independent variable to the model our f-statistic values is also getting increased, it may be due to Relevance of Additional Variables, Overfitting or Multicollinearity.

The model has statistical significance (indicated by the low p-values) but explains only a very small proportion of the variance in the rating.

The low R-squared values indicate that the model may not be a good fit for predicting the rating based on these variables alone. So, I tried even changing the proportion of our sample size but I did not make much impact.

Reference:

- https://www.southampton.ac.uk/passs/confidence_in_the_police/multivariate_analysis/linear_regression.page
- https://www.grammarly.com/grammar-check
- https://web.ma.utexas.edu/users/mks/statmistakes/modelcheckingplots.html
- https://timeseriesreasoning.com/contents/assumptions-of-linear-regression/
- https://www.andrew.cmu.edu/user/achoulde/94842/homework/regression_diagnostics.html