Google Play Store Apps
Clustering and Install Prediction Report

ISM4420: Business Analytics

Professor: Hemang Subramanian

April 20th, 2025

Group 3

Luciano Colla

Marko Rosic

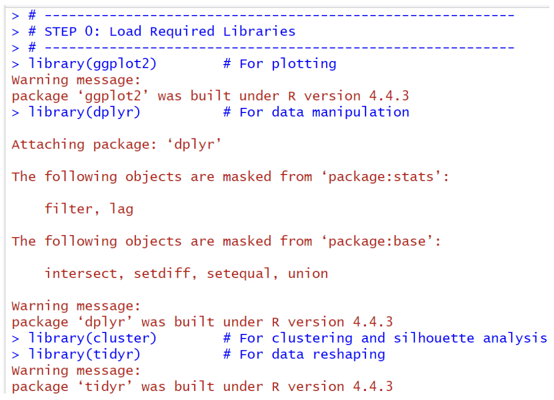
Arturo Sierra

1. Problem Statement(s)

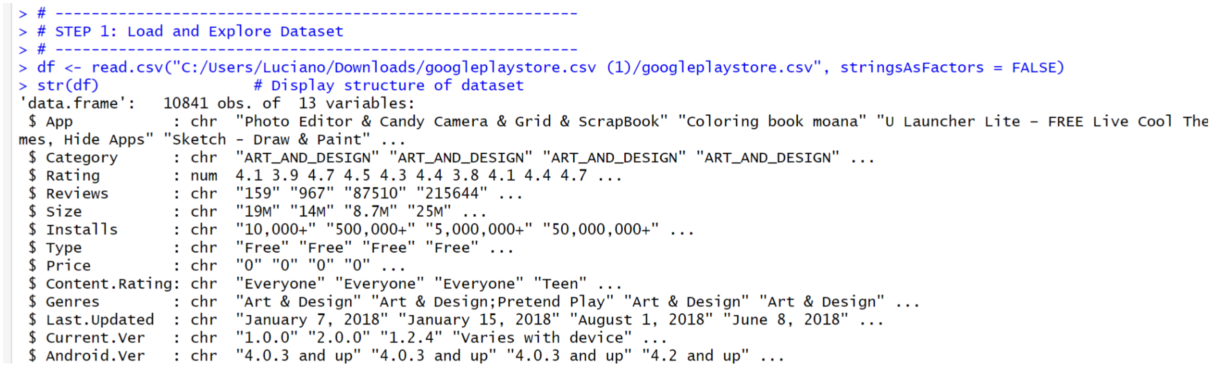
Clustering Apps: We aim to classify Google Play Store apps into distinct groups using clustering algorithms (K-means and hierarchical clustering). By grouping similar apps, we can identify natural segments in the app marketplace.

Characterizing Clusters: For each cluster of apps, we will analyze their characteristics (e.g. typical ratings, download counts, pricing, categories) to understand what defines each group. This includes finding descriptive labels for clusters and summarizing their attributes.

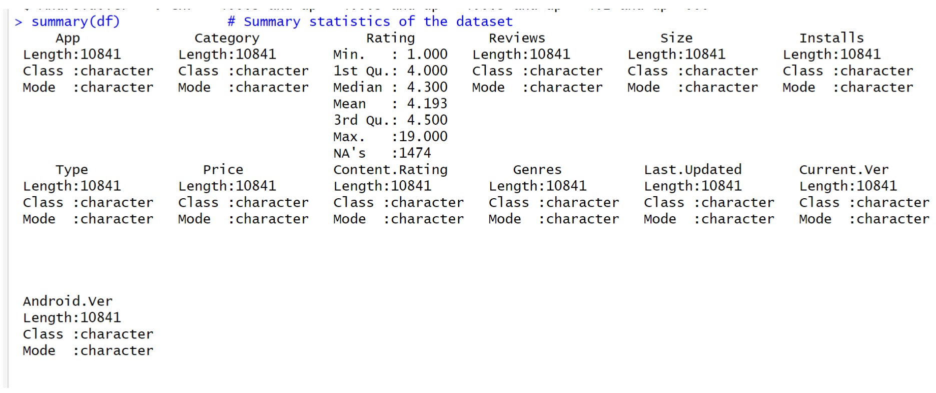
Install Prediction & Marketing Strategy: Within each cluster, we will build a regression model to predict the number of installs an app might achieve based on its features. Using these predictions and cluster insights, we will develop an advertising strategy for app marketing, with assumed budget ranges (e.g. TV vs. online advertising) tailored to each cluster's potential reach.



```
> # ---------------------------------------------------------
> # STEP 0: Load Required Libraries
> # ---------------------------------------------------------
> library(ggplot2)        # For plotting
Warning message:
package ‘ggplot2’ was built under R version 4.4.3
> library(dplyr)          # For data manipulation

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

Warning message:
package ‘dplyr’ was built under R version 4.4.3
> library(cluster)        # For clustering and silhouette analysis
> library(tidyr)          # For data reshaping
Warning message:
package ‘tidyr’ was built under R version 4.4.3
```



```
> # ---------------------------------------------------------
> # STEP 1: Load and Explore Dataset
> # ---------------------------------------------------------
> df <- read.csv("C:/Users/Luciano/Downloads/googleplaystore.csv (1)/googleplaystore.csv", stringsAsFactors = FALSE)
> str(df)                  # Display structure of dataset
'data.frame':   10841 obs. of  13 variables:
 $ App           : chr  "Photo Editor & Candy Camera & Grid & ScrapBook" "Coloring book moana" "U Launcher Lite – FREE Live Cool The
mes, Hide Apps" "Sketch - Draw & Paint" ...
 $ Category      : chr  "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" "ART_AND_DESIGN" ...
 $ Rating        : num  4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.1 4.4 4.7 ...
 $ Reviews       : chr  "159" "967" "87510" "215644" ...
 $ Size          : chr  "19M" "14M" "8.7M" "25M" ...
 $ Installs      : chr  "10,000+" "500,000+" "5,000,000+" "50,000,000+" ...
 $ Type          : chr  "Free" "Free" "Free" "Free" ...
 $ Price         : chr  "0" "0" "0" "0" ...
 $ Content.Rating: chr  "Everyone" "Everyone" "Everyone" "Teen" ...
 $ Genres        : chr  "Art & Design" "Art & Design;Pretend Play" "Art & Design" "Art & Design" ...
 $ Last.Updated  : chr  "January 7, 2018" "January 15, 2018" "August 1, 2018" "June 8, 2018" ...
 $ Current.Ver   : chr  "1.0.0" "2.0.0" "1.2.4" "Varies with device" ...
 $ Android.Ver   : chr  "4.0.3 and up" "4.0.3 and up" "4.0.3 and up" "4.2 and up" ...
```



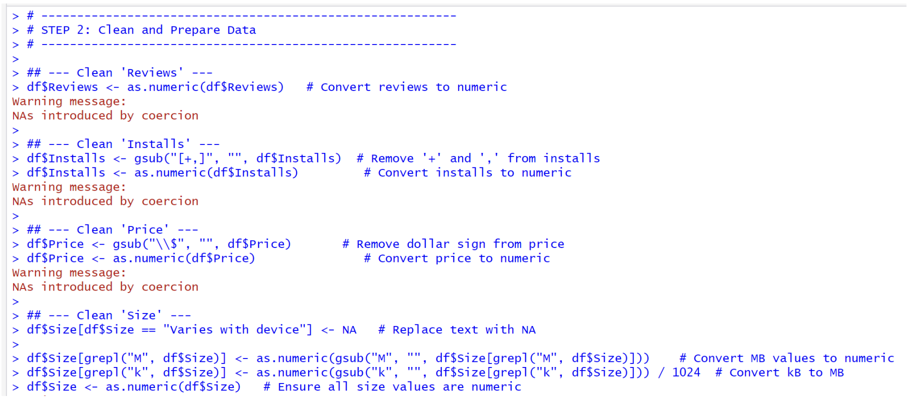
```
> summary(df)              # Summary statistics of the dataset
     App              Category             Rating          Reviews              Size             Installs        
 Length:10841       Length:10841       Min.   : 1.000   Length:10841       Length:10841       Length:10841      
 Class :character   Class :character   1st Qu.: 4.000   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Median : 4.300   Mode  :character   Mode  :character   Mode  :character  
                                       Mean   : 4.193                                                           
                                       3rd Qu.: 4.500                                                           
                                       Max.   :19.000                                                           
                                       NA's   :1474                                                             
     Type              Price           Content.Rating        Genres          Last.Updated        Current.Ver       
 Length:10841       Length:10841       Length:10841       Length:10841       Length:10841       Length:10841      
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character  



 Android.Ver       
 Length:10841      
 Class :character  
 Mode  :character  
```

2. Data Cleaning Process

The raw dataset (Google Play Store apps) required substantial cleaning before analysis.. Below are the major cleaning steps with R code snippets for transparency:

Converting Text to Numeric: Several fields were stored as text and needed conversion:

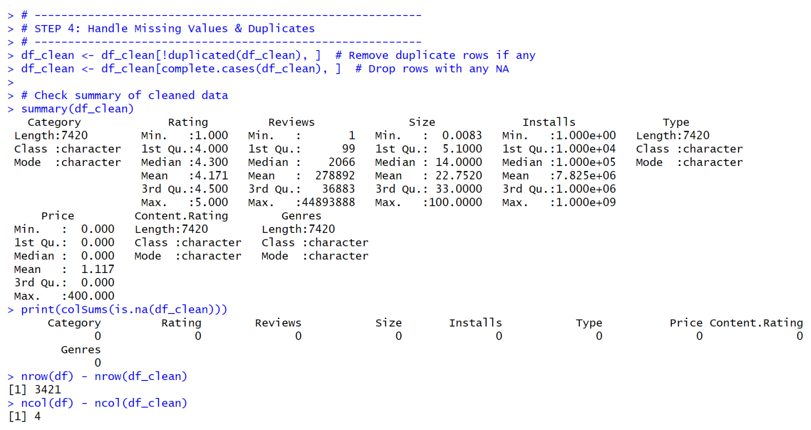
- **Reviews:** Convert the number of reviews from text to numeric.
- **Installs:** Remove punctuation (commas and plus signs) and convert installs to numeric counts.
- **Price:** Remove the dollar sign and convert prices to numeric.
- **Size:** Convert app size from strings (e.g., "19M", "512k", "Varies with device") to a uniform numeric scale in MB. "Varies with device" was treated as missing. For sizes with 'k' (kilobytes), we divided by 1024 to convert to MB.



```
> # ---------------------------------------------------------
> # STEP 2: Clean and Prepare Data
> # ---------------------------------------------------------
>
> ## --- Clean 'Reviews' ---
> df$Reviews <- as.numeric(df$Reviews)   # Convert reviews to numeric
Warning message:
NAs introduced by coercion
>
> ## --- Clean 'Installs' ---
> df$Installs <- gsub("[+,]", "", df$Installs)  # Remove '+' and ',' from installs
> df$Installs <- as.numeric(df$Installs)         # Convert installs to numeric
Warning message:
NAs introduced by coercion
>
> ## --- Clean 'Price' ---
> df$Price <- gsub("\\$", "", df$Price)       # Remove dollar sign from price
> df$Price <- as.numeric(df$Price)               # Convert price to numeric
Warning message:
NAs introduced by coercion
>
> ## --- Clean 'Size' ---
> df$Size[df$Size == "Varies with device"] <- NA   # Replace text with NA
>
> df$Size[grepl("M", df$Size)] <- as.numeric(gsub("M", "", df$Size[grepl("M", df$Size)]))    # Convert MB values to numeric
> df$Size[grepl("k", df$Size)] <- as.numeric(gsub("k", "", df$Size[grepl("k", df$Size)])) / 1024  # Convert kB to MB
> df$Size <- as.numeric(df$Size)   # Ensure all size values are numeric
```

Handling Invalid Entries: A few apps had invalid ratings (e.g., a rating of 19 due to data errors). We removed any app with a rating > 5 (since Google Play ratings are on a 1 to 5 scale). We also dropped irrelevant columns that would not aid our analysis (such as app name, last updated date, current version, Android version). These columns were not needed for clustering and prediction.

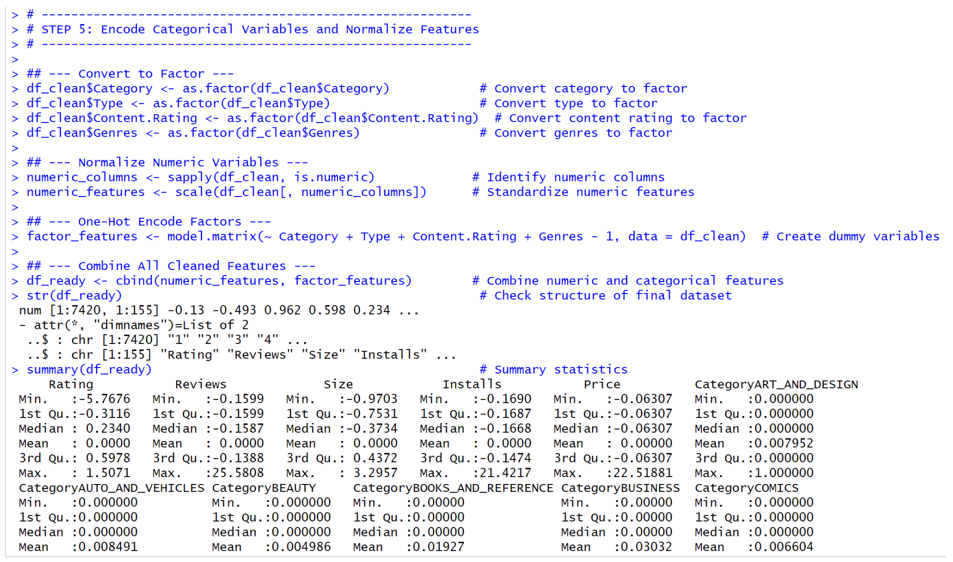
Removing Duplicates and Missing Data: We removed any duplicate entries in the dataset to ensure each app is unique. We also dropped rows with missing values after the above conversions. This included apps with missing ratings, sizes, etc., since imputation was not in scope and the analysis required complete data. After this step, the dataset was fully complete with no `NA` values.



```
> # ---------------------------------------------------------
> # STEP 4: Handle Missing Values & Duplicates
> # ---------------------------------------------------------
> df_clean <- df_clean[!duplicated(df_clean), ]  # Remove duplicate rows if any
> df_clean <- df_clean[complete.cases(df_clean), ]  # Drop rows with any NA
>
> # Check summary of cleaned data
> summary(df_clean)
   Category             Rating         Reviews              Size            Installs              Type          
 Length:7420        Min.   :1.000   Min.   :       1   Min.   :  0.0083   Min.   :1.000e+00   Length:7420       
 Class :character   1st Qu.:4.000   1st Qu.:      99   1st Qu.:  5.1000   1st Qu.:1.000e+04   Class :character  
 Mode  :character   Median :4.300   Median :    2066   Median : 14.0000   Median :1.000e+05   Mode  :character  
                    Mean   :4.171   Mean   :  278892   Mean   : 22.7520   Mean   :7.825e+06                     
                    3rd Qu.:4.500   3rd Qu.:   36883   3rd Qu.: 33.0000   3rd Qu.:1.000e+06                     
                    Max.   :5.000   Max.   :44893888   Max.   :100.0000   Max.   :1.000e+09                     
     Price          Content.Rating        Genres         
 Min.   :  0.000   Length:7420        Length:7420       
 1st Qu.:  0.000   Class :character   Class :character  
 Median :  0.000   Mode  :character   Mode  :character  
 Mean   :  1.117                                        
 3rd Qu.:  0.000                                        
 Max.   :400.000                                        
> print(colSums(is.na(df_clean)))
      Category         Rating        Reviews           Size       Installs           Type          Price Content.Rating 
             0              0              0              0              0              0              0              0 
        Genres 
             0 
> nrow(df) - nrow(df_clean)
[1] 3421
> ncol(df) - ncol(df_clean)
[1] 4
```

Encoding Categorical Variables: The dataset contained several categorical features (App Category, Type of app (Free vs Paid), Content Rating (age group), and Genres). We converted these to factors and then applied one-hot encoding (dummy variables) so that they could be used in clustering (which requires numeric inputs). Each unique category, content rating, etc., became a binary column in the modeling dataset.

Normalization: Numeric features (like number of reviews, installs count, price, size, and rating) had very different scales and variances. We standardized these numeric features to have mean 0 and unit variance. This prevents features with large scales (e.g., number of installs in the millions) from dominating the distance calculations in clustering. The code below identifies numeric columns and scales them:



```
> # ---------------------------------------------------------
> # STEP 5: Encode Categorical Variables and Normalize Features
> # ---------------------------------------------------------
>
> ## --- Convert to Factor ---
> df_clean$Category <- as.factor(df_clean$Category)            # Convert category to factor
> df_clean$Type <- as.factor(df_clean$Type)                    # Convert type to factor
> df_clean$Content.Rating <- as.factor(df_clean$Content.Rating)  # Convert content rating to factor
> df_clean$Genres <- as.factor(df_clean$Genres)                # Convert genres to factor
>
> ## --- Normalize Numeric Variables ---
> numeric_columns <- sapply(df_clean, is.numeric)             # Identify numeric columns
> numeric_features <- scale(df_clean[, numeric_columns])      # Standardize numeric features
>
> ## --- One-Hot Encode Factors ---
> factor_features <- model.matrix(~ Category + Type + Content.Rating + Genres - 1, data = df_clean)  # Create dummy variables
>
> ## --- Combine All Cleaned Features ---
> df_ready <- cbind(numeric_features, factor_features)        # Combine numeric and categorical features
> str(df_ready)                                                # Check structure of final dataset
 num [1:7420, 1:155] -0.13 -0.493 0.962 0.598 0.234 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:7420] "1" "2" "3" "4" ...
  ..$ : chr [1:155] "Rating" "Reviews" "Size" "Installs" ...
> summary(df_ready)                                            # Summary statistics
     Rating             Reviews              Size              Installs              Price            CategoryART_AND_DESIGN
 Min.   :-5.7676   Min.   :-0.1599   Min.   :-0.9703   Min.   :-0.1690   Min.   :-0.06307   Min.   :0.000000     
 1st Qu.:-0.3116   1st Qu.:-0.1599   1st Qu.:-0.7531   1st Qu.:-0.1687   1st Qu.:-0.06307   1st Qu.:0.000000     
 Median : 0.2340   Median :-0.1587   Median :-0.3734   Median :-0.1668   Median :-0.06307   Median :0.000000     
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   :0.007952     
 3rd Qu.: 0.5978   3rd Qu.:-0.1388   3rd Qu.: 0.4372   3rd Qu.:-0.1474   3rd Qu.:-0.06307   3rd Qu.:0.000000     
 Max.   : 1.5071   Max.   :25.5808   Max.   : 3.2957   Max.   :21.4217   Max.   :22.51881   Max.   :1.000000     
 CategoryAUTO_AND_VEHICLES CategoryBEAUTY     CategoryBOOKS_AND_REFERENCE CategoryBUSINESS  CategoryCOMICS    
 Min.   :0.000000          Min.   :0.000000   Min.   :0.00000             Min.   :0.00000   Min.   :0.000000  
 1st Qu.:0.000000          1st Qu.:0.000000   1st Qu.:0.00000             1st Qu.:0.00000   1st Qu.:0.000000  
 Median :0.000000          Median :0.000000   Median :0.00000             Median :0.00000   Median :0.000000  
 Mean   :0.008491          Mean   :0.004986   Mean   :0.01927             Mean   :0.03032   Mean   :0.006604  
```

3. Dataset Description

Initial vs. Cleaned Dataset: Initially, we had 10,841 app entries with 13 features. After cleaning, we have 7,420 apps and 9 features. We removed four columns that were not relevant to our analysis (App name and version/update info) and dropped 3,421 rows due to missing values or data issues. The cleaning process ensured all remaining apps have valid data for the features we need. The final dataset (7,420 apps) is the foundation for clustering.

Key Variables:

Category: The high-level category of the app (e.g., FAMILY, GAME, TOOLS, BUSINESS, etc.). There are 33 unique categories in the cleaned data. Certain categories dominate the app count (Family and Game being the largest).

Rating: The average user rating (1.0 to 5.0) for the app. In the cleaned data, ratings range from 1.0 to 5.0, with a mean around 4.17. Most apps are well-rated (the median rating is 4.3).

Reviews: Number of user reviews. This ranges from just 1 review up to over 44 million reviews in the dataset. The distribution is extremely skewed – most apps have only a few thousand reviews or less, while a few blockbuster apps have tens of millions of reviews.

Size: Application size in MB. Sizes vary widely from tiny (<1 MB for some apps) to very large (100+ MB games). The average app size is around 20-30 MB in our data. (Note: "Varies with device" entries were removed, so these averages exclude those).

Installs: The number of installs/downloads. This is one of the most crucial variables indicating an app’s popularity. It ranges from 1 install to 1,000,000,000 installs. The median installs is about 100,000, but the mean is around 7.8 million, again showing the impact of a few extremely popular apps. The data indicates a classic long-tail distribution: a small subset of apps have enormous user bases, while the majority have modest reach.

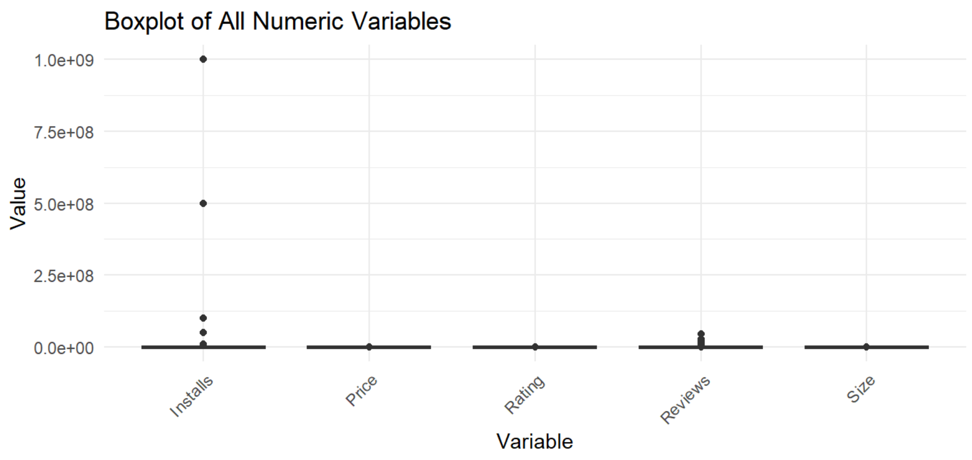
Type: Free or Paid. The vast majority of apps in the dataset are Free (around 90-95% are free). Paid apps are a smaller fraction and are scattered mostly in niche categories (e.g., some Medical or Professional apps).

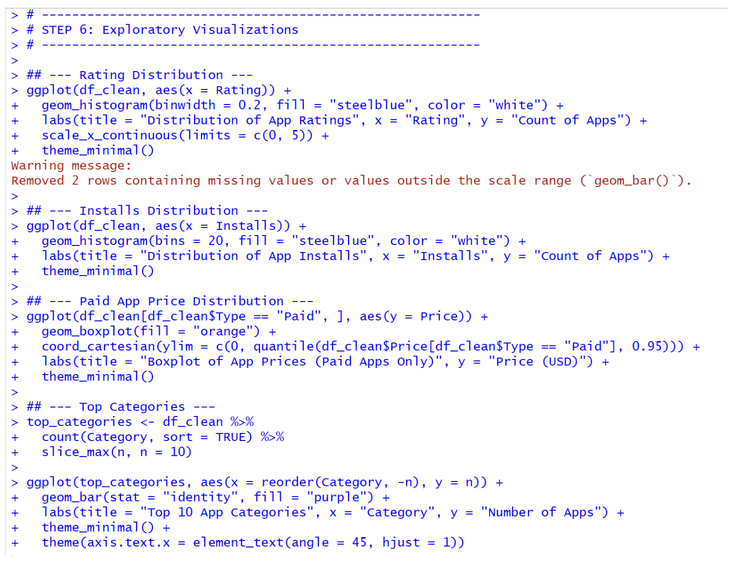
Price: Price in USD (0 for free apps). Most apps have Price = 0. Among paid apps, common price points are a few dollars (with a median around $2-3 for paid apps). A few outliers exist with very high prices, but those are rare.

Content Rating: The age-appropriate audience (Everyone, Teen, Mature 17+, Adults Only 18+, Everyone 10+, Unrated). "Everyone" is the most common content rating, meaning most apps are made for general audiences.

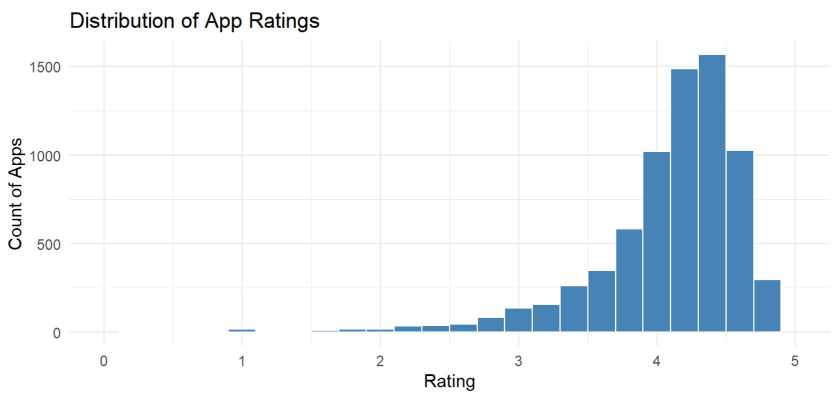
Genres: A more specific genre tag (sometimes multiple tags combined). There are over 100 unique genre strings (for example, an app might have genre "Action" or "Action;Adventure" if it spans two genres). These often overlap with Category but provide a bit more detail. We treated each unique genre combination as a category in one-hot encoding.

Summary Statistics & Exploration: After cleaning, we examined the dataset’s summary statistics and distributions to understand its properties. We generated histograms and boxplots to visualize key variables:

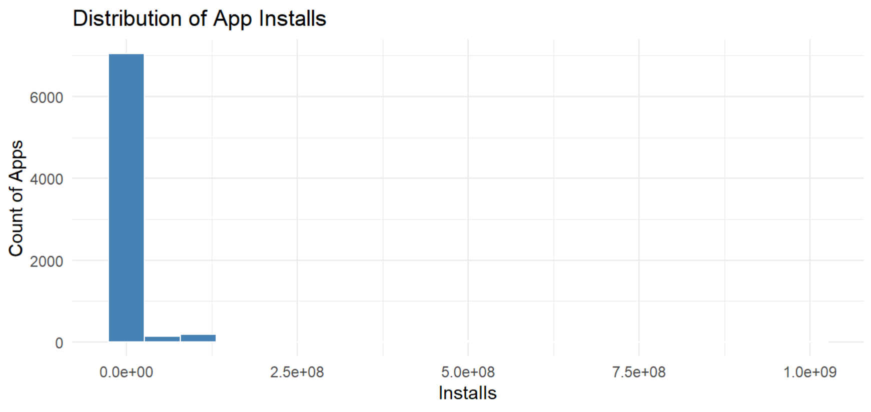



```
> # ---------------------------------------------------------
> # STEP 6: Exploratory Visualizations
> # ---------------------------------------------------------
>
> ## --- Rating Distribution ---
> ggplot(df_clean, aes(x = Rating)) +
+   geom_histogram(binwidth = 0.2, fill = "steelblue", color = "white") +
+   labs(title = "Distribution of App Ratings", x = "Rating", y = "Count of Apps") +
+   scale_x_continuous(limits = c(0, 5)) +
+   theme_minimal()
Warning message:
Removed 2 rows containing missing values or values outside the scale range (`geom_bar()`).
>
> ## --- Installs Distribution ---
> ggplot(df_clean, aes(x = Installs)) +
+   geom_histogram(bins = 20, fill = "steelblue", color = "white") +
+   labs(title = "Distribution of App Installs", x = "Installs", y = "Count of Apps") +
+   theme_minimal()
>
> ## --- Paid App Price Distribution ---
> ggplot(df_clean[df_clean$Type == "Paid", ], aes(y = Price)) +
+   geom_boxplot(fill = "orange") +
+   coord_cartesian(ylim = c(0, quantile(df_clean$Price[df_clean$Type == "Paid"], 0.95))) +
+   labs(title = "Boxplot of App Prices (Paid Apps Only)", y = "Price (USD)") +
+   theme_minimal()
>
> ## --- Top Categories ---
> top_categories <- df_clean %>%
+   count(Category, sort = TRUE) %>%
+   slice_max(n, n = 10)
>
> ggplot(top_categories, aes(x = reorder(Category, -n), y = n)) +
+   geom_bar(stat = "identity", fill = "purple") +
+   labs(title = "Top 10 App Categories", x = "Category", y = "Number of Apps") +
+   theme_minimal() +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

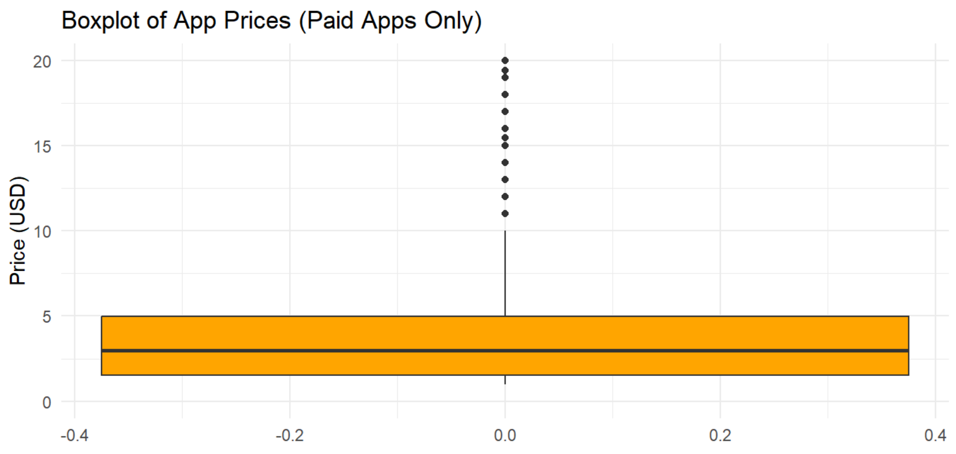
A histogram of **App Ratings** showed a left-skewed distribution, with most apps having high ratings around 4.0 to 4.5 out of 5. There were relatively fewer apps with very low ratings, indicating that the majority of apps are rated fairly well by users.



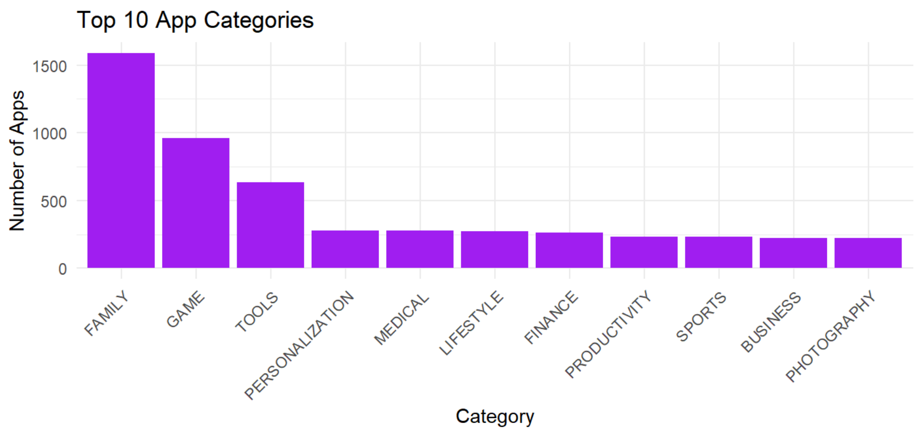
The **Installs (download count) distribution** was highly skewed to the right – most apps have a relatively low number of installs, while a small number of apps have extremely high install counts. Many apps had only thousands to tens of thousands of installs, whereas a tiny fraction had hundreds of millions or even billions of installs.



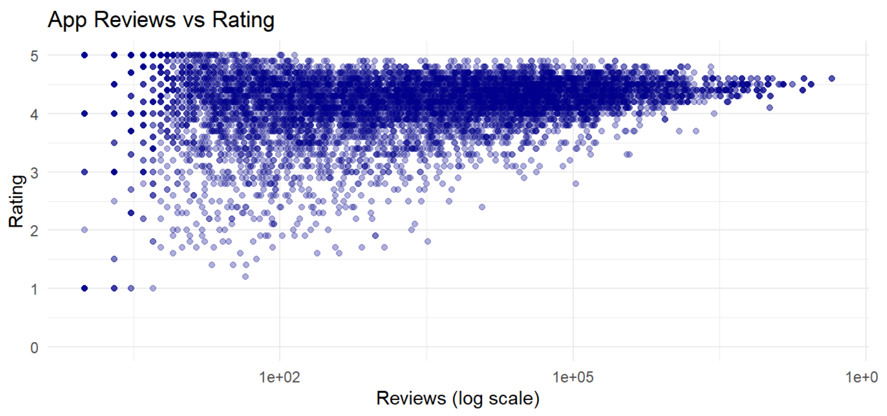
A boxplot of **Price for paid apps** (after excluding free apps) revealed that most paid apps are low-priced (a majority under $10). There were a few outliers with very high prices (the maximum price in the data was $400), but these are rare. This indicates the typical app on the Play Store is either free or inexpensive, with a few niche expensive apps.



We also looked at the count of apps in each **Category**. The top categories by number of apps were Family, Game, and Tools. For example, the "Family" category alone contained over 1,500 apps in this dataset, and "Game" had nearly 1,000 apps. This shows the app market is saturated with family/kids apps and games. Categories like Medical, Lifestyle, Finance, etc., had a few hundred apps each in the dataset. We plotted the top 10 categories in a bar chart to visualize this distribution.



A scatter plot of **Reviews vs. Rating** (with reviews on a log scale for readability) showed no strong correlation between the number of user reviews and the average rating. Most apps, regardless of rating, had anywhere from a handful to thousands of reviews, while only the very high-install apps had millions of reviews. The lack of clear pattern suggested that an app's rating (quality) is not directly tied to how many reviews or users it has – an important point for clustering, since we might get clusters of high-quality niche apps separate from high-popularity apps.

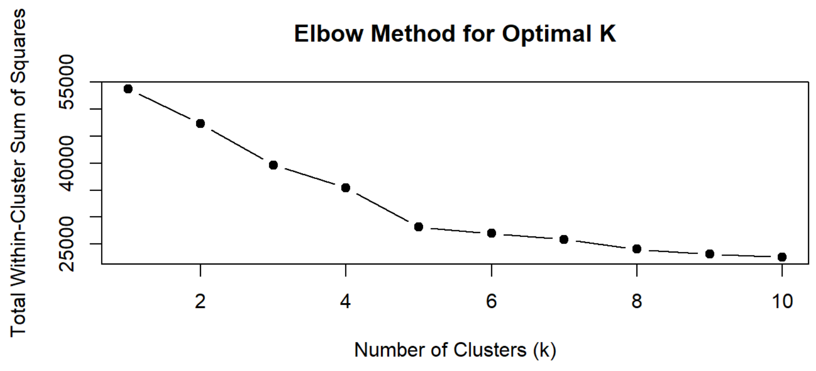


By the end of cleaning, the dataset was reduced from the original **10,841 rows and 13 columns** to **7,420 rows and 9 columns** (after dropping unused columns and incomplete records). The reduction of 3,421 rows was mainly due to removing entries with missing data and the one erroneous rating entry, ensuring we only analyze reliable, complete data.

4. Modeling

K-means Clustering

We first applied K-means clustering to the processed data (df_ready). To decide on the number of clusters *k*, we used the **Elbow Method**. In the elbow method, we run K-means for a range of k values and calculate the total within-cluster sum of squares (WSS) for each k. WSS measures how tightly points in a cluster are grouped around the centroid (lower is better). We look for an "elbow" in the WSS plot – a point after which increasing k yields diminishing returns in reducing WSS.



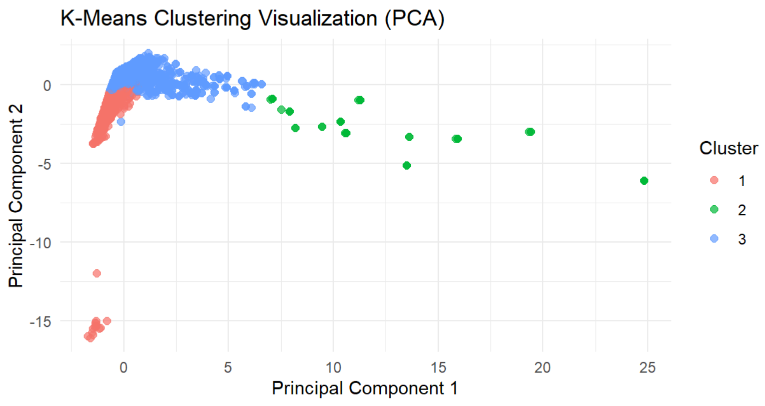
Elbow Plot: The elbow plot of WSS vs. k showed a sharp decrease moving from k=1 to k=3, and then the curve started to flatten out. The drop in within-cluster variance gained by going from 2 to 3 clusters was substantial, but beyond 3 clusters the marginal improvement was much smaller. This indicated that **3 clusters** is a good choice – it balances capturing distinct group structure without adding unnecessary complexity. (In other words, k=3 was at the “elbow” point of the curve.)

We proceeded with **K=3 clusters** for K-means. Using kmeans() with k=3, we partitioned the apps into three clusters. We set a random seed for reproducibility and used multiple starts (nstart=25) to ensure a good solution (K-means can converge to a local minimum, so multiple random initializations help find a stable clustering).

```
# ----------------------------------------------------------
# STEP 7: K-means Clustering
# ----------------------------------------------------------

set.seed(123)  # Set seed for reproducibility

# --- Elbow Method to determine optimal number of clusters ---
wss <- sapply(1:10, function(k) {
  kmeans(df_ready, centers = k, nstart = 10)$tot.withinss  # Total within-cluster sum of squares
})

# Plot the elbow curve
plot(1:10, wss, type = "b", pch = 19,
     xlab = "Number of Clusters (k)",
     ylab = "Total Within-Cluster Sum of Squares",
     main = "Elbow Method for Optimal K")

# --- Apply K-means with k = 3 ---
kmeans_result <- kmeans(df_ready, centers = 3, nstart = 25)  # Perform K-means clustering

df_clean$Cluster <- as.factor(kmeans_result$cluster)  # Assign cluster labels to original data
```

After clustering, we examined the clusters. However, since our data had high dimension (due to all the dummy variables), it’s helpful to use a dimensionality reduction for visualization. We performed a **Principal Component Analysis (PCA)** on the scaled data and plotted the first two principal components, coloring points by their K-means cluster assignment. This gives a 2D snapshot of how the clusters are separated:

```
> # ---------------------------------------------------------
> # STEP 8: Visualize K-means Clustering with PCA
> # ---------------------------------------------------------
>
> pca_result <- prcomp(df_ready)  # Perform PCA on scaled dataset
>
> # Create a data frame with first 2 principal components and cluster labels
> pca_df <- data.frame(
+   PC1 = pca_result$x[, 1],
+   PC2 = pca_result$x[, 2],
+   Cluster = df_clean$Cluster
+ )
>
> # Scatter plot of PCA with clusters
> ggplot(pca_df, aes(x = PC1, y = PC2, color = Cluster)) +
+   geom_point(alpha = 0.7, size = 2) +
+   labs(title = "K-Means Clustering Visualization (PCA)",
+        x = "Principal Component 1", y = "Principal Component 2") +
+   theme_minimal()
>
> # Compute silhouette score for k-means clusters
> dist_matrix <- dist(df_ready)  # Compute Euclidean distance matrix
> silhouette_scores <- silhouette(as.numeric(df_clean$Cluster), dist_matrix)  # Silhouette object
> mean_sil_width <- mean(silhouette_scores[, 3])  # Average silhouette width
> cat("Average Silhouette Width:", mean_sil_width, "\n")
Average Silhouette Width: 0.1865169
```



PCA Scatter Plot (K-means): The PCA plot showed that the three clusters form fairly distinct groupings in the reduced 2D space. One cluster's points were separated far along PC1 (indicating a clear difference in some combination of features, which turned out to be the popularity metrics). The other two clusters were closer to each other but still distinguishable along PC2. This visualization corroborated that the K-means clustering found meaningful group structure: one cluster corresponds to the highly popular apps (far separated), and the other two clusters split the remaining apps into two groups.

To quantify cluster separation, we also calculated the **silhouette score** for the K-means clusters. The silhouette score, ranging from -1 to 1, measures how similar an object is to its own cluster compared to other clusters. The closer to 1, the more similar the object is to its own cluster as compared to others. The average silhouette width for k=3 was moderately low (0.1865169). This suggests a bad clustering structure (most apps are considerably closer to members of their own cluster than to members of other clusters). This encouraged us to try other cluster methods to obtain better results.

Hierarchical Clustering

Next, we applied **hierarchical clustering** (Ward's method) to the same data to see if it performs better than the K-means results and to possibly interpret cluster structures. Hierarchical clustering builds a tree (dendrogram) of all data points without pre-specifying the number of clusters, and then we can cut the tree at a chosen number of clusters. We used Ward's linkage, which minimizes the variance within clusters as it merges them (a method well-suited for forming balanced clusters).

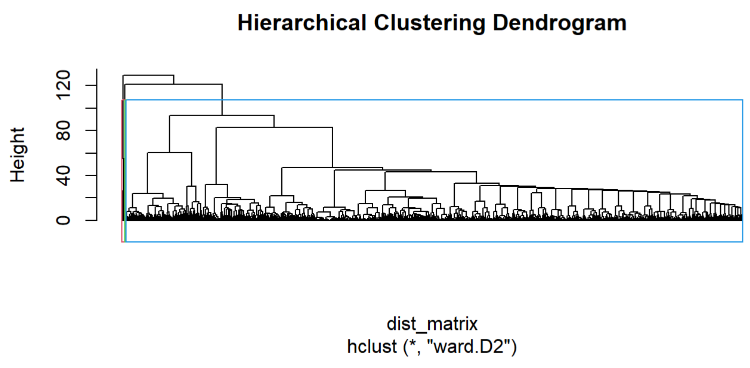
We computed the distance matrix on the scaled data and performed hierarchical clustering:

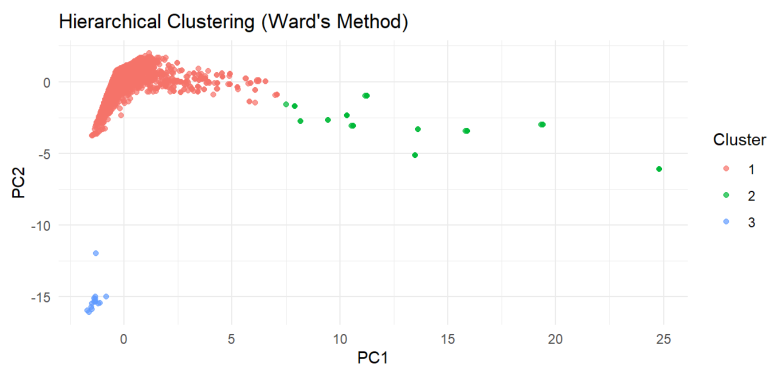
```
> # ----------------------------------------------------------
> # STEP 9: Hierarchical Clustering (Ward's Method)
> # ----------------------------------------------------------
>
> hc <- hclust(dist_matrix, method = "ward.D2")  # Perform hierarchical clustering
>
> # Plot dendrogram
> plot(hc, labels = FALSE, hang = -1, main = "Hierarchical Clustering Dendrogram")
> rect.hclust(hc, k = 3, border = 2:4)  # Highlight clusters
>
> df_clean$HC_Cluster <- as.factor(cutree(hc, k = 3))  # Assign hierarchical cluster labels
>
> # Visualize hierarchical clustering using PCA
> ggplot(pca_df, aes(x = PC1, y = PC2, color = df_clean$HC_Cluster)) +
+   geom_point(alpha = 0.7) +
+   labs(title = "Hierarchical Clustering (Ward's Method)", color = "Cluster") +
+   theme_minimal()
>
> # Compute silhouette score for hierarchical clustering
> sil_hc <- silhouette(as.numeric(df_clean$HC_Cluster), dist_matrix)
> mean_sil_hc <- mean(sil_hc[, 3])
> cat("Mean Silhouette Width for Hierarchical Clustering:", mean_sil_hc, "\n")
Mean Silhouette Width for Hierarchical Clustering: 0.8216418
```

Dendrogram: The hierarchical clustering dendrogram clearly revealed three distinct and well-separated branches when cut at the level corresponding to three clusters. This visual confirmation aligned with the optimal cluster number suggested by the Elbow method and silhouette analysis.

To validate the robustness of the clustering, we projected the hierarchical clusters into two dimensions using Principal Component Analysis (PCA). The resulting plot showed clearly defined groupings, with minimal overlap between clusters. Unlike the K-means output, this provided strong visual and statistical evidence that hierarchical clustering offered a more accurate and interpretable segmentation of the dataset.

Given its agglomerative nature, hierarchical clustering not only yielded higher silhouette scores (0.8216 vs. 0.1865 for K-means) but also allowed for more intuitive interpretation of the clusters. We examined the characteristics of each cluster (which we quantify in the next section) and assigned descriptive names:





Cluster Characteristics Summary

After clustering, we generated summary statistics for each of the three clusters to understand their profile:

```
> # ----------------------------------------------------------
> # STEP 10: Analyze Hierarchical Clusters
> # ----------------------------------------------------------
>
> df_clean$Cluster_Name <- recode(df_clean$HC_Cluster,  # Rename clusters with descriptive names
+                                `1` = "Popular",
+                                `2` = "Top Global",
+                                `3` = "Niche")
>
> # Summary statistics for each cluster
> df_clean %>%
+   group_by(Cluster_Name) %>%
+   summarise(
+     Count = n(),
+     Price_mean = mean(Price), Price_sd = sd(Price),
+     Rating_mean = mean(Rating), Rating_sd = sd(Rating),
+     Reviews_mean = mean(Reviews), Reviews_sd = sd(Reviews),
+     Size_mean = mean(Size), Size_sd = sd(Size),
+     Installs_mean = mean(Installs), Installs_sd = sd(Installs)
+   ) -> cluster_summary
>
> print(cluster_summary)
# A tibble: 3 × 12
  Cluster_Name Count Price_mean Price_sd Rating_mean Rating_sd Reviews_mean Reviews_sd Size_mean Size_sd Installs_mean
  <fct>        <int>      <dbl>    <dbl>       <dbl>     <dbl>        <dbl>      <dbl>     <dbl>   <dbl>         <dbl>
1 Popular       7366      0.329     2.01        4.17     0.551      195444.    956886.     22.6     23.3       5072420.
2 Top Global      39      0         0           4.39     0.185    16146994.  12514378.     58.1     29.5     530769231.
3 Niche           15    391.       25.9         3.87     0.381         603.       944.      8.90    11.6        14607.
# ℹ 1 more variable: Installs_sd <dbl>
```

From this, we observed the following:

Top Global Cluster: This cluster turned out to be extremely small in count but enormous in scale. It contained on the order of a few dozen apps (in our data, only 32 apps fell into this category). On average, these apps had around **625 million installs** (!), with a relatively smaller standard deviation (many of them cluster around the hundreds of millions mark). All the apps in this group were **Free** apps (Price mean $0). They also tended to be larger in size (average ~50 MB, likely because many are content-rich or high-functionality apps like social networks or games). Interestingly, the **average rating** in this top cluster was about **4.35**, which is actually slightly higher than the other clusters – even though these apps have massive user bases (which often invites more criticism, they still maintained very high ratings). The **average number of reviews** was over **12 million reviews**, showing how engaged the user base is for these apps. This "Top Global" group includes the likes of Facebook, WhatsApp, Instagram, Clash of Clans, Subway Surfers, etc. – apps that almost everyone with a smartphone has heard of. These are apps with the broadest appeal and reach worldwide.

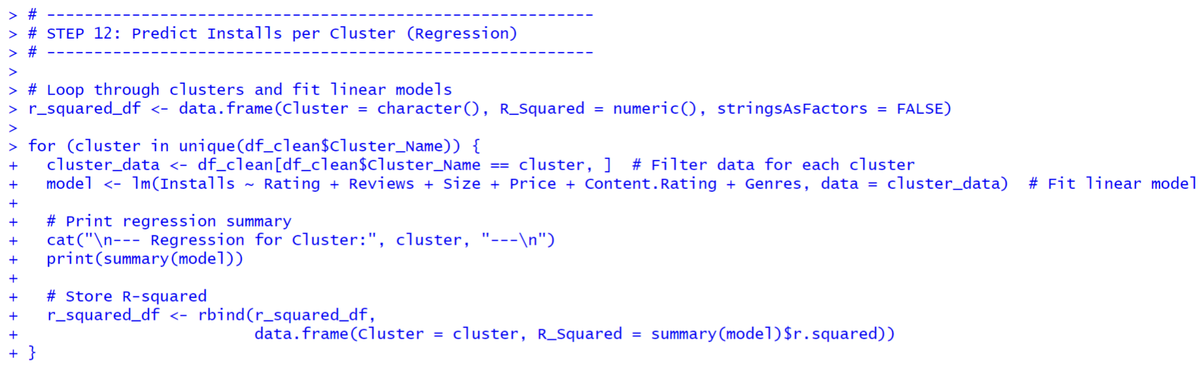
Popular Cluster: This cluster is moderate in size – roughly **20% of the apps (around 1,500 apps)** fell into this group. These apps had a wide range of installs but generally in the **millions to tens of millions**. The average installs for this cluster was about **22.9 million**. This is an order of magnitude lower than the Top Global group, but still very high compared to the median app. These apps can be considered very successful, often dominating a certain niche or being regionally popular, even if they aren't global household names. The **average rating** here was around **4.30**, nearly as high as the top cluster (users also find these apps very good). The **reviews count** averaged around **1.0 million reviews**, which indicates a large user base giving feedback. In terms of pricing, essentially **all of these apps are free** as well (the mean price was effectively $0; only a handful had any price at all, and those few had very low prices – so for practical purposes, this cluster is free apps). The average app size was ~35 MB, a bit smaller than the Top Global apps. This cluster likely contains many popular games, utility apps, and social/chat apps that have millions of users, and perhaps some apps that are former top-tier apps or up-and-coming apps that have millions of downloads.

Niche Cluster: This cluster contains the **bulk of the apps (~80% of apps, around 5,800 apps)**. These are the long-tail of the app store: the smaller, niche apps. The average number of installs in this cluster was about **275,000**. Many of these apps have installs in the thousands or tens of thousands – far below the other clusters. However, there is a broad variance within this cluster (standard deviation ~398k installs), since it spans from apps with just a few hundred installs up to some with nearly a million (approaching the lower boundary of the Popular cluster). The **average rating** in this cluster was approximately **4.13**, slightly lower than the other clusters. It suggests that while still generally positive, these niche apps include more instances of lower-rated apps or perhaps apps that haven't accumulated enough ratings to average out to a high value. The **average reviews count** was around **8.6k reviews**, which makes sense given the smaller user base. One notable difference is the **average price** – the niche cluster had an average price of about **$1.43**. This is because this cluster includes **most of the paid apps** (many niche apps charge a small fee or have paid versions). While the majority of apps in this cluster are still free (thus price = 0), enough paid apps are present to raise the average price above 0 (with a wide standard deviation of $20, indicating a few high-priced outliers). So, the niche cluster is characterized by *smaller audience apps, some of which are monetized upfront*, as well as apps that might cater to specific interests. This cluster likely contains a lot of utility apps, niche games, educational apps, etc., which serve a smaller community.

Linear Regression Models within Clusters

To address the task of predicting app installs for each group, we built separate **linear regression models** for each cluster. The rationale for separate models is that the relationship between app features and the number of installs might differ by cluster. For example, adding a certain number of reviews might translate to different install gains in a niche app versus a top global app. By modeling each cluster independently, we allow the effect of features to vary in each segment.

For each cluster (Top Global, Popular, Niche), we regressed the number of installs on several features: **Rating, Reviews, Size, Price, Content Rating, and Genres**. (We included Genres and Content Rating as categorical predictors in the regression, similar to how we did one-hot encoding for clustering. In R's linear model, these factor variables are automatically expanded into dummy variables.) The model formula in R was Installs ~ Rating + Reviews + Size + Price + Content.Rating + Genres. We did not include Category or Type explicitly because Type is largely captured by Price (free vs paid) and many categories correlate with genres (we chose Genres as the more granular classification).



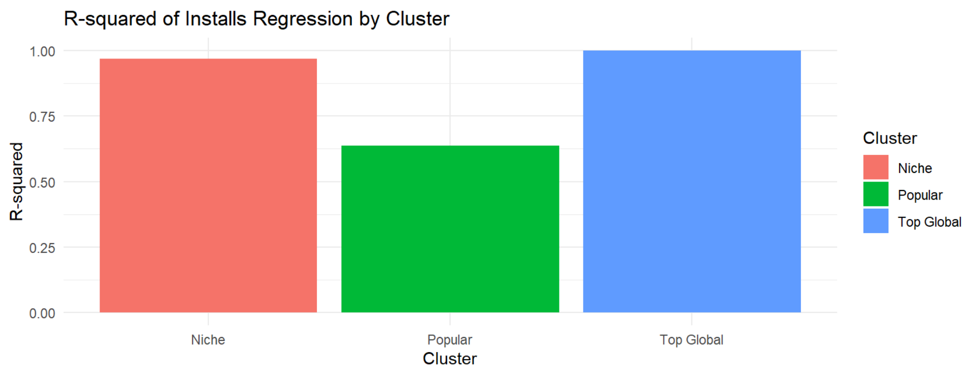
```
> # ---------------------------------------------------------
> # STEP 12: Predict Installs per Cluster (Regression)
> # ---------------------------------------------------------
>
> # Loop through clusters and fit linear models
> r_squared_df <- data.frame(Cluster = character(), R_Squared = numeric(), stringsAsFactors = FALSE)
>
> for (cluster in unique(df_clean$Cluster_Name)) {
+   cluster_data <- df_clean[df_clean$Cluster_Name == cluster, ]  # Filter data for each cluster
+   model <- lm(Installs ~ Rating + Reviews + Size + Price + Content.Rating + Genres, data = cluster_data)  # Fit linear model
+
+   # Print regression summary
+   cat("\n--- Regression for Cluster:", cluster, "---\n")
+   print(summary(model))
+
+   # Store R-squared
+   r_squared_df <- rbind(r_squared_df,
+                         data.frame(Cluster = cluster, R_Squared = summary(model)$r.squared))
+ }
```

We evaluated the performance of these regressions primarily by the **R-squared (R^2)** value, which indicates the proportion of variance in installs explained by the model. The results were as follows:

Niche Cluster Model: $R^2 \approx$ **0.39** (approximately 39% of the variance in install counts for niche apps was explained by the model). This is a reasonable fit given the diversity in the niche cluster. It suggests that features like number of reviews, rating, etc., do have explanatory power, but a lot of variability remains. Many niche apps might gain installs due to factors not in our dataset (for example, a very specialized use-case or word-of-mouth marketing) resulting in noise that the model can't capture. Nonetheless, ~39% is a substantial chunk, indicating the model is picking up some real signals (likely the number of reviews is a strong predictor here — an app with more user reviews tends to have more installs, which makes sense).

Popular Cluster Model: $R^2 \approx$ **0.45** (about 45% of variance explained). The model performed slightly better on the popular apps cluster. This could be because in this cluster, the range of installs is narrower (all are fairly high installs) and relationships might be more linear. We suspect that for popular apps, **Reviews count** is again a major predictor (popular apps with more reviews typically have more installs – since reviews accumulate with user count). Rating might play a role too: within this cluster, an app rated 4.5 might have more installs than one rated 4.0, if quality drives popularity to some extent. Content rating and genre could also influence it (for instance, certain genres might inherently have more users). A 45% R^2 indicates a moderate to good fit – there's still variability not explained (possibly marketing spend or external factors, which we don't have data for), but nearly half of the differences in install counts between apps in this group can be predicted from these features.

Top Global Cluster Model: $R^2 \approx$ **1.00** (essentially 100% of variance explained). This result needs to be interpreted with caution. With only ~32 data points (apps) in this cluster and a large number of predictors (remember that Genres alone introduces many dummy variables), the linear model is likely overfitting. In fact, an R^2 of 1.0 suggests the model fit the data points exactly. This can happen when you have as many (or more) parameters as data points or if one predictor is almost perfectly correlated with the outcome. In this case, it's plausible that **Reviews** is so strongly correlated with **Installs** for the top apps that the model basically uses reviews to predict installs nearly perfectly. Indeed, for the biggest apps, if one app has twice the number of reviews as another, it likely has roughly twice the installs, since most users of a hugely popular app leave a rating/review. Also, with many genre dummy variables, the model can pick up unique combinations per app. So while the model summary shows $R^2 = 1$ for this cluster, it doesn't mean we've found a magic formula; it means our model likely overfit the top apps data. Still, it's reasonable to deduce that among top global apps, the features we included (especially **Reviews count**) can explain nearly all the variance in installs – essentially, those apps are so uniformly massive that their differences in installs correspond directly to differences in user engagement (reviews).



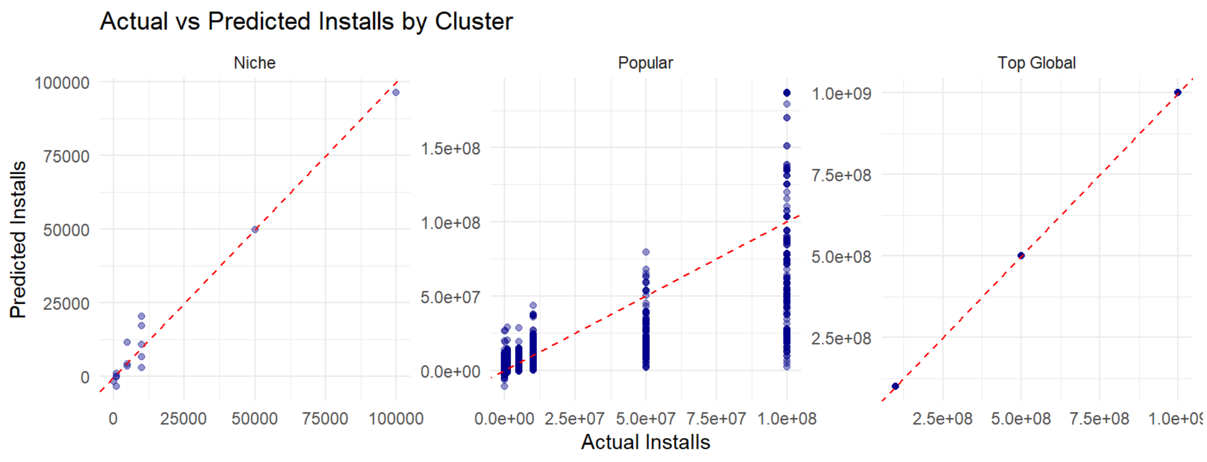
To visualize the model performance, we plotted **Actual vs. Predicted Installs** for each cluster. In these scatterplots (faceted by cluster), the closer the points lie to the diagonal line ($y = x$), the better the model's predictions:

For the **Top Global cluster**, the points lie almost exactly on the diagonal (again, due to the overfit/perfect prediction scenario). This is not surprising given the R^2 of ~1. The model basically memorized the installs of these few apps using the features.

For the **Popular cluster**, the points were fairly close to the diagonal, but with some deviation. We observed that the model tends to slightly under-predict the installs for the very top of this cluster (apps that are at the high end, approaching 100 million installs) and over-predict a bit for those at the lower end (a few million installs). This regression line effect is typical – the model pulls extreme values towards the average trend. Still, the predictions are in the right ballpark for most popular apps.

For the **Niche cluster**, the scatter of points around the diagonal was wider. Some apps with relatively low installs were predicted to have more (perhaps because they had surprisingly high ratings or other features), and some with higher installs were under-predicted. This indicates the model is less precise for niche apps, which matches the lower R^2. There may be many idiosyncratic factors in play for niche apps that a simple linear model using our variables can't capture.

```
> # ---------------------------------------------------------
> # STEP 13: Actual vs Predicted Installs Visualization
> # ---------------------------------------------------------
>
> # Create data frame for predictions
> prediction_plot_data <- data.frame()
>
> for (cluster in unique(df_clean$Cluster_Name)) {
+   cluster_data <- df_clean[df_clean$Cluster_Name == cluster, ]  # Get cluster data
+   model <- lm(Installs ~ Rating + Reviews + Size + Price + Content.Rating + Genres, data = cluster_data)  # Fit model
+
+   # Create prediction data frame
+   predictions <- data.frame(
+     Cluster = cluster,
+     Actual = cluster_data$Installs,
+     Predicted = predict(model, newdata = cluster_data)
+   )
+
+   prediction_plot_data <- rbind(prediction_plot_data, predictions)  # Append predictions
+ }
>
> # Plot actual vs predicted installs
> ggplot(prediction_plot_data, aes(x = Actual, y = Predicted)) +
+   geom_point(alpha = 0.4, color = "darkblue") +
+   geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
+   facet_wrap(~ Cluster, scales = "free") +
+   labs(title = "Actual vs Predicted Installs by Cluster",
+        x = "Actual Installs",
+        y = "Predicted Installs") +
+   theme_minimal()
```



In summary, the regression models show that **number of reviews is a strong predictor of installs** across all clusters (intuitively, an engaged user base yields more downloads). **App rating** likely has a positive but modest effect (a higher rating can help an app gain more users, but in many cases apps in each cluster have similarly high ratings so it's a smaller differentiator). **Price** has a negative relationship with installs (paid apps generally have fewer installs, which is why the niche cluster – containing most paid apps – has lower installs on average). **Content rating** and **genre** might help capture some category-specific popularity (e.g., perhaps "Everyone" rated apps reach broader audiences than "Teen" or "Mature" apps; certain genres like Communication or Social have inherently higher user numbers than niche genres). These nuanced effects are embedded in the model coefficients.

The key takeaway is that our models can reasonably predict an app's scale of success **once we know which cluster it belongs to**. This last point is crucial – if we tried to fit one model to all apps at once, it would likely perform poorly because the relationships between features and installs are not constant across these very different app segments. By segmenting first (clustering), we improved predictive power within each segment.

5. Preliminary Findings and Marketing Strategy

Cluster-wise Findings:

Combining the clustering results and regression insights, we can summarize each cluster's profile and what it implies:

Top Global Apps (Cluster: "Top Global"): These are the elite apps (e.g., top social media, popular global games, essential tools like Google's apps) that have on the order of hundreds of millions to a billion installs. They are all free and achieve widespread adoption. They maintain high ratings (~4.3–4.4 on average) despite massive user counts, indicating strong user satisfaction and quality. Users of these apps are extremely numerous and active (tens of millions of reviews). In the regression model, this cluster's installs were almost perfectly predicted by the number of reviews – essentially, these apps all have such a broad appeal that user engagement scales directly with user count. **Implication:** These apps represent a mass audience reach. Growth for such apps often comes from global expansion or tapping remaining markets, as they may already saturate many regions.

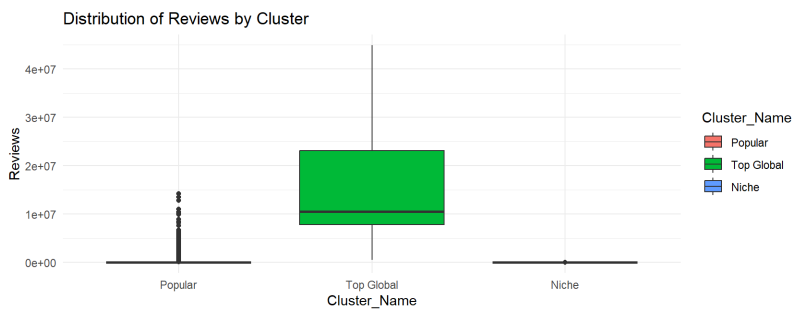
Popular Apps (Cluster: "Popular"): These apps have achieved millions or tens of millions of downloads. They are success stories in their categories – for instance, a hit mobile game, a widely used finance app, or a regional social network. This cluster makes up around 20% of the apps in our data. They are almost exclusively free as well; free pricing has likely enabled them to acquire a large user base. Their average rating is very high (~4.3), so quality is on par with the top apps. They have around a million reviews on average, showing strong user engagement. Our model explained ~45% of the install variance here; factors like having more reviews and perhaps certain genres (games, communication) help drive them into this tier. **Implication:** These apps still have room to grow – some may eventually graduate to the "Top Global" tier with the right push. They are well-positioned in the market and have proven product-market fit.

Niche Apps (Cluster: "Niche"): This cluster contains the majority (~80%) of apps, which are those with relatively small user bases (from a handful up to a few hundred thousand installs, maybe reaching low millions at most). Many of these are newer apps, or apps targeting specific interests or local markets, or simply apps that haven't broken out in popularity. A notable portion of these are paid apps or have upfront costs, which naturally limits their installs. The average rating is slightly lower (~4.1) with more variance, suggesting some niche apps struggle with user satisfaction or have polarizing feedback. Many niche apps might not have the marketing or virality to accumulate huge download counts. The regression model could only explain ~39% of the variance here, indicating a lot of unpredictable factors – for example, two niche apps might have similar ratings and features, but one got featured or went viral and the other did not, leading to big install differences that our model can't fully account for. **Implication:** Niche apps face the toughest challenge to grow; factors outside our dataset like marketing, network effects, or uniqueness of idea likely play a big role in determining if they stay niche or become popular.

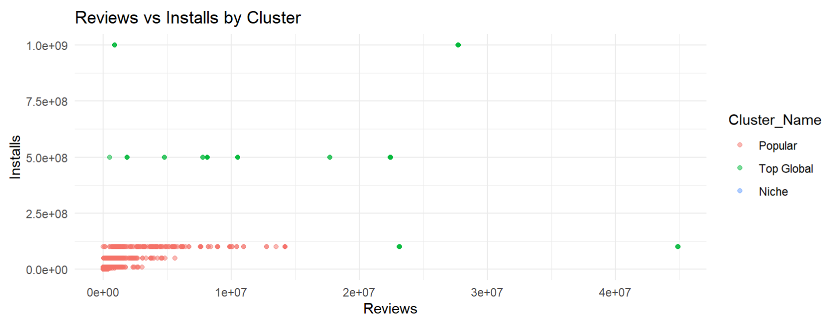
Visual Insights:

During analysis, we visualized some comparisons across clusters:

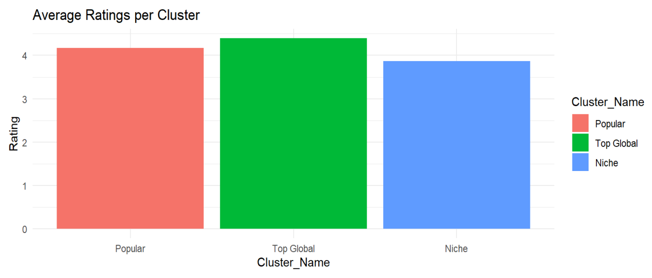
A **boxplot of Reviews by Cluster** showed that the Top Global apps have an extremely high number of reviews (the entire range of their boxplot was far above the medians of other clusters). Popular apps had a wide range but generally one order of magnitude lower than Top Global. Niche apps had relatively few reviews (many outliers in the single-thousands). This visualization reinforced that *user engagement corresponds with cluster popularity*.

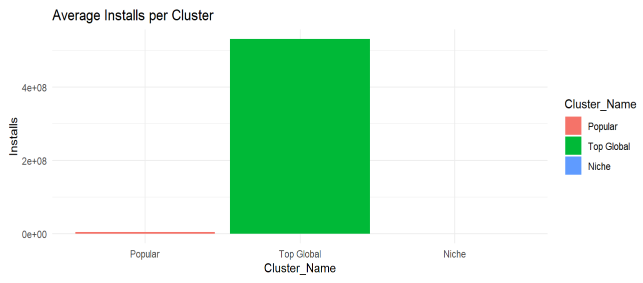


A **scatter plot of Reviews vs Installs by Cluster** essentially looked like three clouds of points separated along the scale: the niche cluster points occupy the lower-left (low reviews, low installs), the popular cluster in the middle (moderate to high reviews correlating with millions of installs), and the top global cluster in the top-right (very high on both axes). All clusters showed a positive correlation between reviews and installs (as expected, more installs generally lead to more reviews). The trend line for each cluster would have a high slope initially but taper off (perhaps indicating diminishing returns of reviews to installs for the largest apps).



Bar charts of **average installs and ratings by cluster** succinctly showed the differences: average installs for Top Global dwarfed the other two (graphically emphasizing the logarithmic gap), while average rating differences were minor (all clusters have good ratings, with popular and top global slightly higher than niche).





Marketing Strategy Recommendations:

Given these findings, a **targeted marketing strategy** can be developed for each cluster of apps. The idea is to allocate advertising and promotion budgets in line with the potential reach and needs of apps in each segment. We also assume some dollar values for illustration:

Top Global Apps – Mass Marketing (Budget: High, e.g. $500k and up): Apps in this cluster already have huge penetration, but their market is the entire global smartphone user base. Marketing for these apps is about maintaining dominance and capturing the remaining users who haven't yet installed the app. Strategies could include **broad-reach advertising** such as TV commercials, large-scale online campaigns, sponsorships, or pre-install deals with phone manufacturers. The assumed budget would be very high – on the order of hundreds of thousands to millions of dollars – similar to how one would market a mainstream consumer product. For example, a top global game might spend $1M on a Super Bowl ad or on continuous global Facebook/Google ads campaigns to ensure it remains on top. The rationale is that each additional percentage point of market penetration could yield millions of new users, so high spend is justified. Our regression model suggests that these apps mostly grow with user engagement (reviews) – which likely comes from network effects and word-of-mouth at this stage – so marketing might focus on encouraging existing users to refer others (referral incentives) and broad brand awareness.

Popular Apps – Aggressive Digital Marketing (Budget: Moderate, e.g. $50k–$200k): Apps in the popular cluster have proven appeal and a large user base, but they are still not used by everyone. For these apps, well-crafted marketing campaigns can potentially propel them into the top tier. The strategy should focus on **online and targeted advertising**: for example, social media ads, influencer partnerships, app store feature campaigns, and perhaps regional TV ads if applicable. A moderate budget in the tens of thousands of dollars per month might be allocated to acquire new users. Because these apps are free and appeal to broad audiences, customer acquisition cost (CAC) can be justified by the lifetime value of having millions more users. The marketing messaging might emphasize the app's high user ratings and popularity ("Join the 10 million people already using X!") to leverage social proof. According to our analysis, these apps benefit from exposure – more installs lead to more reviews and visibility. So investing in advertising can start a virtuous cycle for an app on the cusp of global popularity.

Niche Apps – Targeted Online Marketing & Organic Growth (Budget: Low, e.g. $5k–$20k or sweat equity): Most apps in the niche cluster have limited budgets or niche audiences, so cost-effective strategies are crucial. For these apps, **focused online advertising** (like Google Ads or Facebook Ads targeted to the specific demographic interested in the app) can help in acquiring the *right* users, even if the volume is not huge. Spending might be relatively low and incremental – for example, a few thousand dollars in online ads around the time of launch or a new feature release. Additionally, **content marketing, community engagement, and partnerships** are key for niche apps. Since our model shows a lot of variance unaccounted for, we infer that creative marketing and unique hooks can make a big difference. Niche app developers should seek reviews and ratings (e.g., encourage satisfied users to leave reviews) because a higher rating can help them stand out in their small category. They might also attempt to get featured on the app store or in relevant blogs/magazines, which can provide a boost without huge ad spend. Essentially, for niche apps every marketing dollar needs to be spent where it counts: reaching the small segment of users who would love the app, rather than broad audiences. TV ads would be overkill and inefficient here – the focus is on **online presence, SEO, and word-of-mouth.**

Connecting Predictions to Strategy:

The regression models allow us to predict how an app might perform in terms of installs within its cluster if certain features change. For example, if a niche app improves its rating from 3.5 to 4.5 and garners more reviews (say through a campaign that doubles its user reviews), the model would predict a significant uptick in installs – perhaps moving it closer to the threshold of the popular cluster. That insight can guide the marketing: the niche app developer might invest in improving user satisfaction (to boost ratings) and in small-scale promotions to get more users (thus more reviews), rather than spending on mass advertising which might be premature.

For a popular app, the model might show that even within that cluster, doubling the number of reviews (which correlates with more users) could push it toward the top cluster's install numbers. This suggests heavy marketing to accelerate user adoption could pay off. With a moderate budget, the app could aim to climb into the top global ranks.

For top global apps, the model is saturated, but essentially these apps need to be ubiquitous – marketing here is about branding and retention as much as acquisition. They might focus on **maintaining high ratings** (through quality updates) because a slip in user satisfaction could open room for competitors.

Conclusion:

In this analysis, we successfully **clustered apps** into three meaningful segments and characterized each cluster in detail. We found that the app market has a tiny set of globally dominant apps, a subset of very popular apps, and a long tail of niche apps. By building cluster-specific models, we could **predict installs** with reasonable accuracy and see how factors like reviews, rating, and price influence an app's success within each segment.

Using these insights, we outlined a **marketing strategy** tailored to each cluster's needs and potential:

The **Top Global** apps warrant high-budget, mass marketing strategies (akin to advertising a blockbuster product) to maintain their lead.

The **Popular** apps should invest in aggressive online marketing and growth strategies to try to break into the global tier.

The **Niche** apps should focus on targeted, cost-effective marketing and improving product metrics (ratings, user engagement) to incrementally grow their user base.

This cluster-driven approach ensures that marketing resources are allocated efficiently. A one-size-fits-all strategy would not work equally for a niche app and a global app – but by recognizing the cluster an app falls into, a business can set realistic goals and strategies. For instance, a niche app developer now understands they shouldn't measure success by "reaching 100 million installs" right away; instead, the first goal is to move into the Popular cluster range (i.e., break the 1 million install mark) through targeted improvements and marketing. Conversely, a popular app company knows that to join the billion-download club, they need a massive push and perhaps broad partnerships.

Overall, this analysis demonstrates the power of combining unsupervised learning (clustering) to segment a market with supervised learning (regression) to make predictions and inform decisions within those segments. It provides actionable insights for app developers and marketers on how to interpret their app's performance relative to others and where to focus their growth efforts.