Polytechnic University of Catalonia

Multivariate Analysis

D4. Final delivery

Authors

Campeny, Eloi
Chriki, Fatima Zohra
Dai, Zhongkai
González, Victor
Moure, Ximena
Xu, Ange

Teachers

Conti, Dante
Gibert, Karina
Ramírez, Sergi

December ◇ QT - 2022/23

Index

1. Motivation and problem definition

Google Play store is a digital distribution platform for mobile applications for devices with Android operating system, as well as an online store.
There are more than 2.5 million apps on the store and the service is used by millions of users on a daily basis. What's more, every day more than 2500 apps are added to it.

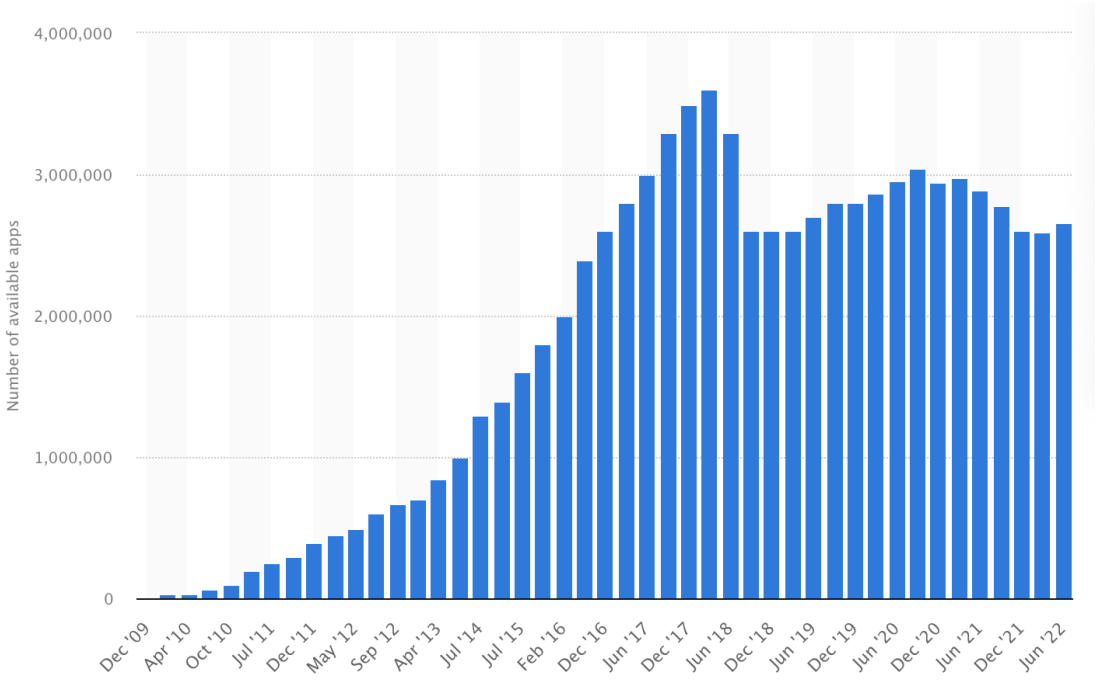


Figure 1: Number of available applications in the Google Play Store

The number of app downloads has been increasing throughout the years. In 2021, users worldwide downloaded approximately 111 billion mobile apps.
A vast majority of apps in the store are free to download so the monetization strategy may vary. This is why an application's success is often measured by the number of installations and the user reviews it has.
An application rating is based on voluntary feedback from users. This can lead to biased ratings due to insufficient or missing votes.
The aim of this study is to analyze which factors can influence the number of downloads and the rating of an app. This can potentially help developers to better understand the mobile application market.

2. Data source presentation

The dataset used in this project is taken from Kaggle and was collected in June 2021. It is publicly accessible and can be retrieved from the following url: https://www.kaggle.com/datasets/gauthamp10/google-playstore-apps

As the original dataset was quite large, containing 2312944 observations, we decided to reduce it to 20000 observations. The final dataset, as well as the script used to obtain it, can be found at the following url: https://drive.google.com/drive/folders/1B-6fwHjjXAG-JSsAIGmQOihYE80M2_3m?usp=sharing

3. Data structure and metadata description

3.1 Description of initial data matrix

The dataset contains 20000 entries and 24 variables, out of which 5 are numerical, 4 are binary and 15 are categorical, which accounts for 480000 data entries. Some of the categorical variables were transformed during the preprocessing phase into numerical ones (Installs, Size, Minimum.Android) in order to fulfill the requirement of having at least 7 numerical variables.

In total there are 10911 missing values in the dataset for a total of 480000 values. Therefore, 2.27 % of the values are missing.

Numerical variables

- Rating 344 missing 1.72%
- Rating.Count 344 missing 1.72%
- Minimum.Installs 1 missing 0.005%
- Maximum.Installs 0 missing 0%
- Price 0 missing 0%

Binary variables

- Free 0 missing 0%
- Ad.Supported 0 missing 0%
- In.App.Purchases 0 missing 0%
- Editors.Choice 0 missing 0%

Qualitative variables

- App.Name 0 missing 0%
- App.Id 0 missing 0%
- Category 0 missing 0%

● Installs	1 missing	0.005%
● Currency	1 missing	0.005%
● Size	0 missing	0%
● Minimum.Android	49 missing	0.245%
● Developer.Id	0 missing	0%
● Developer.Website	6097 missing	30.485%
● Developer.Email	0 missing	0%
● Released	750 missing	3.75%
● Last.Updated	0 missing	0%
● Content.Rating	0 missing	0%
● Privacy.Policy	3324 missing	16.62%
● Scraped.Time	0 missing	0%

Furthermore, the dataset contains some variables that do not provide useful information, such as the developer email and website, app identification, privacy policy.

3.2 Metadata table

URLs
- Original source
- Reduced dataset

Inclusion criteria: Google Play Store Apps collected in the month of June 2021.

No. of rows: 20000
No. of columns: 24

Variable	Modalities	Meaning	Type	Measuring unit	Missing code	Measuring procedure	Range	Role
App.Name		Name of the app	Categorical nominal (factor)					Explanatory
App.Id		Package name	Categorical nominal (factor)					Explanatory
Category	48 modalities in total	App category	Categorical nominal (factor)					Explanatory
Rating		Average rating	Numerical continuous (numeric)	Star			[0, 5]	Response
Rating.count		Number of ratings	Numerical discrete (integer)	Rating				Explanatory
Installs		Approximate install count	Categorical nominal (factor)					Explanatory
Minimum.Installs		Approximate minimum app install count	Numerical discrete (integer)	Install				Explanatory
Maximum.Installs		Approximate maximum app install count	Numerical discrete (integer)	Install				Explanatory
Free	True, False	Whether app is Free or Paid	Categorical binary [True, False] (factor)					Explanatory
Price		App price	Numerical continuous (numeric)	Currency				Explanatory
Currency		App currency	Categorical nominal (factor)					Explanatory
Size		Size of application	Categorical					Explanatory

		package	nominal (factor)					
Minimum.Android	32 in total	Minimum android version supported	Categorical nominal (factor)					Explanatory
Developer.Id		Developer Id in Google Playstore	Categorical nominal (factor)					Explanatory
Developer.Website		Website of the developer	Categorical nominal (factor)					Explanatory
Developer.Email		Email-id of developer	Categorical nominal (factor)					Explanatory
Released		App launch date on Google Playstore	Temporal (Date)					Explanatory
Last.Updated		Last app update date	Temporal (Date)					Explanatory
Content.Rating	Everyone, Teen, Mature 17+, Everyone 10+, Adults only 18+	Maturity level of app	Categorical nominal (factor)					Explanatory
Privacy.Policy		Privacy policy from developer	Categorical nominal (factor)					Explanatory
Ad.Supported	True, False	Ad support in app	Categorical binary [True, False] (factor)					Explanatory
In.App.Purchases	True, False	In-App purchases in app	Categorical binary [True, False] (factor)					Explanatory
Editors.Choice	True, False	Whether rated as Editor Choice	Categorical binary [True, False] (factor)					Explanatory
Scraped.Time		Scraped date-time in GMT	Temporal (Date)					Explanatory

3.3 Final scope of study

After a thoughtful consideration of which variables were adequate for our study, we decided to discard the following variables: `App.Id`, `Developer.Id`, `Developer.Website`, `Developer.Email`, `Installs` and `Privacy.Policy`. We consider that these variables do not provide any insight into what we are trying to accomplish. This was made during preprocessing.
Additionally, we created new variables and transformed other ones to numerical. All of this is explained in detail in the preprocessing section.
The cleansed dataset contains 12 variables, out of which 7 are numerical, 3 are categorical and 2 are binary.

4. Preprocessing

This section includes all the different preprocessing tasks carried out in this project together with the results obtained. Several preprocessing methods were applied in order to improve the quality of the dataset, which are:

- Feature selection
- Feature derivation
- Variable transformations
- Segmentation of population
- Identifying and handling missing data
- Detection of outliers
  - Univariate outliers
  - Multivariate outliers

In the following subsections, we present in detail each of the preprocessing methods that were carried out.

4.1 Feature selection

The aim of feature selection is filtering the uninteresting variables and to remove the noise in the dataset. After analyzing the `Playstore` dataset, many irrelevant variables were detected. The criteria followed to apply feature selection is the following:

- **Variables with unique value for each observation:** there were detected five variables in the dataset, which are `App.Id`, `Developer.Id`, `Developer.Website`, `Developer.Email` and `Privacy.Policy`. All these features were removed.
- **Variables with unique value for all observations (Constants):** one logical variable detected in the dataset, called `Editors.Choice`, was removed because 98% of the observations had the same value.
- **Variables highly correlated with other variables:** the dataset includes two columns with the same values, one called `Installs` and the other one `Minimum.Installs`, the difference between these columns is that the first one is categorical and the second one numerical. Since `Minimum.Installs` includes the same values as `Installs`, this last one was removed. After studying the correlation between the numerical data, the results show that `Minimum.Installs` is highly correlated with `Maximum.Installs` with a 0.98 correlation coefficient. In order to decide which column to keep and which one to remove, we studied the distribution of each column.

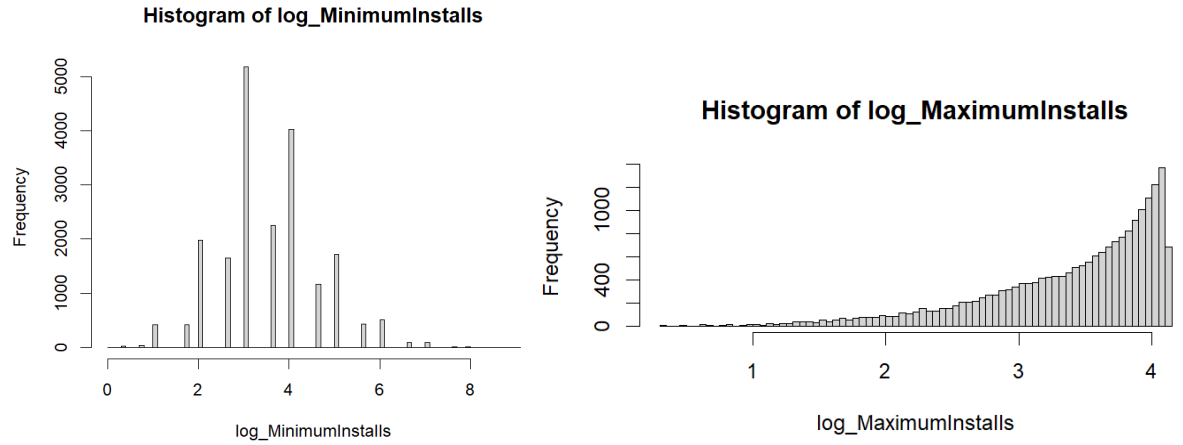


Figure 2: Distribution comparison

Comparing the distributions shown in Figure 2, `Minimum.Installs` includes dispersed values and distant ranges while `Maximum.Installs` seems to follow an exponential distribution. Therefore, `Minimum.Installs` was removed and `Maximum.Installs` was kept and named as `Installs`.

The unnecessary columns mentioned before were removed from the dataset, therefore, the dimension of the dataset was reduced from 24 to 15 columns. Throughout the following preprocessing process, the dimension of the dataset will be further reduced due to the transformation of some columns.

4.2 Feature derivation

We carried out some variable derivations in order to have more numerical variables in the dataset.

First, we decided to derive a new numerical variable called `DaysLastUpdate` from the variables `Scraped.Time` and `Last.Updated`. This new variable indicates the number of days passed since the last app update until the time it was scraped.

Afterwards, we derived from `Scraped.Time` and `Released` a new numerical variable called `ReleasedDays`. This other variable indicates the number of days since the app was released until the scrapped time.

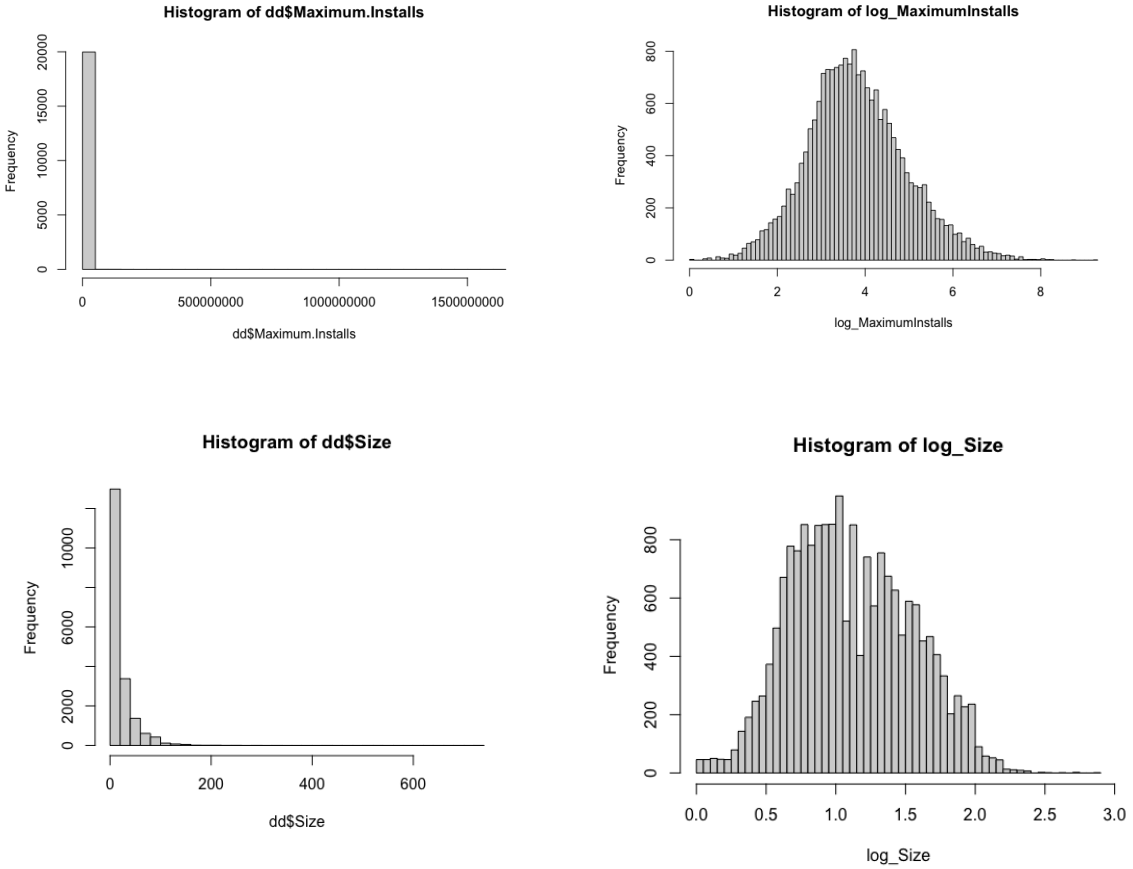
After the derivation of these variables, it did not make any sense to keep the variables `Released`, `Last.Updated` and `Scraped.Time` so we decided to remove them from the dataset.

Finally, we also researched whether an app's name has any impact on its visibility in the `Appstore`. We concluded that it certainly does. Therefore, we decided to do one more derivation which was to derive a new numeric variable called `AppNameLen` from the variable `App.Name` which accounts for the length of the name of the app.

4.3 Variable transformations

After analyzing the dataset we found that the variable called `Size` was a categorical one and it had different measures. The Size of data was in GB, MB and KB. So we converted all the data to MB and then we transformed it to a numerical variable.

Another decision we made, after looking at the histograms in Figure 3 to see the distribution of the values in the dataset, was to apply a logarithmic transformation to the variables `Size`, `Installs` and `Rating.Count`. The reason behind this is that the large range between their values could be a problem for some algorithms and that it is easier to visualize the data.



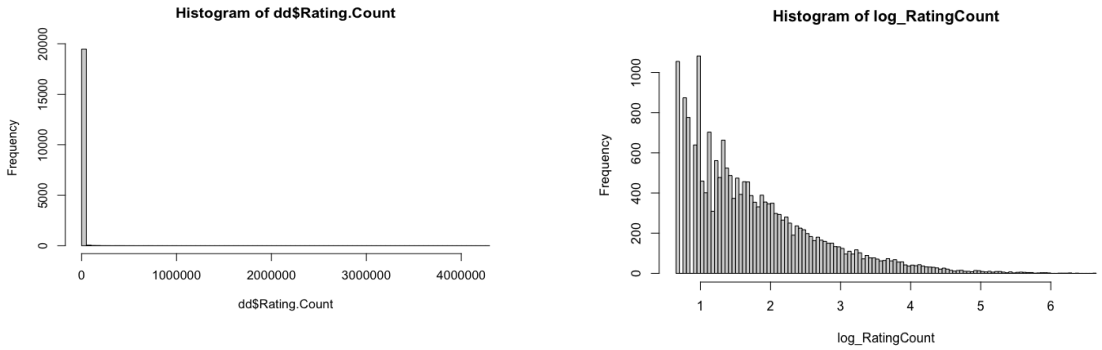


Figure 3: Histograms of the variables before and after applying a logarithmic transformation

Finally, the last transformation we did was to reduce the modalities of the categorical variables `Category` and `Minimum.Android`.
`Category` had 48 levels and we group them in 6 levels and `Minimum.Android` had 32 levels and we group them in 9 levels.

4.4 Segmentation of population

To assess if we should keep free and paid apps we performed a Kolmogrov-Smirnov test (KS test). This test is used to test whether two samples follow the same distribution or not. If they do not follow the same distribution they are two different populations.

In order to do this, we separated the data into two groups: one which contained the paid apps and another one with the free ones. After this, we performed the KS test for each numerical variable using both datasets.

As depicted in Figure 4, for almost every numerical variable, the KS test gave us a p-value less than 0.05. For this reason we concluded that there is more than one population (free and paid apps) and therefore we decided to analyze only free apps as it has the biggest population.

Following this decision, we eliminated from the dataset the two variables called `Currency` and `Price` as we no longer need them since we are going to analyze only free apps.

```
        Two-sample Kolmogorov-Smirnov test

data:  pay$Rating and free$Rating
D = 0.061604, p-value = 0.08275
alternative hypothesis: two-sided
```

```
        Two-sample Kolmogorov-Smirnov test

data:  pay$Rating.Count and free$Rating.Count
D = 0.12337, p-value = 0.000005664
alternative hypothesis: two-sided
```

```
        Two-sample Kolmogorov-Smirnov test

data:  pay$Minimum.Installs and free$Minimum.Installs
D = 0.20459, p-value = 0.0000000000000009992
alternative hypothesis: two-sided
```

```
        Two-sample Kolmogorov-Smirnov test

data:  pay$Maximum.Installs and free$Maximum.Installs
D = 0.2214, p-value < 0.00000000000000022
alternative hypothesis: two-sided
```

```
        Two-sample Kolmogorov-Smirnov test

data:  pay$DaysLastUpdate and free$DaysLastUpdate
D = 0.19473, p-value = 0.00000000000002776
alternative hypothesis: two-sided
```

```
        Two-sample Kolmogorov-Smirnov test

data:  pay$Size and free$Size
D = 0.11604, p-value = 0.00005687
alternative hypothesis: two-sided
```

Figure 4: Test results for Kolmogorov-Smirnov Test

4.5 Identifying and handling of missing values

To obtain clean data it is necessary to deal with the missing values that could be present in it, so the first step for the treatment of missing data would be to identify the variables affected. Figure 5 shows the variables affected with missing values and the amount they have.

$Continuous

	label	var_type	n	missing_n	missing_percent	mean
Rating	Rating	<dbl>	19227	342	1.7	4.1
Rating.Count	Rating.Count	<int>	19227	342	1.7	3342.7
Size	Size	<dbl>	18646	923	4.7	20.3
DaysLastUpdate	DaysLastUpdate	<dbl>	19569	0	0.0	562.5
ReleasedDays	ReleasedDays	<dbl>	18829	740	3.8	1195.8
AppNameLen	AppNameLen	<int>	19569	0	0.0	23.1
Installs	Installs	<int>	19569	0	0.0	431313.5

$Categorical

	label	var_type	n	missing_n	missing_percent	levels_n
Category	Category	<fct>	19569	0	0.0	6
Minimum.Android	Minimum.Android	<fct>	18921	648	3.3	8
Content.Rating	Content.Rating	<fct>	19569	0	0.0	6
Ad.Supported	Ad.Supported	<fct>	19569	0	0.0	2
In.App.Purchases	In.App.Purchases	<fct>	19569	0	0.0	2

Figure 5: Variables with missing values

The next step would be to determine the type of missing data present. The little test was performed to determine if the data was missing completely at random, but as the results obtained in Figure 6 shows, that was not the case, so it was necessary to determine if the data was random missing or not.

A tibble: 1 × 4

	statistic	df	p.value	missing.patterns
	<dbl>	<dbl>	<dbl>	<int>
1	551.	29	0	7

Figure 6: Result of the Little test

Analyzing the Figures 7, 8, 9 and 10, the type of missing data can be determined.

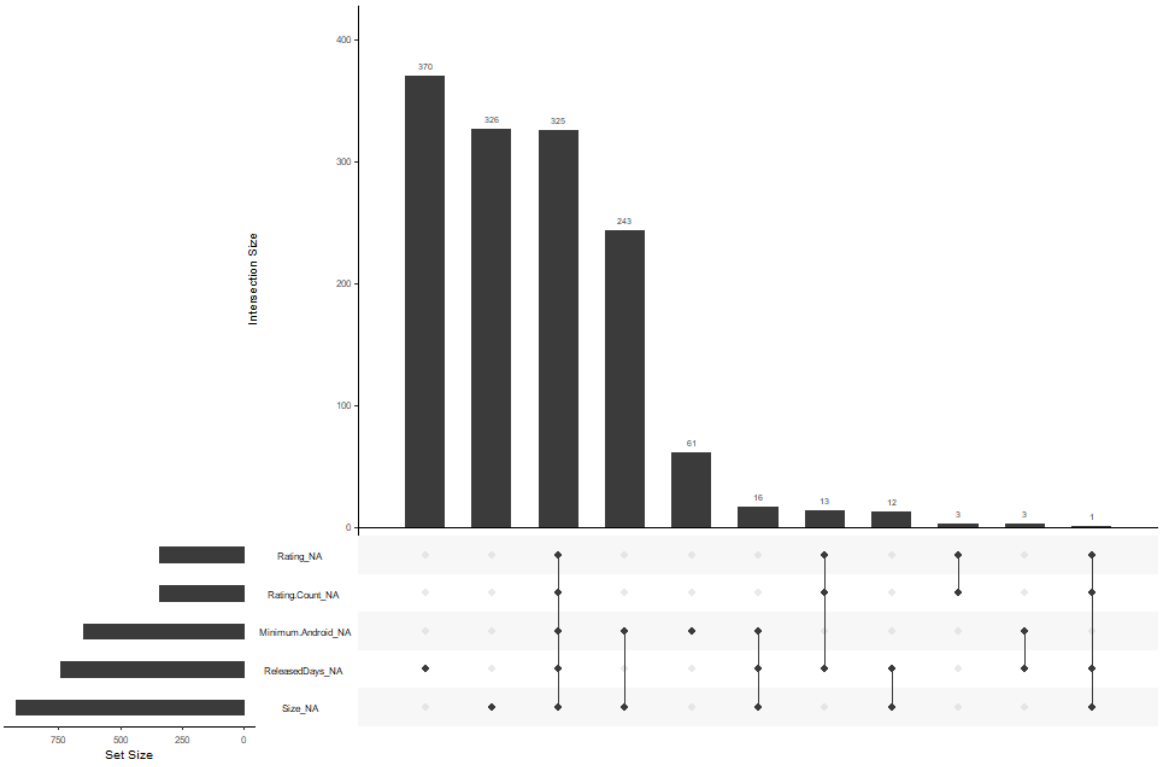


Figure 7: Amount of missing values for every pattern present in the data

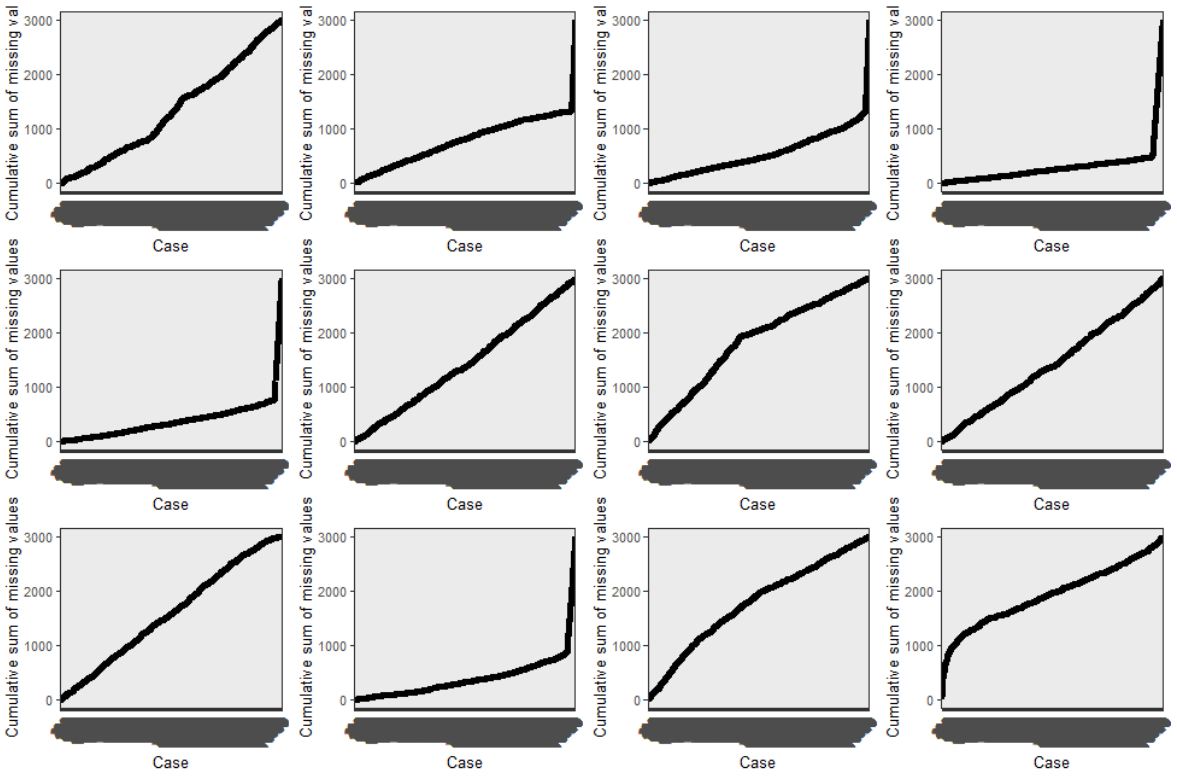


Figure 8: The graphs count the accumulative missing values for each variable ordered in ascending order: Category, Rating, Rating.Count, Size, Minimum.Android, Content.Rating, Ad.Supported, In.App.Purchases, DaysLastUpdated, ReleasedDays, AppNameLen, Installs

Missing Data analysis				
Rating		Not missing	Missing	p-value
Content.Rating	only 18+	1	0	0297
	Everyone	16591	286	
	Everyone +10	332	10	
	Mature 17+	567	7	
	Teen	1735	39	
	Unrated	1	0	
In.App.Purchases	FALSE	16874	301	0.955
	TRUE	2353	41	
Installs	Mean	438640.9	19371	0.545
Rating.Count		Not missing	Missing	p-value
Content.Rating	only 18+	1	0	0.297
	Everyone	16591	286	
	Everyone +10	332	10	
	Mature 17+	567	7	
	Teen	1735	39	
	Unrated	1	0	
In.App.Purchases	FALSE	16874	301	0.955
	TRUE	2353	41	

Installs	Mean	438640.9	19371	0.545
Size		Not missing	Missing	p-value
Installs	Mean	399106.2	1081950.9	0.111
Minimum.Android		Not missing	Missing	p-value
Content.Rating	only 18+	1	0	0.511
	Everyone	16314	563	
	Everyone +10	328	14	
	Mature 17+	563	11	
	Teen	1714	60	
	Unrated	1	0	
Installs	Mean	399106.2	108195.,9	0.11
ReleasedDays		Not missing	Missing	p-value
Content.Rating	only 18+	1	0	0,833
	Everyone	16245	632	
	Everyone +10	327	15	
	Mature 17+	556	18	
	Teen	1699	75	
	Unrated	1	0	
Installs	Mean	429944.8	466139.5	0.939

Figure 9: Missing data analysis

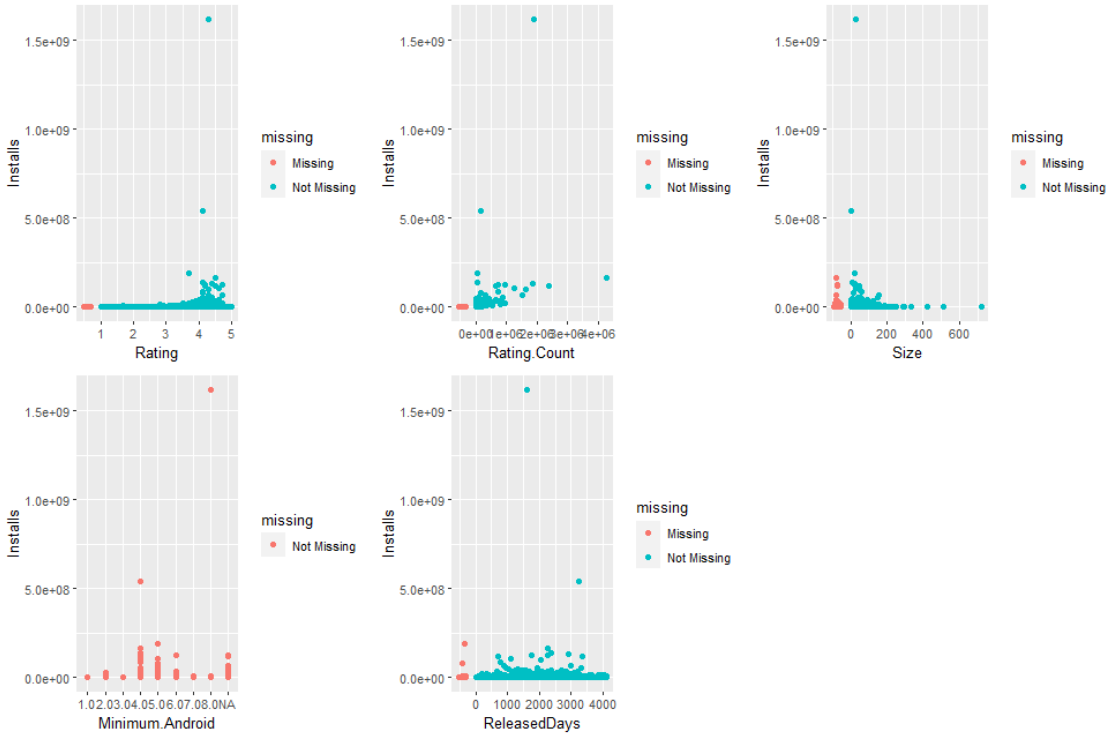


Figure 10: Missing values for every variable with missing data depending of number of installs

It can be appreciated that all the missing values have some dependency with the `Installs` variable, because for low values of `Installs` there is a concentration of missing values, mostly due to the concentration of rows following the pattern of 5 missing values per row. Because of this pattern of missing data can be explained by the number of installs (observed variable), having much less information for low installs, making the data to be missing at random.

To treat the missing values first the data is separated into numerical and categorical, then to perform the numerical imputation the MICE method is applied. In Figure 11 the frequency of the variables before and after the imputation can be observed.

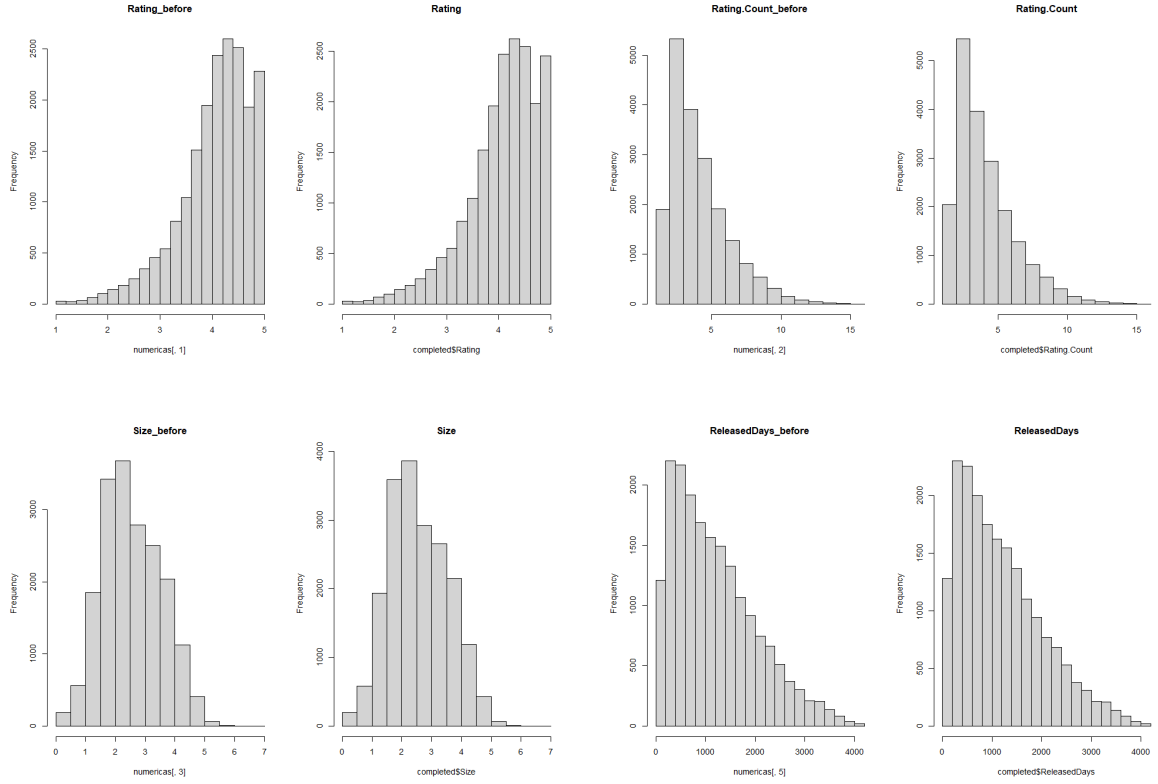


Figure 11: Histograms of Rating, Rating.Count, Size and ReleasedDays before and after the imputation

For the categorical values the MICE method is used and once all the data has been imputed then both tables are rejoined. Figure 11 shows the comparison before and after the imputation.

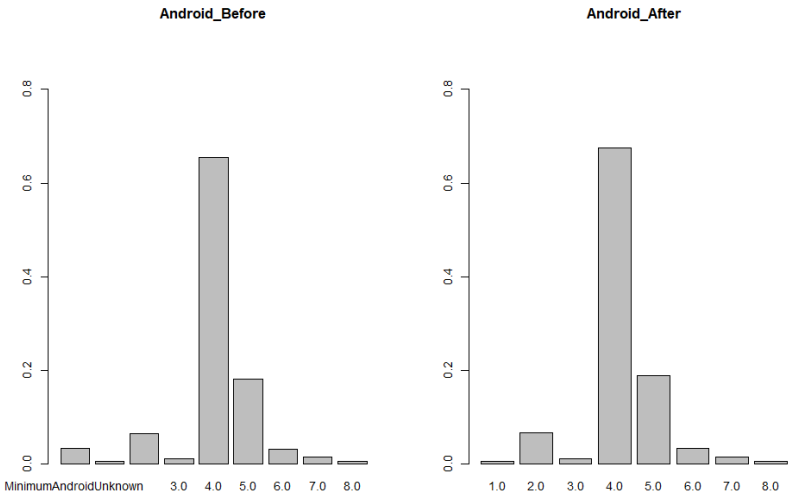


Figure 12. Comparison of `Minimum.Android` before and after de imputation

All the imputed variables have the same distribution after the imputation, which implies that the imputation has been done correctly.

4.6 Univariate detection of outliers

The aim of this section is to detect univariate outliers on numerical features and decide if it is better to keep outliers or just remove them from the main dataset and store them into the outlier dataset for further analysis.

Before moving to the analysis, it is better to point out that we only analyze the variables that we did not remove before (eg. `Minimum.Installs`) and if the variable was transformed we analyze its transformation.

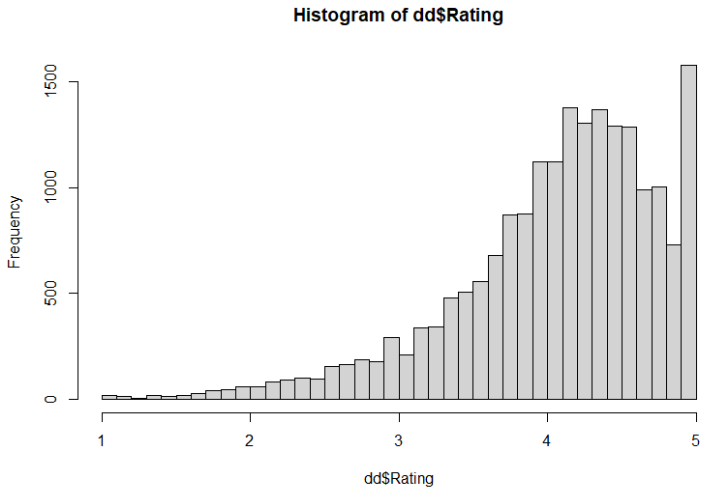


Figure 13: Histogram of `Rating`

First, we analyze the target value `Rating`, as the Figure 13 shows, `Rating` has a distribution that looks like an χ^2 except for the apps that are rated with 5. In this value there are more individuals than there should be if `Rating` follows this type of distribution.

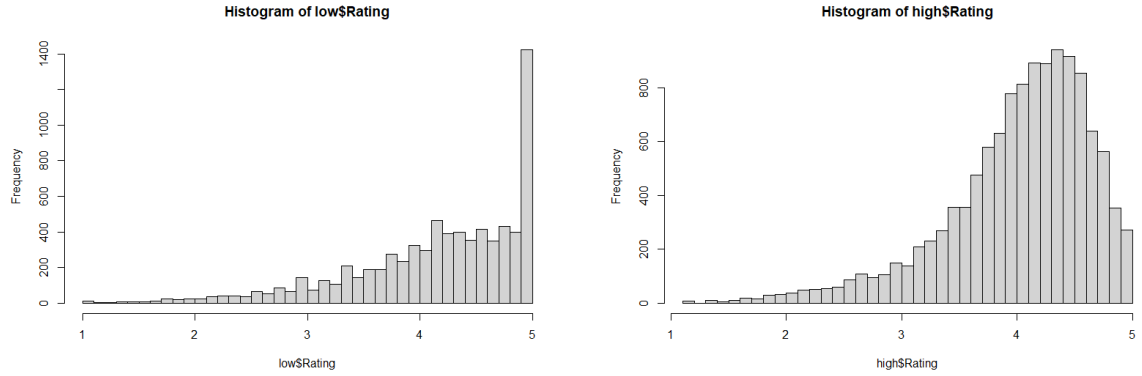


Figure 14: Histogram of Low Rating (left) and High Rating (Right)

We try to find out what was the reason behind this unusual behavior. We discovered that the apps with less than 20 votes have a completely different distribution, we checked this visually (Figure 14) and with Kolmogorov–Smirnov test. So we decided to separate these two populations, for our analysis we kept the apps with more or equal than 20 votes and the others were moved into the outlier dataset.

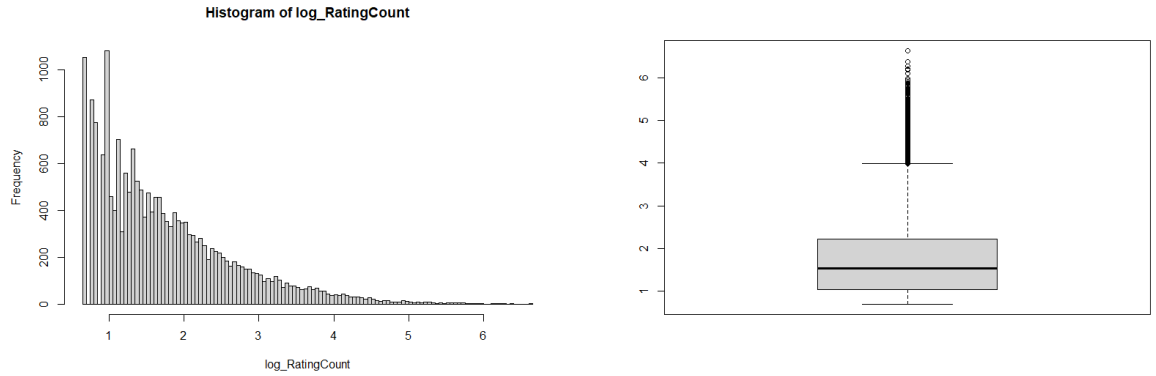


Figure 15: Histogram and box plot of the logarithm of `Rating.Count`

For analyzing the logarithm of the `Rating.Count`, we plotted an histogram and a box plot (Figure 15). In the first one we can see that it follows an exponential distribution, the only remarkable thing about the histogram is that it has some hollows but probably done by artifacts. Box plot has more interesting information, the first one is the tail that sticks out of the IQR method. This can be outliers, but it is important to remember that we have a large dataset of 20000 individuals. Since we

have this huge amount of individuals and they are close together, we decided to keep them. However, the second interesting thing is that on top of the tail there is a lonely individual. In this case we consider this individual as an outlier, and we changed its value for a missing value.

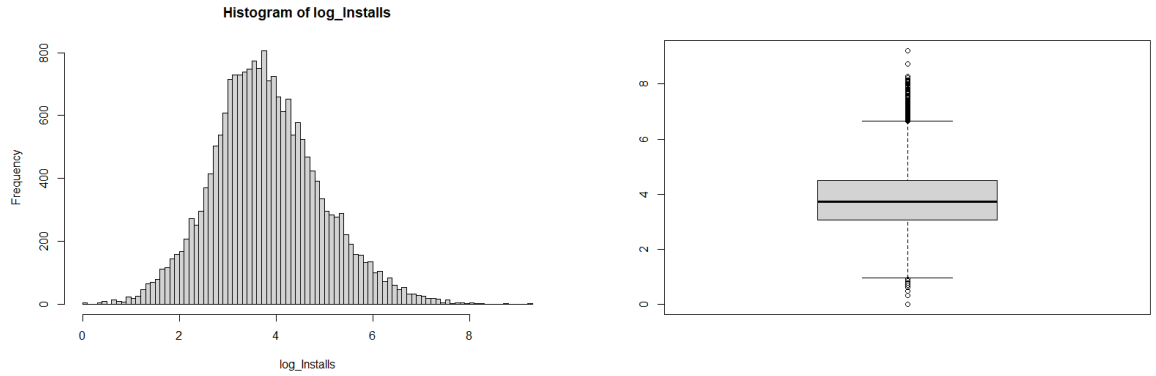


Figure 16: Histogram and box plot of the logarithm of `Installs`

In the analysis of the logarithm of the `Install`, we plotted a histogram and box plot (Figure 16). In the first one we can see that it follows a normal distribution, however the plot is not centered, this gives some clues about some outlier presence. Box plot has more interesting information, like the tails that stick out of the IQR method. This is the same case as the analysis of the logarithm of `Rating.Count`. Moreover, we have the same lonely individuals, in this case we have two on top and one on the bottom. We follow the same procedure as the case of the logarithm of `Rating.Count`, we erase the values and put a missing.

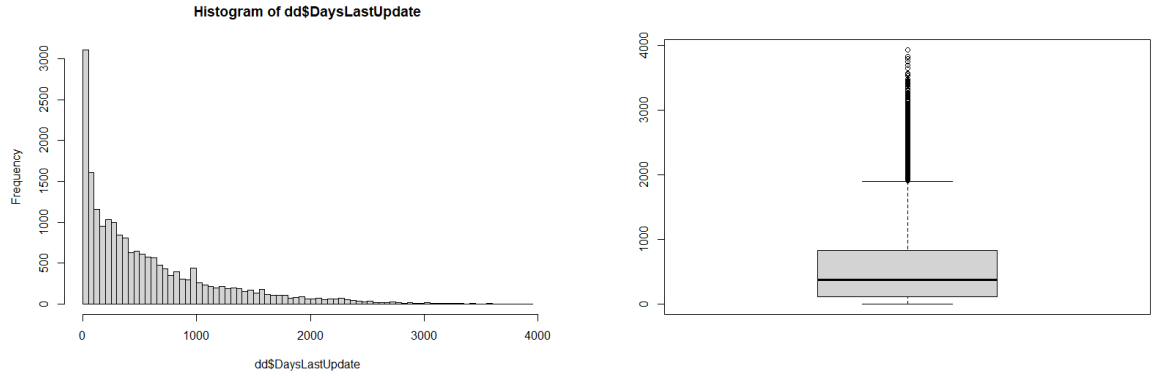


Figure 17: Histogram and box plot of `DaysLastUpdate`

In the analysis of `DaysLastUpdate`, we plotted a histogram and a box plot (Figure 17). In the first one, we can observe that it follows an exponential distribution. Box plot has the same tail that we have seen in other variables. Like in the other cases we kept the tail, however, in this case we did not erase the top value because the individuals are closer than in the other cases.

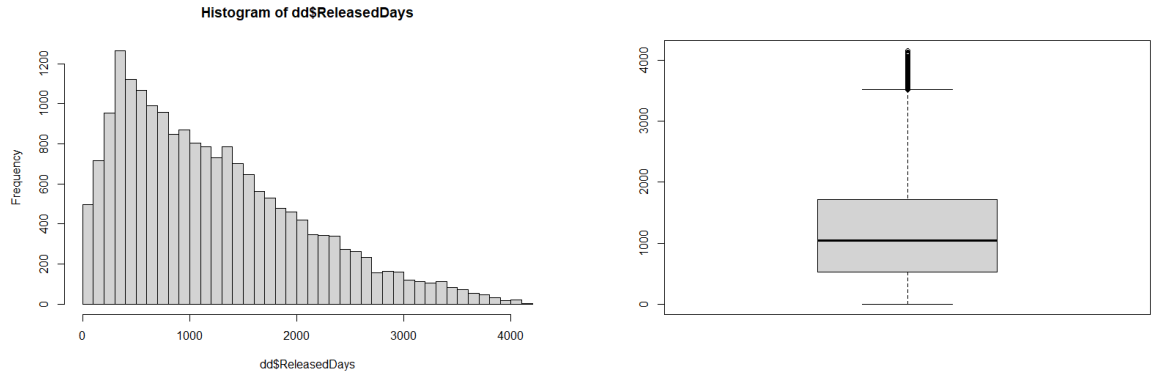


Figure 18: Histogram and box plot of `ReleasedDays`

The last variable that we analyze is `ReleasedDays`, as we have done with the other variables we make a histogram and box plot (Figure 18). The histogram shows an unusual distribution that does not look like any well known distribution. In addition, the box plot has a top tail like many other variables, so we kept these individuals on the main dataset.

4.7 Detection of multivariate outliers

In order to detect multivariate outliers we use the Mahalanobis distance. We use the r function of Mahalanobis distances for detecting and moving these individuals from the main dataset to the outliers dataset.

5. Basic initial descriptive statistics of preprocessed variables and conclusions

5.1 Univariate analysis

In this section, our cleaned and preprocessed dataset will be visualized by means of box plots, histograms and pie charts. This will help us to understand how the variables are distributed.

As a first step, we started by simply visualizing some basic information like mean, median, min, max for the numeric variables. We didn't notice any value out of the expected.

	Min	1st Q.	Median	Mean	3rd Q.	Max
Rating	1.600	3.800	4.200	4.078	4.500	5.000
Rating.Count	3.045	3.738	4.654	5.148	6.116	13.013
Size	0.01094	1.85630	2.56495	2.63017	3.36730	6.23637
DaysLastUpdate	0.0	91.0	325.0	534.4	787.0	3069.0
ReleasedDays	8	584	1146	1286	1837	4085
AppNameLen	1.00	14.00	22.00	24.18	31.00	50.00
Installs	3.664	8.661	9.863	10.048	11.303	18.165

Figure 19: Numerical variables summary

We then plotted the histograms and the box plot for all the numerical variables. From them we can see that people tend to not give a rating to apps. Another thing that results from these plots is that when they do tend to give a rating of 4+.

Most apps have a name with a length between 10 and 30 characters, but there are also apps with very long names according to the conventions of app naming.

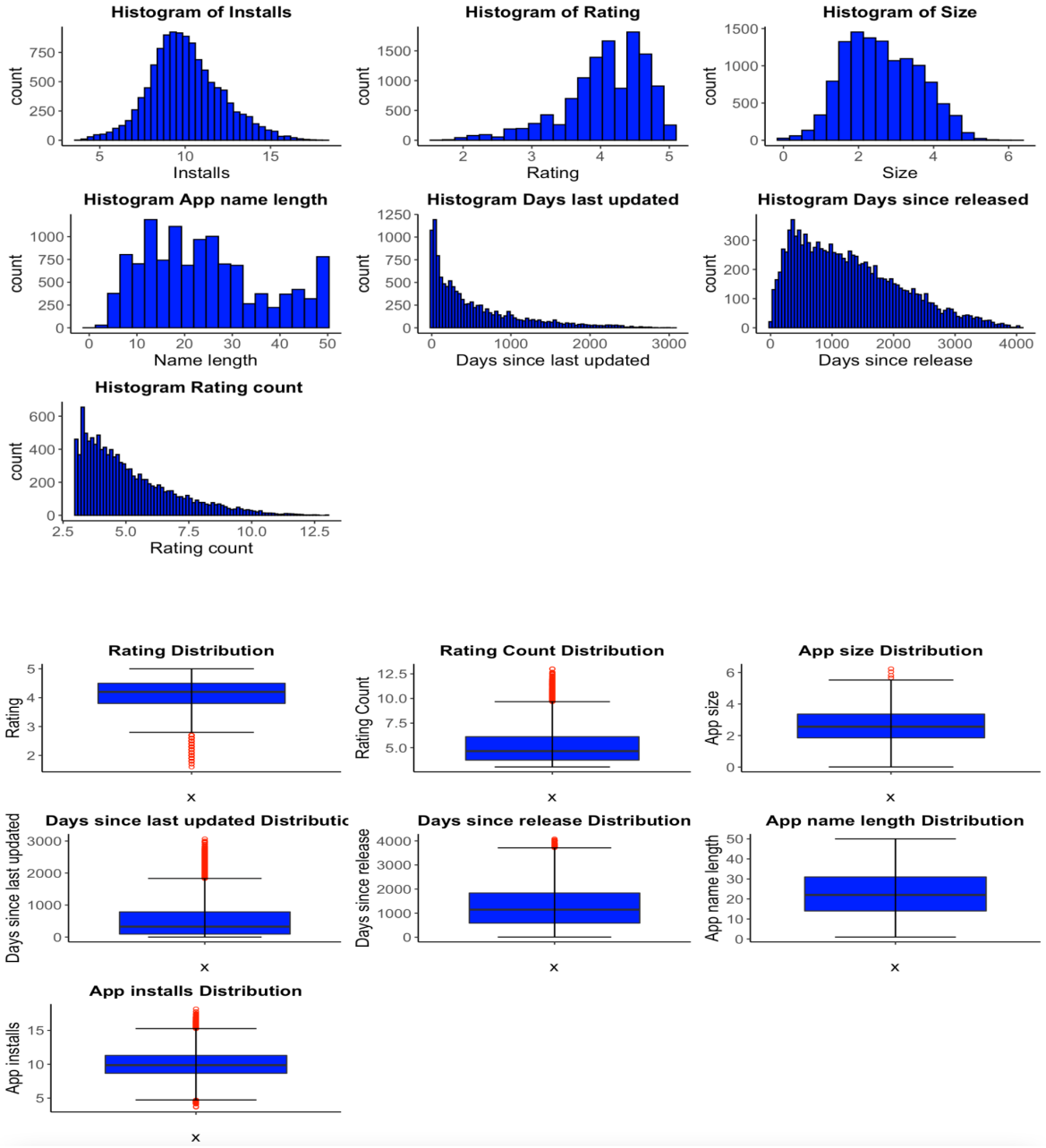


Figure 20: Histograms and box plot for numerical variables

If we look at Figure 21 it can be seen that the vast majority of apps have in-app purchases and that almost 60% of the apps support advertisements on them.

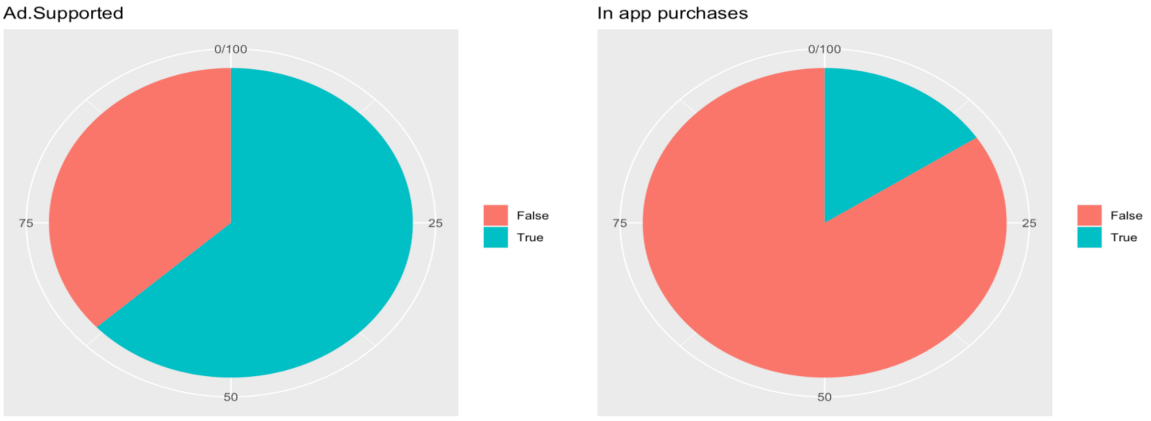


Figure 21: Ad supported and in app purchases pies

From Figure 22 we can conclude that most of the apps on the store belong to the Educational and the Lifestyle categories. Followed by the ones belonging to the Game and the Entertainment ones.

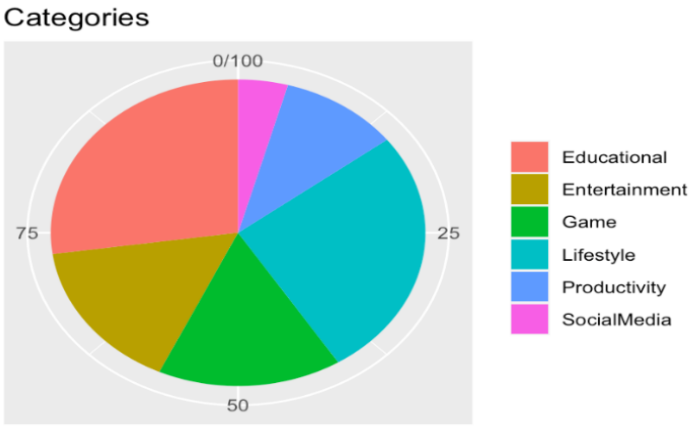


Figure 22: Pie chart for Category

As we can appreciate in figure 23 the vast majority of apps on the store, more than 80%, are suitable for every age, since their content rating is Everyone.

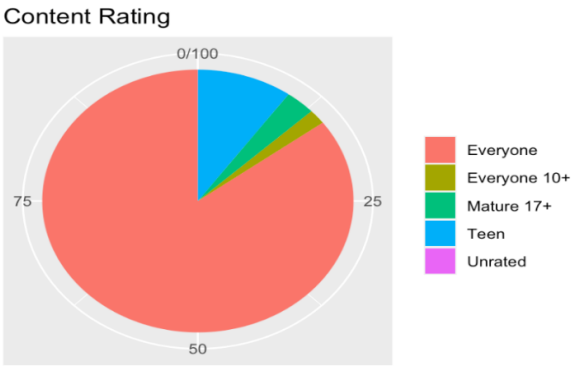


Figure 23: Pie chart for Content Rating

In **Figure 24**. we can see that the vast majority of the apps on the store require at least a version 4 of Android.

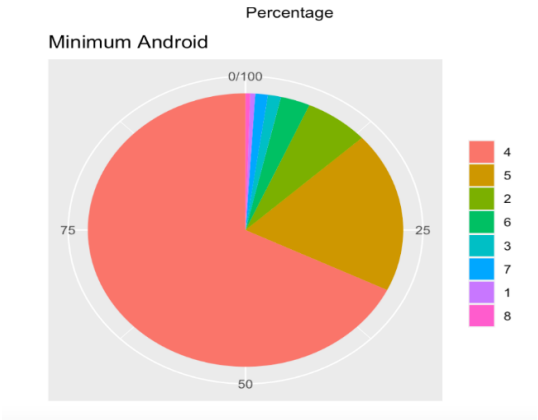


Figure 24: Pie chart for Content Rating

5.2 Bivariate analysis

In this section, we analyze the relationship between different pairs of variables in our data in order to determine if there are some patterns between the features of the different apps. The analysis was made mainly using visualization tools.

5.2.1 Correlation matrix

Firstly, in order to have a global scope of the relation of all variables of our data, we print the correlation matrix. From it we can establish that there is a significant positive correlation between `Rating.Count` and `Installs`. So the more `Installs` the more `Rating.Count`. There is also a light correlation between `ReleasedDays` and `DaysLastUpdate`, the “older” an app is, the longer the time since the last update.

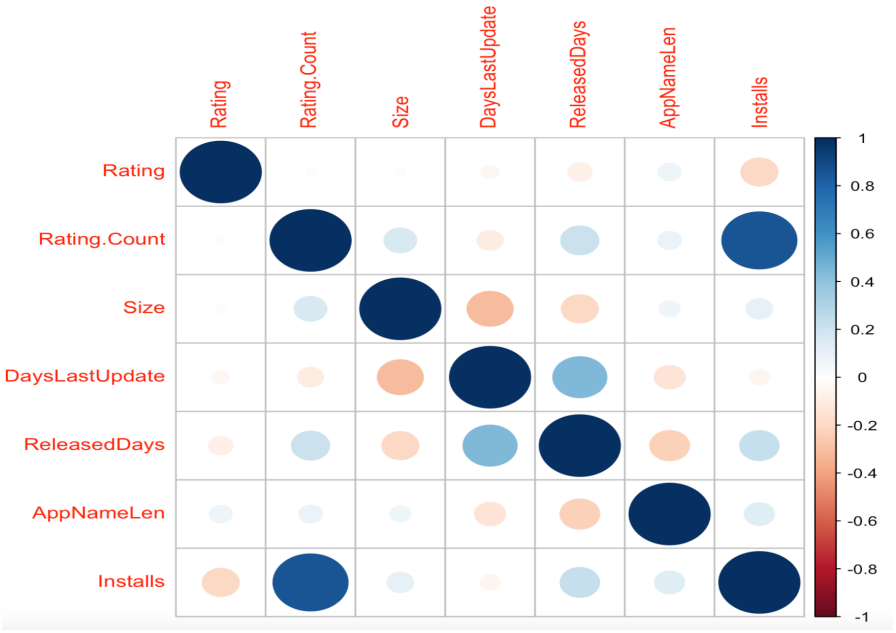


Figure 25: Correlation matrix

5.2.1.1 Rating vs Installs

To study the relationship between Rating and Installs, we print a scattershot using these two variables. From the plot we can see that an app that has a lower number of installs has a rating above 4. This can be explained by the fact that when an app has few downloads, it will generally have few reviews and they are usually good reviews, so the rating number is going to be high. Once an app starts getting more downloads it also starts having more reviews, so the rating is more dispersed. Finally, the apps that achieve more installs have higher scores than the ones in the center of installs.

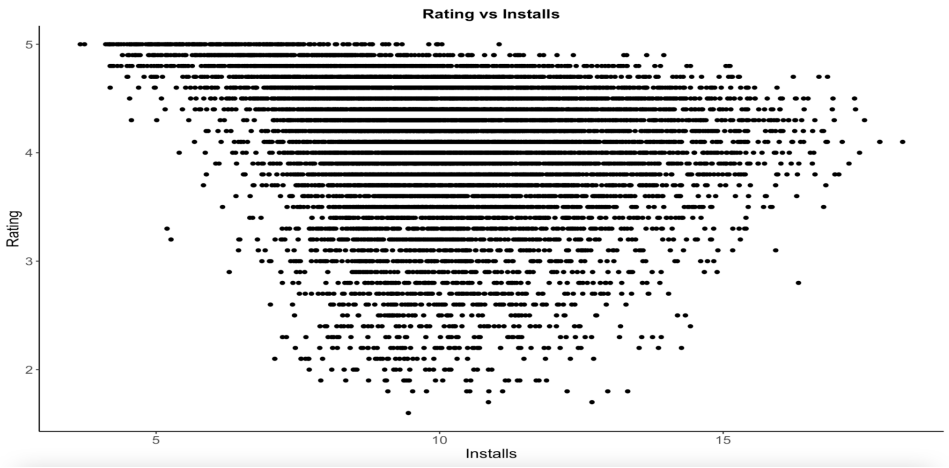


Figure 26: Rating vs Installs

5.2.1.2 Installs vs Size

We also tried to see if there is any relationship between `Installs` and `Size`. In this case, the results show, as we have already seen in the correlation matrix, that there is not a clear correlation between `Size` and `Installs`.

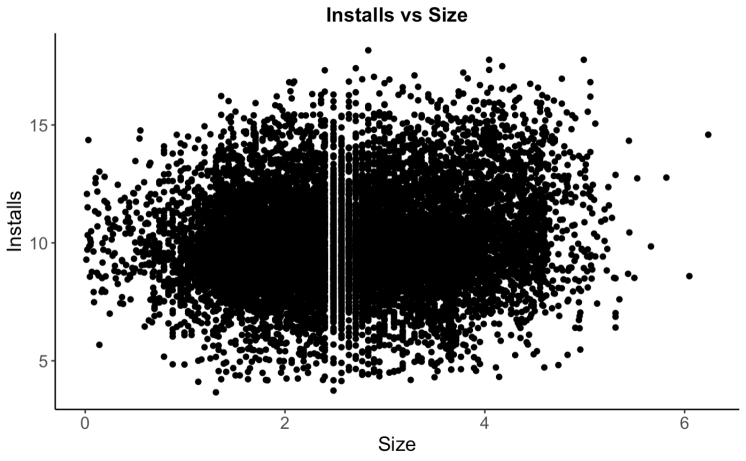


Figure 27: Installs vs Size

5.2.1.3 Installs vs Name Length

As we stated before, we did some research and we found out that there are some guidelines an app developer has to follow to name an app. Therefore, what we wanted to find out was if there is some relationship between the number of installs and the length of the app name.
But from Figure 28 we can see that the length of the app name does not affect the number of installs.

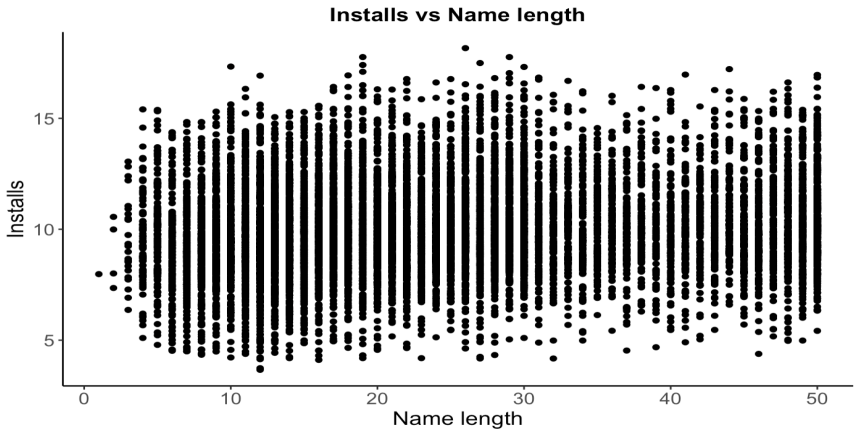


Figure 28: Installs vs Name length

5.2.1.4 Rating vs Category

We also were interested to know how the apps were rated according to its category. From Figure 29 we can establish that the distribution of apps rating for the different categories is quite similar.

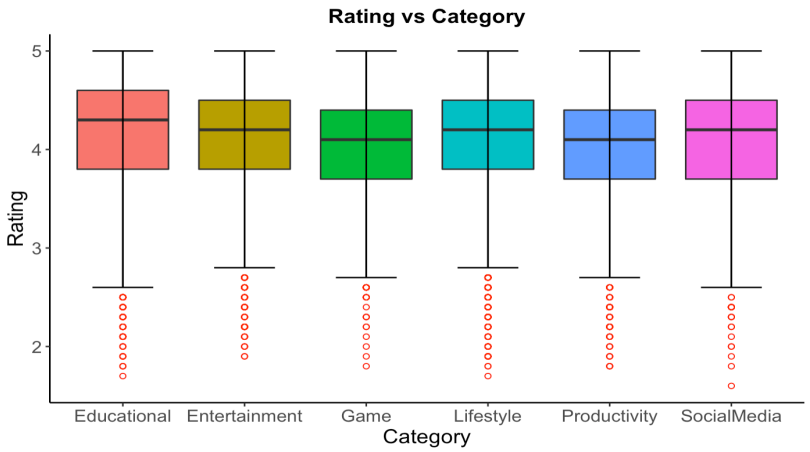


Figure 29: Rating vs Category

5.2.1.5 Size vs Category

We also tried to study if there is any relationship between the size of the apps according to their category Figure 30. The plot shows that apps that belong to the category Game are the ones with greater size. Which makes a lot of sense since game apps need more graphics among other things. While the apps with lower size are Productivity.

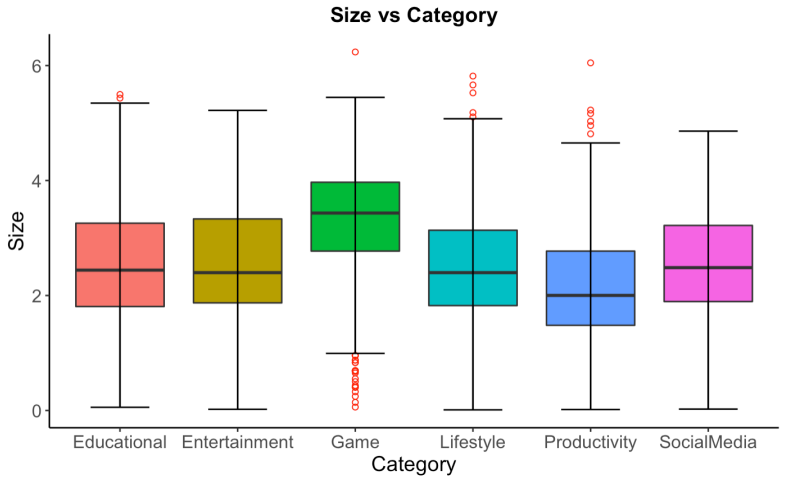


Figure 30: Size vs Category

6. PCA Analysis for numerical variables

PCA is a technique used in numerical data to reduce the dimensionality of the data by transforming it into a new coordinate system where most of the variation of the data can be described with fewer dimensions.

The aim of this section is to perform Principal Component Analysis (PCA) to summarize information in a dataset with multiple inter-correlated quantitative variables. Furthermore, to better visualize variation in a dataset with fewer variables than the original dataset.

6.1 Scree plot

Before starting PCA, the numerical variables need to be separated from the rest of the variables. Once this step is completed, the PCA process can be done.

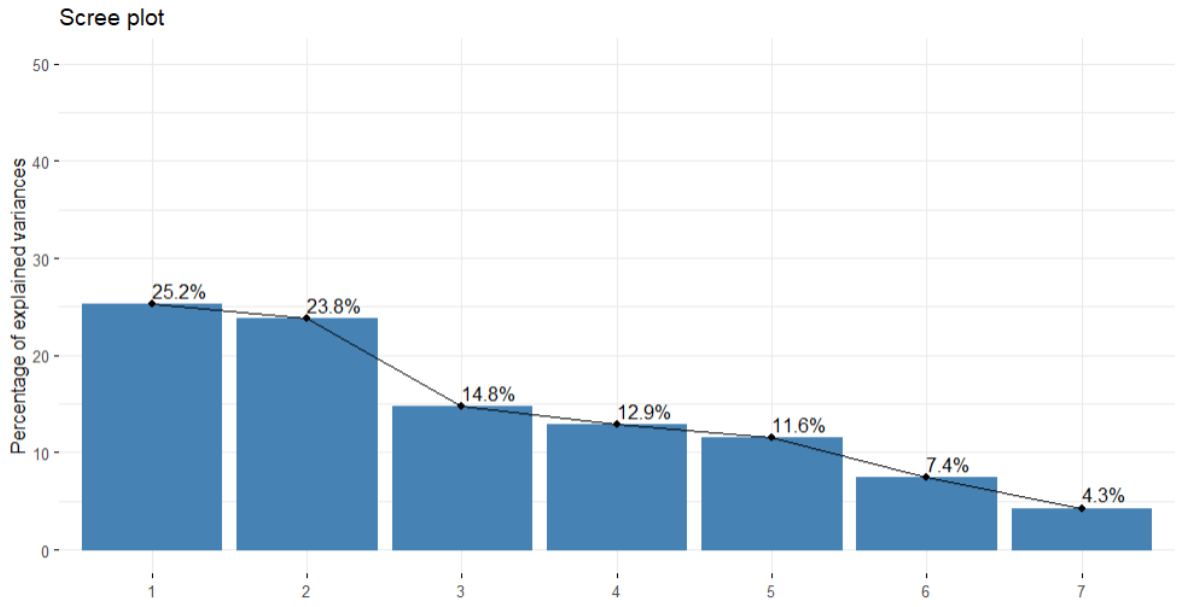


Figure 31: Scree plot

The proportion of variation explained by each dimension is given in the above Figure 31. As we can see, 25,2% of the variation is explained by the first dimension and 23,8% by the second one. As for the number of principal components to retain after PCA, in our analysis we only take the first three components that after adding up the successive proportions of variation, satisfied with 63.91% of the total variance, nearly two-thirds of the information in our dataset. From the plot above, we can see that after the point of dimension 3, all the remaining eigen-values decrease more slowly.

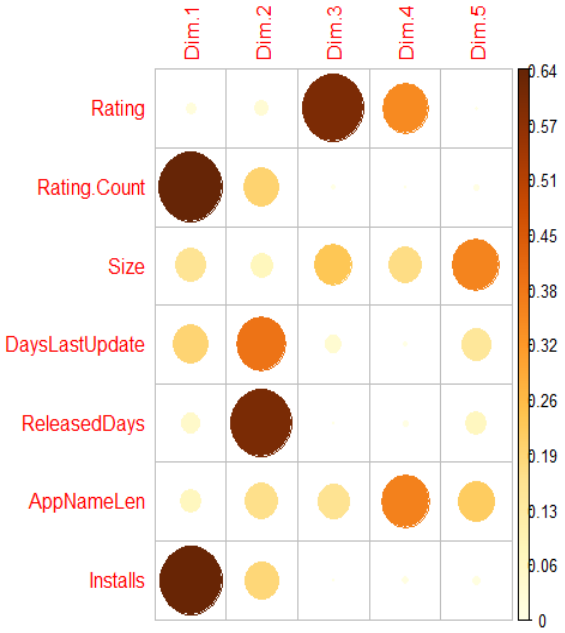


Figure 32: Correlation map

From Figure 32, it is clear to see that for both dimensions 3 and 4, we have the same list of most correlated variables. Due to this behavior, we can conclude that dimension 4 is just a reflection of dimension 3. We can also see that the variables `Rating.Count`, `DaysLastUpdate` and `Installs` are correlated with both PC1 and PC2, thus they are the most important in explaining the variability in the dataset.

From the figure 32, we can find the following highly correlated variables for each component:

- **Dim1:** `Rating.Count` and `Installs` → The meaning of this principal component can be interpreted as “Popularity”.
- **Dim2:** `DaysLastUpdate`, `ReleasedDays`→ Its meaning can be interpreted as “Active time && Update frequency”
- **Dim3:** `Rating`, `AppNameLen` and `Size` → Its meaning can be interpreted as “Rating && characteristics”

6.2 Factorial map visualization

6.2.1 Individuals projections

- Quality of representation of individuals on the factor map

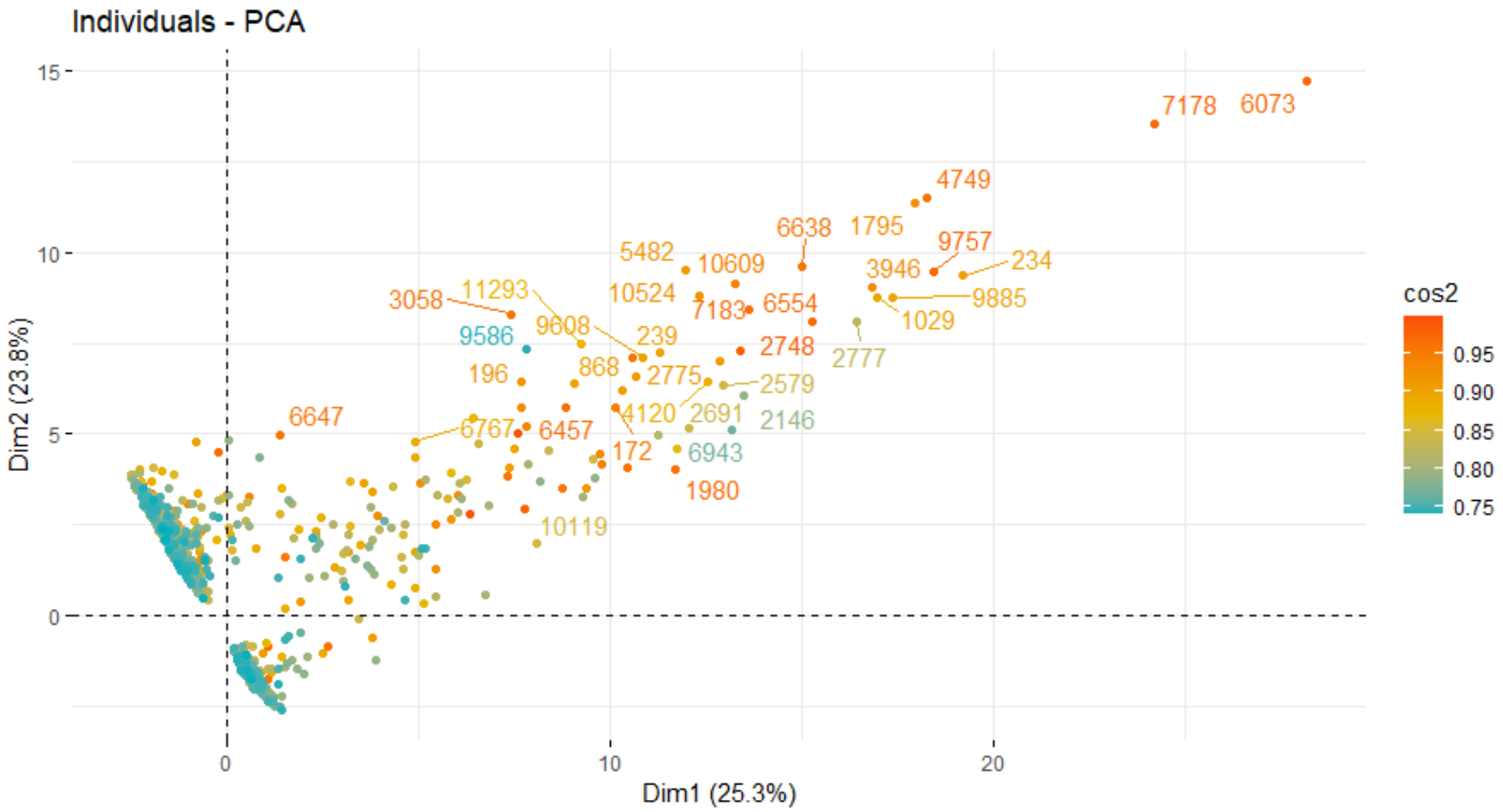


Figure 33: Quality of representation of the top 1000 individuals Dim1-2

From the Figure 33 of top 1000 individuals with better representation on the factor map we can clearly find three groups of individuals:

- Group 1: Apps with higher number of installs, votes, frequency of update and longer existing days, they are the ones with better quality of representation.
- Group 2: Apps with a poor or even zero number of installs and votes, but still exist in the market in which some release of updates were made recently.
- Group 3: Recently released or updated apps that haven't achieved much number of installs and votes

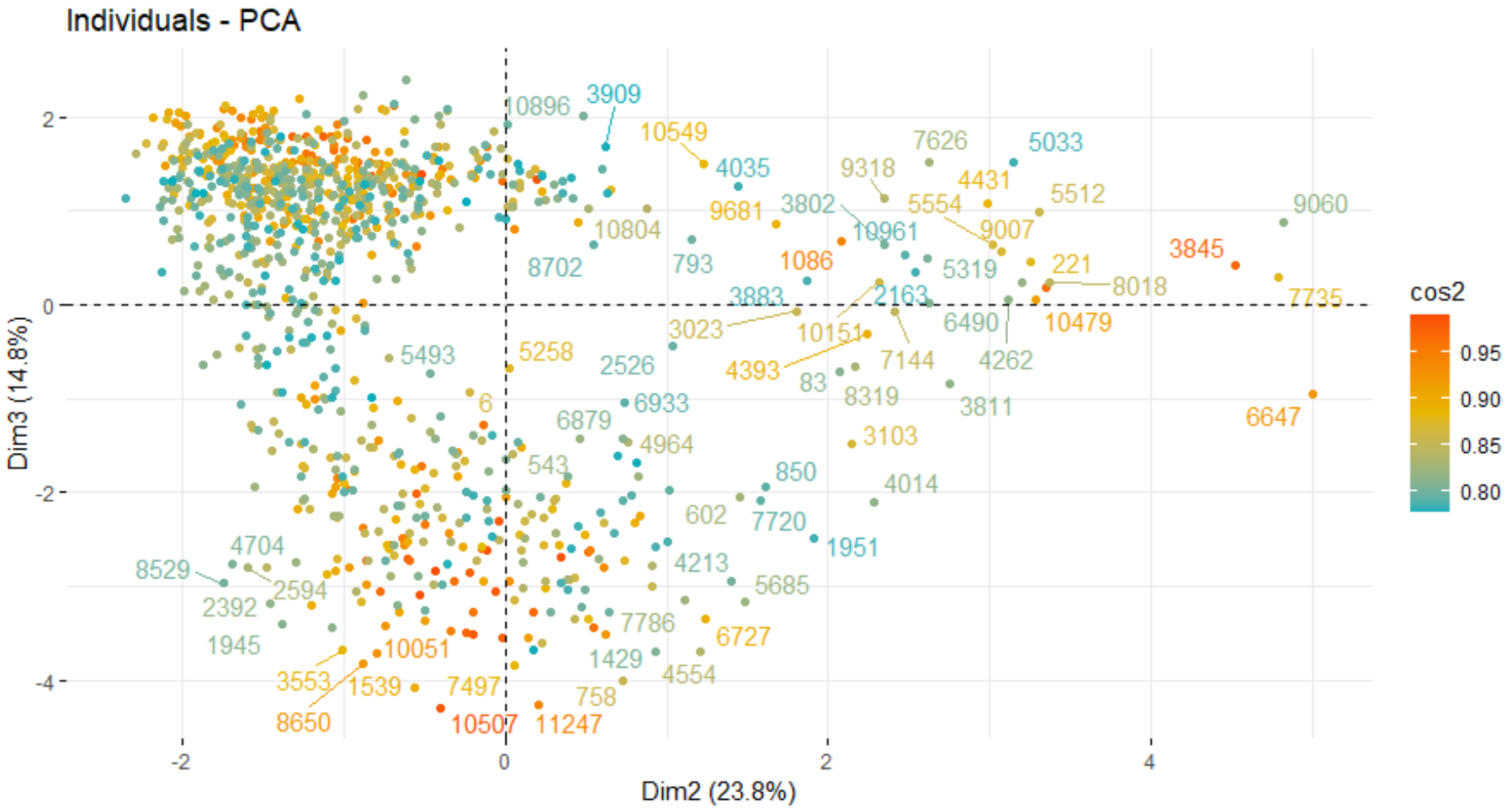


Figure 34: Quality of representation of top 1000 individuals Dim2-3

From Figure 34 we can see that `Rating` is not much correlated with number of existing days or frequency of update. Having a high `Rating` can't lead us to conclude that it is because of the longer existing days or frequency of update of an app. There are other factors that have more influence on the `Rating`.

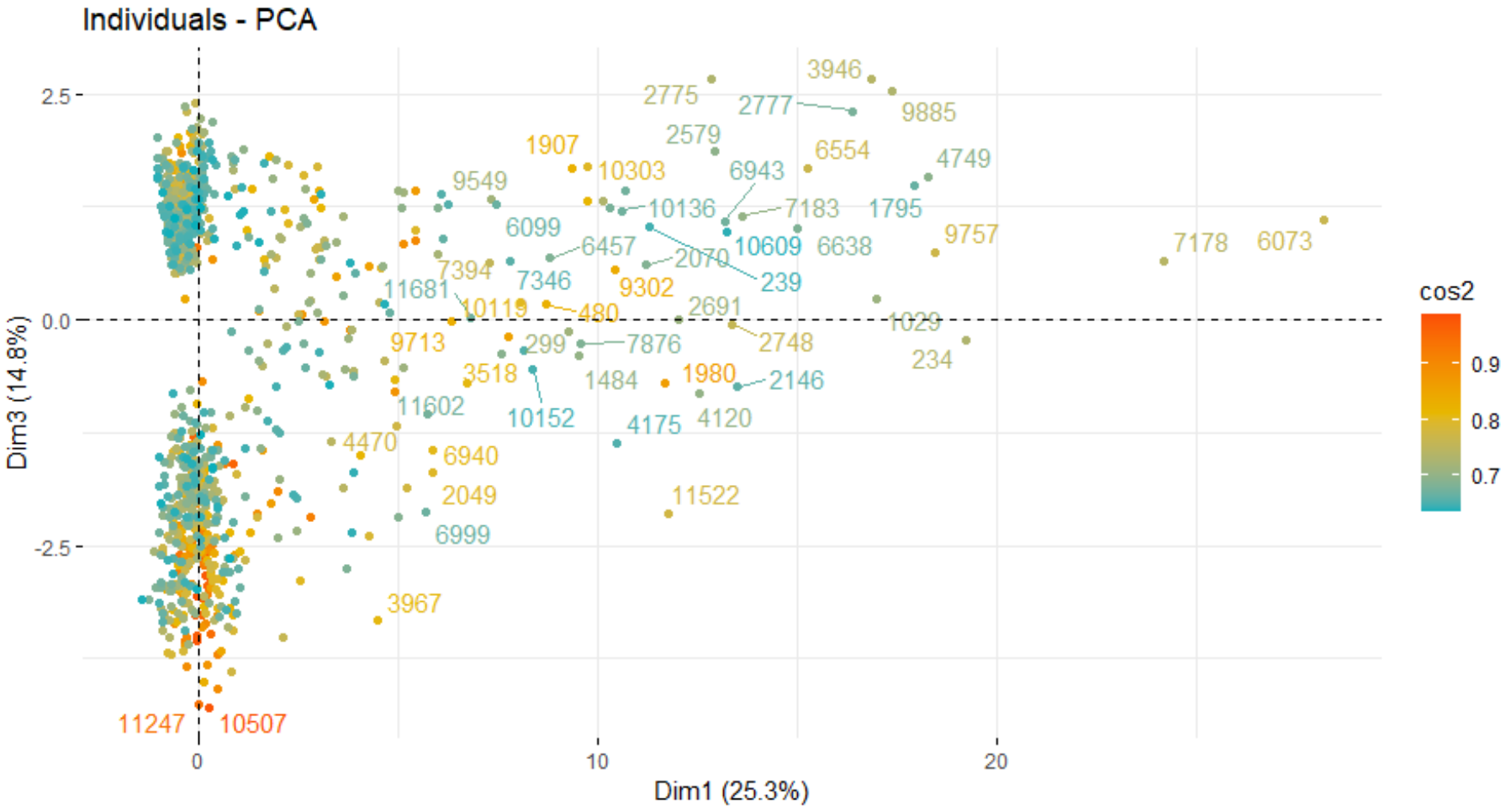


Figure 35: Quality of representation of top 1000 individuals Dim1-3

From figure 35, we can classify the individuals into the following groups:

- Group 1: Apps that have a higher number of installs and votes, in general, also have a good rating.
- Group 2: Apps with a poor number of installs and votes, but have an acceptable rating.
- Group 3: Apps with a poor number of installs and votes that receive bad ratings.
- Group 4: Apps with an acceptable number of installs and votes, but still their rating is bad.

- **Contribution of individuals to the first two principal components**

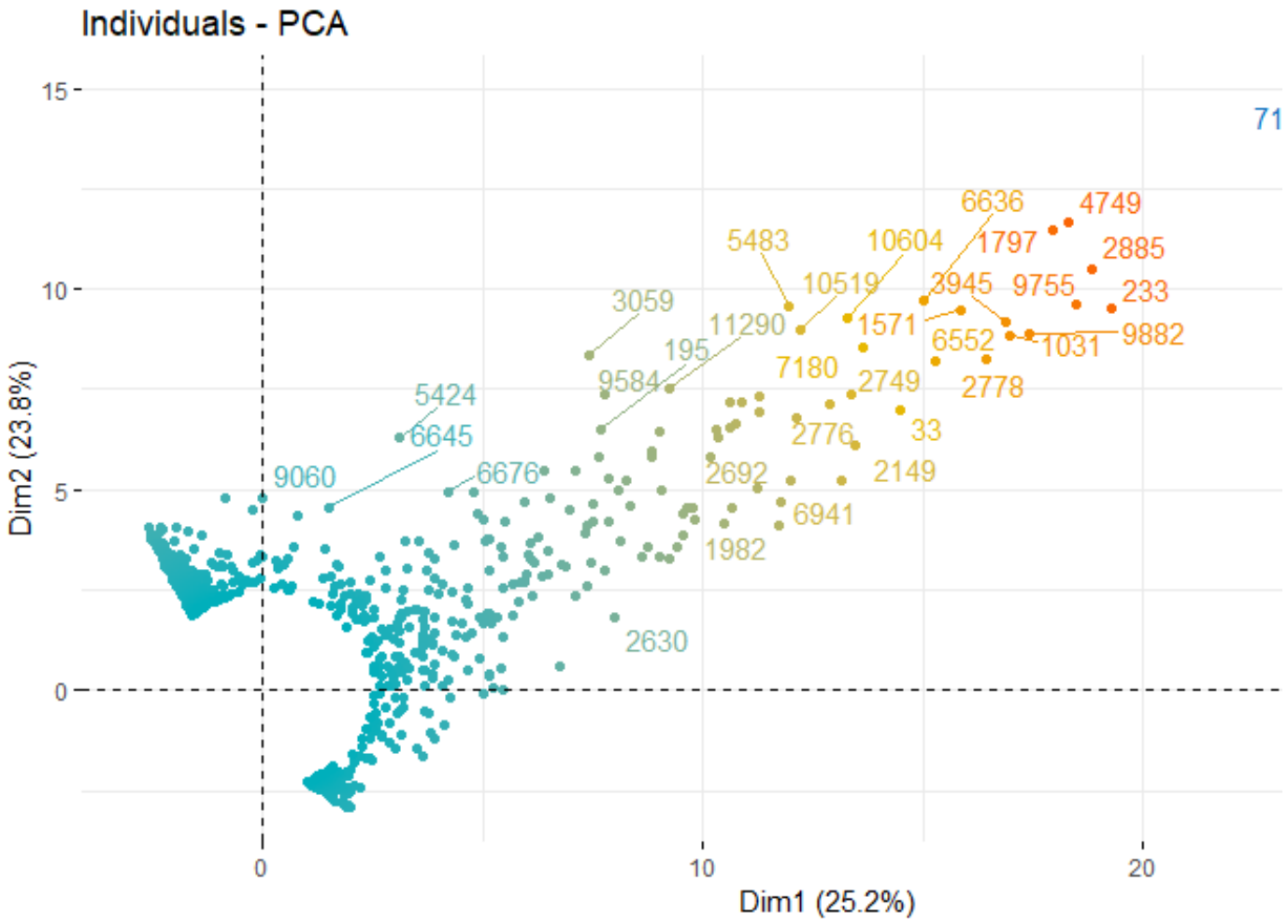


Figure 36: Contribution of individuals to Dim1-2

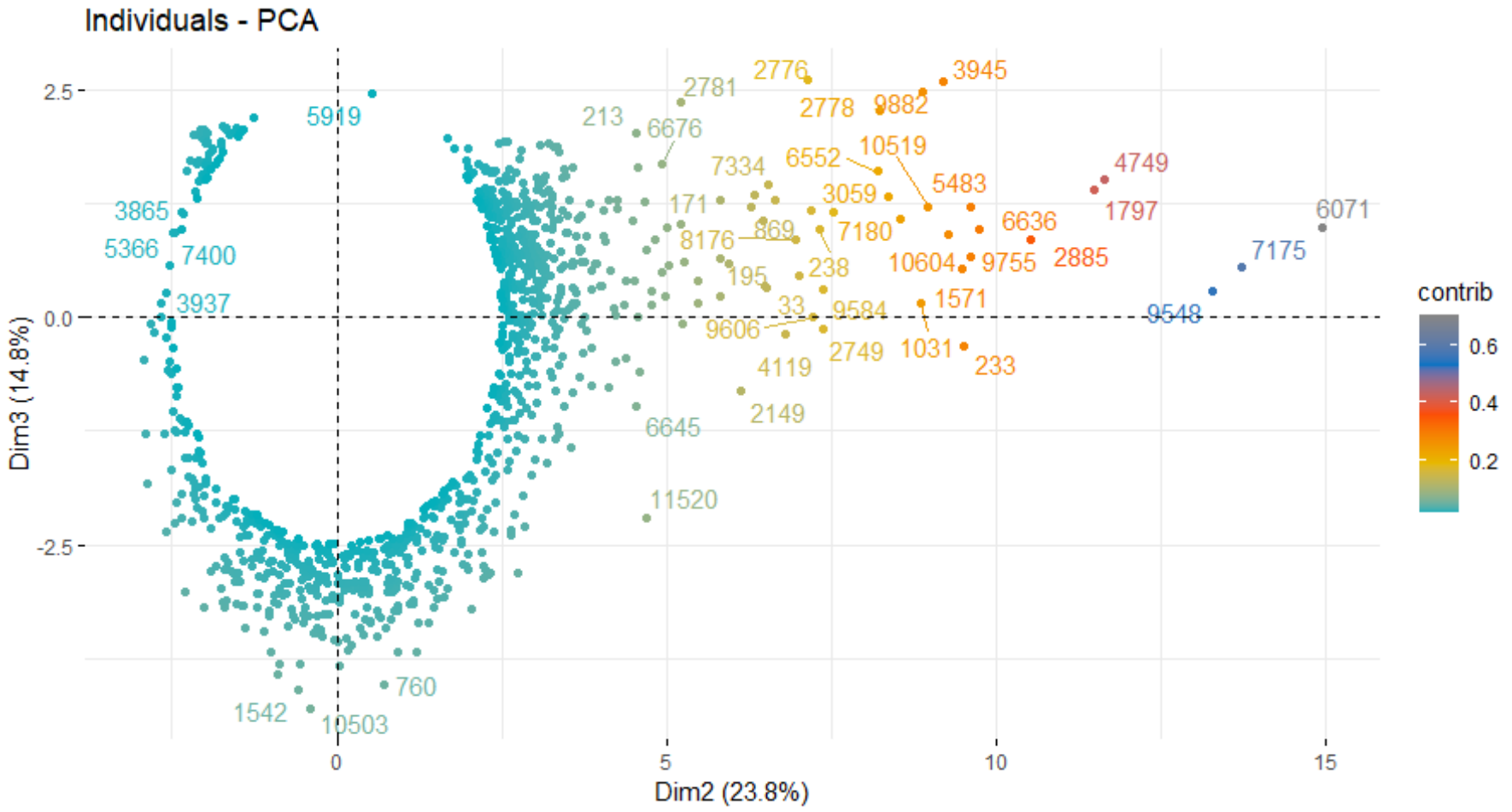


Figure 37: Contribution of individuals to Dim2-3

From figures 36 and 37 about the top most contributed 1000 individuals, we can see the same groups of individuals as before.

6.2.2 Common projection of numerical variables and modalities of qualitative variable

- Contribution of variables to PCs

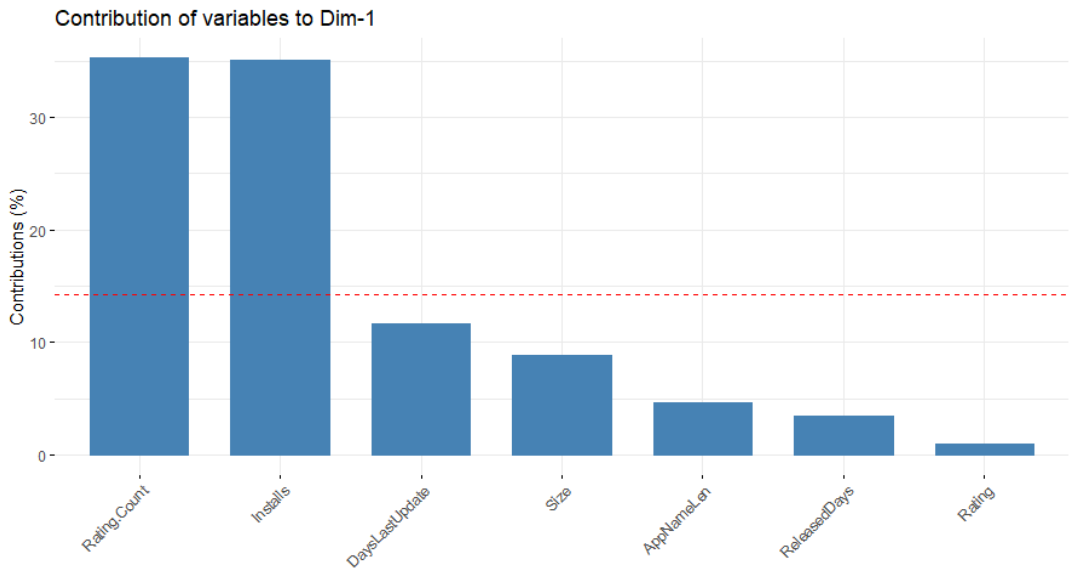


Figure 38: Contribution of variables to Dim1

From the above Figure 38 we can see that for dimension 1, the most contributed variables are `Rating.Count` and `Installs`.

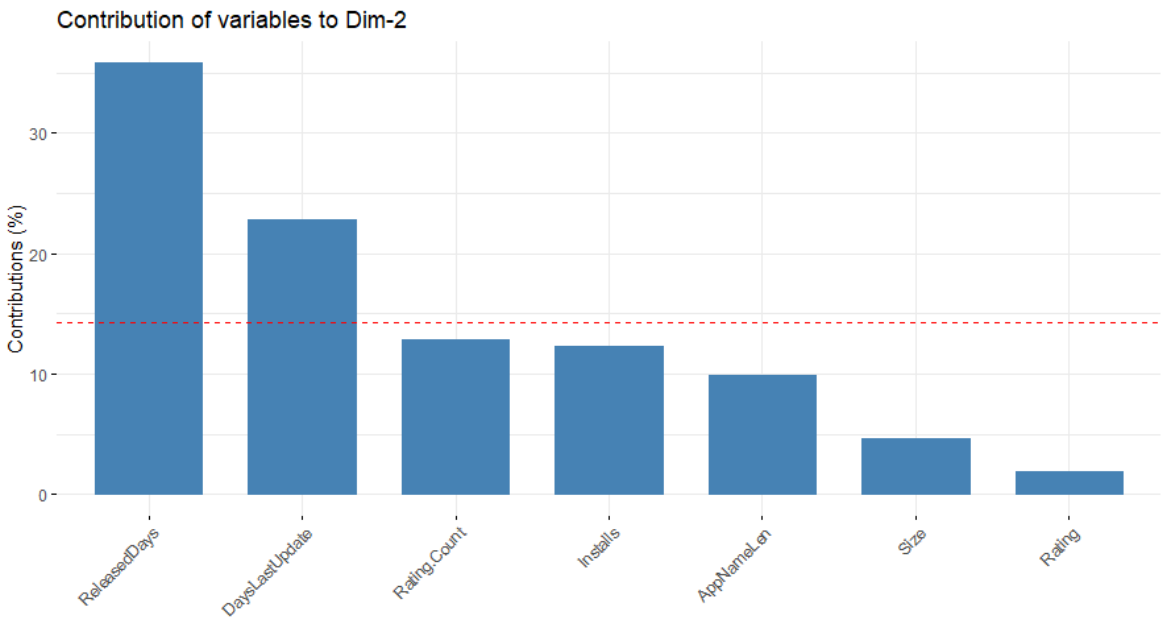


Figure 39: Contribution of variables to Dim 2

From the above Figure 39 we can see that `Released Days` and `DaysLastUpdate`, are the most contributed variables for dimension 2.

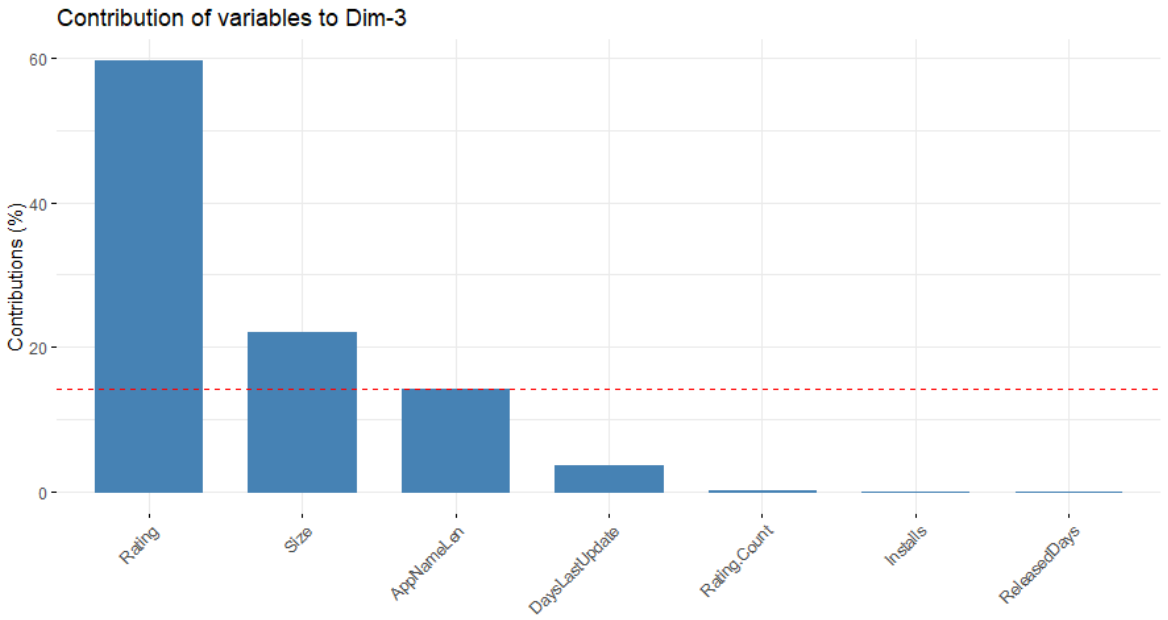


Figure 40: Contribution of variables to Dim 3

From the above Figure 40 we can see that `Rating`, `Size` and `AppNameLen` are the most contributed variables for dimensions 3.

- Biplots of individuals and variables

In the plot below (Figure 41) we use the variable `Category` as a supplementary qualitative variable and it is used for coloring individuals by groups.

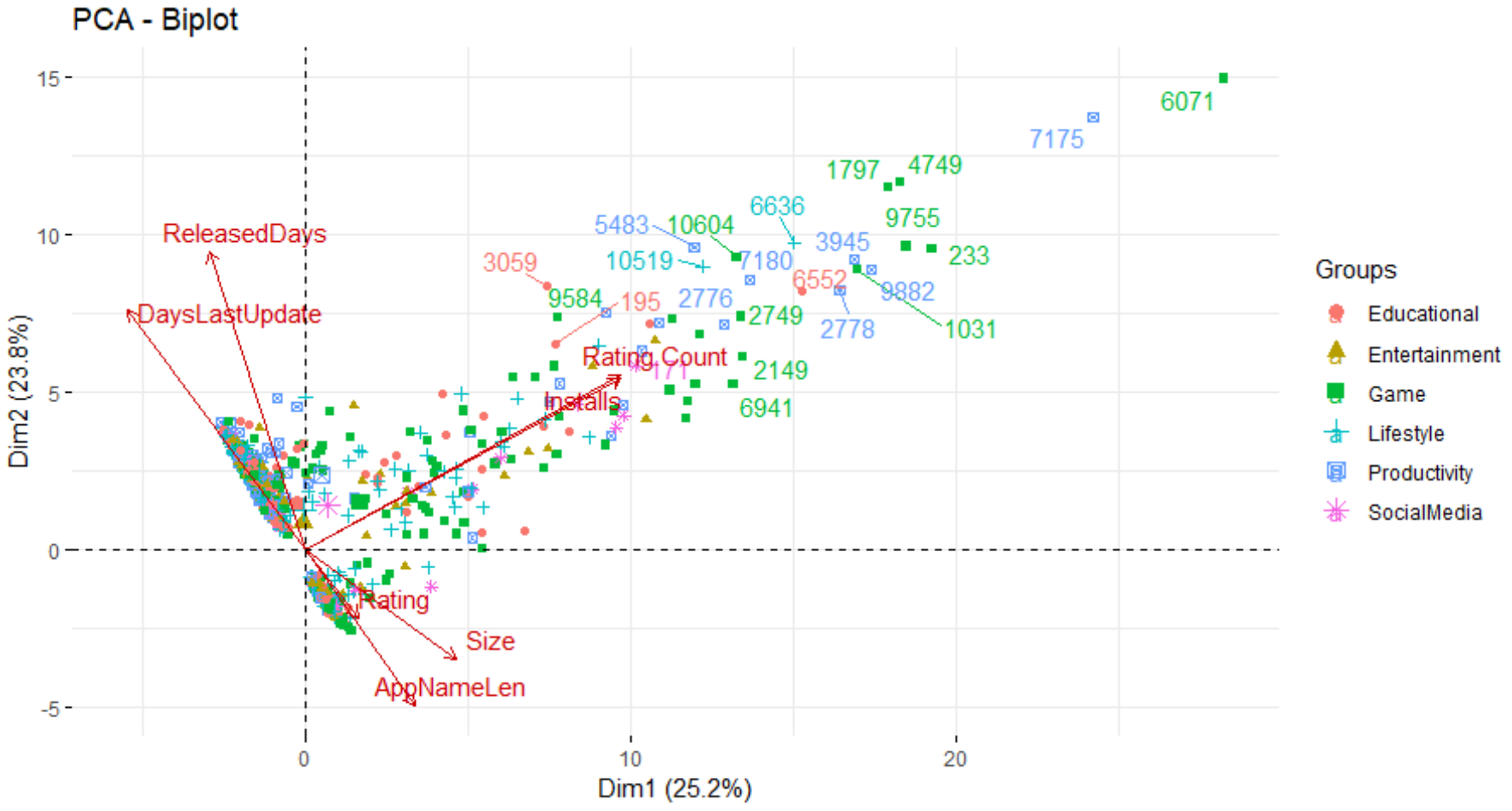


Figure 41: Biplot of individuals and variables Dim1-2

From Figure 41 we can observe that on the one hand, apps of categories `Game` and `Lifestyle` are always the ones with higher number of installs and votes, thus apps of these categories tend to be active longer and have a higher frequency of updates.

On the other hand, educational category tend to be less popular than other categories.

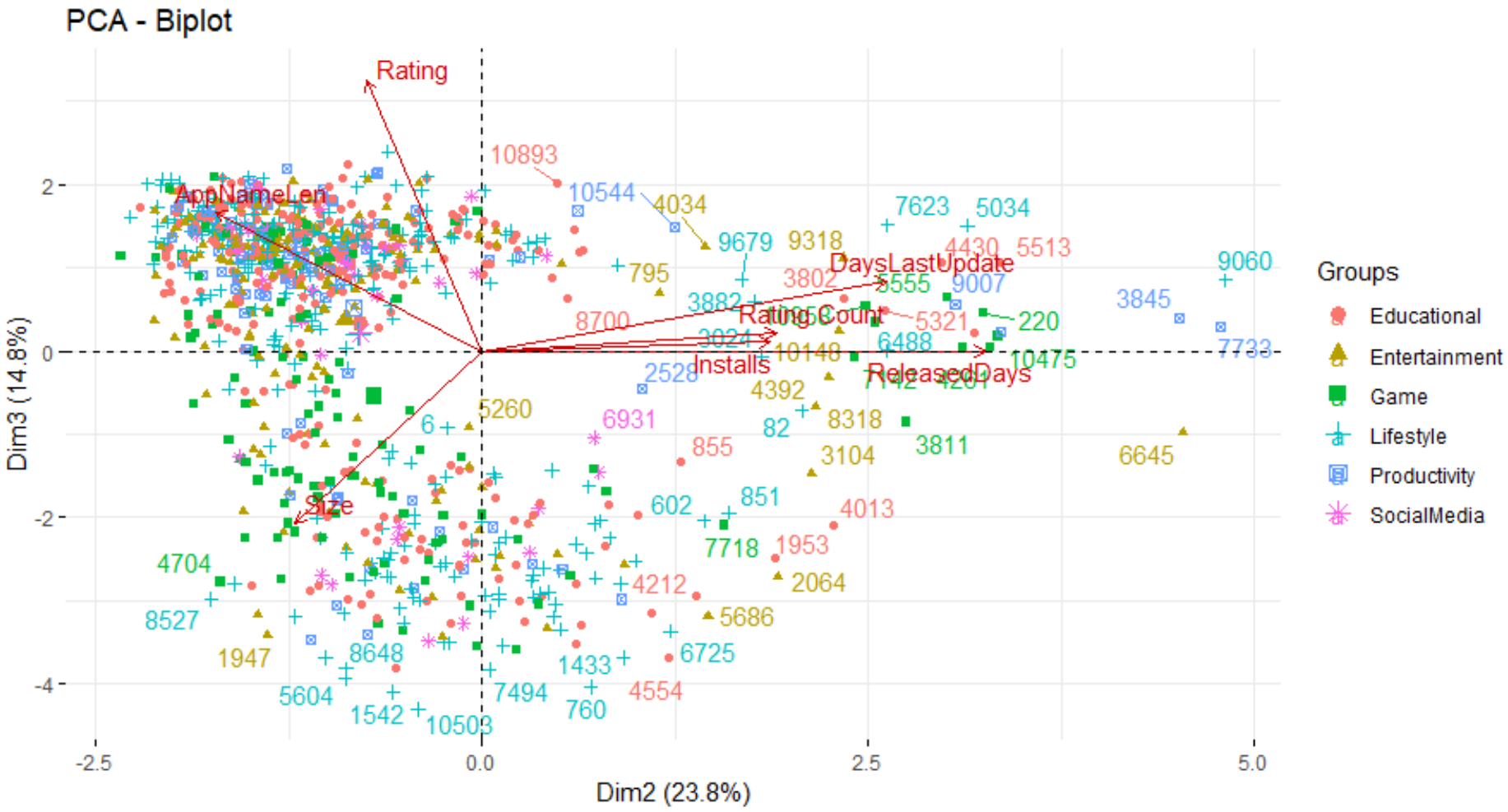


Figure 42: Biplot of individuals and variables Dim 2-3

From the Figure 42 we get the following interpretation:
For apps of categories `Game` and `Lifestyle` which are recently released to the market or have a low frequency of update, the probability of getting a not good or poor rating is relatively high.

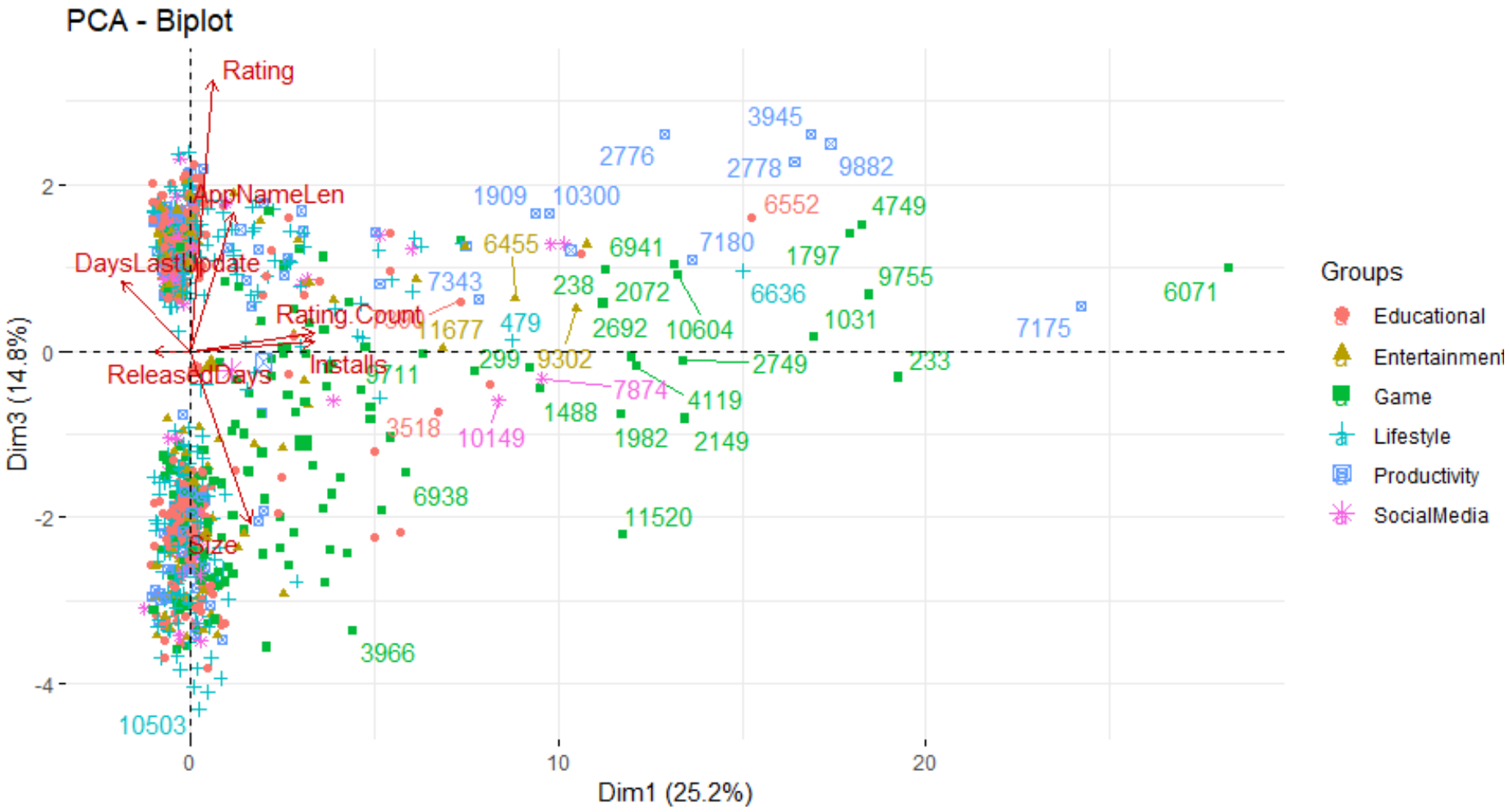


Figure 43: Biplot of individuals and variables Dim1-3

From the above Figure 43 we can also notice that although apps of category `Game` can always get a higher number of installs, it's not correlated with the rating. There are cases in which apps have higher popularity but poor ratings. What is noticeable is that unpopular `Game` apps tend to receive more bad ratings than good ratings.

6.2.3 Interpretation of relationships among variables observed

In order to understand the meaning of our principal components and to be able to assign a name to identify each component, it is necessary to identify those variables whose correlations are the highest with components.

The following plots can help us to better understand the relationship between all variables.

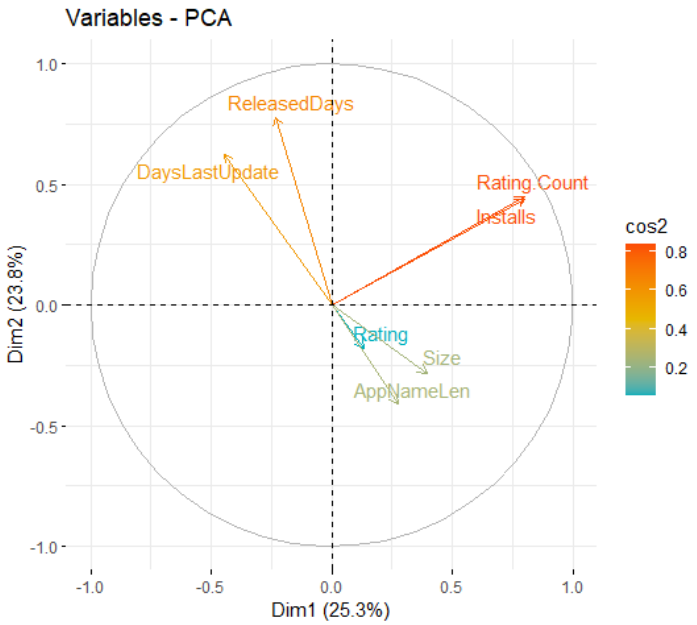


Figure 44: variable correlation Dim1-2

From the above variable correlation plot we observe the following behaviors:

- `Rating.Count` and `Installs` are positively correlated, because they are strongly grouped together. As the distance between the variables and the origin measures the quality of the variables on the factor map, and from the plot it's clear to see that they are far away from the origin, we can conclude that variables `Rating.Count` and Installs are well represented on the principal component.
- `ReleasedDays` and `DaysLastUpdate` are also positively correlated but not strongly grouped together. As `ReleasedDays` is closer to the circle of

correlation and the angle it forms with respect to the axis Dim2 is smaller, it has better representation on the factor map than `DaysLastUpdate`.

- `Rating`, `Size` and `AppNameLen`(Rating && characteristics) are negatively correlated with `ReleasedDays` and `DaysLastUpdate`(Active time && Update frequency), which means that "Rating && characteristics" has the opposite behavior than "Active time && Update frequency".

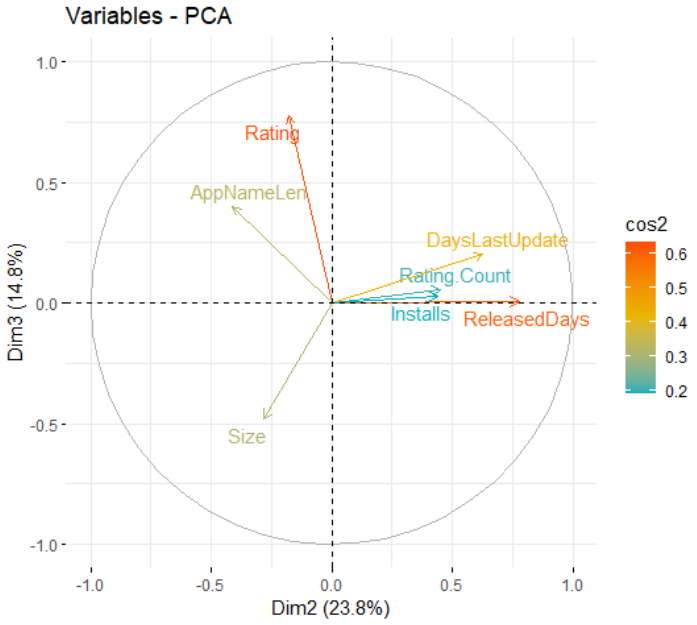


Figure 45: variable correlation Dim2-3

From the above variable correlation plot we observe the following behaviors:

- `Rating.Count` and `Installs` (Popularity) are positively correlated with `DaysLastUpdate` and `ReleasedDays` (Active time & update frequency) but as they are closer to the center of the circle, they are less important for the components 2-3, which means having a low quality of representation (cos2).
- `Rating` and `AppNameLen` are positively correlated, but `AppNameLen` contribution to the building of PC3 is less than `Rating` because it forms a bigger angle with respect to the axis dim3.

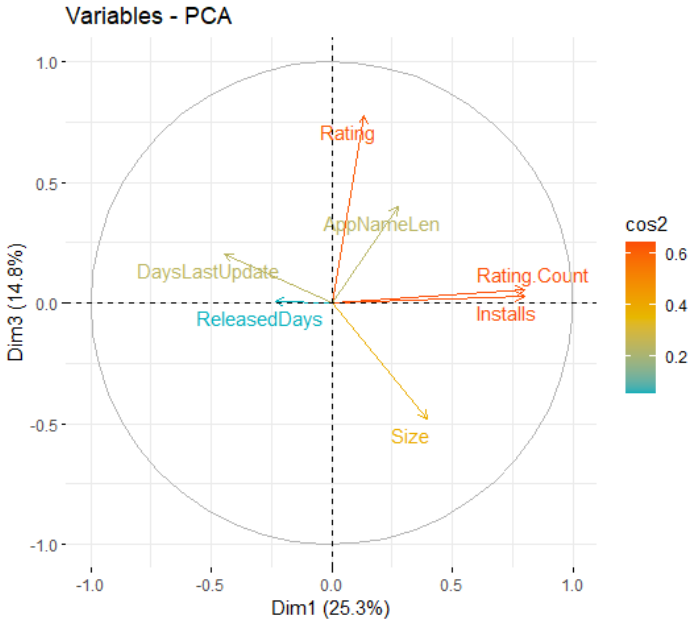


Figure 46: variable correlation Dim1-3

From the above variable correlation plot we observe the following behavior. `DaysLastUpdate` and `Size` are negatively correlated, this means that the more days without updating an app the smaller is its size, which makes sense because usually the larger is the size, the more bugs needed be fixed or new features to be included to the app, thus higher is the frequency of update.

As variables can be represented as points (coordinates) in components space by using their correlation with the components. We tried to apply the clustering algorithm Kmeans to classify variables into 3 groups. Thus, we get the same division of variables as before.

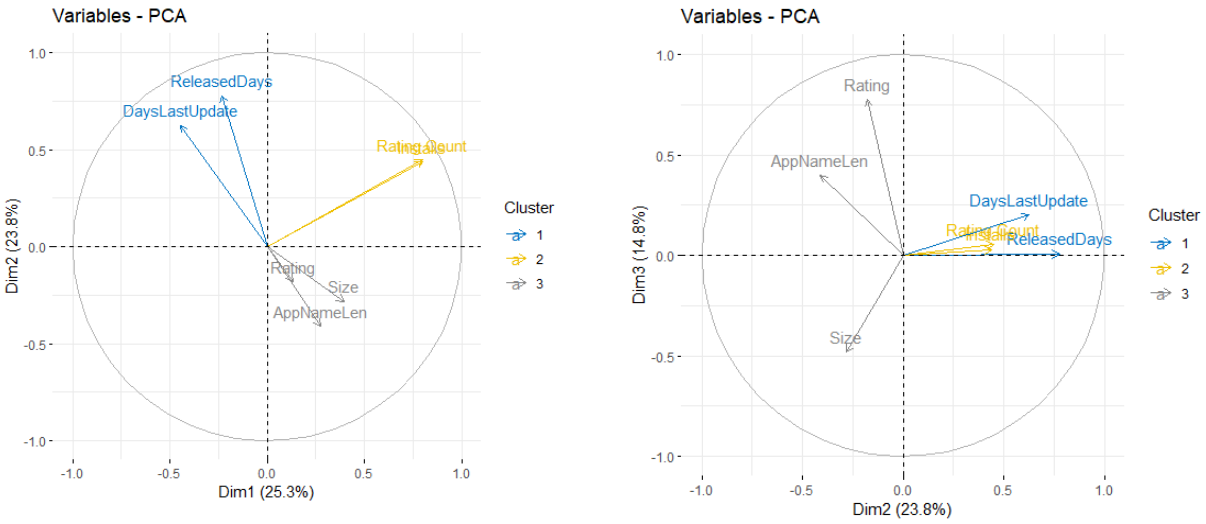


Figure 47: Correlation maps after applying Kmeans

6.3 Conclusions

After performing PCA and various types of analysis about variables and individuals behavior on factor map through plots, we get the following conclusions:

- Variables `Rating.Count`, `DaysLastUpdate` and `Installs` are correlated with both PC1 and PC2, therefore they are the most important in explaining the variability in the dataset.
- The frequency of updates doesn't have any effect on the rating. We can consider in this case the possibility that not all apps information in this dataset was updated correctly or with the relevant information.
- There are apps with fewer votes but high ratings(on the figures of individuals we can observe this behavior), there seems to be a non-linear relationship between `Rating.Count` and `Rating`.
- `Game` and `Lifestyle` are the categories with the most number of installs.
- `Rating.Count` and `Installs` are highly correlated, the total variance would not be affected if we remove one of these variables when performing PCA.
- A higher number of installs or votes doesn't mean that the rating is also high.

7. MCA of multiple qualitative variables

MCA is a data analysis technique for categorical data, used to detect and represent underlying structures in a dataset. It does this by representing data as points in a low-dimensional Euclidean space.

7.1 Detection of low frequency variable categories

Before starting MCA, we need to identify variable categories with a very low frequency as these types of variables can distort the analysis. In Figure 48, we can see that the variables `Minimum.Android` and `Content.Rating` have low frequency categories.

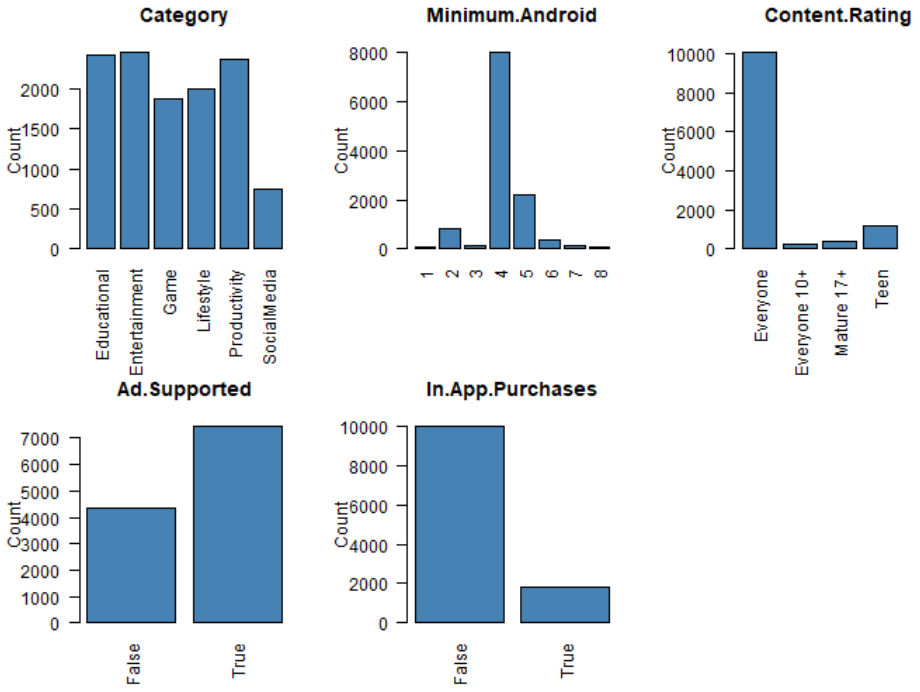


Figure 48: Distribution of the variable categories

For `Minimum.Android` and `Content.Rating` we aggregated the categories so that we have <4, 4, 5 and >5 and Everyone and AgeRestricted, respectively (Figure 49).

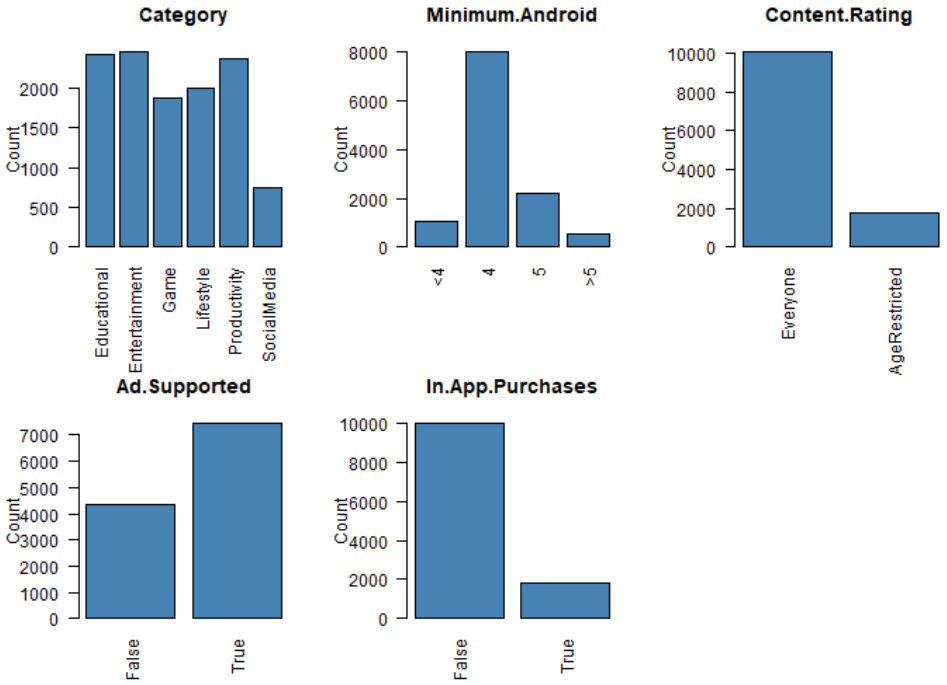


Figure 49: Distribution of the variable categories after aggregation of low frequency categories

7.2 Eigen values

After performing MCA using the **logical table**, we obtained the eigen values (Figure 50). We kept the dimensions that had an eigen value bigger than 1/p, where p is the number of categorical variables. In our case p=5, so 1/p = 0.2.
In the end, we decided to keep 5 dimensions.

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	0.3226002	14.663646	14.66365
Dim.2	0.2530929	11.504222	26.16787
Dim.3	0.2278697	10.357712	36.52558
Dim.4	0.2084731	9.476049	46.00163
Dim.5	0.2022985	9.195386	55.19701
Dim.6	0.1976014	8.981880	64.17889
Dim.7	0.1962563	8.920743	73.09964
Dim.8	0.1750459	7.956633	81.05627
Dim.9	0.1521349	6.915224	87.97150
Dim.10	0.1380674	6.275793	94.24729
Dim.11	0.1265597	5.752712	100.00000

Figure 50: Eigen values of the dimensions

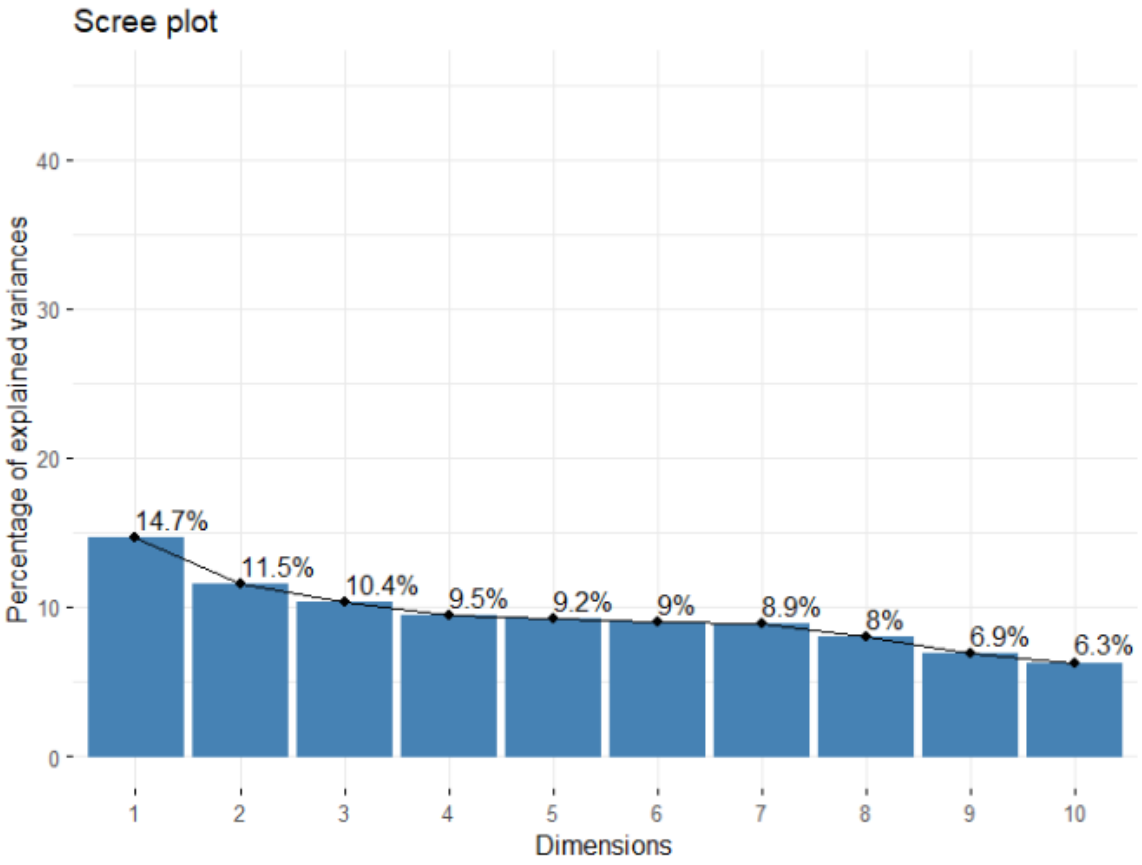


Figure 51: Scree plot of the dimensions

7.3 Biplots of individuals and variable categories

In this section, we can see the biplots of individuals and variable categories for each combination of dimensions.

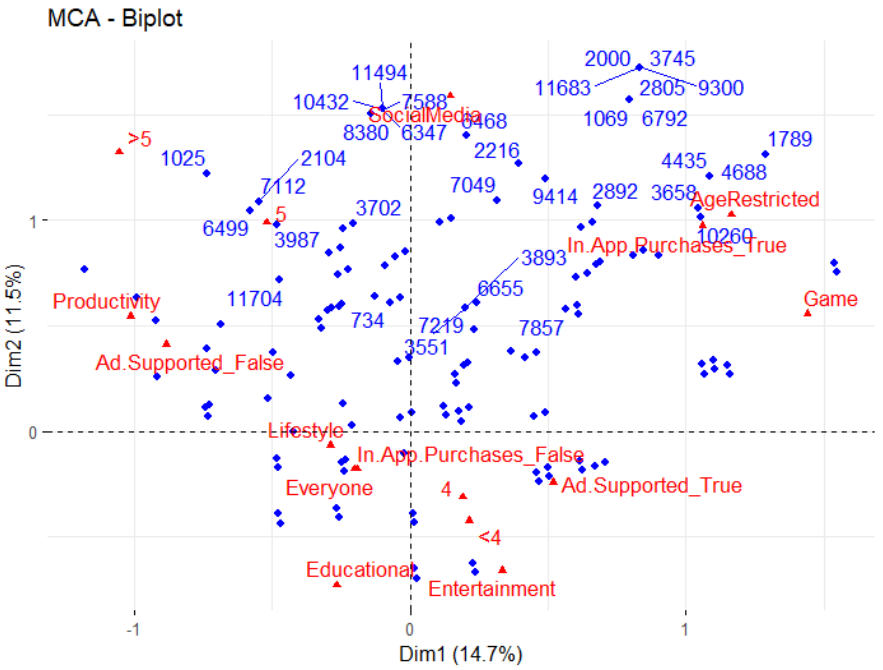


Figure 52: Biplot of individuals and variable categories in dimension 1-2

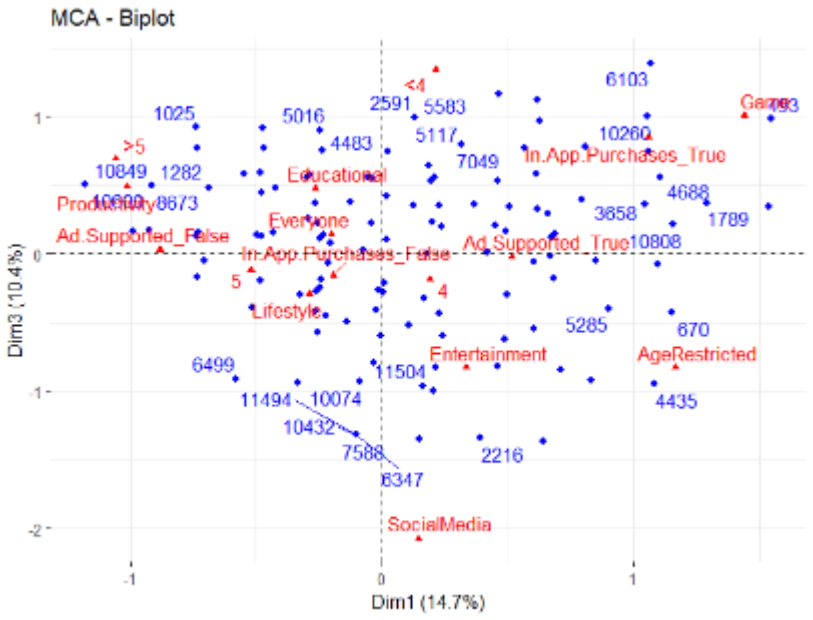


Figure 53: Biplot of individuals and variable categories in dimension 1-3

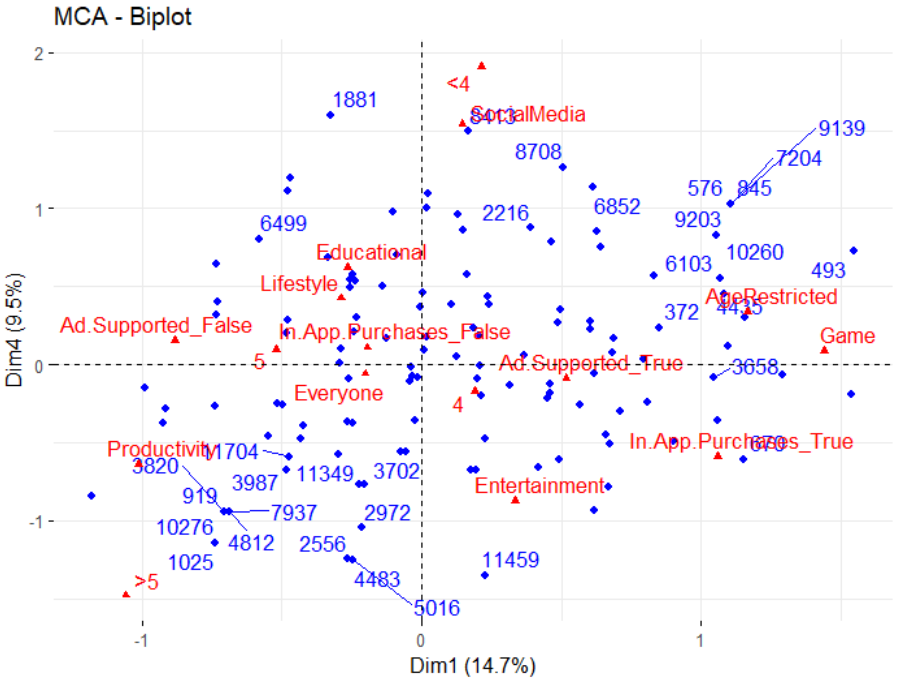


Figure 54: Biplot of individuals and variable categories in dimension 1-4

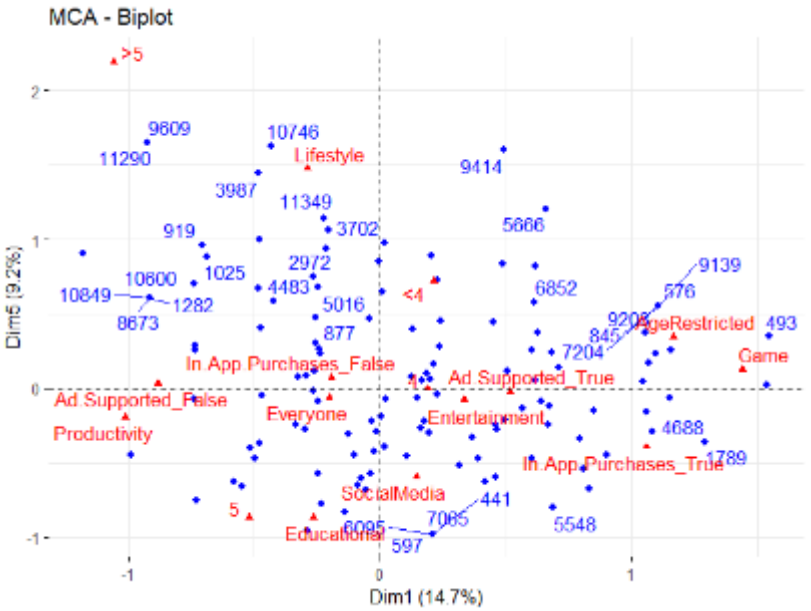


Figure 55: Biplot of individuals and variable categories in dimension 1-5

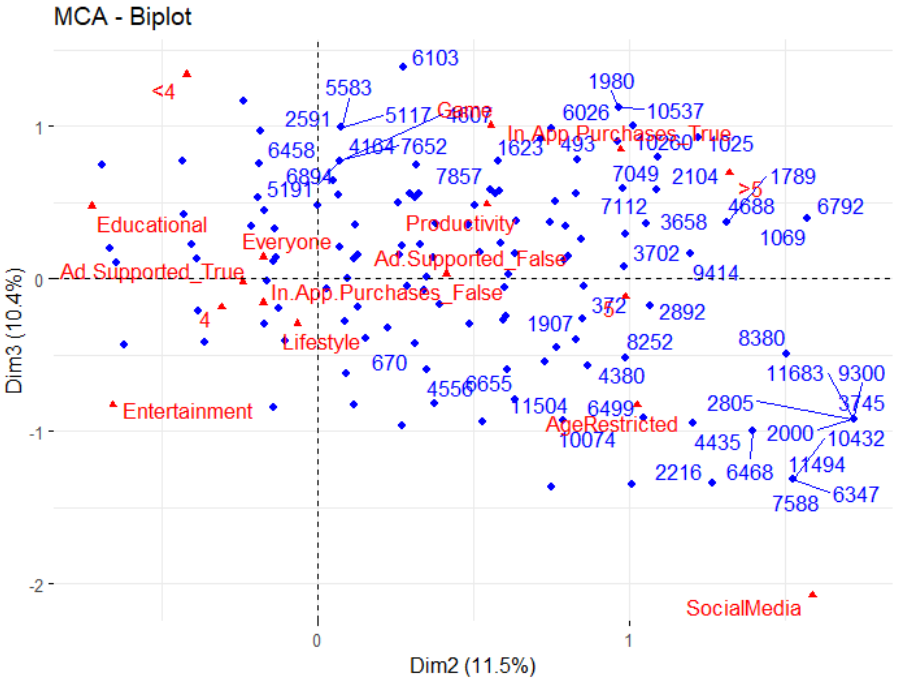


Figure 56: Biplot of individuals and variable categories in dimension 2-3

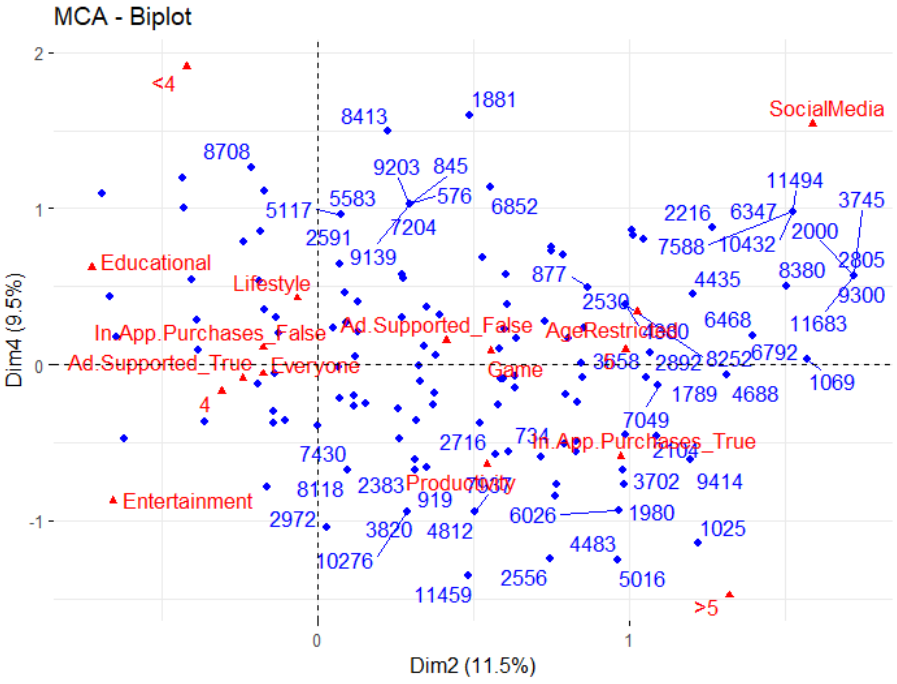


Figure 57: Biplot of individuals and variable categories in dimension 2-4

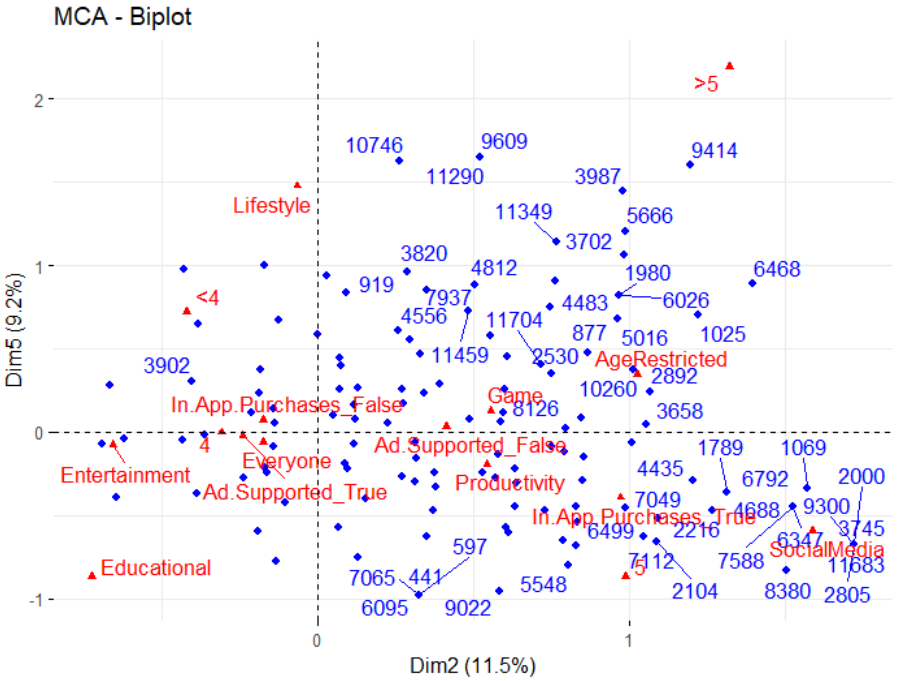


Figure 58: Biplot of individuals and variable categories in dimension 2-5

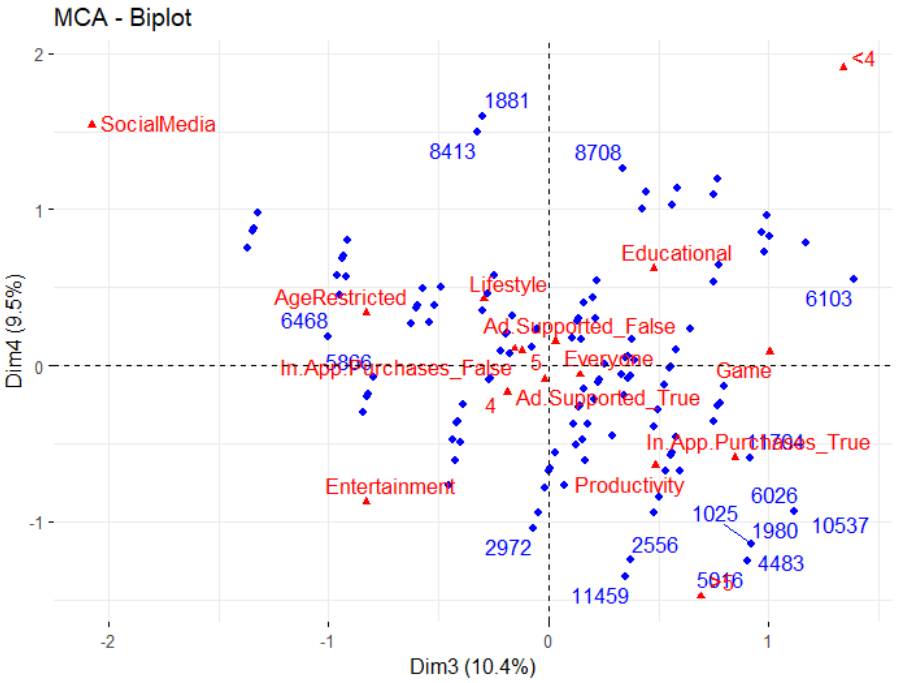


Figure 59: Biplot of individuals and variable categories in dimension 3-4

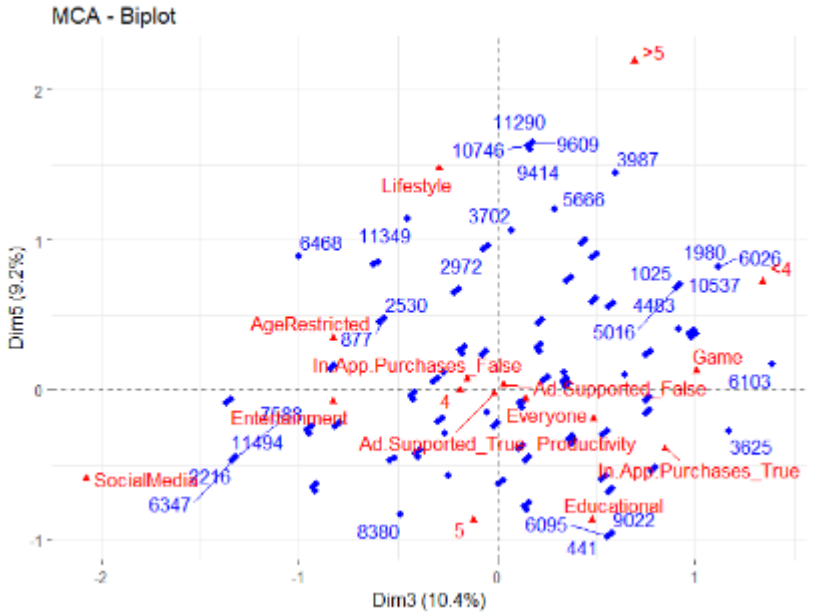


Figure 60: Biplot of individuals and variable categories in dimension 3-5

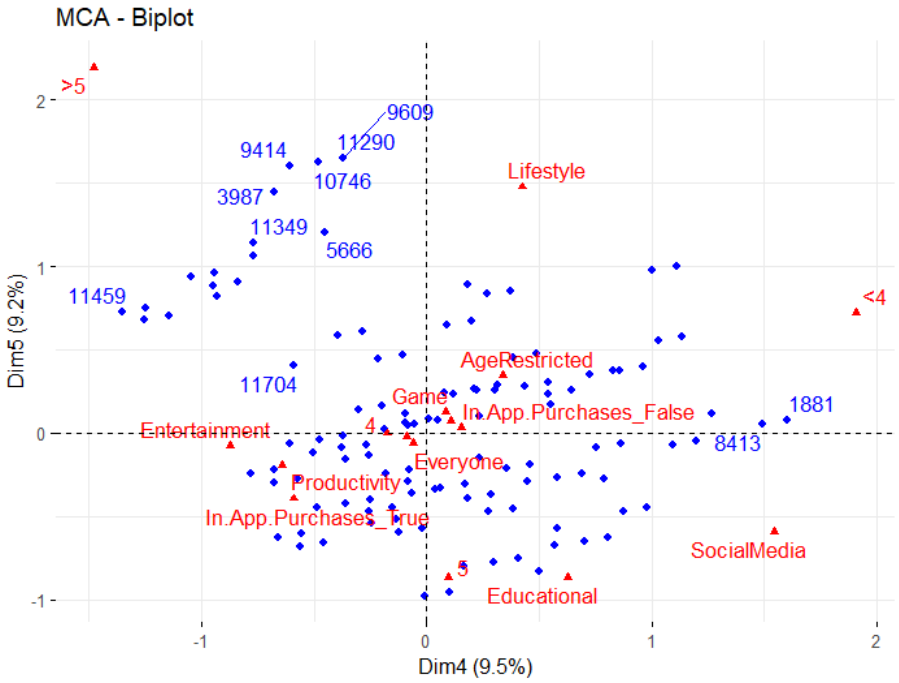


Figure 61: Biplot of individuals and variable categories in dimension 4-5

7.4 Correlation between variables and principal dimensions

In this section, we can see the correlation between variables and MCA principal dimensions for every combination of dimensions. The plots are quite similar because the selected dimensions have more or less the same inertia.

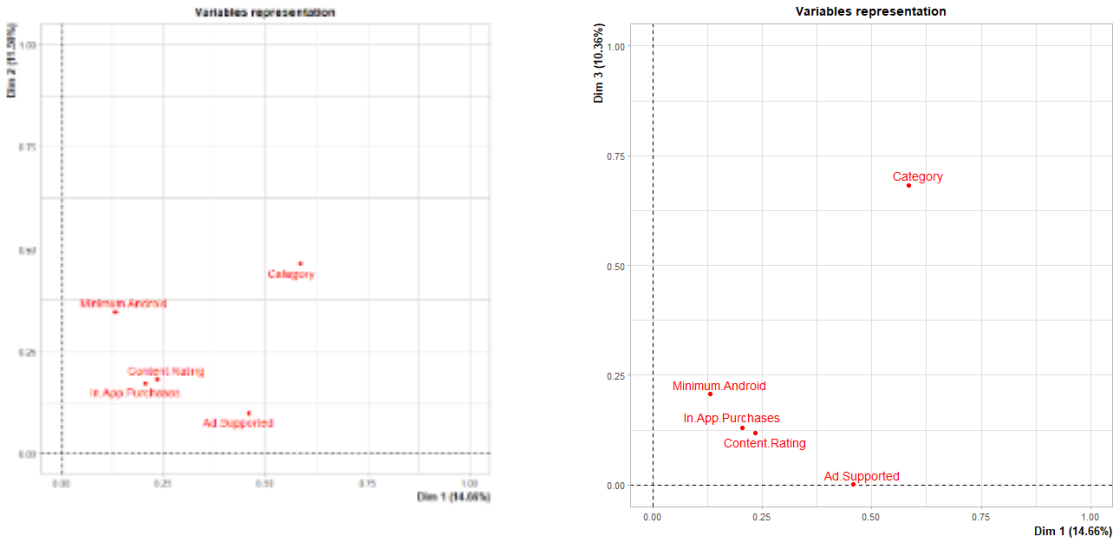


Figure 62: Correlation between variables and principal dimensions in dimension 1-2 and 1-3

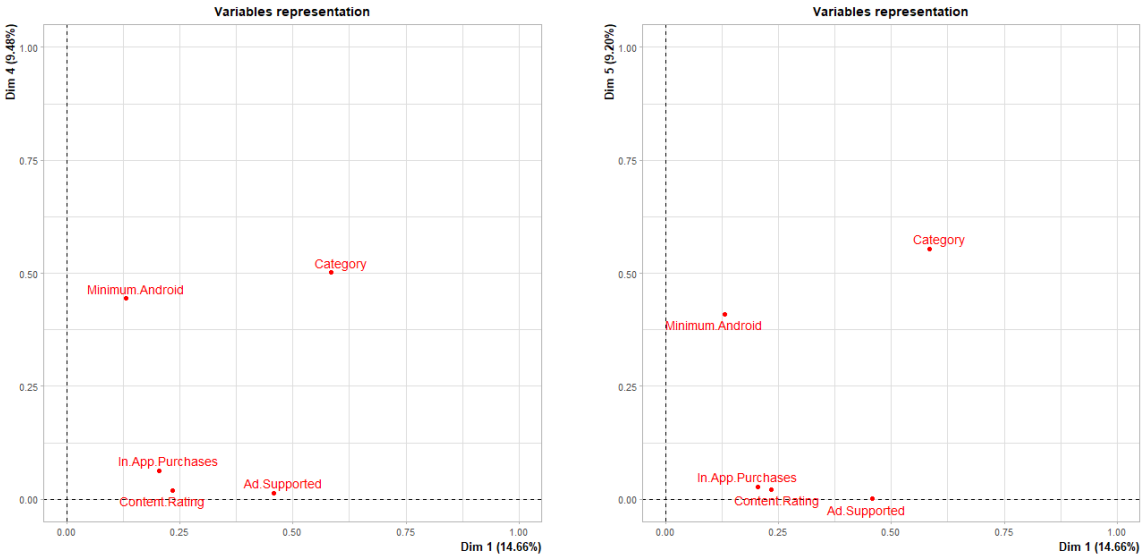


Figure 63: Correlation between variables and principal dimensions in dimension 1-4 and 1-5

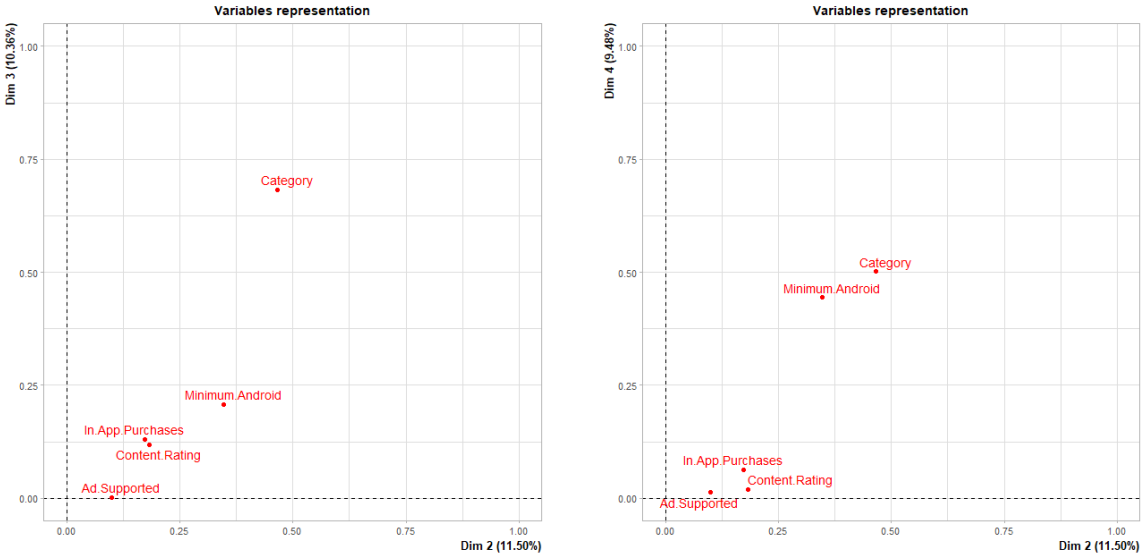


Figure 64: Correlation between variables and principal dimensions in dimension 2-3 and 2-4

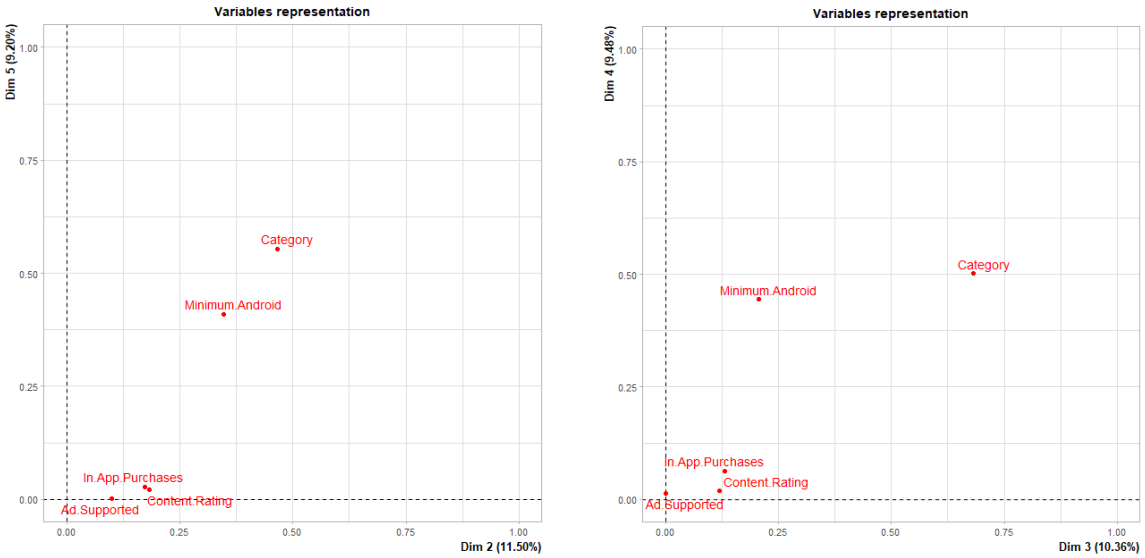


Figure 65: Correlation between variables and principal dimensions in dimension 2-5 and 3-4

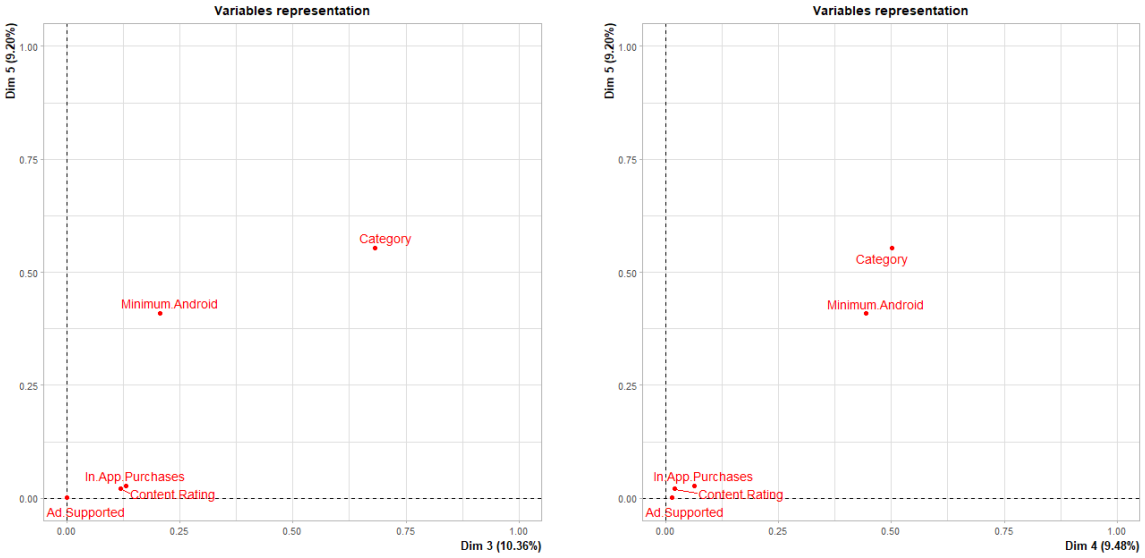


Figure 66: Correlation between variables and principal dimensions in dimension 3-5 and 4-5

7.5 Quality of representation of variable categories

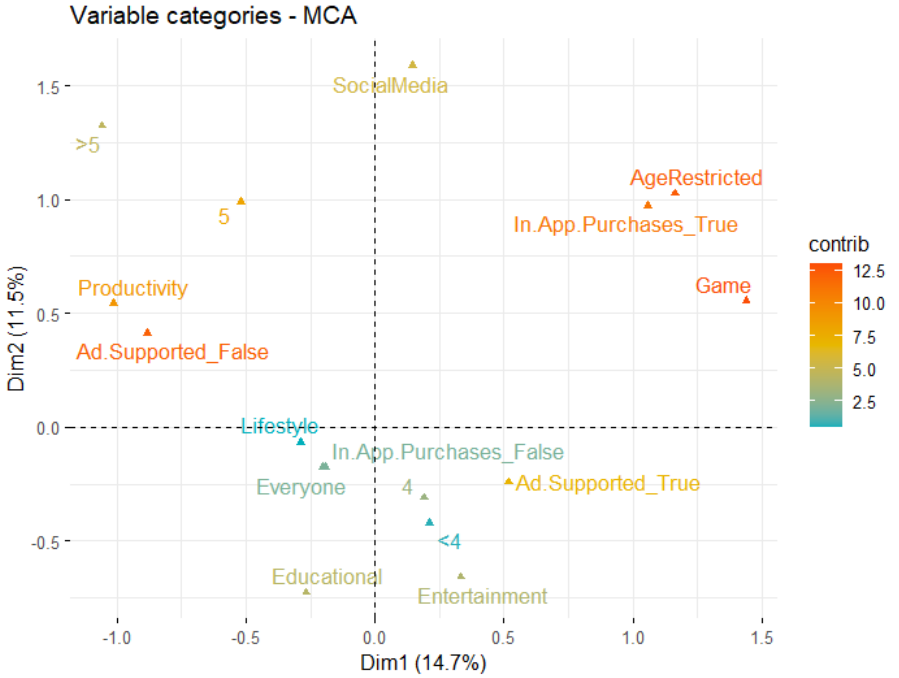


Figure 67: Quality of representation of variable categories in dimension 1-2

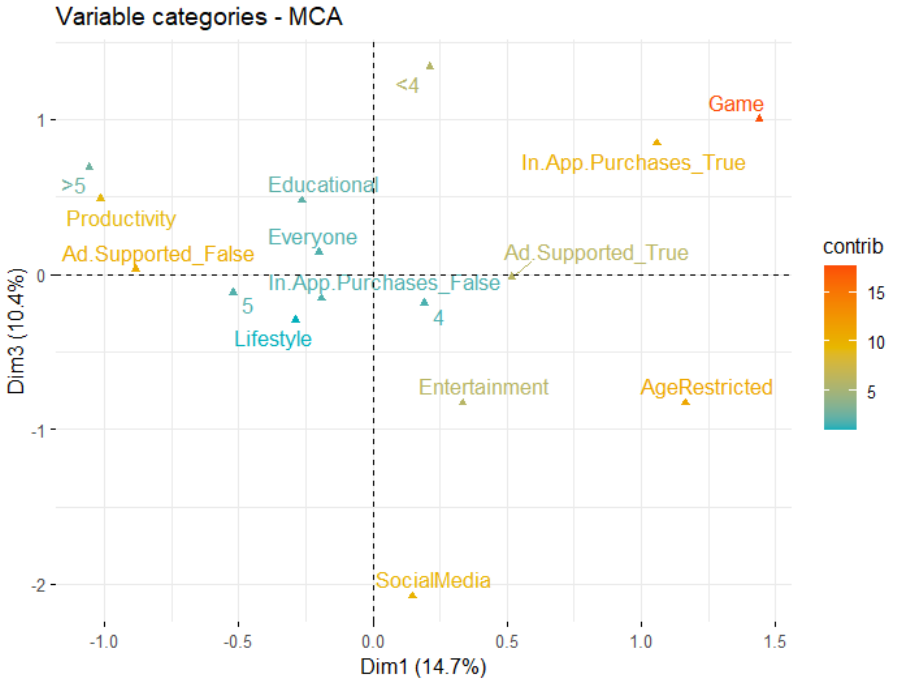


Figure 68: Quality of representation of variable categories in dimension 1-3

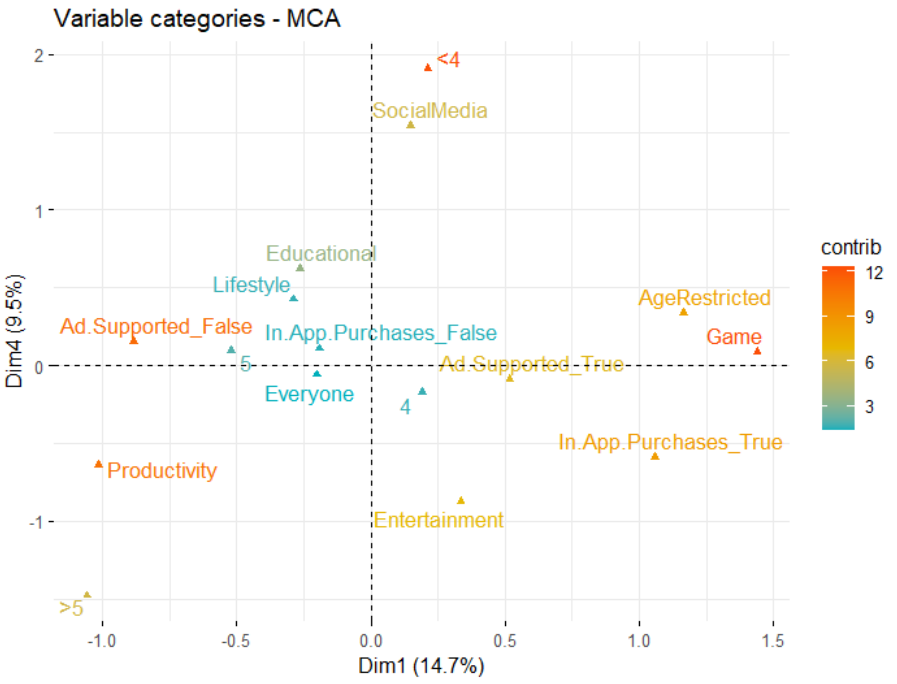


Figure 69: Quality of representation of variable categories in dimension 1-4

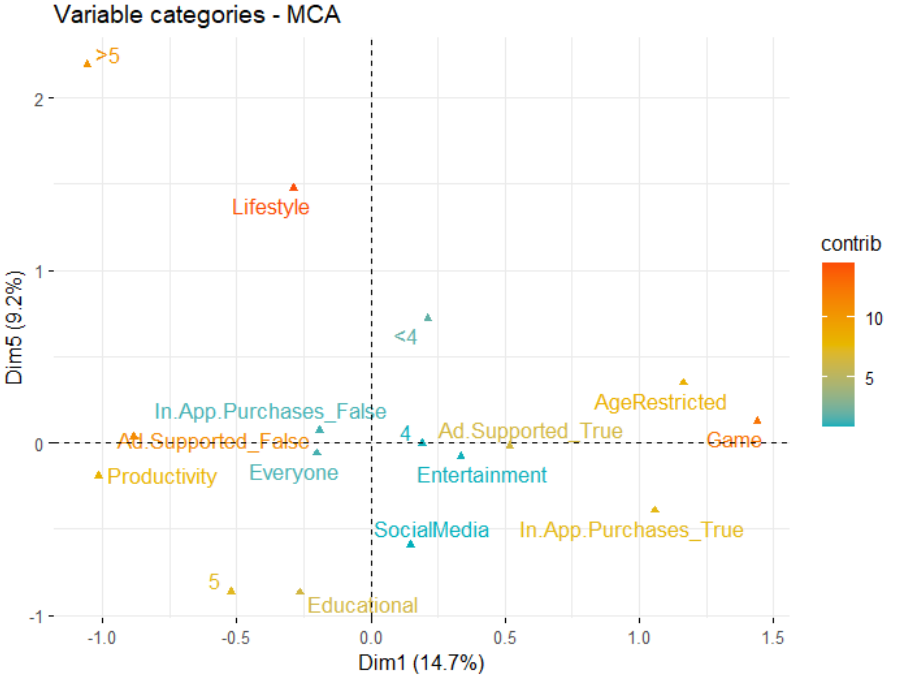


Figure 70: Quality of representation of variable categories in dimension 1-5

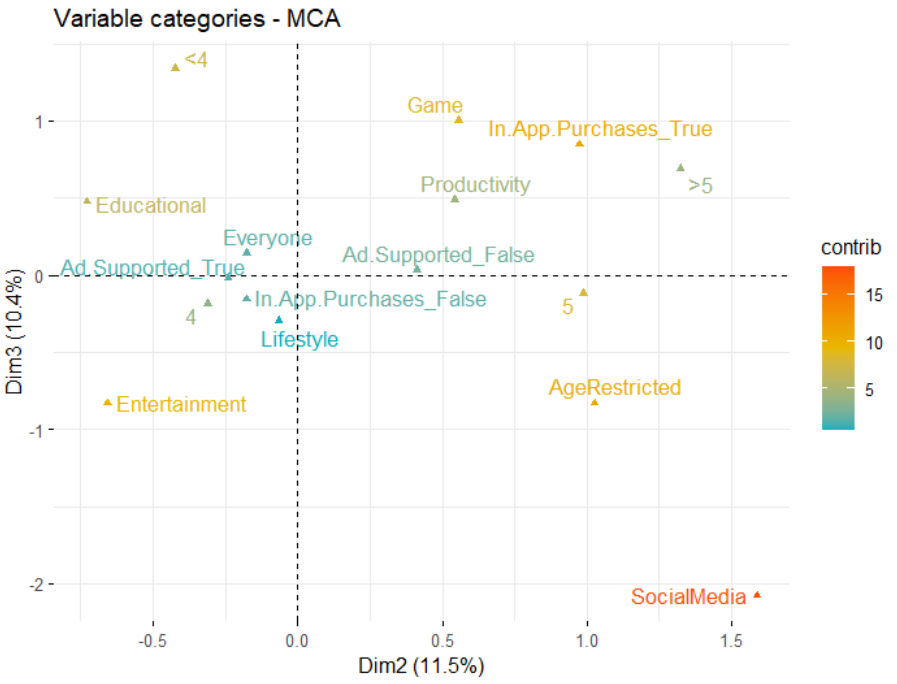


Figure 71: Quality of representation of variable categories in dimension 2-3

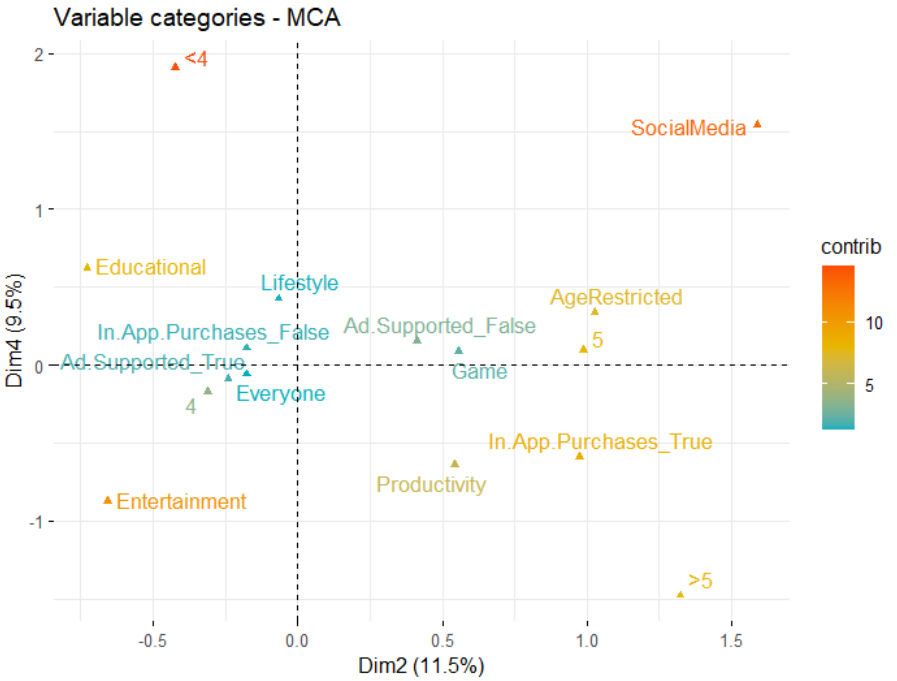


Figure 72: Quality of representation of variable categories in dimension 2-4

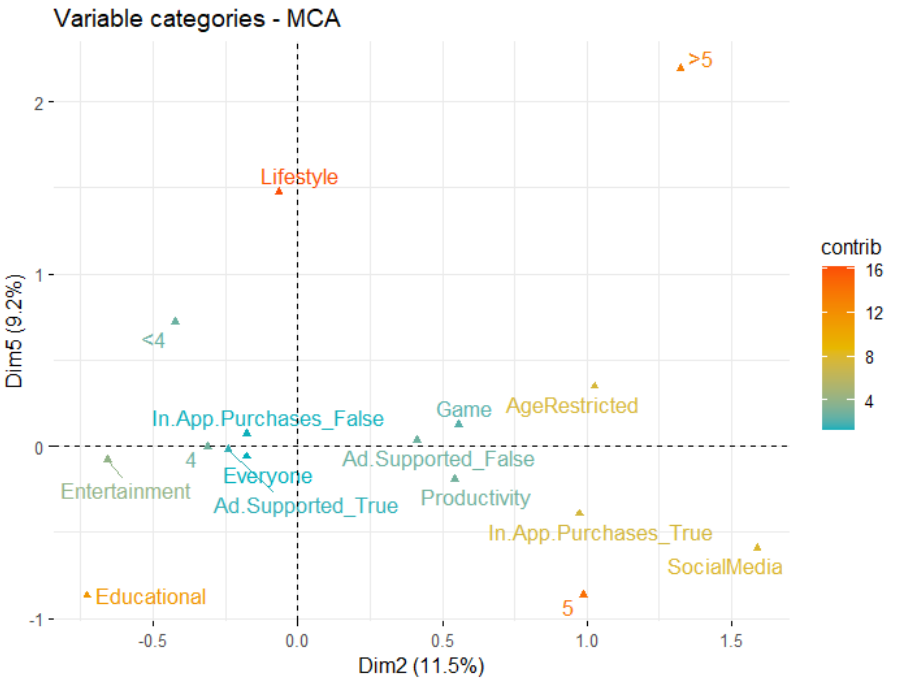


Figure 73: Quality of representation of variable categories in dimension 2-5

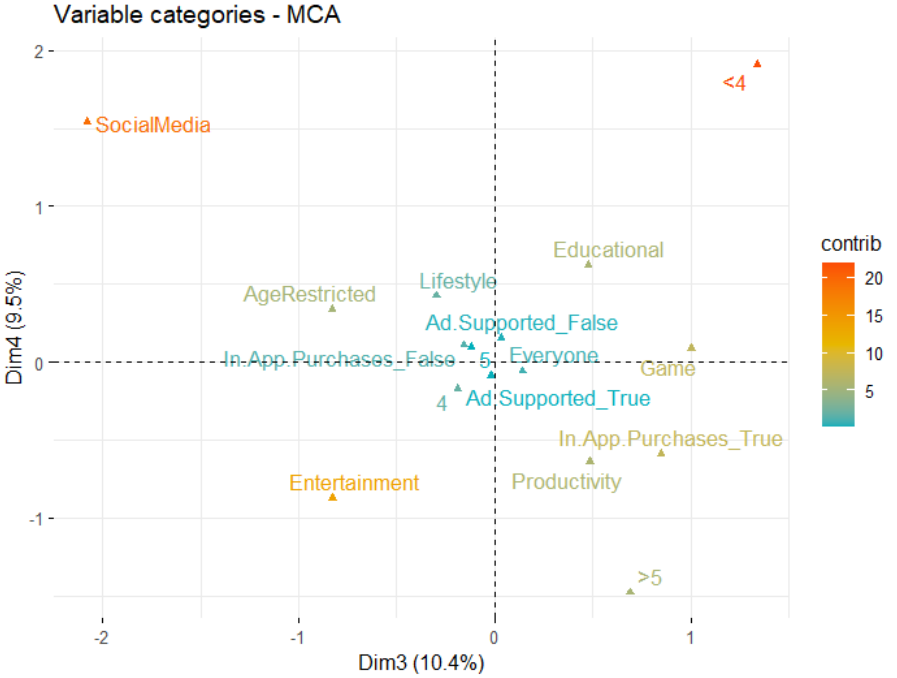


Figure 74: Quality of representation of variable categories in dimension 3-4

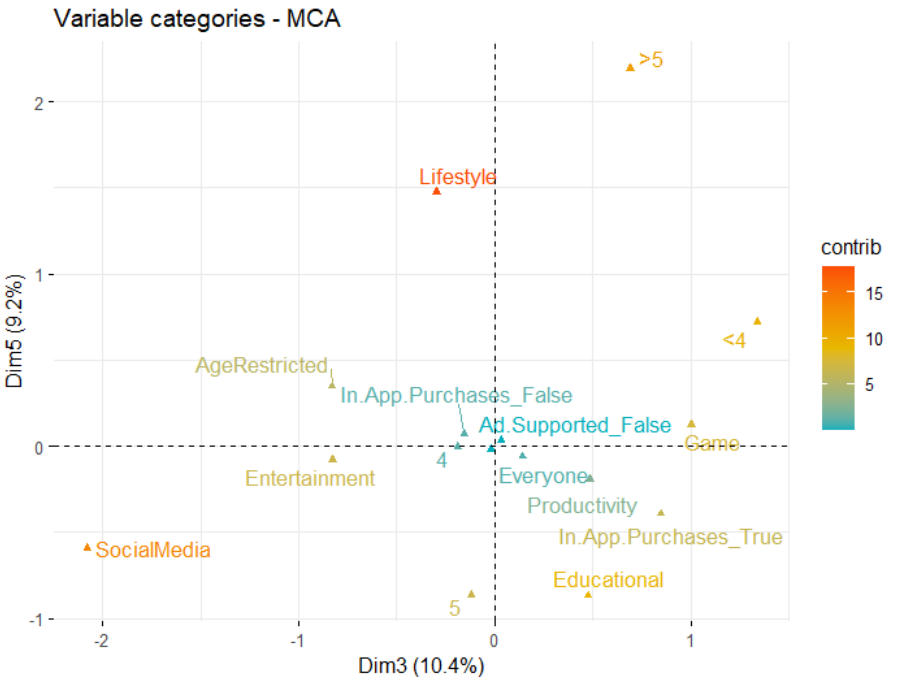


Figure 75: Quality of representation of variable categories in dimension 3-5

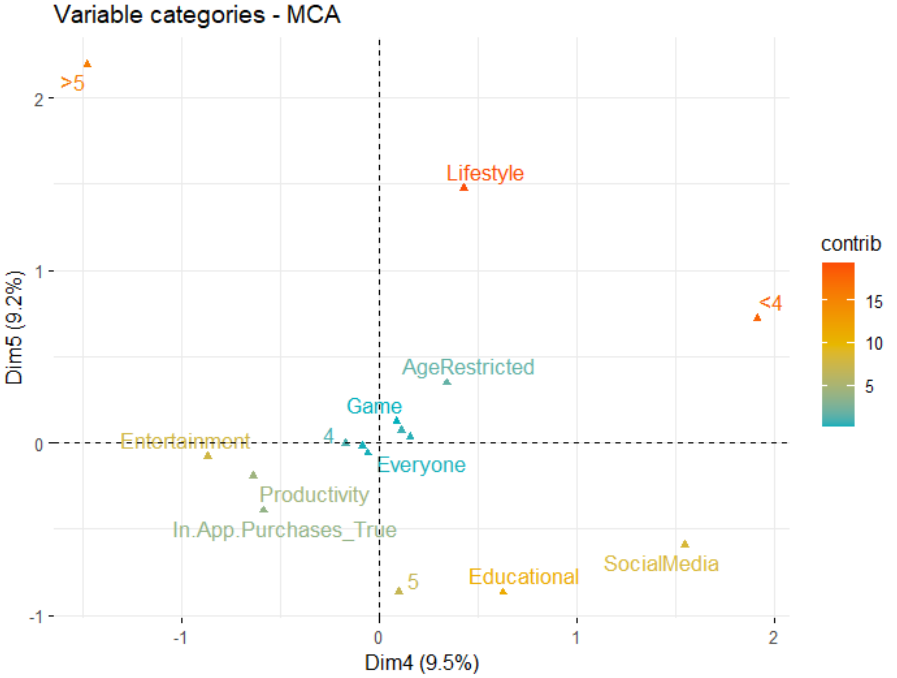


Figure 76: Quality of representation of variable categories in dimension 4-5

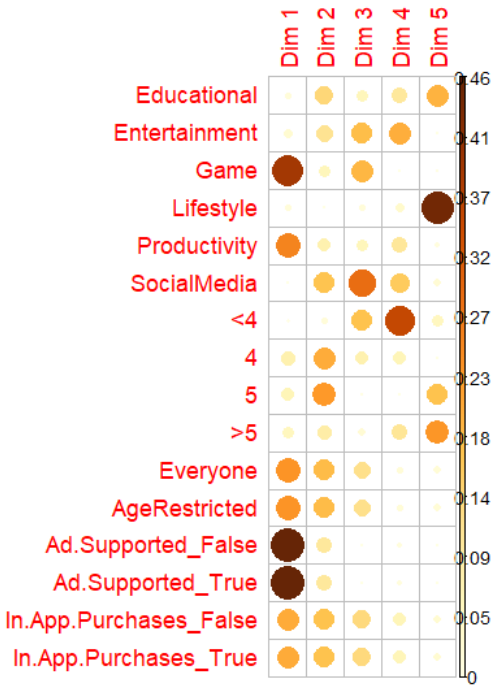


Figure 77: Correlation plot between variable categories and dimensions

Looking at the plots above, on the positive side of dimension 1 there are age restricted gaming apps with in-app purchases and ads. On the negative side of dimension 1 there are productivity apps for everyone with no in-app purchases and ads.
On the positive side of dimension 2 we have social media apps and on the negative side we have educational and entertainment apps.
On the positive side of dimension 3 we have gaming apps that require a minimum android version of less than 4, in other words, old apps and on the negative side we have social media apps.
On the positive side of dimension 4 we have old apps (minimum android version <4) and on the negative side we have new apps (minimum android version >5).
On the positive side of dimension 5 we have lifestyle apps and on the negative side we have educational apps.

7.6 Contribution of variable categories to the dimensions

The plots below show the contribution of variable categories for each dimension.

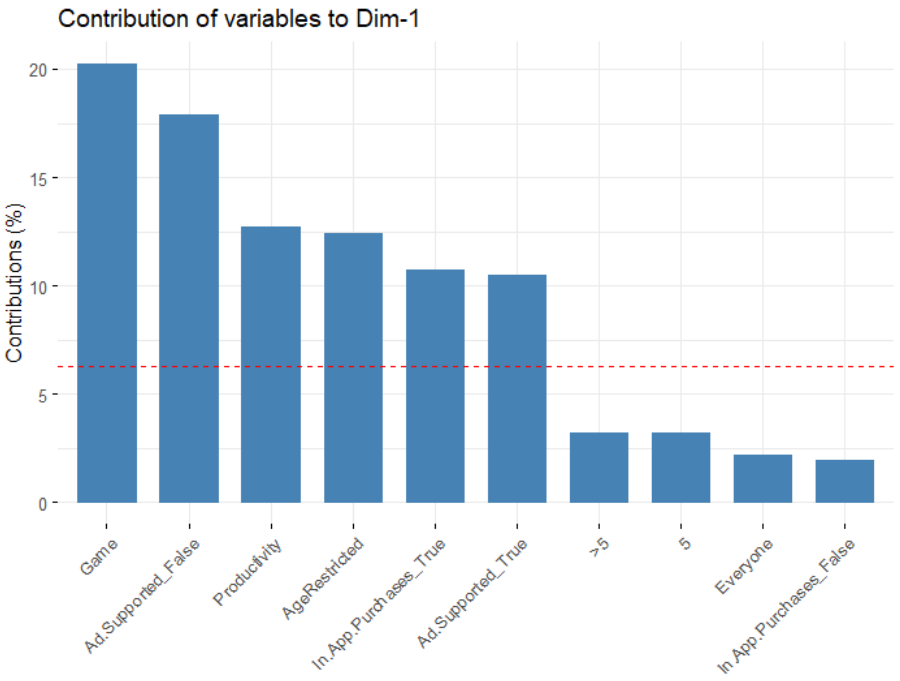


Figure 78: Contribution of variable categories in dimension 1

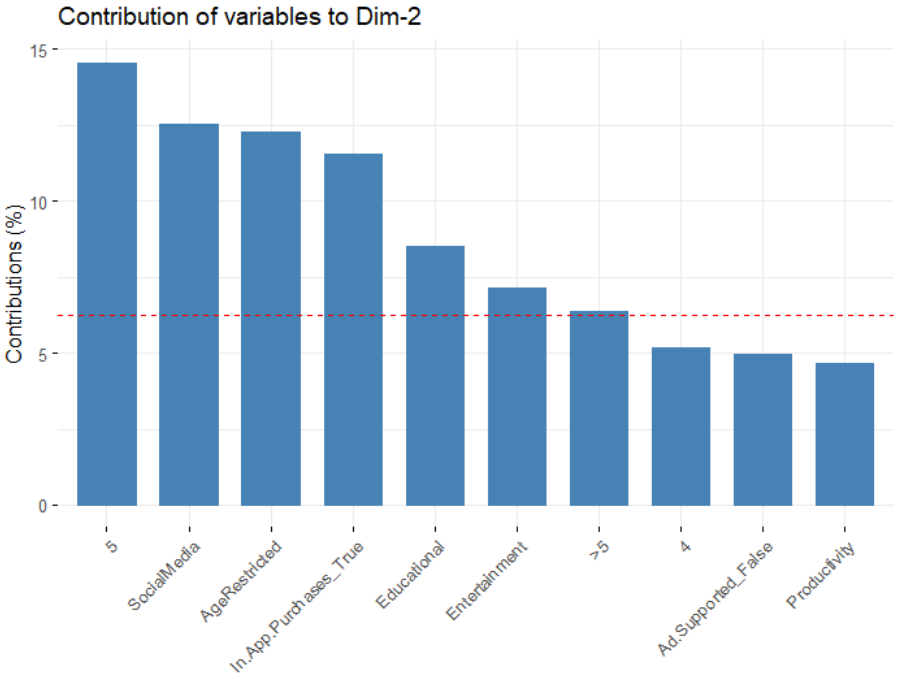


Figure 79: Contribution of variable categories in dimension 2

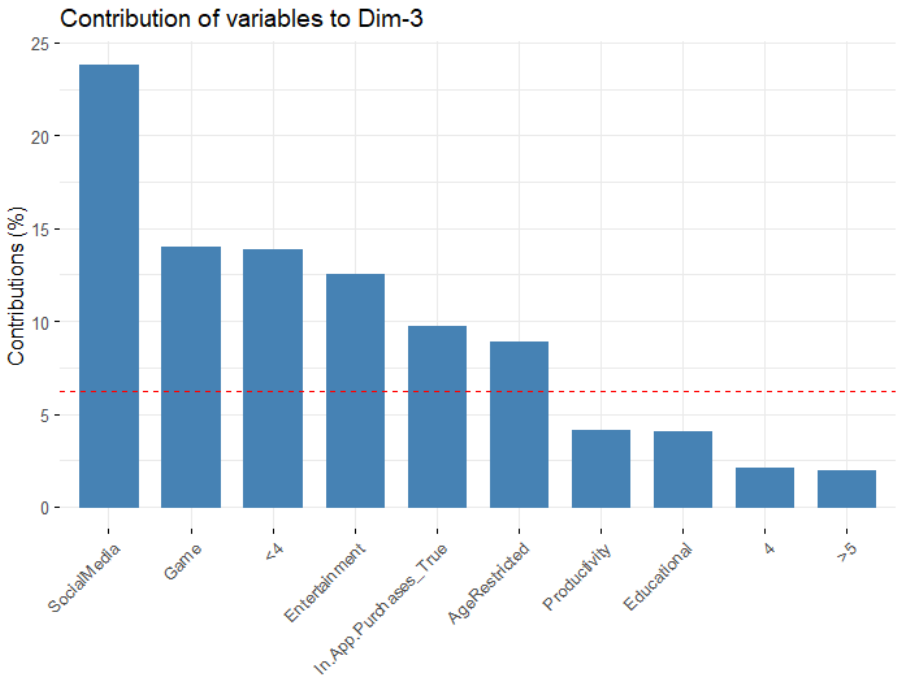


Figure 80: Contribution of variable categories in dimension 3

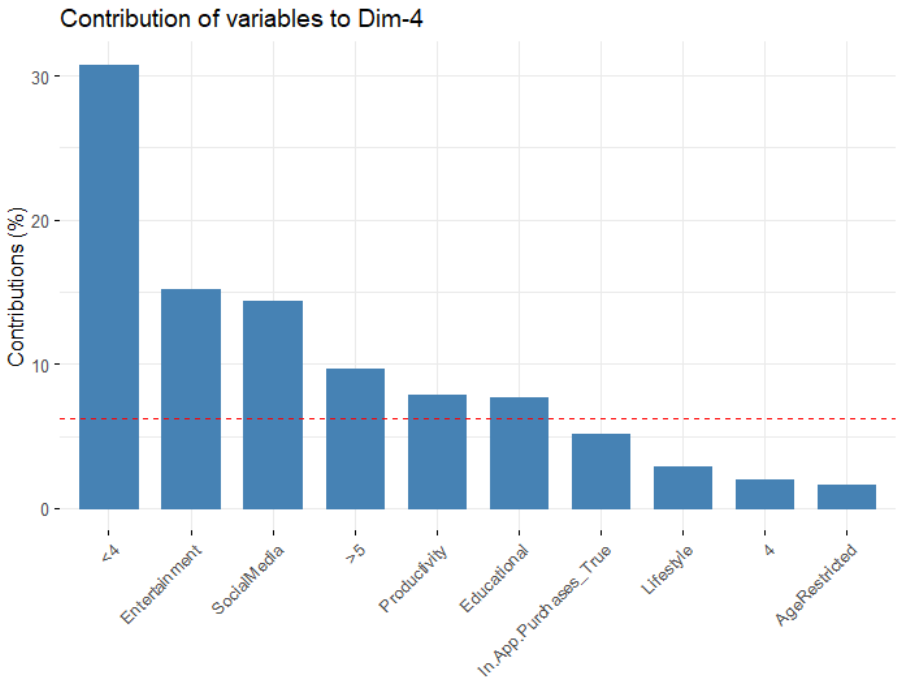


Figure 81: Contribution of variable categories in dimension 4

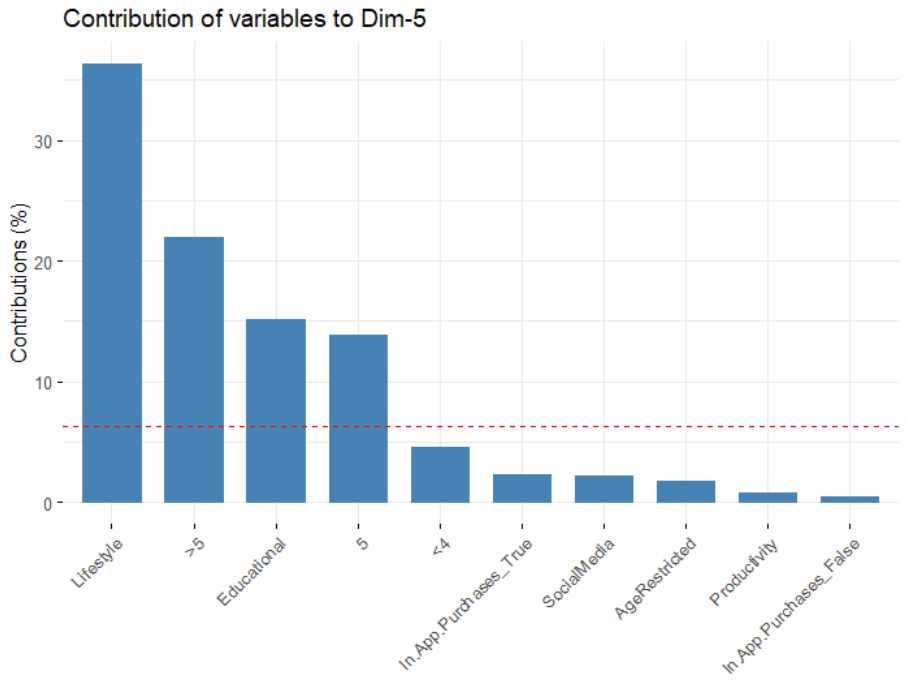


Figure 82: Contribution of variable categories in dimension 5

7.7 Color individuals by groups

Looking at the following plots we can differentiate individuals by groups if their ellipses are separated between them.
We can clearly see all the groups except for `Minimum.Android` in dimension 1-2 and `Ad.Supported` in dimensions 3-4, 3-5 and 4-5.
These plots can help us with the future clustering analysis, because we can find out the group of individuals which are similar to each other in the group but are different from the individuals in other groups.

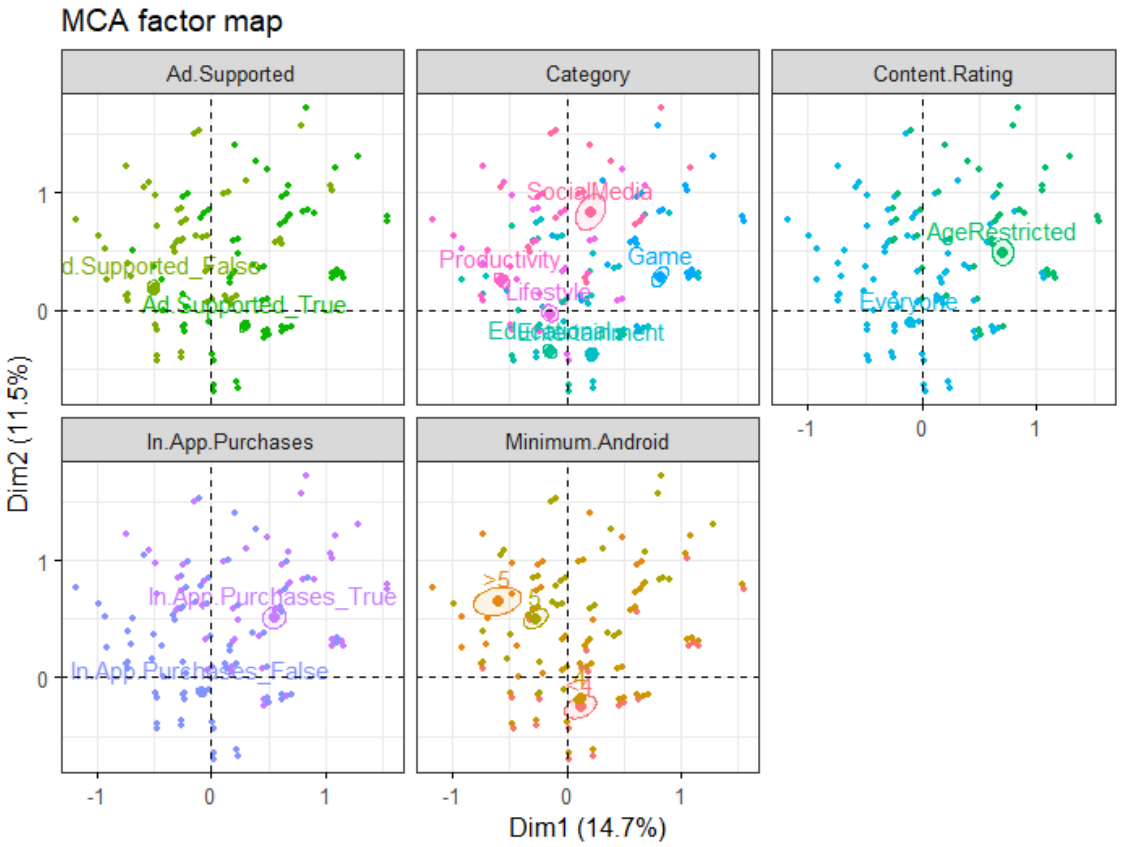


Figure 83: Groups of individuals for each variable in dimension 1-2

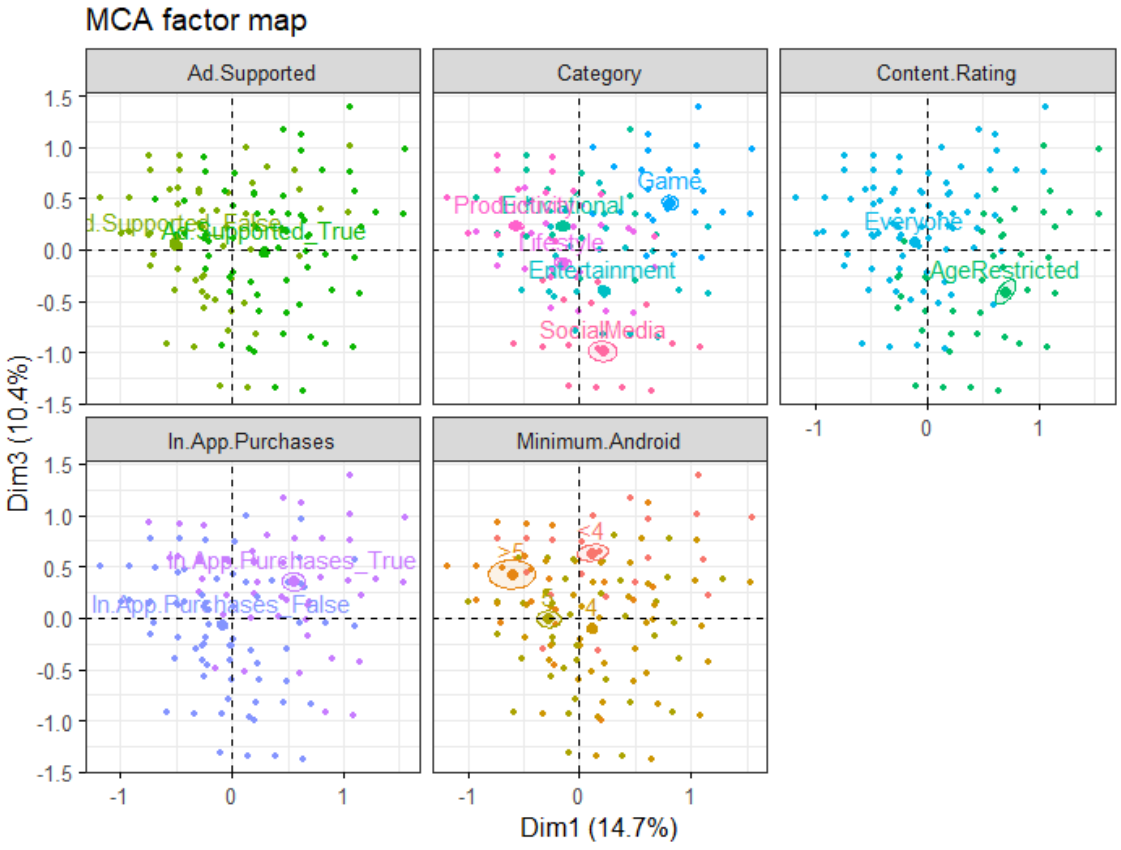


Figure 84: Groups of individuals for each variable in dimension 1-3

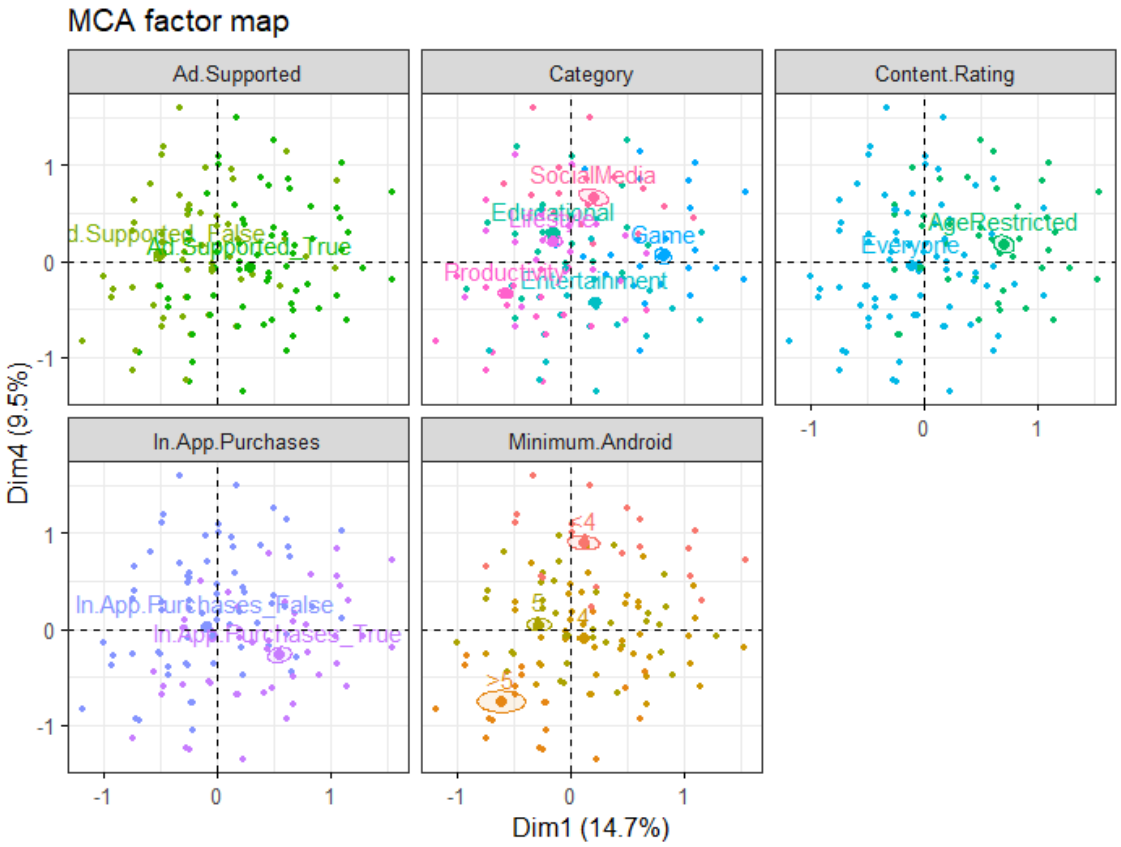


Figure 85: Groups of individuals for each variable in dimension 1-4

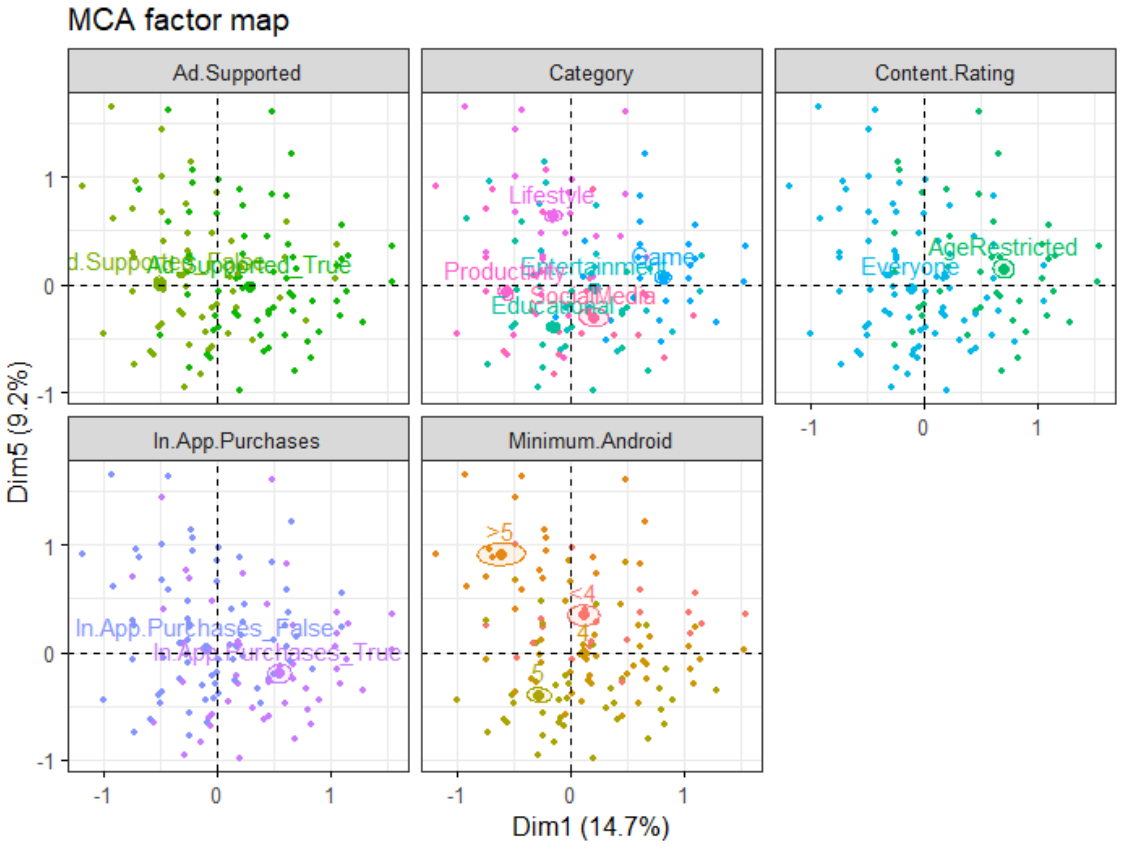


Figure 86: Groups of individuals for each variable in dimension 1-5

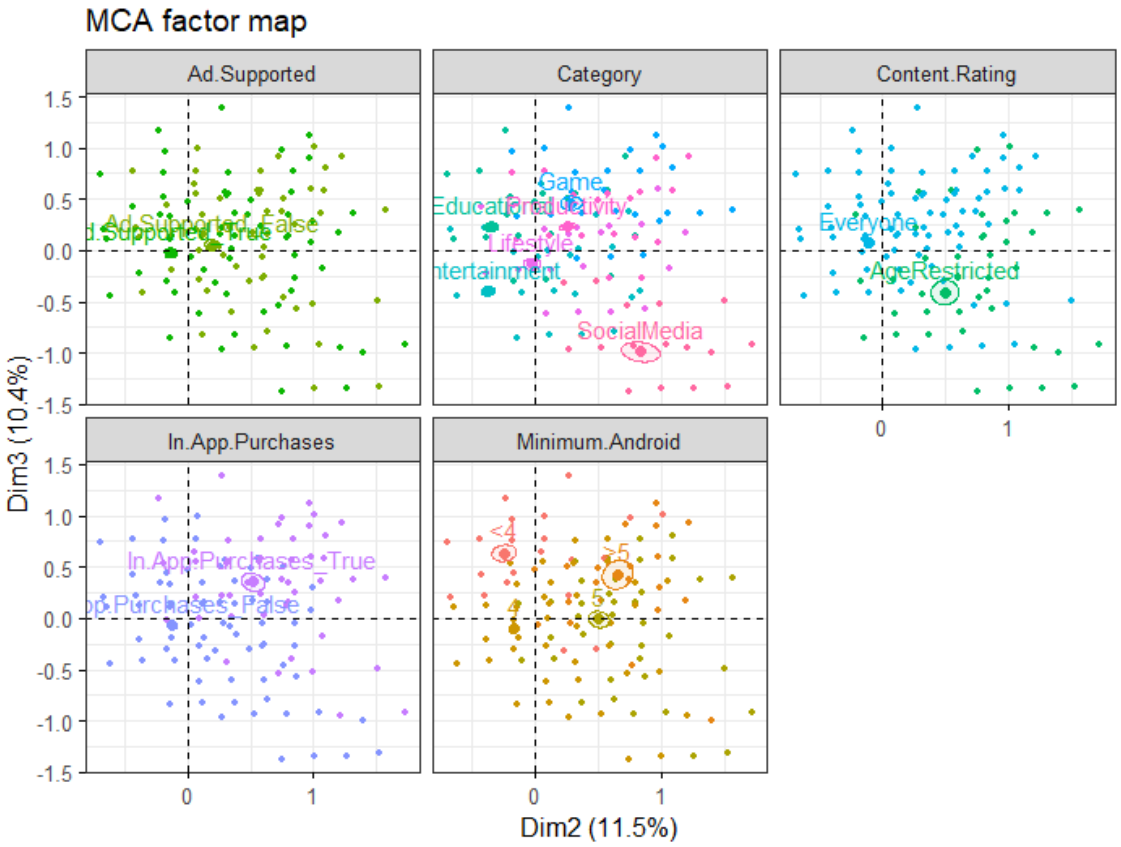


Figure 87: Groups of individuals for each variable in dimension 2-3

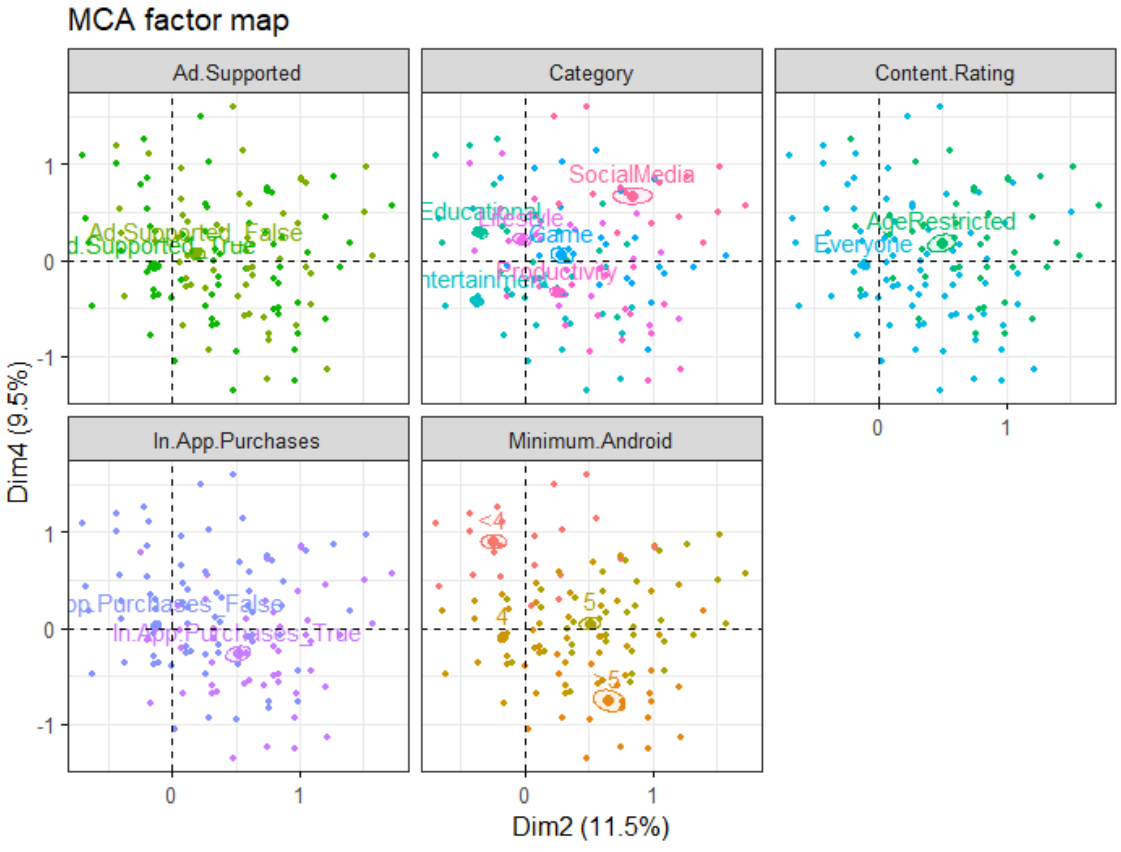


Figure 88: Groups of individuals for each variable in dimension 2-4

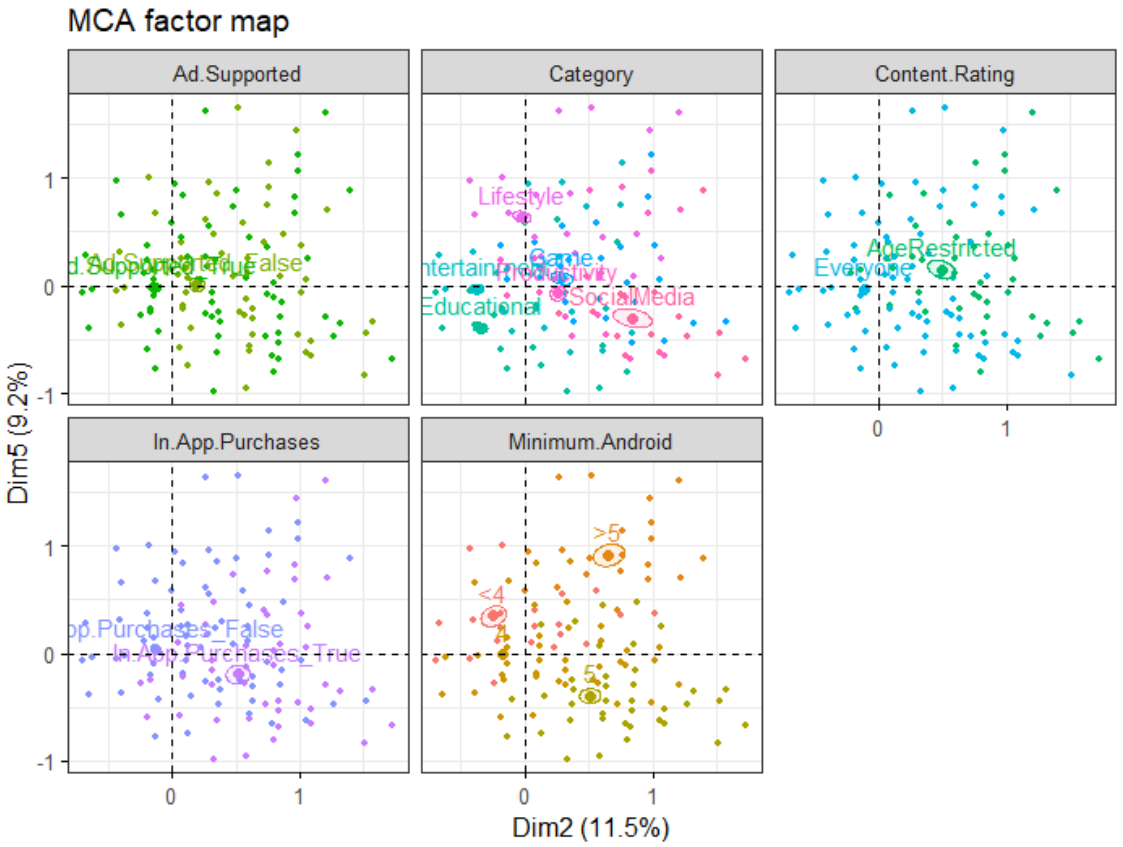


Figure 89: Groups of individuals for each variable in dimension 2-5

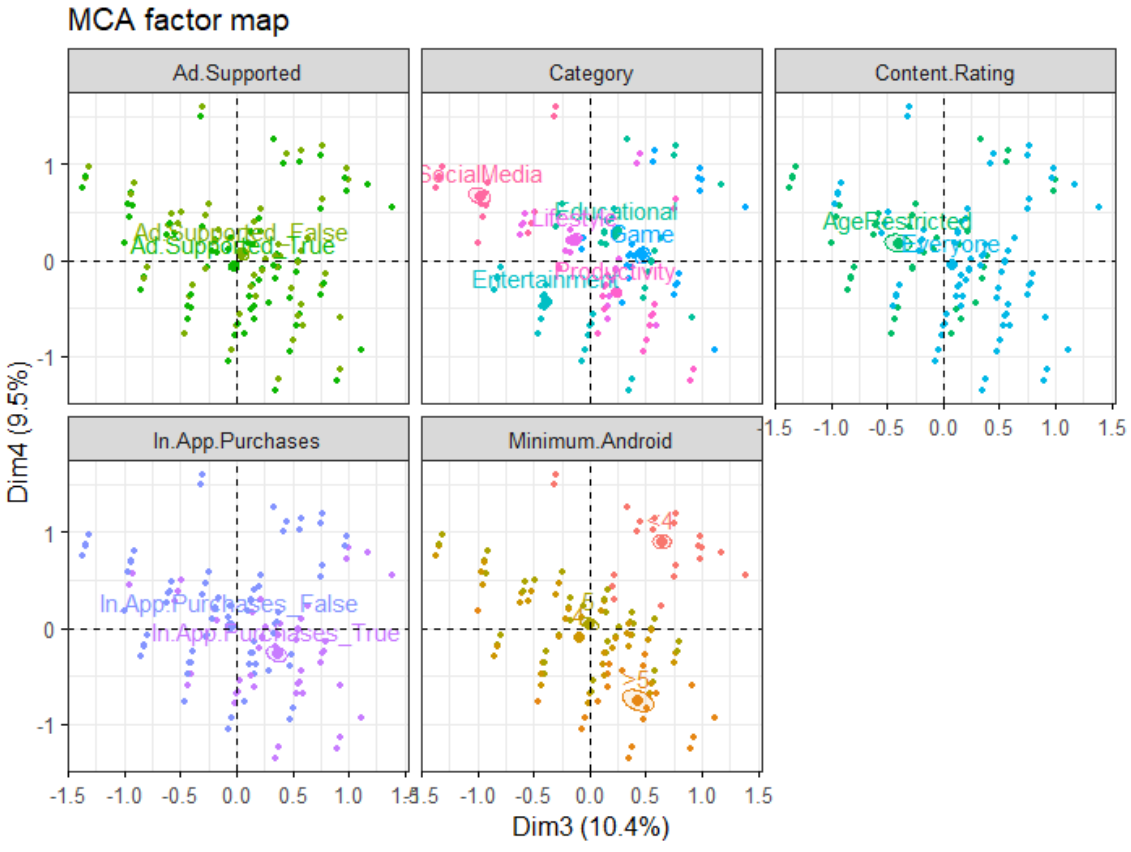


Figure 90: Groups of individuals for each variable in dimension 3-4

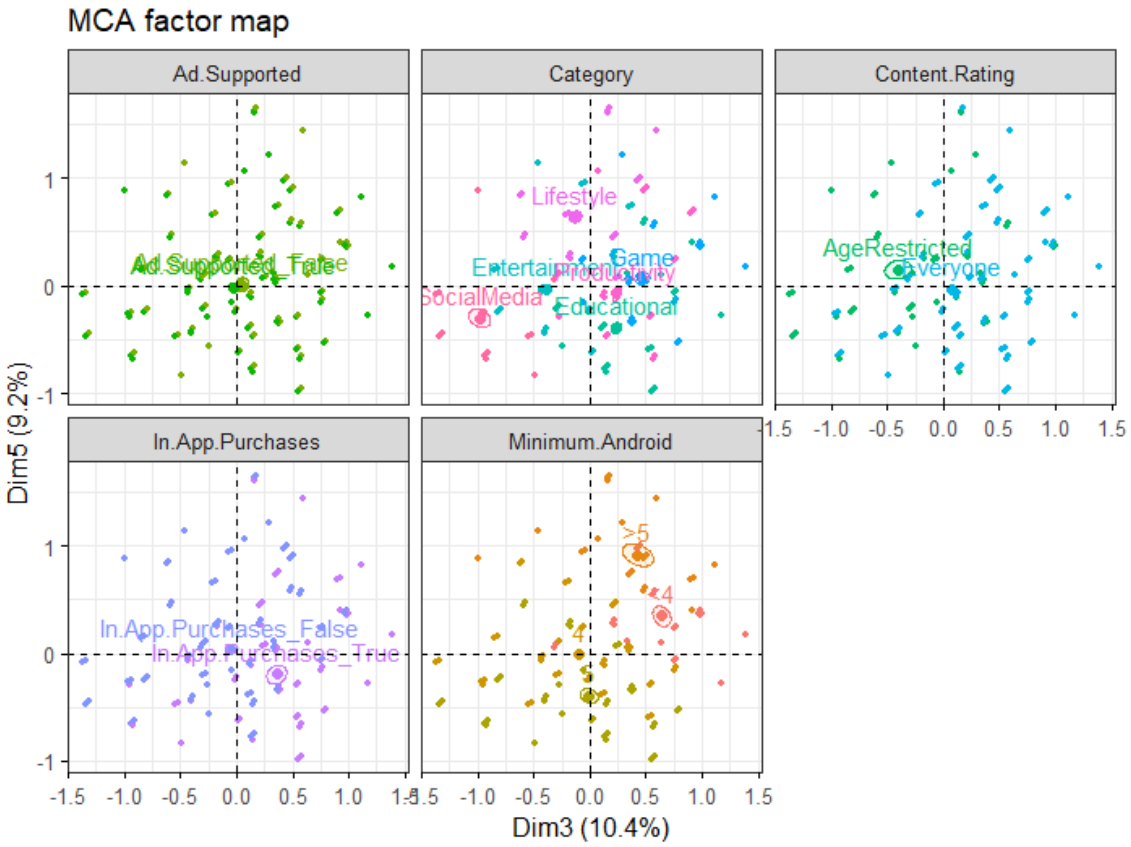


Figure 91: Groups of individuals for each variable in dimension 3-5

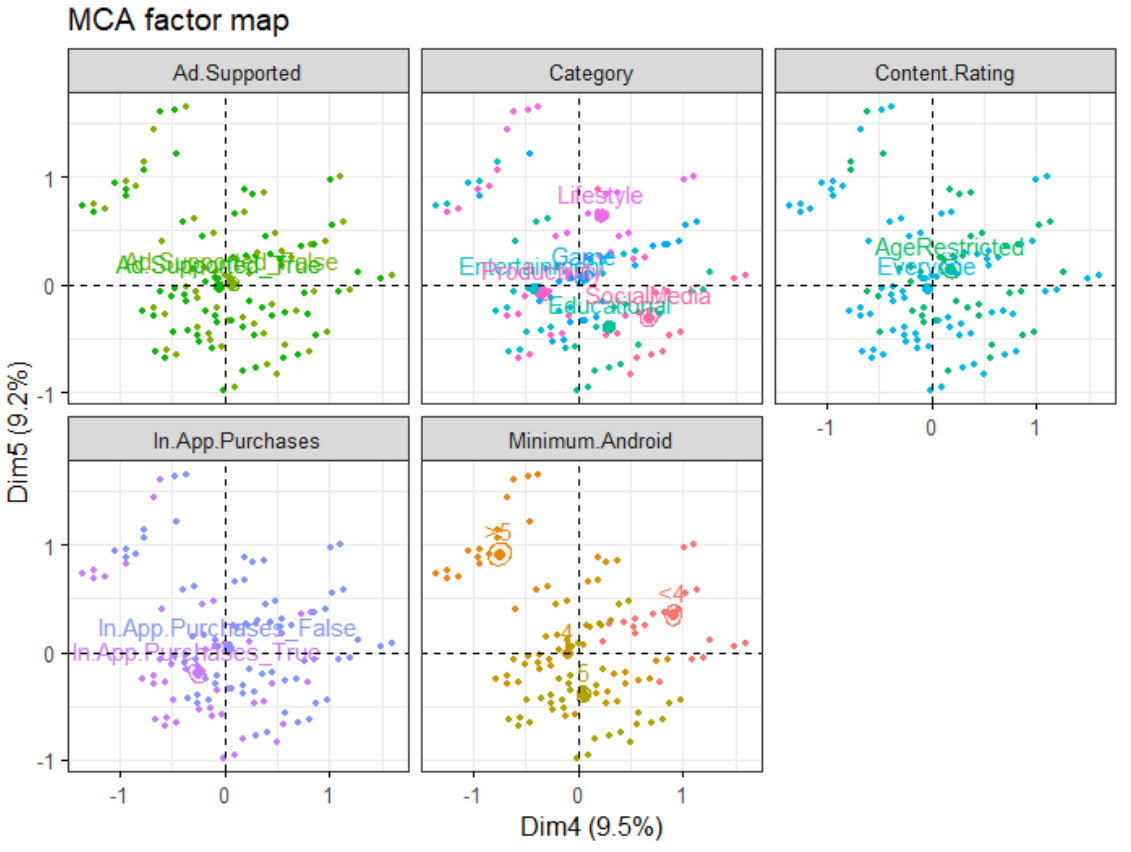


Figure 92: Groups of individuals for each variable in dimension 4-5

7.8 Conclusions

In this section, we are going to try to label our dimensions after performing MCA.
Dimension 1 can be the **level of entertainment** of an app, as on the positive side there are age restricted gaming apps with in-app purchases and ads and on the negative side there are productivity apps for everyone with no in-app purchases and ads.
Dimension 2 can be the **level of procrastination** of an app, as on the positive side we have social media apps and on the negative side we have educational and entertainment apps.
Dimension 3 can be the **level of companionship** of an app, as on the positive side we have gaming apps and on the negative side we have social media apps. Normally, we feel more connected to society using social media apps rather than gaming apps.
Dimension 4 can be the **longevity** of an app, as on the positive side we have old apps (minimum android version <4) and on the negative side we have new apps (minimum android version >5).
Finally, dimension 5 can be **the helpfulness in a person's lifestyle**, as on the positive side we have lifestyle apps (like diet and gym apps) and on the negative side we have educational apps.

8. Multiple Factor Analysis

Multiple Factor Analysis is a factorial method that allows analyzing data with mixed features typed classified in groups. This analysis was performed using the `FMA()` function offered by the `FactoMiner` Package. In order to perform a FAM analysis, we must group the features of the dataset in order to study the data. In total, there were five groups created that describe a specific aspect of the data:

- **Group 1 [Antiquity]:** describes the antiquity of the apps, includes two numerical variables, `DaysLastUpdate` and `ReleasedDays`.
- **Group 2 [Popularity]:** this group also includes numerical features, `Rating.Counts`, `Installs`, and `Rating`, and describes the popularity of the apps.
- **Group 3 [App Features]:** includes two features, `AppNameLen` and `Size`, which express the length of the apps' names and the size in memory.
- **Group 4 [Topic]:** this group includes factorial features, `Category`, `Content.Rating` and `Minimum.Android`, which state the categorization of the apps.
- **Group 5 [Monetization]:** specifies the monetization of the apps, includes two logical features, `Ad.Supported` and `In.App.Purchase`.

The first and fourth groups, `Antiquity` and `Topic`, were declared as supplementary groups, while the rest of the groups remain as active groups. Antiquity and the numeric features were scaled during the analysis in order to compare them in the same units and obtain meaningful results. As a result, we obtain 100% of accumulated variance with 7 dimensions, Figure 93.

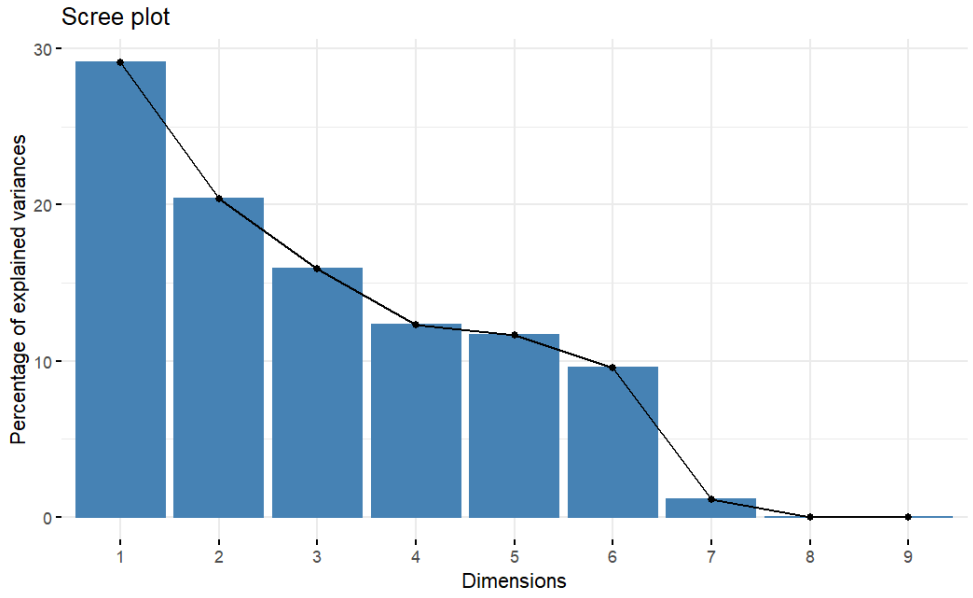


Figure 93: Percentage of explained variances

In the following section, we analyze the results taking the first three dimensions, which achieve 65.35% of cumulative variance.

8.1 Dimensions analysis

In this section, we study the contribution of each active group, and their variables, to each one of the three main dimensions. The plots below show the contribution of each group for each dimension. All the dimensions share the same groups, the differences reside in the contribution of each group in each dimension, but have different contributions. In the case of dim1, the group with the higher contribution is `Monetization`, in dim2 `App Features`, and dim3 `Popularity`. In all dimensions, we can see that the group `Topic` and `Antiquity` do not have a strong contribution compared with the other groups.

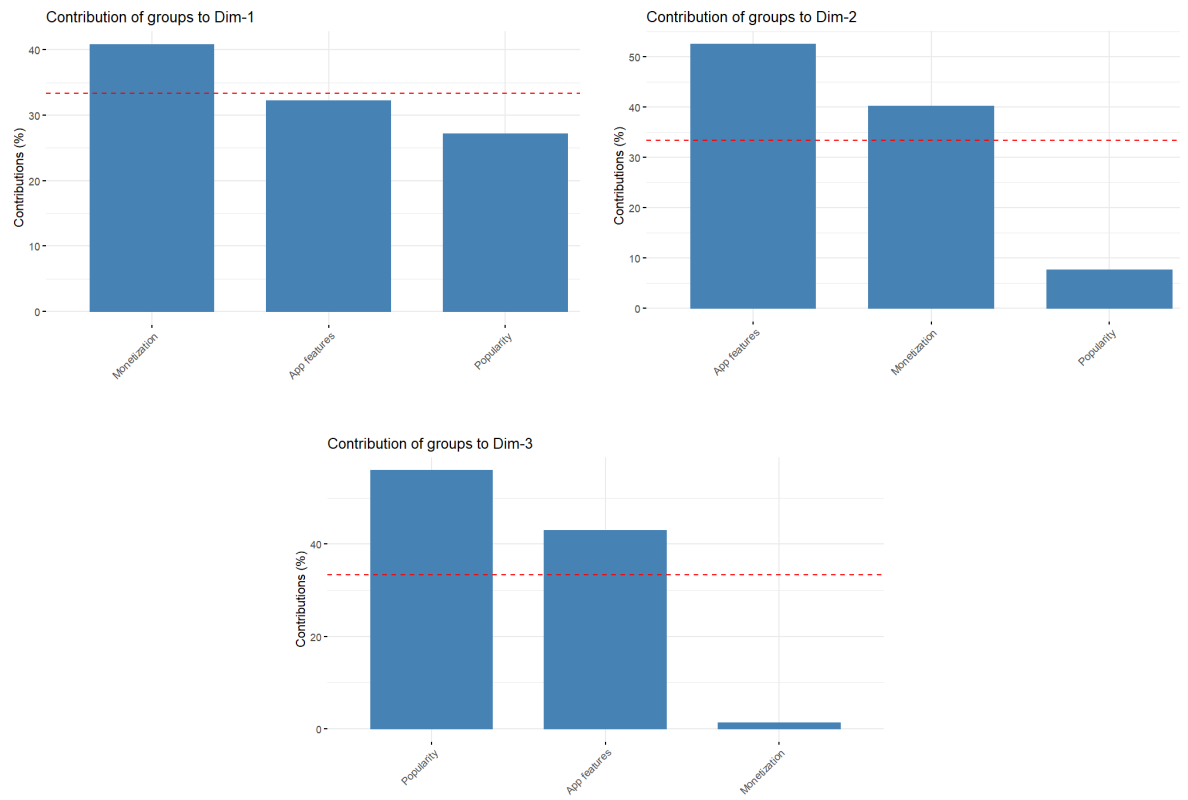


Figure 94: Groups contribution to dim1, dim2 and dim3

Meanwhile, the plots below show the contribution of each feature for each dimension. Variables that contribute the most to dimX are the most important in explaining the variability in the dataset. As we can see in Figure 95, there are variables that contribute more to a specific dimension and less to others. `AppNameLen` has the highest contribution in two dimensions: dim1 and dim2. Another important variable is `Installs`, which appears in dim1 and dim3 as the second main contributor. Another important variables are `Size` and `Rating.Count`, which appear in multiple dimensions as contributors.

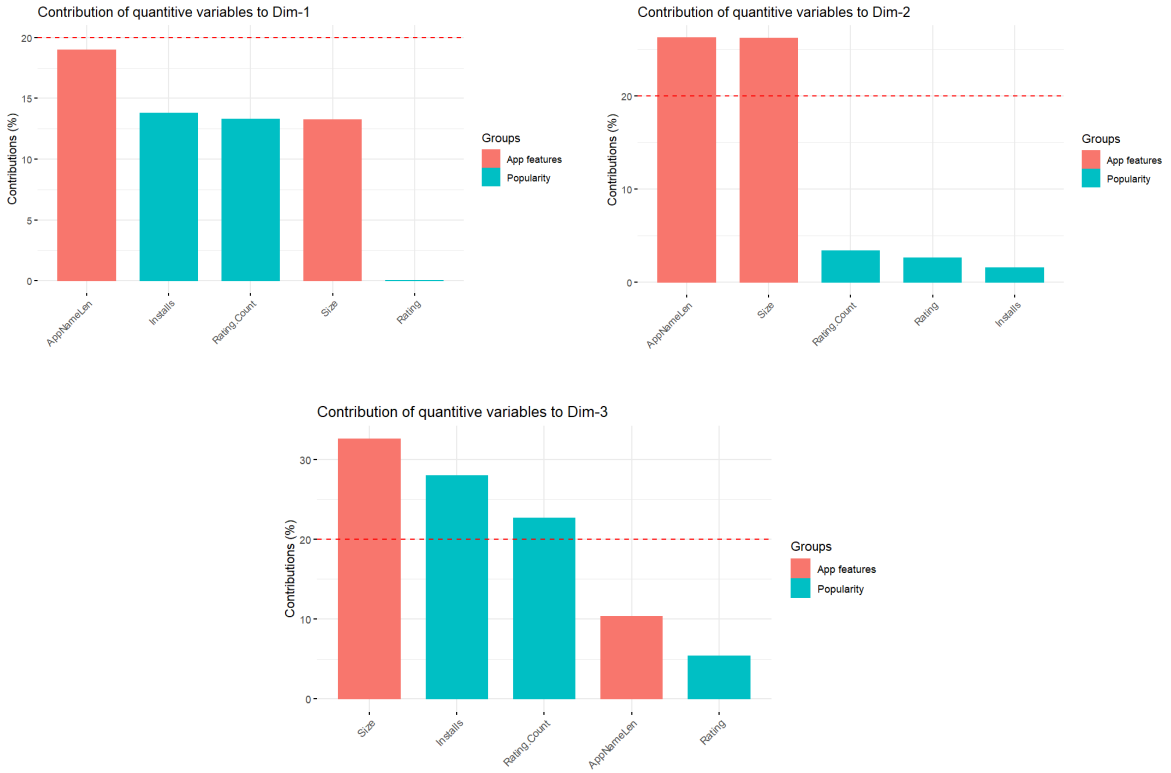


Figure 95: Variables contribution to dim1, dim2 and dim3.

After studying each dimension separately, next we will analyze dimensions by pairs: dim1 & dim2, dim1 & dim3, and dim2 & dim3. In the analysis, we make use of synthetic plots to compare the groups and variables contribution.

- **DIMENSION 1 & 2**

From the perspective of groups, the main contributors of dim1 are, in order of importance, `Monetization`, `App Features` and `Popularity`. While for dim2 are `App Features`, `Monetization` and `Popularity` Figure 96. From the perspective variables, we can see that both dim and dim2 have correlation with `AppNameLength`, as we saw before. Other main contributors for dim1 are `Install` and `Rating.Count`, while for dim2 is `Size`.

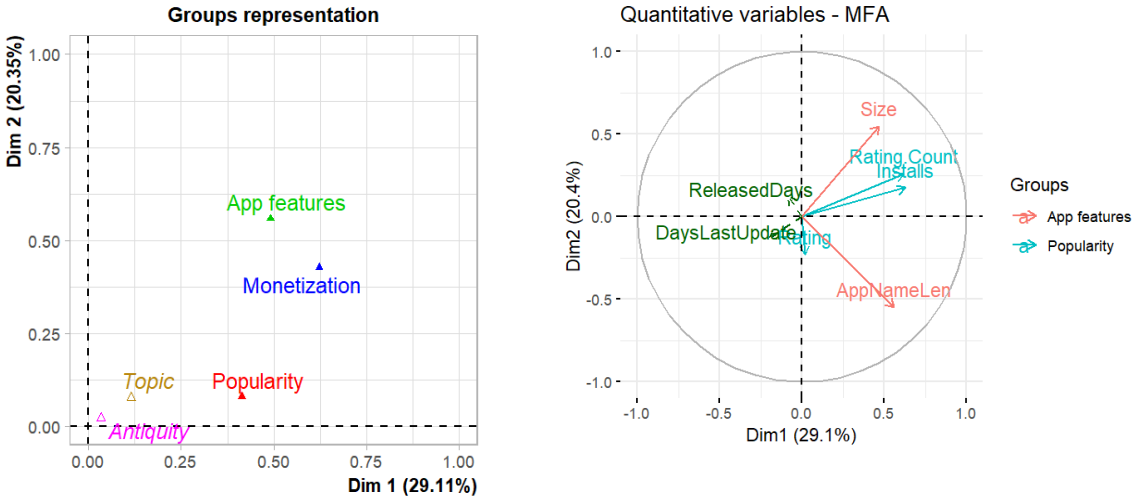


Figure 96: Groups and variables contribution to dim1 and dim2

The plot Figure 96 shows that there is a strong correlation between `Installs` and `Rating.Count`, this last variable also has a light correlation with `Size`. Moreover, `Size` and `AppNameLen` seem to be independent variables. So, the variables of the group `Popularity` are correlated, except `Rating`. And the variables of the `App Features` are independent.

Figure 97 shows the plot of the top 1500 individuals with the highest contribution to dim1 and dim2. We can see a dense cloud of points on the positive axis of dim1. Those individuals have a strong popularity and recent updates. We can also see small cloud points in the negative dim1 side, with a poor popularity and old updates. Other small clusters appear in dim2 negative axis, those are the individuals with higher ratings and middle-long names. This means that apps with shorter names are more popular but have lower ratings, while the unpopular ones have better ratings. This may be explained by the number of ratings. So, the rating does not completely define the popularity of an app, but the number of users does.

We also study how the individuals are seen by different groups. Our aim is to analyze if individuals are seen in the same way by different groups, or there are individuals specific to some groups. As we can see in the plot of Figure 98, the groups, specialty `App Features`, have a different perspective view in respect to the other groups.

The difference between Figure 97 and Figure 98, is that the first one plots the point that corresponds to the center of gravity of the partial points of the individual. That is, the individual viewed by all groups of variables. While Figure 98 plots how each app is viewed by each group and its barycenter.

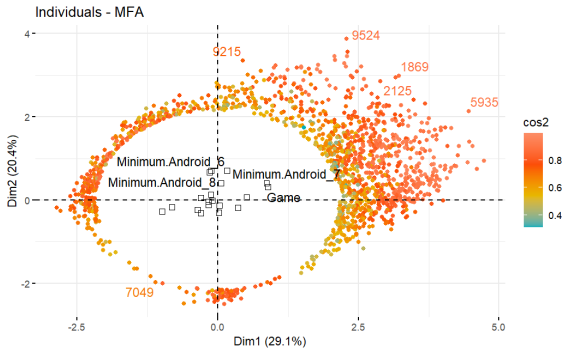


Figure 97: Individuals contribution to dim1&2

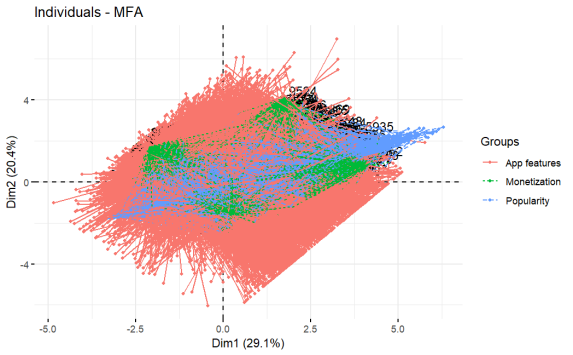


Figure 98: Partial points representation dim1&2

- **DIMENSION 1 & 3**

Figure 99 shows the group contribution for dim1 and dim3. On one hand, dim 3 is correlated mainly, as it does dim1, with `Popularity`, and `App Features`. The fact that `Popularity` and `App Features` are close to each other means that they have several dimensions in common. On the other hand, we have `Monetization`, which is highly correlated with dim1 but very poorly with dim3.
The second plot shows us the contribution of variables for dim1 and dim3. In this case, `Size`, `Installs` and `Rating.Count` have a considerable contribution for both dim1 and dim3. The difference remains in `AppNameLen`, wish has strong correlation with dim1 but no that much with dim3.

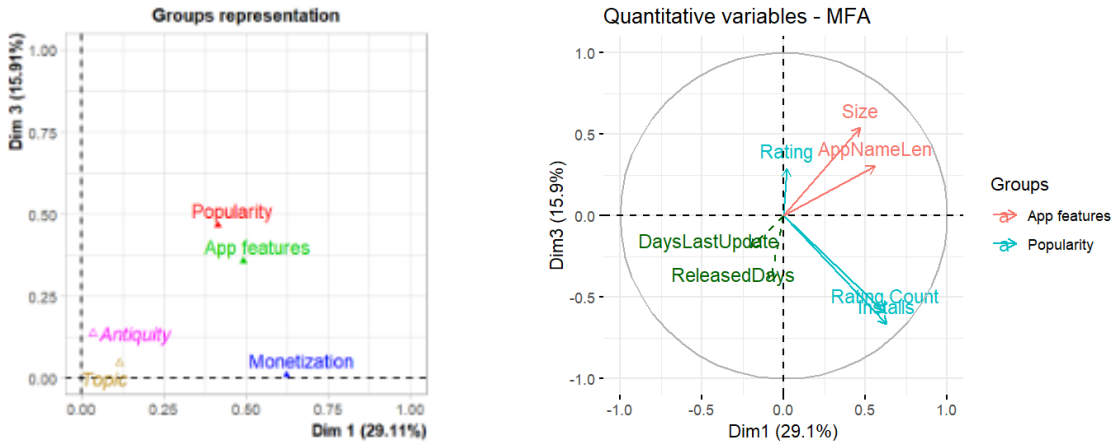


Figure 99: Groups and variables contribution to dim1 and dim3

In Figure 99 we plot the contribution of each variable with dim1 and dim3. This plot shows that dim3 is highly correlated mainly with the `Size`, `Installs` and `Rating.Counts` of the apps, these variables also have a considerable correlation with dim1 as we have seen before. From the plot, it seems that in this case `Size` and `AppNameLen` do have a positive correlation. So, there is a partial correlation between the variables of the `App Features` group.

The Figure 100 below plots the individuals, as the same case we saw before, there is not a clear group of clusters between the individuals, so we are going to summarize the individuals based on the axes. What we can see is that there is a difference in cloud points density between dim1 positive side and negative side. On the positive side of dim1, we have the popular apps (-dim3) and the apps with high size and name length (+dim3), while on the other side we have the unpopular apps (+dim3) and older apps (-dim3). This also indicates to us that the newer apps tend to have shorter names and lower sizes than the older ones.

In Figure 101 we plot the partial points on the individuals in dim1 and dim3. The first thing we can see is that `Monetization`, almost, has the same "view" to all the individuals. `App features` have higher "views" of individuals on axes (+dim1, +dim3) and (-dim1, -dim3).

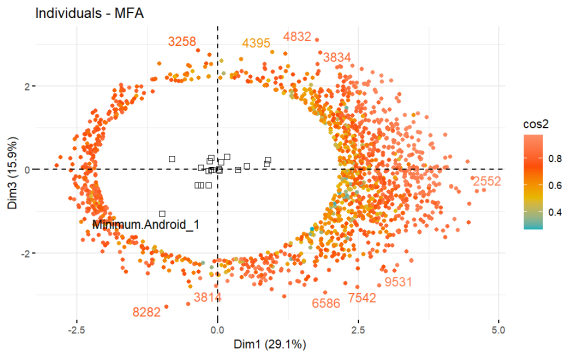


Figure 100: Individuals contribution to dim1&3

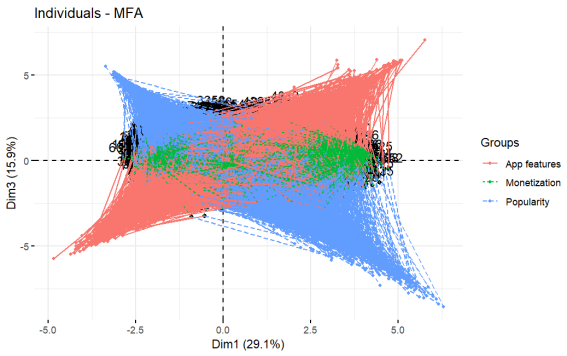


Figure 101: Partial points representation dim1&3

- **DIMENSION 2 & 3**

Once again, we plot the group and variable plots, but in this case we are comparing dim2 with dim3. The main difference between dim2 and dim3 is that while the main contributors of dim2, from highest to lowest, are `App Features`, `Monetization` and `Popularity`, while for dim3 is the order.

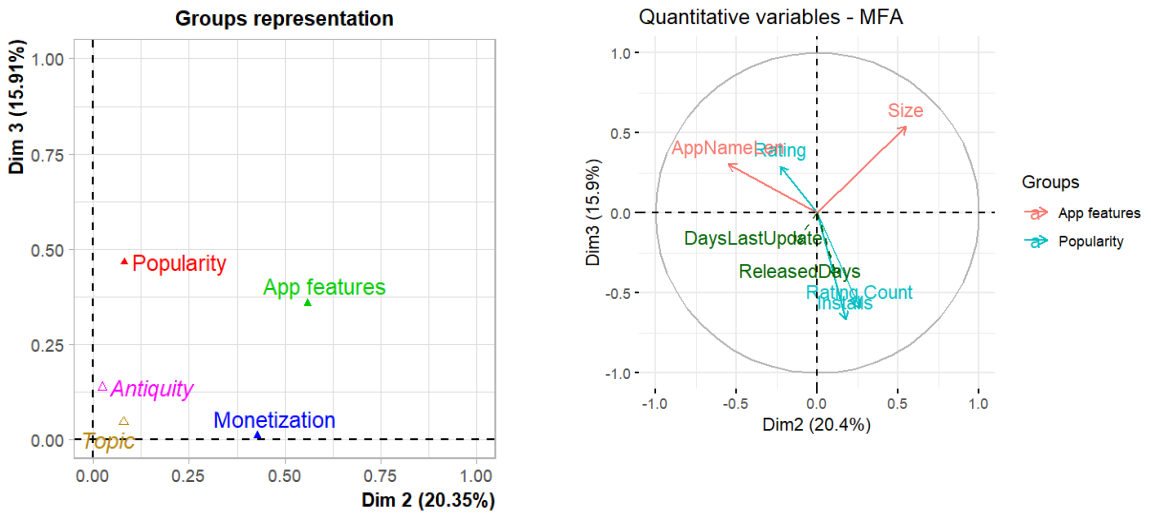


Figure 102: Groups and variables contribution to dim2 and dim3

On the other plot, we have a comparison of the variables of dim2 and dim3. In this case, we see the same scenario as we saw comparing dim1-dim2: the `AppNameLen` variable is independent of the `Size`. What is more is that in this case, `Rating` has an inverse correlation with `Rating.Count` and `Installs`.

Finally, plotting the individuals, we can see a similar behavior compared to what we have seen before. The individuals do not really have clusters groups, all the individuals are spread all over the four axes Figure 103. In the other plot Figure 104, we see how group `Population` and `App Feature` have “strong views” compared to `Monetization`.

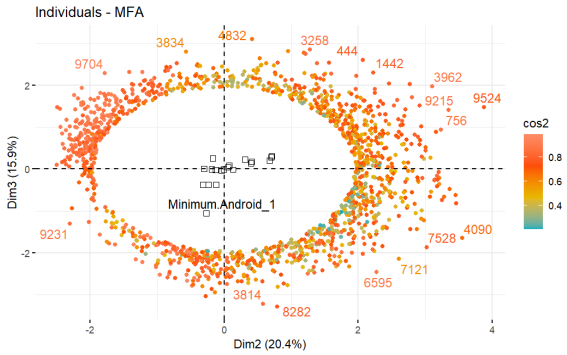


Figure 103: Individuals contribution to dim2&3

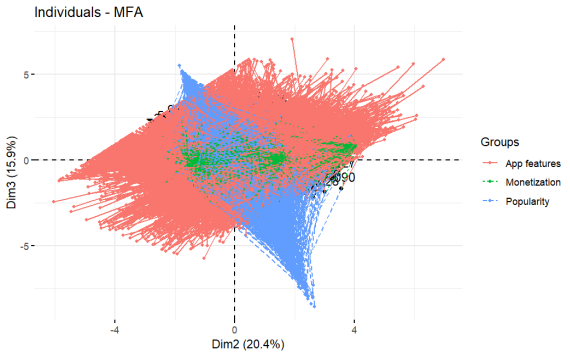


Figure 104: Partial points representation dim2&3

8.2 Conclusions

After the analysis conducted to our data with MFA, setting five groups: `Monetization`, `App Feature`, `Popularity`, `Topic` and `Antiquity`, we conclude the following:

- **About groups:**
  - The `App Feature` group contains more variety of apps compared to other groups, followed by `Monetization` and `Popularity`.
  - The groups are, mostly, not correlated between them.
  - `Monetization` is close to the mean configuration MFA, followed by `App Feature` and `Popularity`.
- **About variables:**
  - There is a relation between the number of installs and the number of ratings of an app, the number of ratings increases proportionally with the number of installs. So, the popularity of an app is defined by the number of installs and number of ratings.
  - In some cases, when the number of installs increases (and so the number of ratings), the `Rating` decreases, this is because there are unpopular apps that do have higher ratings than the popular ones. This means that rating should not be part of the group Popularity.
  - Usually, the size of an app is independent of the light length of an app name, but in some cases there is a positive correlation between them.
  - There is a light correlation between the days since the last release and days since the last update.
  - Usually, the older an app is, the more popular it is. We can also sometimes see that the newer apps tend to have less size and short names.
- **About individuals:**
  - There are no clear clusters of individuals in the data.
  - In general, not all individuals are seen the same by all the groups, there is a high difference, specially between `App Features` and `Popularity`.

9. Association rules mining analysis

Before we can start we did the same transformations for low frequency modalities as we had done in MCA. Moreover, in order to include numerical variables for our analysis we discretize the numerical values dividing the range of values approximately in 3 equally frequent ranges. The modalities names are the name of the feature plus low, mid or high.

9.1 Identification of the frequent itemsets and the extraction association

First, we identify the most frequent modalities that are listed in the table below.

Modality	**Absolute frequency**	**Relative frequency**
Content.Rating=Everyone	10080	0.852
In.App.Purchases=False	10001	0.846
Minimum.Android=4	8009	0.677
Ad.Supported=True	7459	0.631
RatingAPP=Rating High	4463	0.377

In order to give a comprehensive vision of the modalities we can take a look at Figure 105. There are a lot of modalities that have a frequency of 0.33, this is caused by the numerical features discretization.

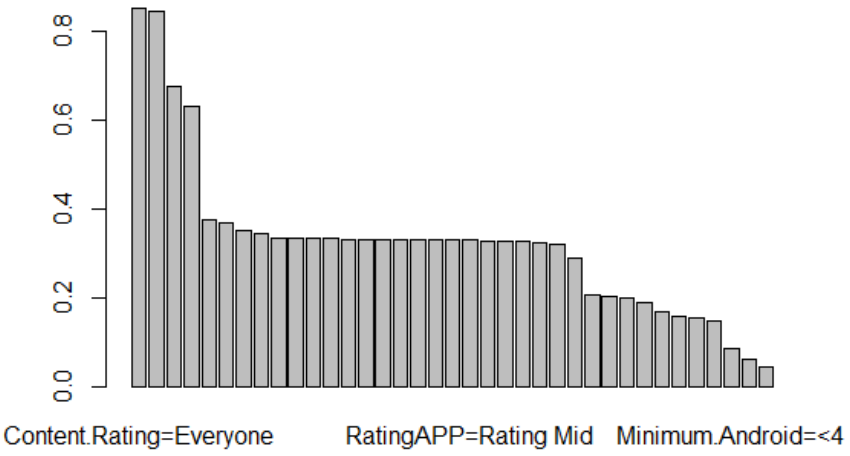


Figure 105: Barplot of modalities frequency

9.2 Rules from the dataset using Apriori

First we check the rules that have our response rule on the right side. As Figure 106 shows, there are few rules and they have poor confidence and low lift. So we can say that these rules are not reliable rules. Moreover all rules have Rating High on the right side and use only 3 modalities for the 4 rules.

```
    lhs                                rhs                        support confidence  coverage     lift count
[1] {Content.Rating=Everyone,
     In.App.Purchases=False,
     Installs=Installs Low}     => {RatingAPP=Rating High} 0.1489092  0.5535995 0.2689836 1.466921  1761
[2] {Content.Rating=Everyone,
     Installs=Installs Low}     => {RatingAPP=Rating High} 0.1608321  0.5478111 0.2935904 1.451583  1902
[3] {In.App.Purchases=False,
     Installs=Installs Low}     => {RatingAPP=Rating High} 0.1660748  0.5472276 0.3034838 1.450037  1964
[4] {Installs=Installs Low}     => {RatingAPP=Rating High} 0.1804499  0.5413496 0.3333333 1.434461  2134
```

Figure 106: Rules for `Rating`

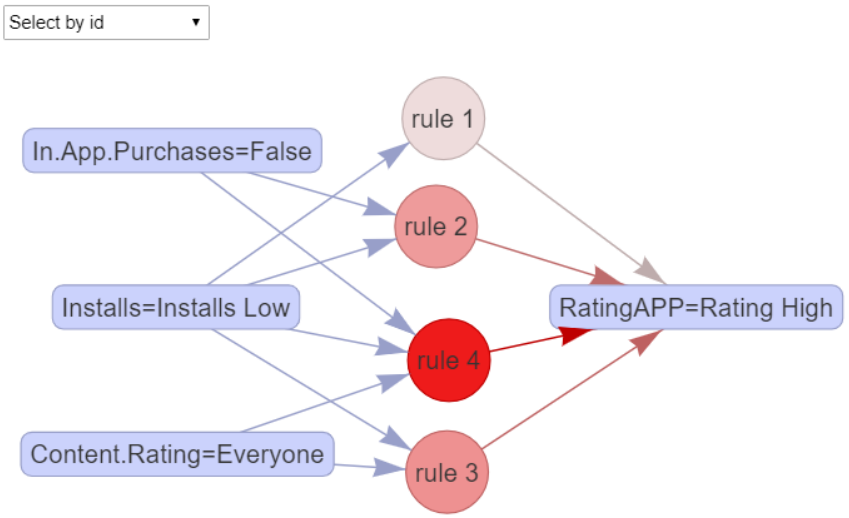


Figure 107: Graph of rules for `Rating`

The 4 rules have `Installs` low and `Rating` high. We have seen this behavior when we analyze the correlation between `Installs` and `Rating`.

An interesting variable for our analysis is `Installs`, due to that is like `Rating`, we want to extract knowledge for maximes these variables. So we checked the rules for this variable. As is shown in Figure 108 there are more rules and with higher confidence and lift. However, it is important to point out that on the left side of these rules there are always `Rating.Count`. As we have seen in previous sections there is a huge correlation between these two variables. Moreover if in future studies we want to predict the rating or installs before the app is released we do not have access to `Rating.Count`. For this reason we try to take more rules for `Installs`, but without using `Rating.Counts`.

```
     lhs                                    rhs                          support confidence  coverage     lift count
[1]  {Rating.Count=Rating.Count High,
      ReleasedDays=ReleasedDays High}  => {Installs=Installs High} 0.1313208  0.8724719 0.1505158 2.617416  1553
[2]  {Rating.Count=Rating.Count High,
      Minimum.Android=4,
      Ad.Supported=True}               => {Installs=Installs High} 0.1456959  0.8606394 0.1692880 2.581918  1723
[3]  {Rating.Count=Rating.Count High,
      Content.Rating=Everyone,
      Ad.Supported=True}               => {Installs=Installs High} 0.1592254  0.8598174 0.1851852 2.579452  1883
[4]  {Rating.Count=Rating.Count High,
      Ad.Supported=True}               => {Installs=Installs High} 0.1998140  0.8580247 0.2328767 2.574074  2363
[5]  {Rating.Count=Rating.Count High,
      Minimum.Android=4}               => {Installs=Installs High} 0.1849315  0.8315589 0.2223913 2.494677  2187
[6]  {Rating.Count=Rating.Count High}  => {Installs=Installs High} 0.2753256  0.8249303 0.3337561 2.474791  3256
[7]  {Rating.Count=Rating.Count Low,
      Content.Rating=Everyone}         => {Installs=Installs Low}  0.2124979  0.7363024 0.2886014 2.208907  2513
[8]  {Rating.Count=Rating.Count Low,
      In.App.Purchases=False}          => {Installs=Installs Low}  0.2200237  0.7282396 0.3021309 2.184719  2602
[9]  {Rating.Count=Rating.Count Low}   => {Installs=Installs Low}  0.2400643  0.7279487 0.3297818 2.183846  2839
```

Figure 108: Rules for `Installs`

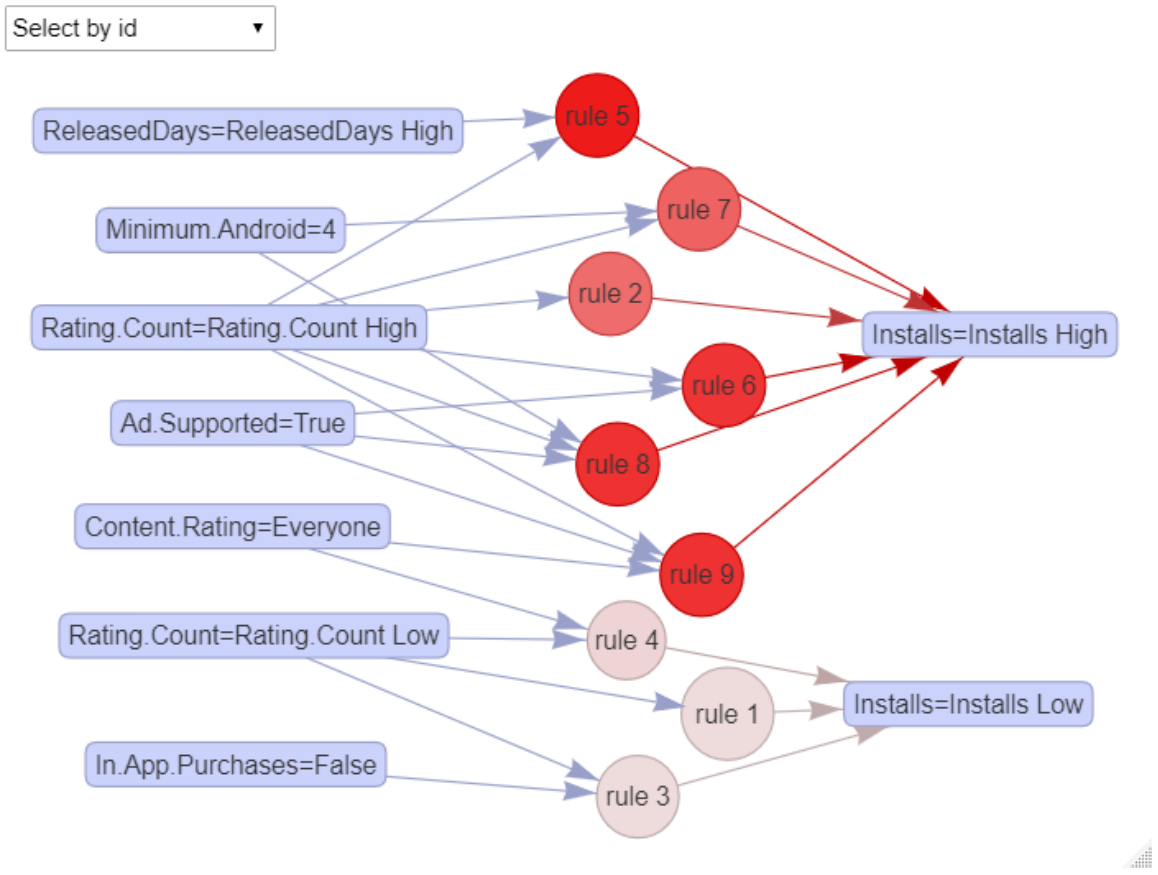


Figure 109: Graph of rules for `Installs`

Now that we have filtered the rules of Installs without using `Rating.Count` (Figure 110), there are fewer rules and with less accuracy and lift. The two rules have `In.App.Purchase` false and give low `Installs`.

```
    lhs                          rhs                       support confidence  coverage     lift count
[1] {Content.Rating=Everyone,
     In.App.Purchases=False,
     RatingAPP=Rating High}   => {Installs=Installs Low} 0.1489092  0.5222420 0.2851344 1.566726  1761
[2] {In.App.Purchases=False,
     RatingAPP=Rating High}   => {Installs=Installs Low} 0.1660748  0.5115916 0.3246237 1.534775  1964
```

Figure 110: Rules for `Installs` without `Rating.Count`

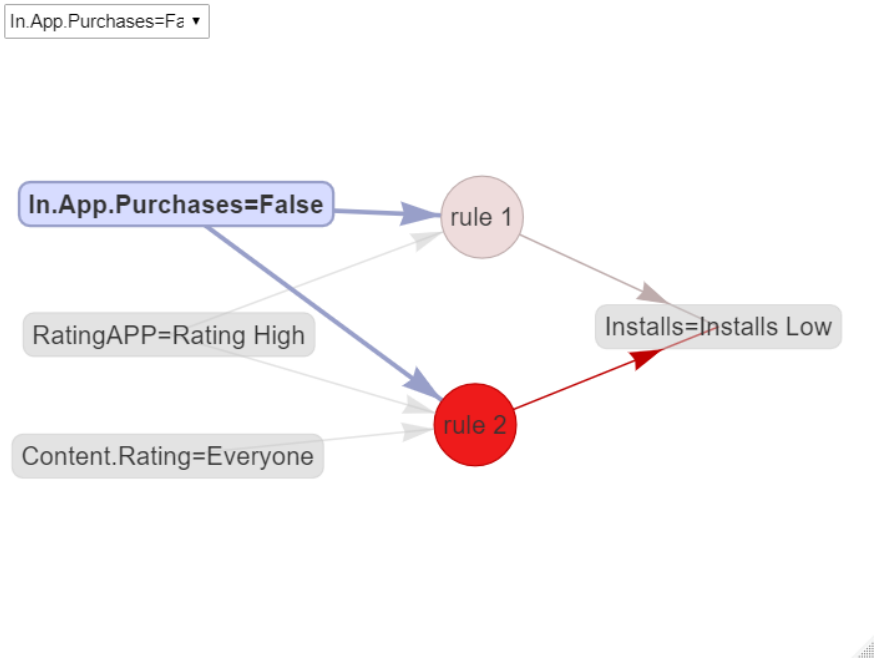


Figure 111: Graph of rules for `Installs` without `Rating.Count`

9.3 Top 20 rules explanation (sorted by decreasing confidence)

```
     lhs                                                rhs                                  support confidence  coverage     lift count
[1]  {ReleasedDays=ReleasedDays High,
      Installs=Installs High}                        => {Rating.Count=Rating.Count High} 0.1313208  0.8849003 0.1484018 2.651338  1553
[2]  {Rating.Count=Rating.Count High,
      ReleasedDays=ReleasedDays High}                => {Installs=Installs High}         0.1313208  0.8724719 0.1505158 2.617416  1553
[3]  {Category=Game}                                => {Ad.Supported=True}              0.1367326  0.8679549 0.1575342 1.376114  1617
[4]  {Rating.Count=Rating.Count High,
      Minimum.Android=4,
      Ad.Supported=True}                            => {Installs=Installs High}         0.1456959  0.8606394 0.1692880 2.581918  1723
[5]  {Rating.Count=Rating.Count High,
      Content.Rating=Everyone,
      Ad.Supported=True}                            => {Installs=Installs High}         0.1592254  0.8598174 0.1851852 2.579452  1883
[6]  {Rating.Count=Rating.Count High,
      Ad.Supported=True}                            => {Installs=Installs High}         0.1998140  0.8580247 0.2328767 2.574074  2363
[7]  {Ad.Supported=True,
      In.App.Purchases=False,
      DaysLastUpdate=DaysLastUpdate Mid}            => {Minimum.Android=4}              0.1574497  0.8376068 0.1879756 1.236801  1862
[8]  {Minimum.Android=4,
      AppNameLen=AppNameLen High}                   => {Ad.Supported=True}              0.2120751  0.8373957 0.2532555 1.327663  2508
[9]  {Rating.Count=Rating.Count High,
      Minimum.Android=4}                            => {Installs=Installs High}         0.1849315  0.8315589 0.2223913 2.494677  2187
[10] {ReleasedDays=ReleasedDays Low,
      AppNameLen=AppNameLen High}                   => {Ad.Supported=True}              0.1302215  0.8315335 0.1566041 1.318369  1540
[11] {Ad.Supported=True,
      In.App.Purchases=False,
      ReleasedDays=ReleasedDays Mid}                => {Minimum.Android=4}              0.1437511  0.8308895 0.1730086 1.226882  1700
[12] {Content.Rating=Everyone,
      Ad.Supported=True,
      ReleasedDays=ReleasedDays Mid}                => {Minimum.Android=4}              0.1444275  0.8263183 0.1747844 1.220132  1708
[13] {Installs=Installs High}                       => {Rating.Count=Rating.Count High} 0.2753256  0.8259767 0.3333333 2.474791  3256
[14] {Ad.Supported=True,
      ReleasedDays=ReleasedDays Mid}                => {Minimum.Android=4}              0.1736005  0.8258246 0.2102148 1.219403  2053
[15] {Rating.Count=Rating.Count High}               => {Installs=Installs High}         0.2753256  0.8249303 0.3337561 2.474791  3256
[16] {Ad.Supported=True,
      DaysLastUpdate=DaysLastUpdate Mid}            => {Minimum.Android=4}              0.1840014  0.8220627 0.2238289 1.213849  2176
[17] {Category=Entertainment,
      Minimum.Android=4}                            => {Ad.Supported=True}              0.1258245  0.8144499 0.1544901 1.291284  1488
[18] {Minimum.Android=4,
      Installs=Installs High}                       => {Ad.Supported=True}              0.1815491  0.7987351 0.2272958 1.266368  2147
[19] {AppNameLen=AppNameLen High}                   => {Ad.Supported=True}              0.2766785  0.7882438 0.3510063 1.249735  3272
[20] {Category=Entertainment,
      In.App.Purchases=False}                       => {Ad.Supported=True}              0.1426518  0.7738532 0.1843396 1.226919  1687
```

Figure 112: Top 20 rules

The rules 1, 2, 4, 5, 9, 13 and 15 all have the correlation between `Installs` and `Rating.Count` that we have seen. The rules 3, 8, 10, 17, 18, 19 and 20 all have the `Ad.Supported` as TRUE, but all of them have a low lift. A quick summary of the more important rules are that the apps are games, have a long name or have a minimum android 4 have ad supported. The rules 7, 11, 12, 14 and 16 all have the `Minimum.Android` 4, but all of them have a low lift. A quick summary of the more important rules are that the apps that don't have in app purchases, have ads or his release date is mid have minimum android 4.

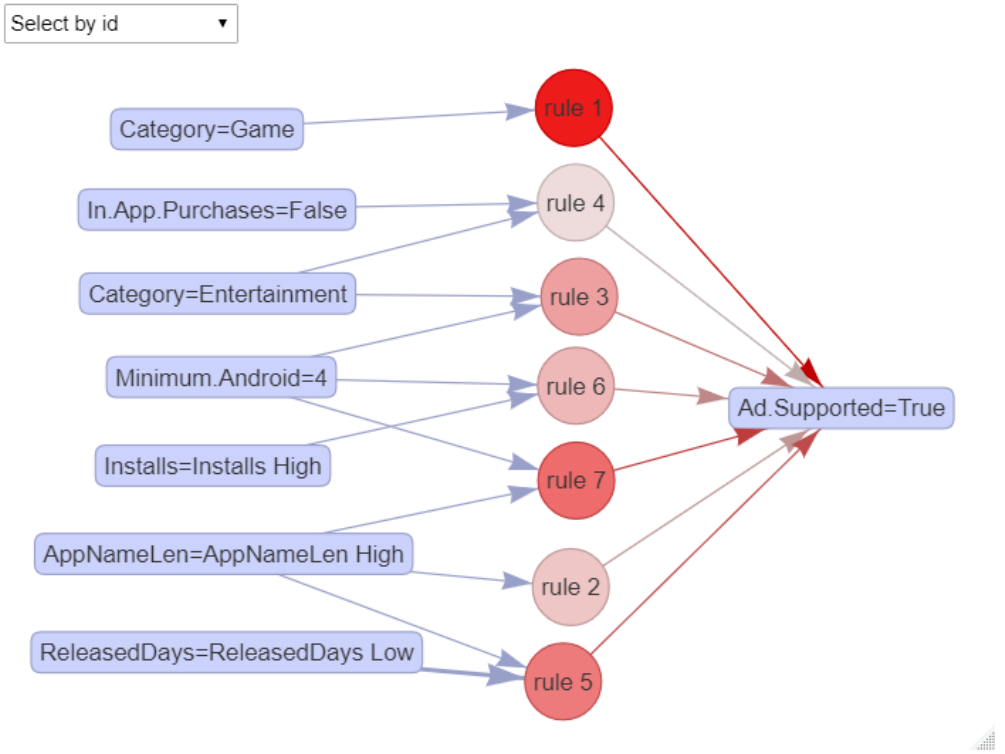


Figure 113: Graph of rules for `Ad.Supported`

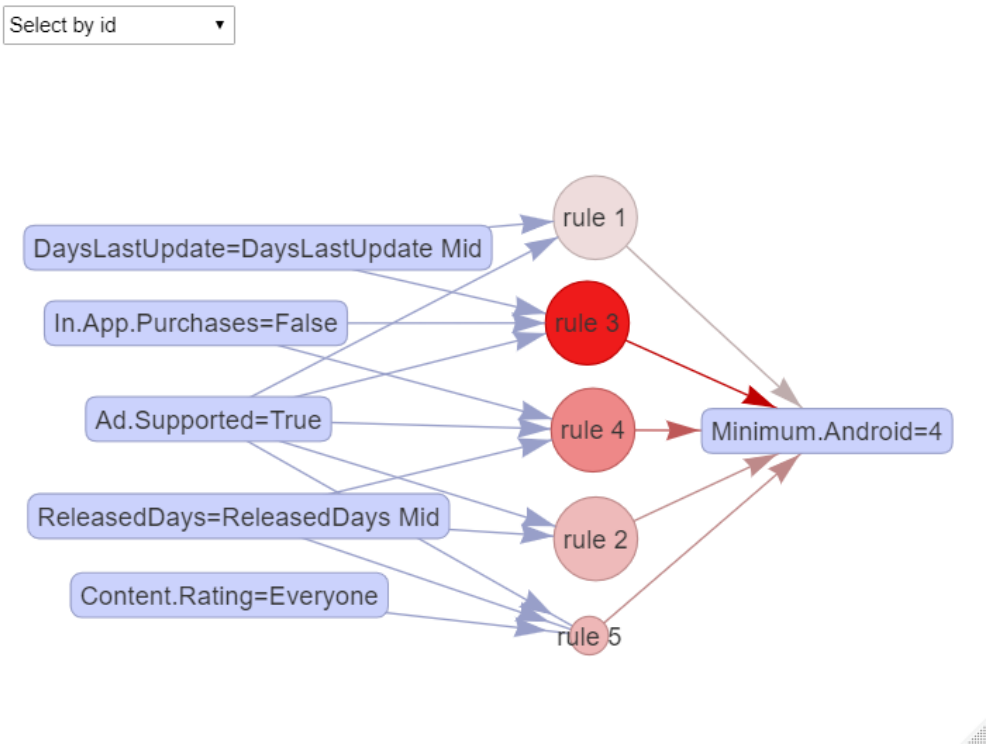


Figure 114: Graph of rules for `Minimum.Android`

10. Partitional clustering

10.1 Data preparation

In this section, we only used the numerical variables from our data as k-means uses only numerical features.

We also applied logarithm to the features `Rating.Count`, `Size`, `Installs`, `DaysLastUpdate` and `ReleasedDays` because their histograms followed a negative exponential distribution.

We scaled our data except for `Rating`, which is our response variable and thus we do not use it for clustering, to get rid of the fact that some features might be more important than others in terms of variance.

10.2 Selecting the optimal number of clusters

Due to the fact that we are using k-means, the number of clusters (k) is an input we have to give to the algorithm.

To determine k we used the Silhouette method and Elbow method.
Looking at Figure 115, the Silhouette method gave us k=2.

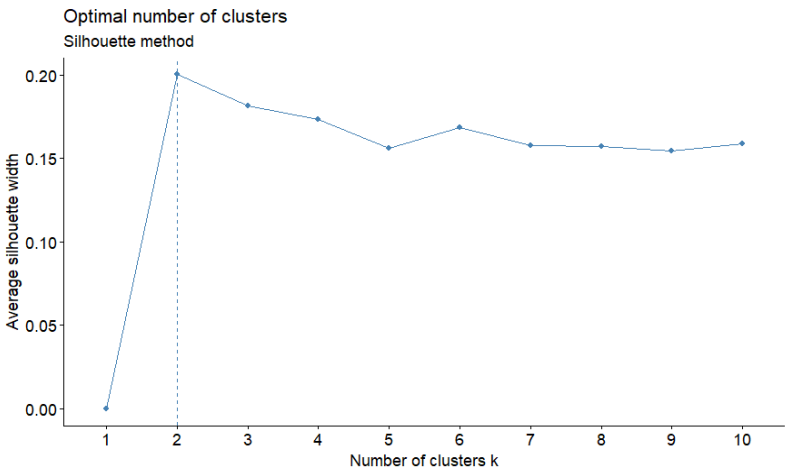


Figure 115: Optimal number of clusters using the Silhouette method

Looking at Figure 116, the Elbow method gave us k=4.

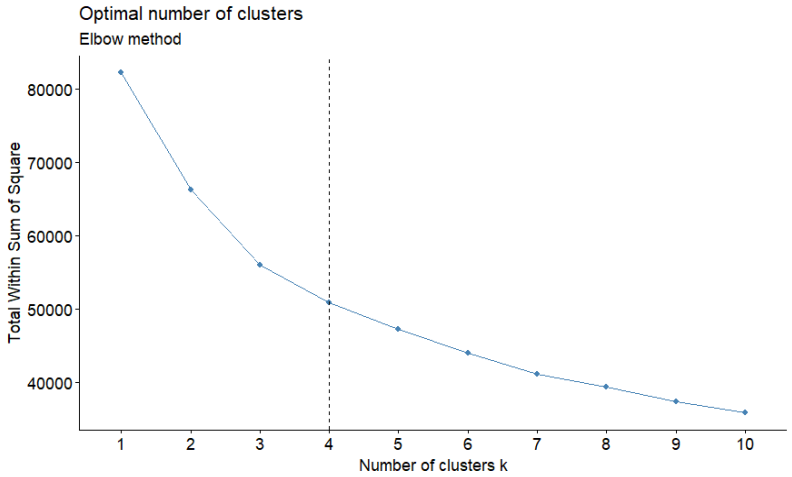


Figure 116: Optimal number of clusters using the Elbow method

10.3 K-means

Using k=2 for k-means we obtained the clusters in Figure 117. The clusters are not overlapping, so we can say that the results are good; but maybe there are clusters that we are ignoring, so we tried using k=4.

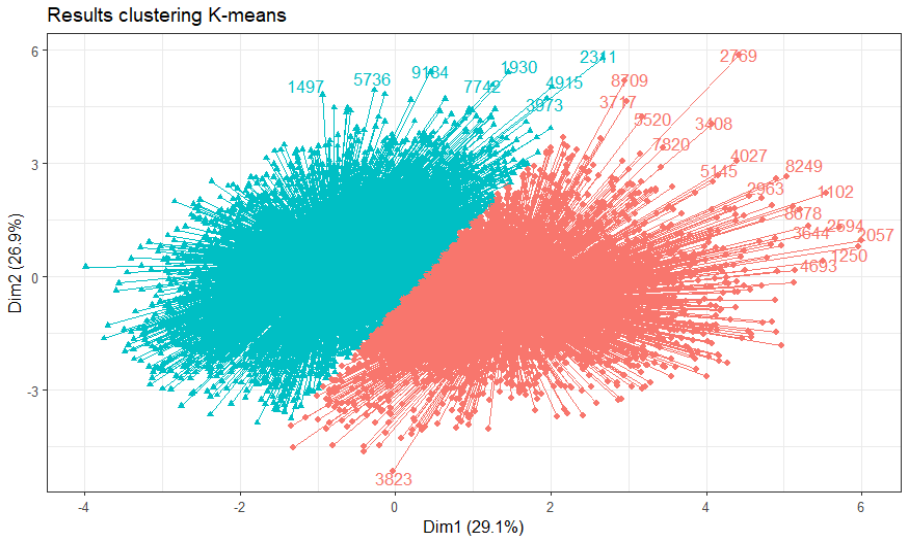


Figure 117: Results of K-means with k=2

Looking at Figure 118, we can see the green and red clusters overlap in the first 2 dimensions, but the others are fine.

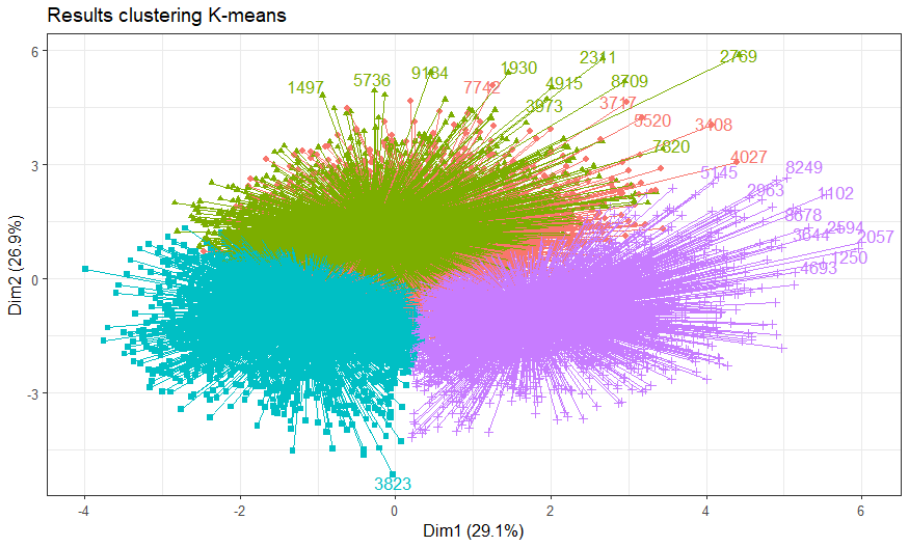


Figure 118: Results of K-means with k=4

Using k=3 we obtain the best result, where the 3 clusters do not overlap with each other (Figure 119).

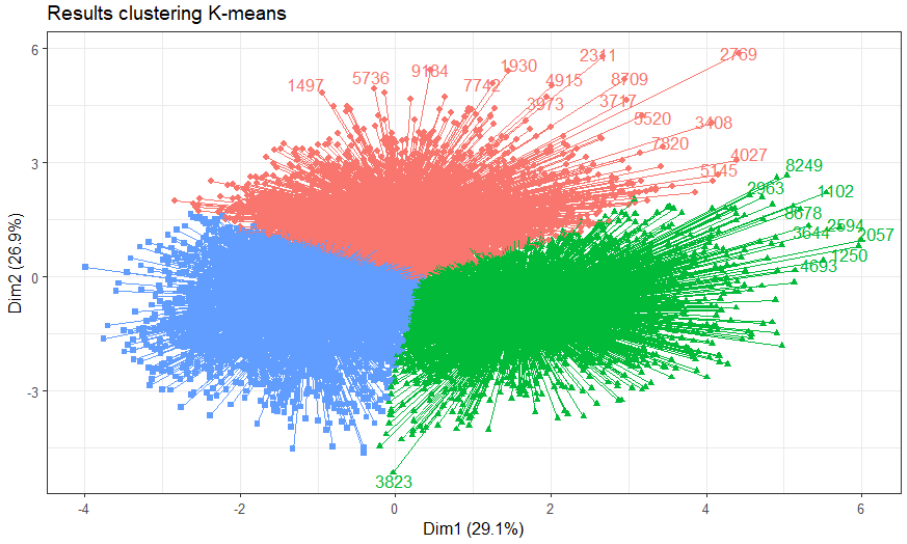


Figure 119: Results of K-means with k=3

10.4 CLARA

We tried using PAM for clustering but it took a long time, so we used CLARA. Instead of finding medoids for the entire data set, CLARA considers a small sample of the data with fixed size (sampsize) and applies the PAM algorithm to generate an optimal set of medoids for the sample.

For k=2 we obtain the clusters in Figure 120.

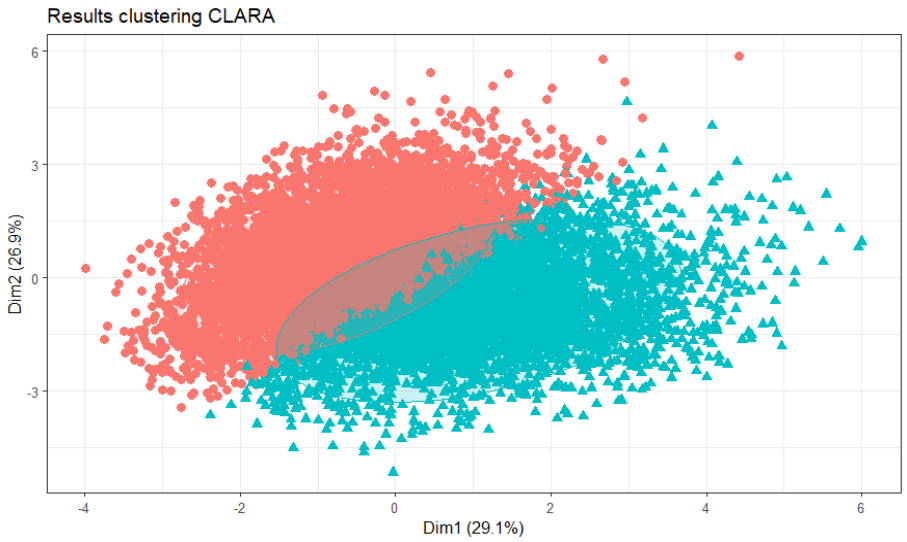


Figure 120: Results of CLARA with k=2

For k=4 we obtain the clusters in Figure 121.

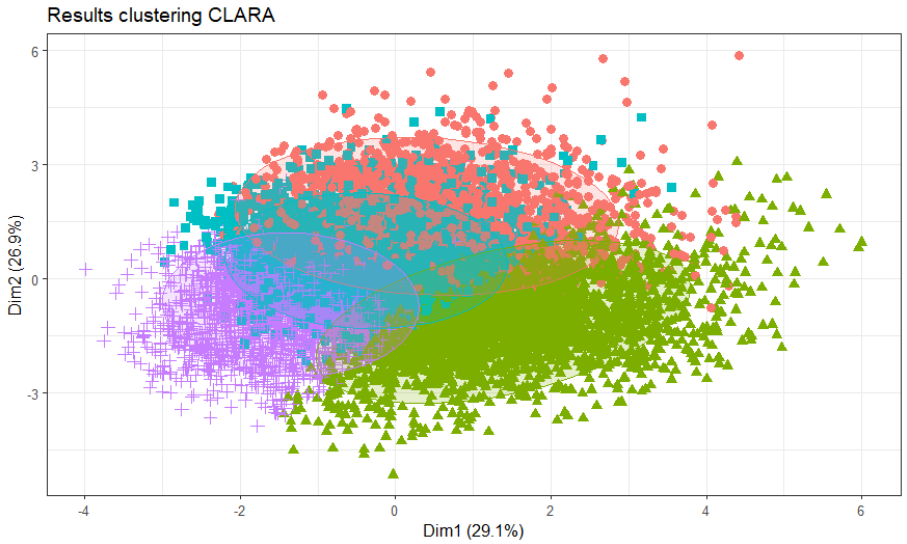


Figure 121: Results of CLARA with k=4

For k=3 we obtain the clusters in Figure 122.

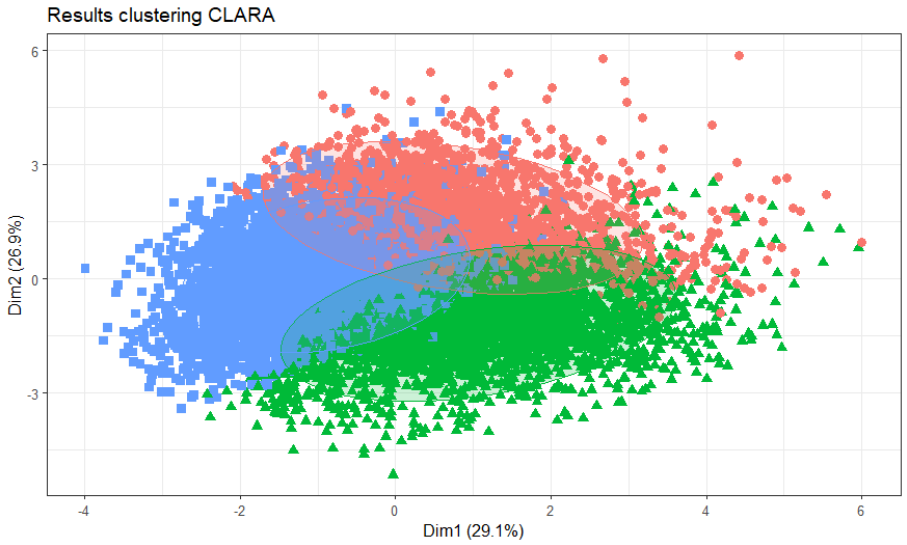


Figure 122: Results of CLARA with k=3

Considering the different results, we can say that the best k for CLARA is also 3.

11. Hierarchical clustering

11.1 Description of data used

For this part of the project we decided to use all variables, except the target `Rating`. As with partitional clustering, we also applied a logarithmic transformation to the features `Rating.Count`, `Size`, `Installs`, `DaysLastUpdate` and `ReleasedDays` and we also scaled the numerical variables.

11.2 Clustering method used

When working with Hierarchical clustering we have at our hand different agglomerative linkage algorithms. In this case, we make use of Ward's minimum variance method, which minimizes the total within cluster variance. At each step, the pair of clusters with minimum between-cluster distance are merged.

As our dataset is composed of both numerical and categorical data, we used the metric Gower square distance to measure the dissimilarity between two individuals.

11.3 Resulting dendrogram

Once we calculated the dissimilarity matrix we used the `hclust` function and we obtained the dendrogram depicted in Figure 125. Looking at it, we can see that the best option is to cut the dendrogram at a height of approximately 145 and split it into 3 clusters.

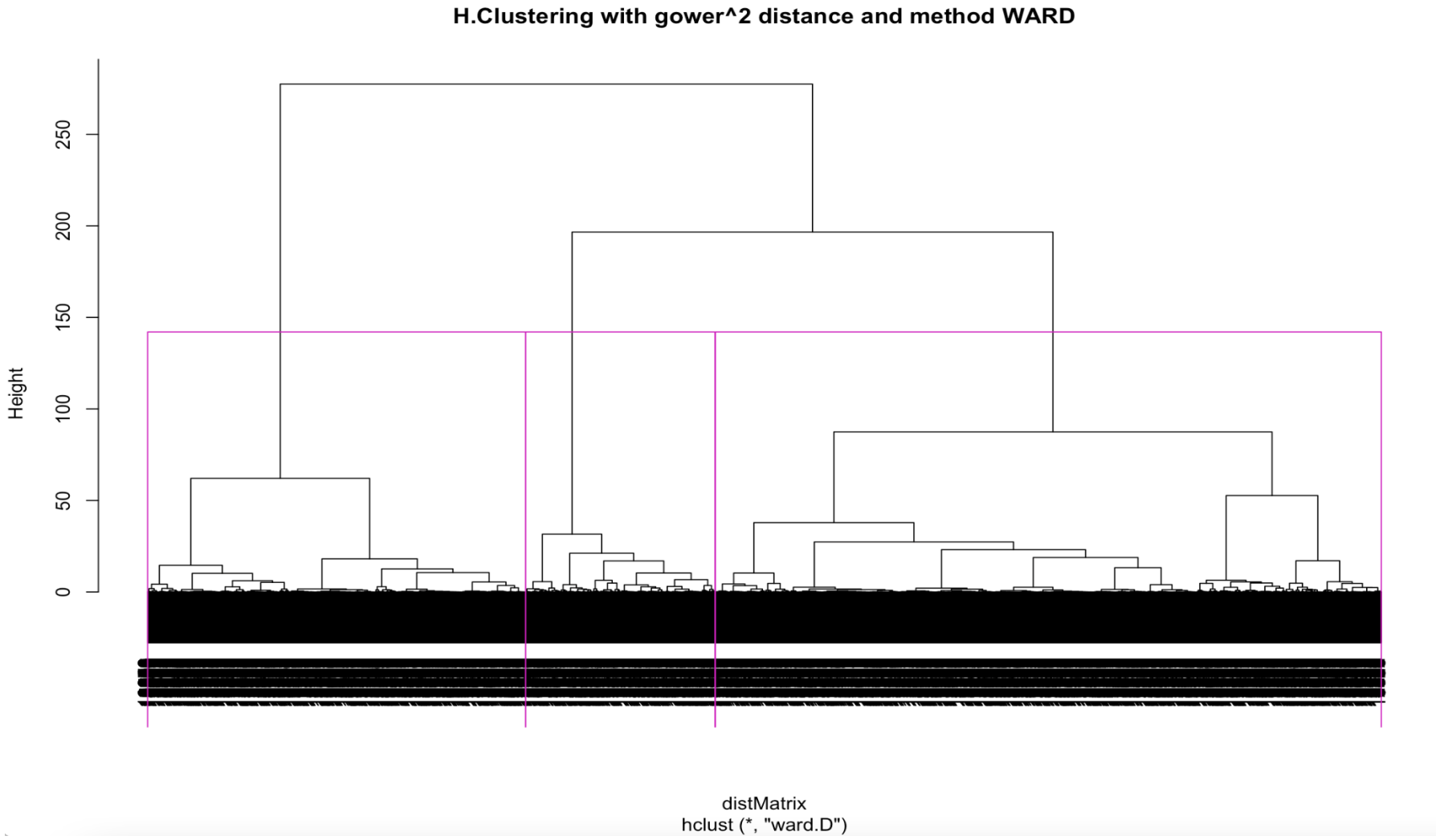


Figure 125: Resulting dendrogram

11.4 Final number of clusters

We decided to use the package `NbClust` for determining the best number of clusters. To do so, we use the following indexes: `kl, ch, hartigan, cindex, db, silhouette, ball, dunn` and `sindex`.

After using this, we obtained the following results:

Index	#Clusters
kl	6
ch	2
hartigan	3
cindex	6
db	3
silhouette	3
ball	3
dunn	2
sindex	2

We get k=2 for 3/9, k=3 for 4/9 and k=6 for 2/9. Consequently, we decided to use k=3.

After this, we used the `Silhouette` analysis to study the separation distance between the resulting clusters, see Figure 10. The `silhouette` plot displays a measure of how close each point in one cluster is to points in the neighboring clusters. This measure has a range of [-1, 1].

Observations with a large `silhouette`, almost 1, are very well clustered. A small silhouette value, close to 0, means that the observation lies between two clusters and observations with a negative silhouette are probably placed in the wrong cluster.

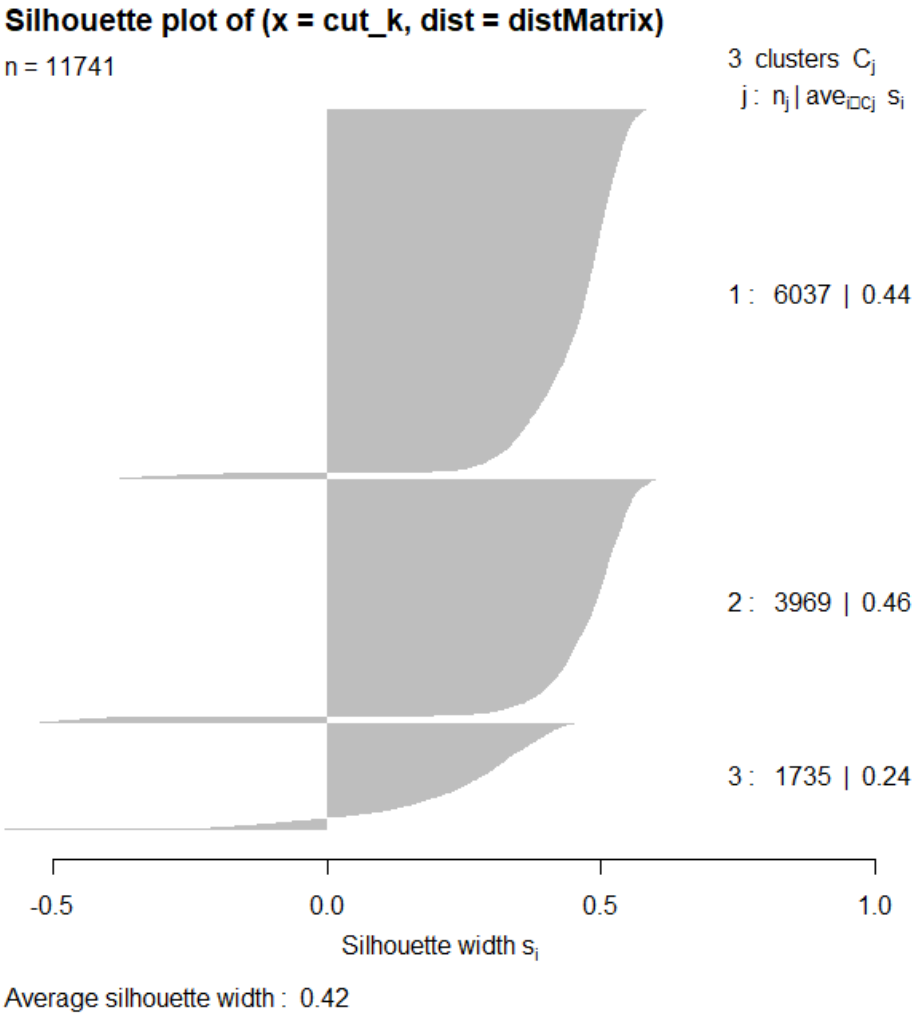


Figure 126: Clusters silhouette plot

From Figure 126 we can see that there are some samples which have negative silhouette values but they are not significant. The average silhouette width is acceptable.

In the following table it can be seen each cluster and its size.

Cluster	Size	Avg. sil width
1	6037	0.44
2	3969	0.46
3	1735	0.24

11.5 Using weights

We tried using weights to recalibrate the dataset. Taking into account all the previous analysis made during the project we decided to re-weight the data as follows:

- `Installs` weight is 40%
- `DaysLastUpdate` weight is 20%
- `ReleasedDays` weight is 20%
- `AppNameLen` weight is 10%
- `Rating.Count` weight is 5%
- `Size` weight is 5%

After applying the weights, we obtained the dendrogram depicted in figure 127. From it, we can see that the best option is to use k=3.

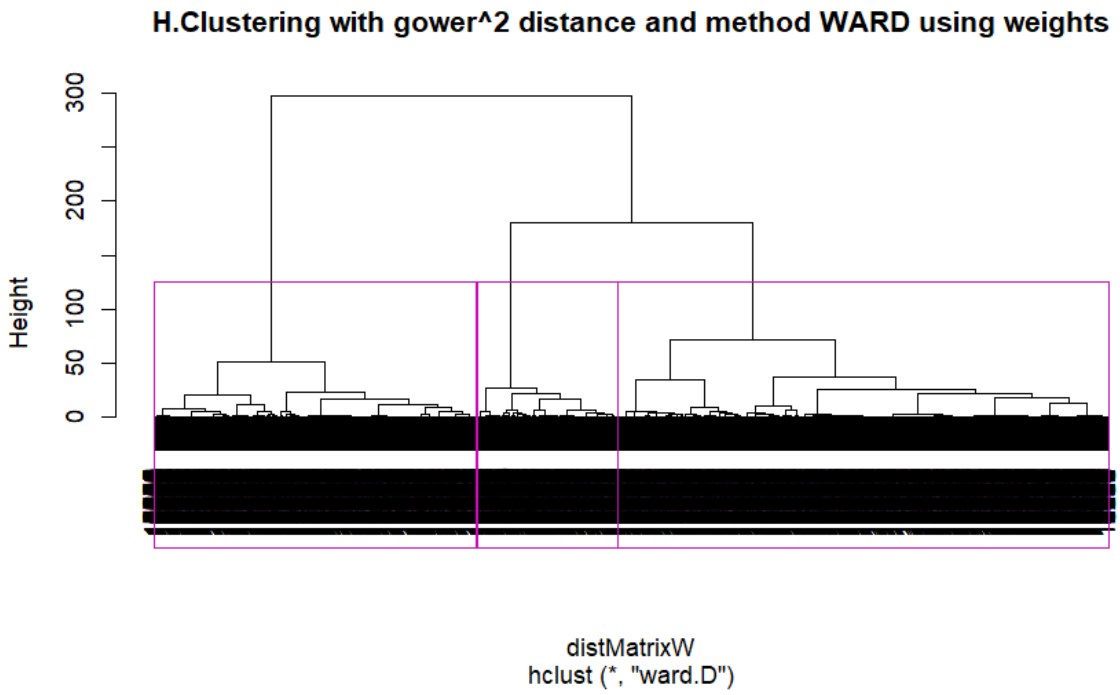


Figure 127: Dendrogram using weights

We then followed the same steps described before for hierarchical clustering and did not get much different results.

11.6 HCPC

In this section we try to apply HCPC (Hierarchical Clustering on Principal Components), a strategy that combine the three methods we have seen until now:

- Principal component methods (PCA, MCA, MFA)
- Hierarchical clustering
- Partitioning clustering (k-means)

The steps we followed are:

1. Perform PCA, here we chose only 4 dimensions to be retained in the output because with 4 dimensions the total amount of the variance explained is almost 90%.
2. Apply Hierarchical Clustering on the PCA outputs (coordinates of the individuals on the principal components).

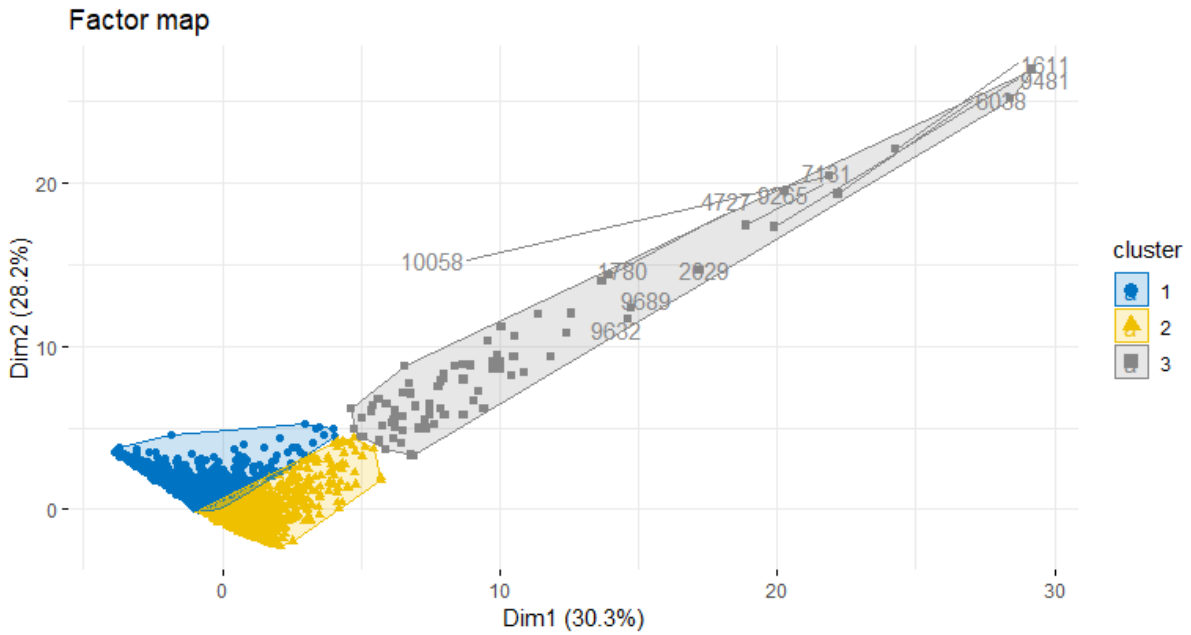


Figure 128: Factor map in Dim1-2

The above plot shows how the individuals are distributed on the factorial map. We can see that there is some overlap between the boundaries of clusters 1 and 2, which indicates a low intercluster distance and a high degree of similarity between them.

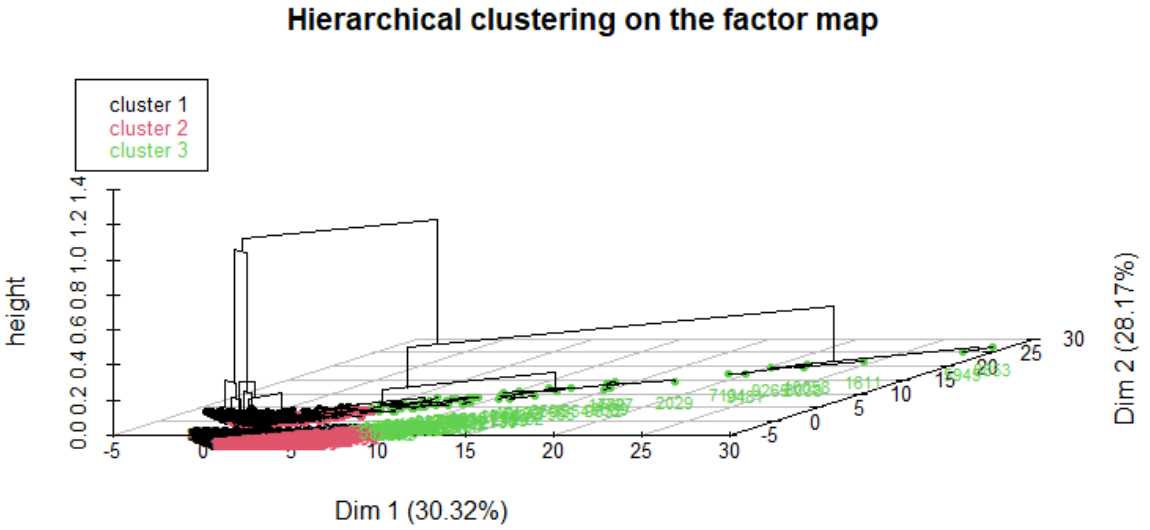


Figure 129: Dendrogram on the factor map

The above figure is a combination of the hierarchical clustering and the principal components, it suggests a solution of 3 clusters.

12. Profiling

12.1 Graphs

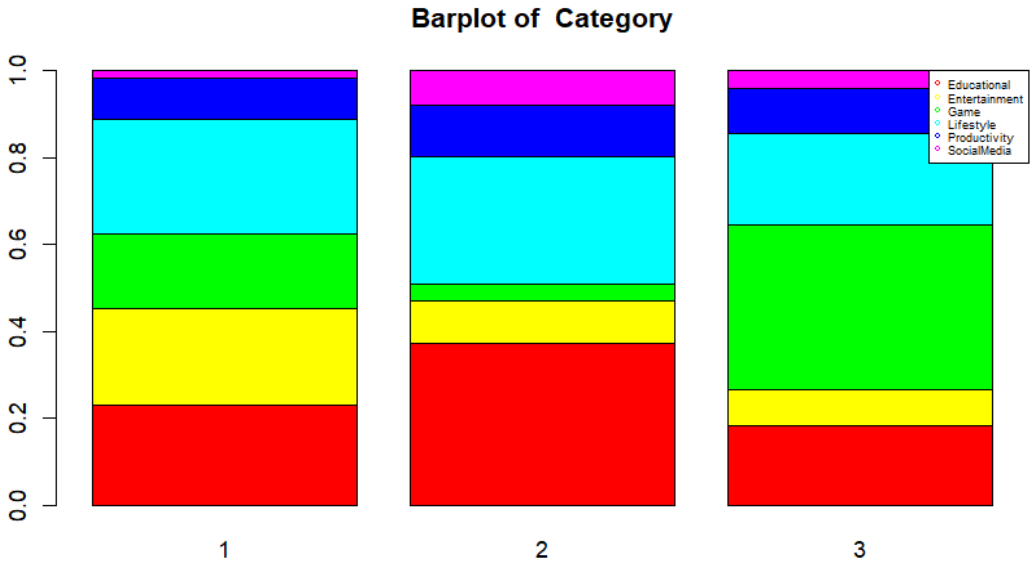


Figure 130: Barplots of Category for each cluster

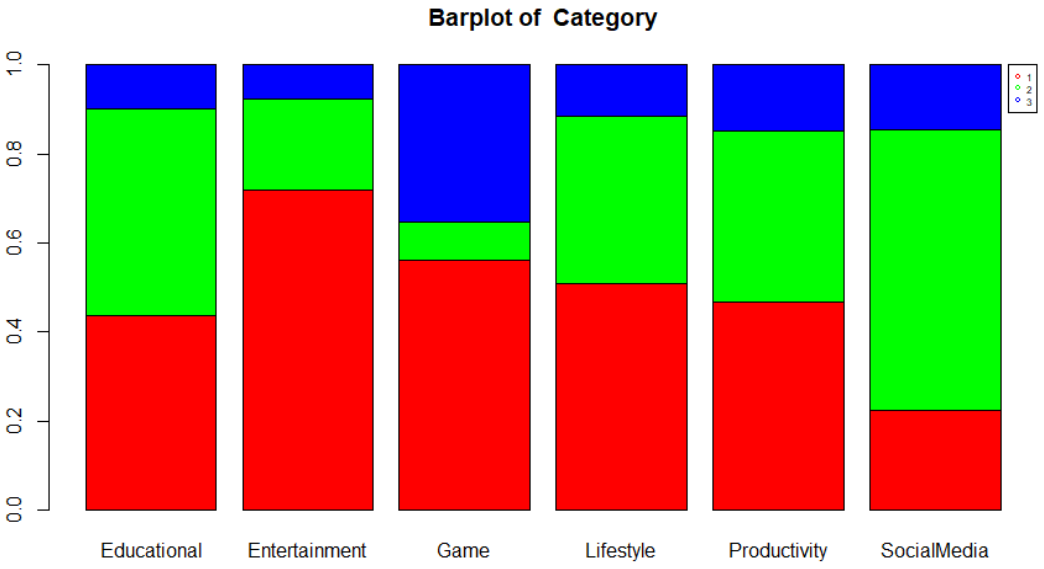


Figure 131: Barplots of Category for each category

As Figure 130 shows, the distribution of categories among the three clusters is different. The most relevant differences are that Games proportion is bigger in the 3rd cluster and very small in the second. Entertainment has similar results, a huge proportion on the 1st cluster and almost no social media in this cluster. Finally, the education proportion is bigger in the 2nd cluster.

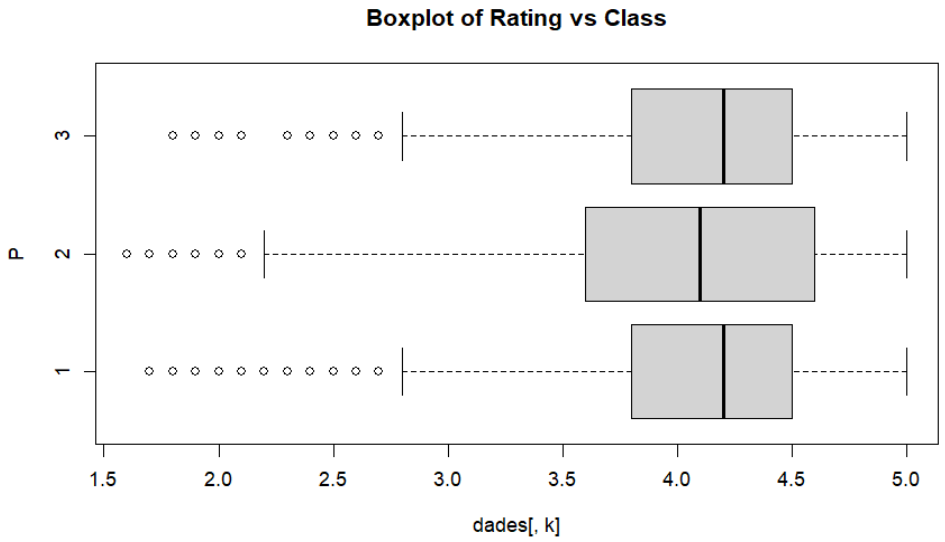


Figure 132: Boxplots of Rating for each cluster

Figure 133: Means of Rating for each cluster

The Figures 132 and 133 show that there is not a big difference in the `Rating` between clusters, especially between 1st and 3rd clusters.

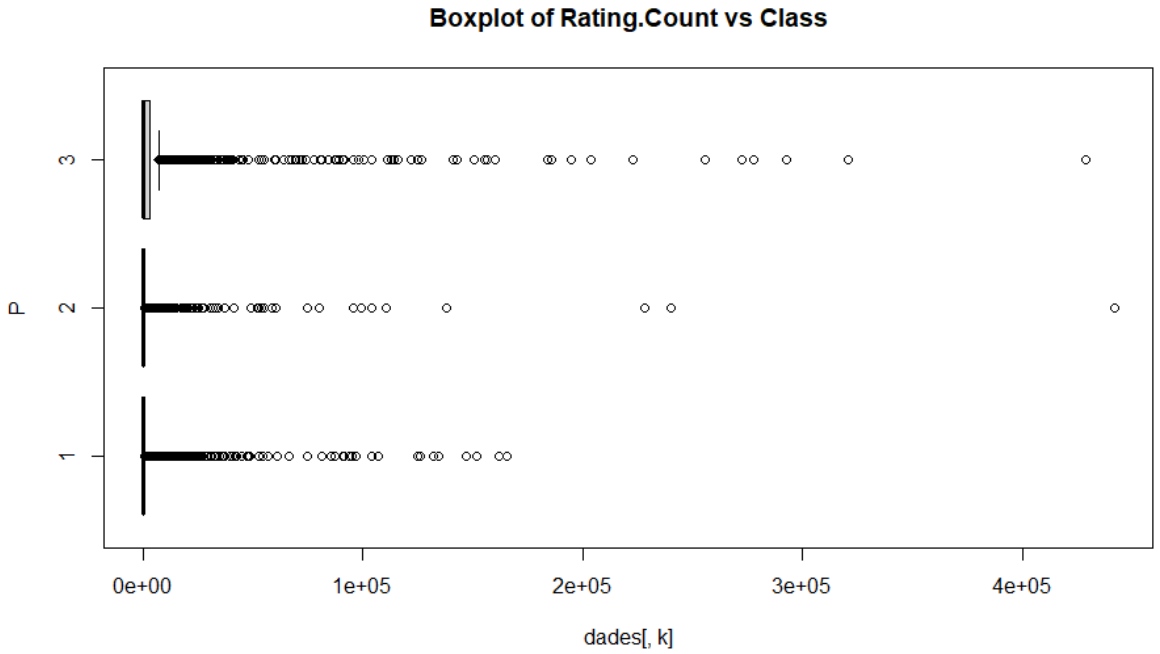


Figure 134: Boxplots of Rating.Count for each cluster

Figure 135: Means of Rating.Count for each cluster

Looking at Figures 134 and 135, we can see that there is a difference between the `Rating.Count` of the different clusters, the third cluster has a huge average compared with the other clusters.

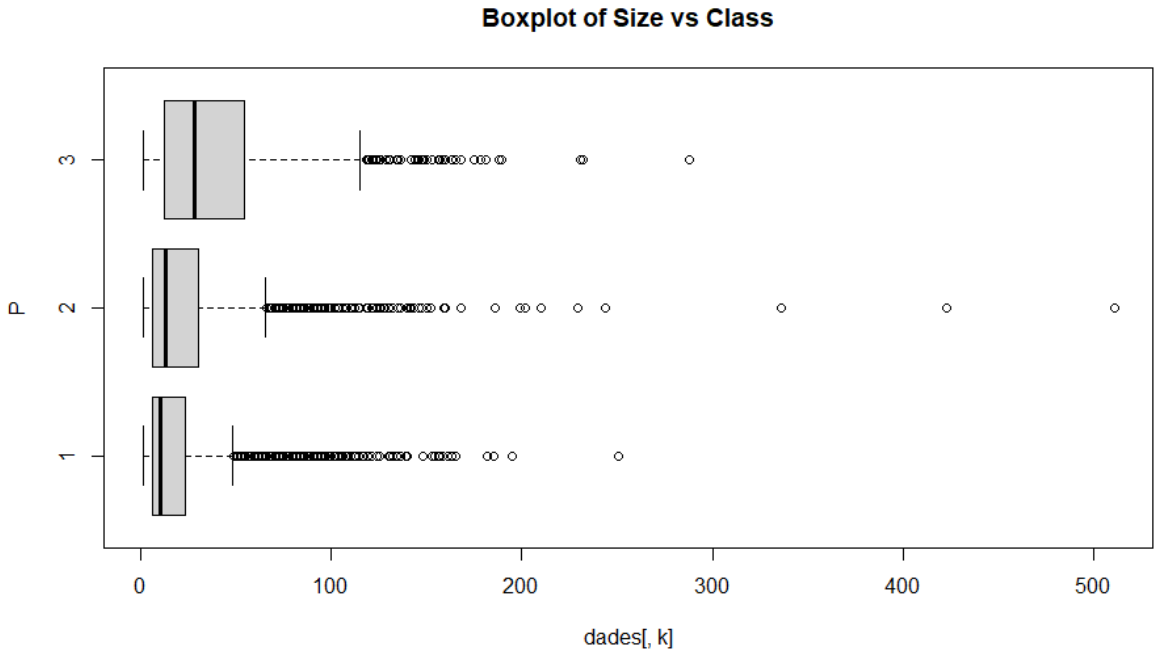


Figure 136: Boxplots of Size for each cluster

Figure 137: Means of Size for each cluster

Figures 136 and 137 show some differences between the clusters with the feature `Size`. The first cluster is below the global mean, the second cluster is in the global mean and the third cluster is quite above it.

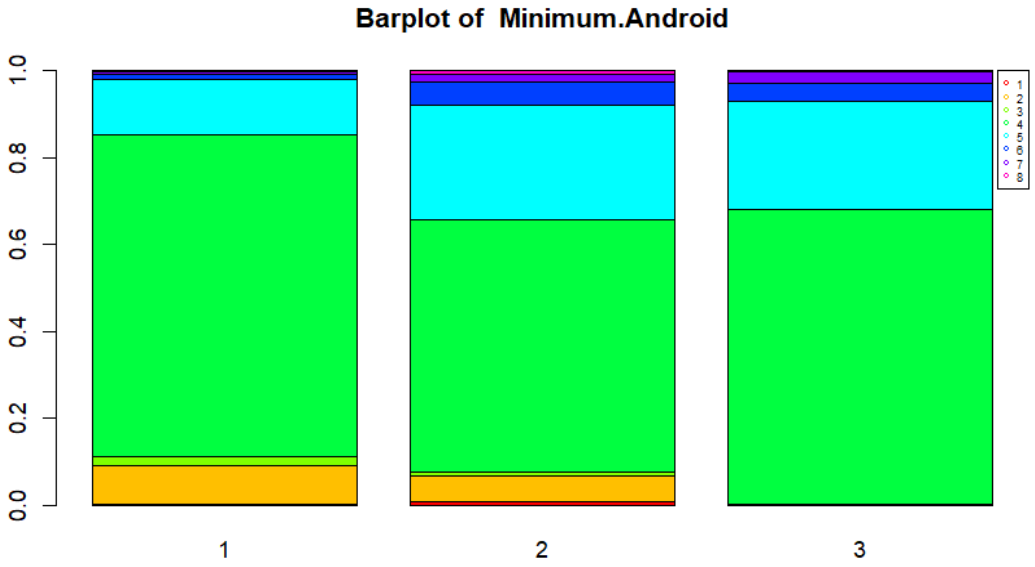


Figure 138: Barplots of Minimum.Android for each cluster

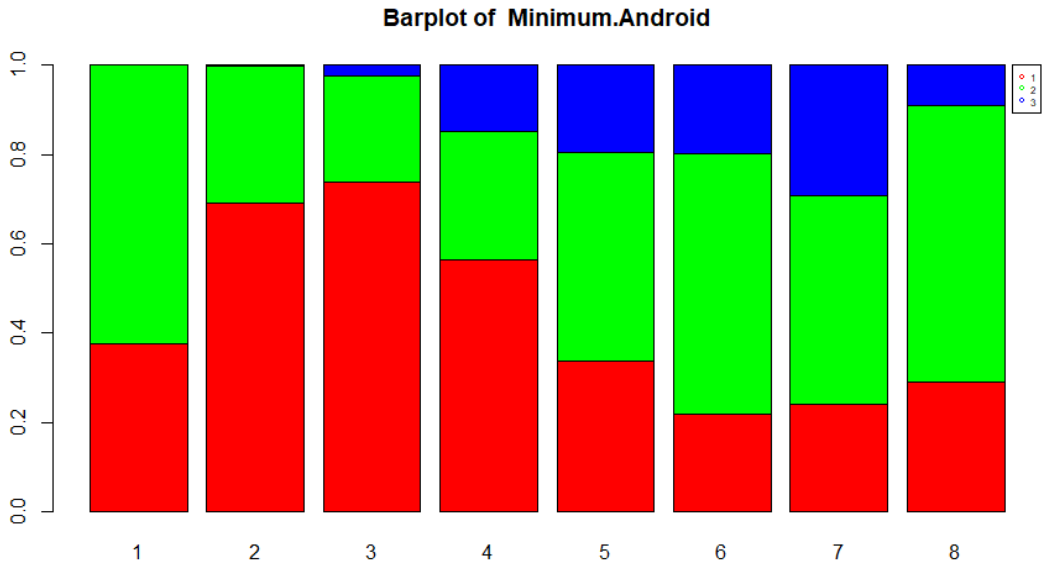


Figure 139: Barplots of Minimum.Android for each android version

Looking at Figure 138, we can't really extract conclusions, because for all of the clusters there are a lot of apps with `Minimum.Android` 4 and 5.

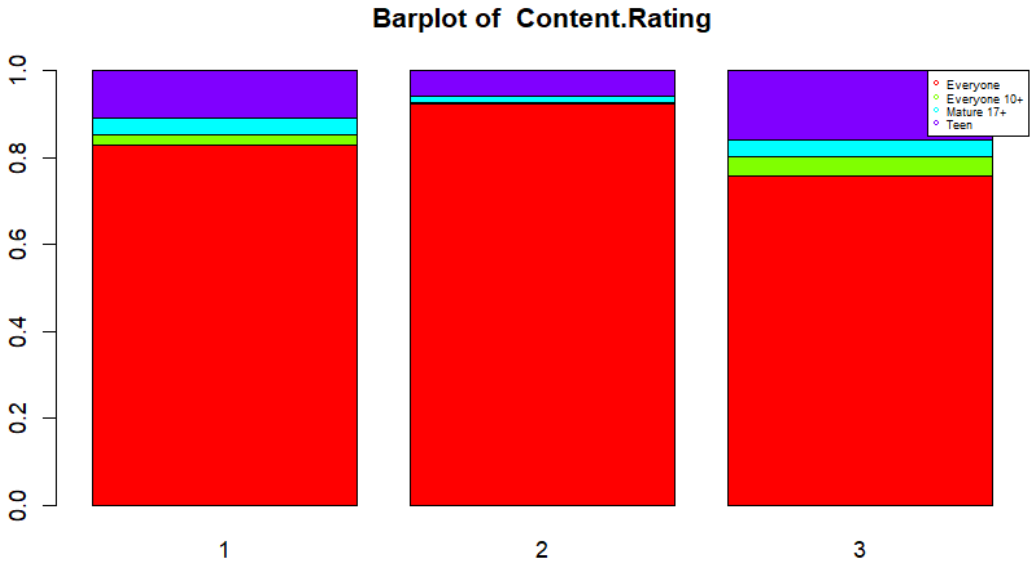


Figure 140: Barplot of Content.Rating for each cluster

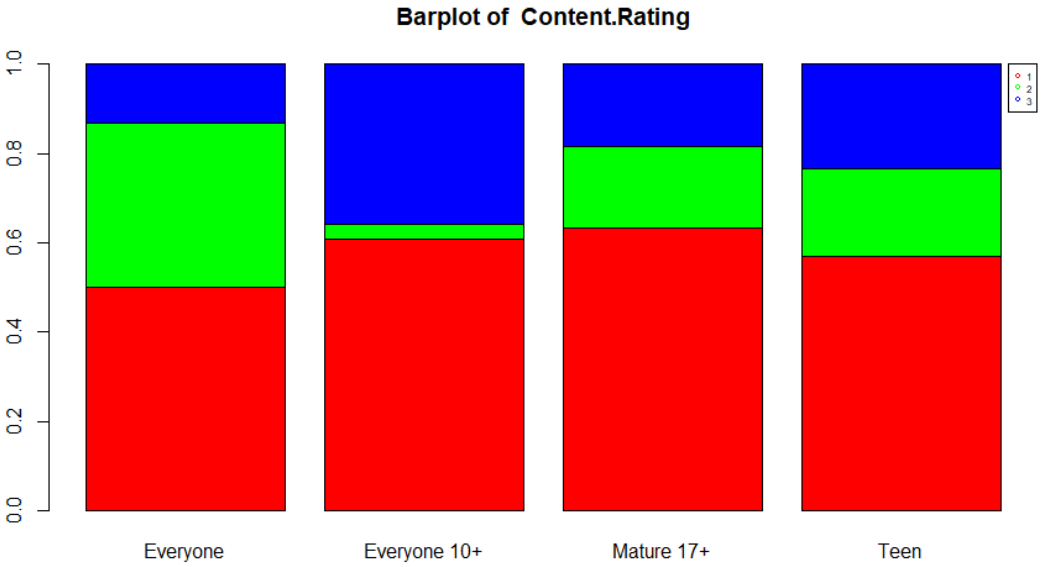


Figure 141: Barplot of Content.Rating for each content rating

Looking at Figure 140, we can say that there are a lot of apps for everyone in all of the clusters. One difference is that there are more apps for teens in cluster 3, which is also the cluster with more gaming apps.

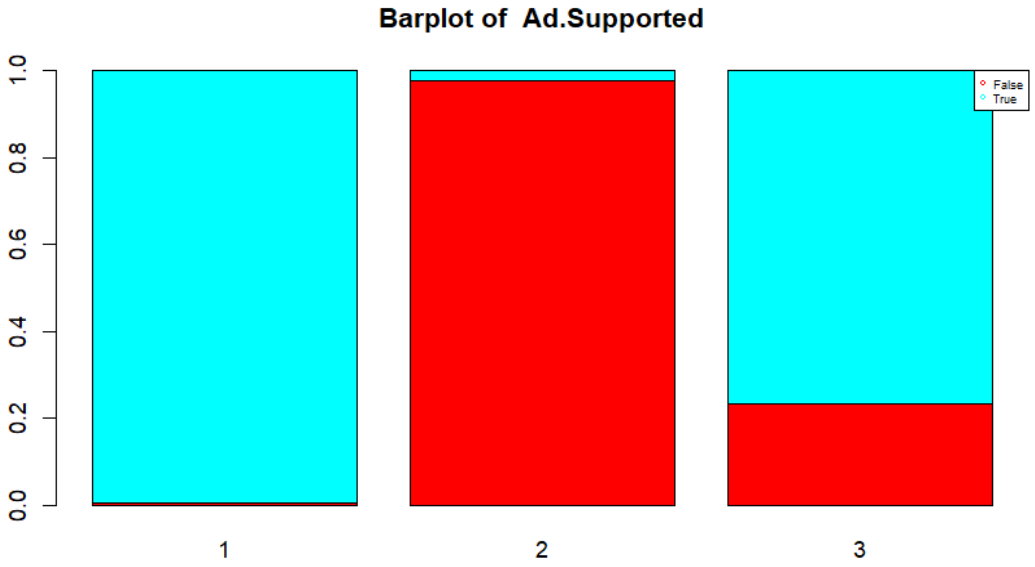


Figure 142: Barplot of Ad.Supported for each cluster

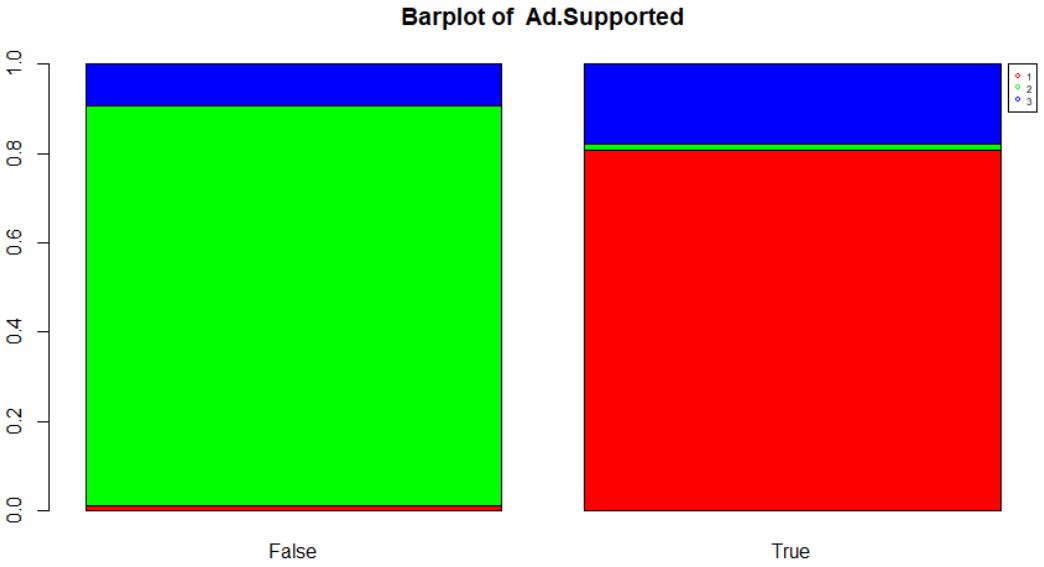


Figure 143: Barplot of Ad.Supported for each ad.supported

Figure 142 clearly shows some patterns. The vast majority of the apps in the first cluster have ads, the majority of the apps in the second cluster don't have ads and 80% of the apps in the third cluster have ads.

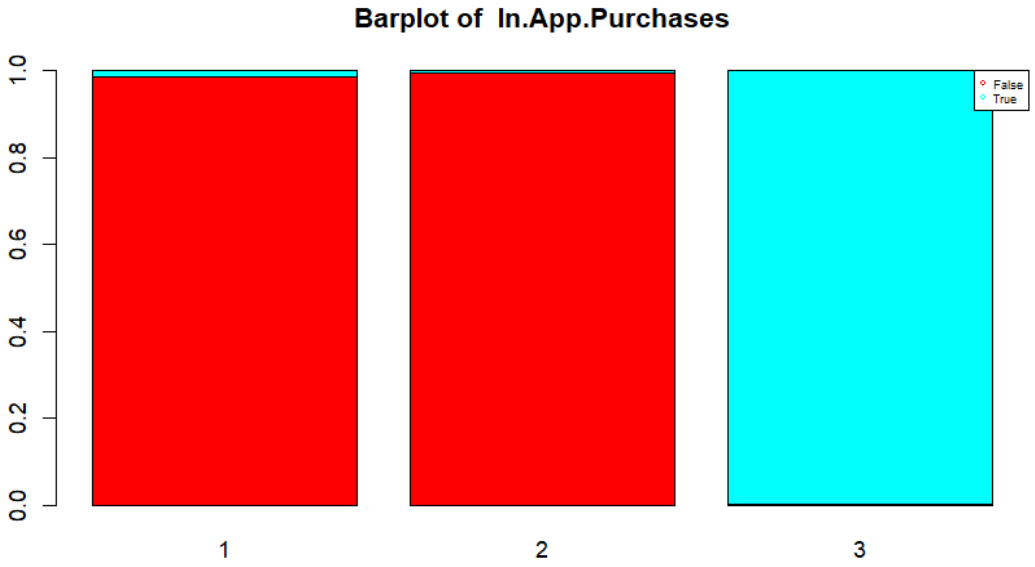


Figure 144: Barplot of In.App.Purchases for each cluster

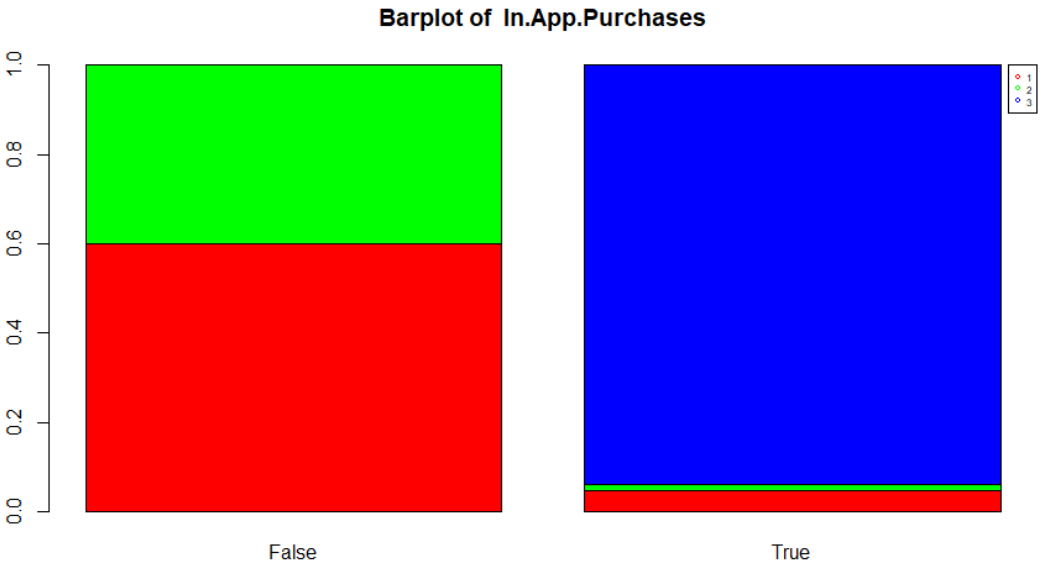


Figure 145: Barplot of In.App.Purchases for each in.app.purchases

Figure 144 also clearly shows some patterns. The majority of the apps in the first and second cluster do not have in-app purchases and the majority of the apps in the third cluster have in-app purchases.

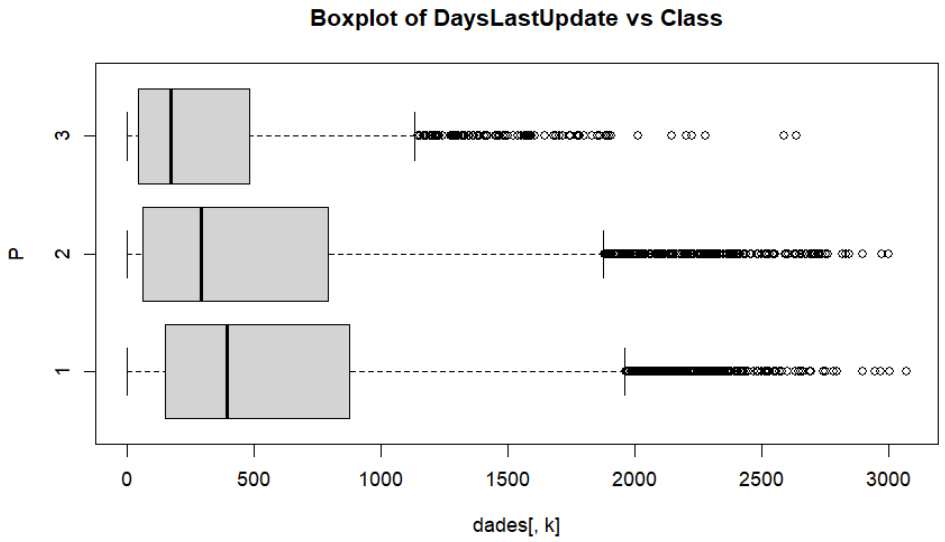


Figure 146: Boxplots of DaysLastUpdate for each cluster

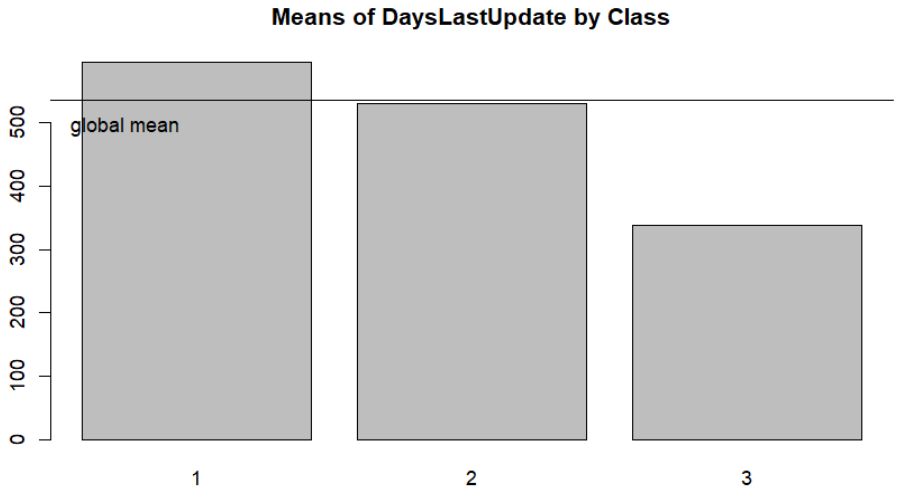


Figure 147: Means of DaysLastUpdate for each cluster

Figures 146 and 147 show differences between the clusters with the feature `DaysLastUpdate`. The first cluster has the highest days past since the last update (above the global mean) and the third cluster has the lowest days past since the last update (below the global mean).

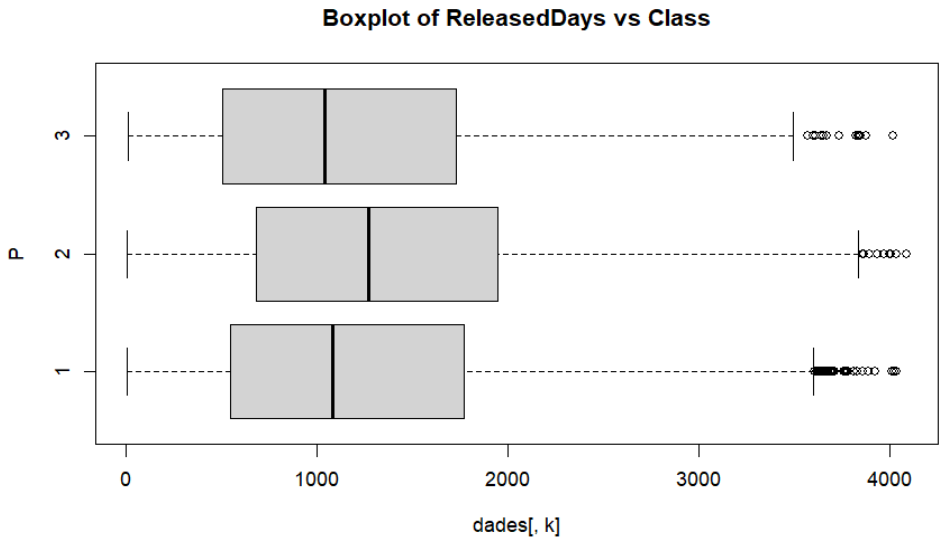


Figure 148: Boxplots of ReleasedDays for each cluster

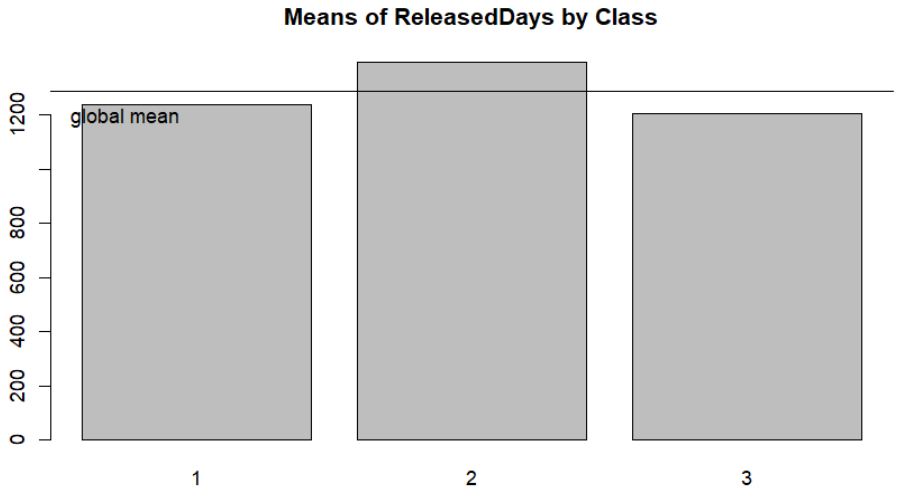


Figure 149: Means of ReleasedDays for each cluster

Looking at Figures 148 and 149 we can say that for the feature `ReleasedDays` there is not much difference for the cluster 1 and 3, but the cluster 2 is above the global mean.

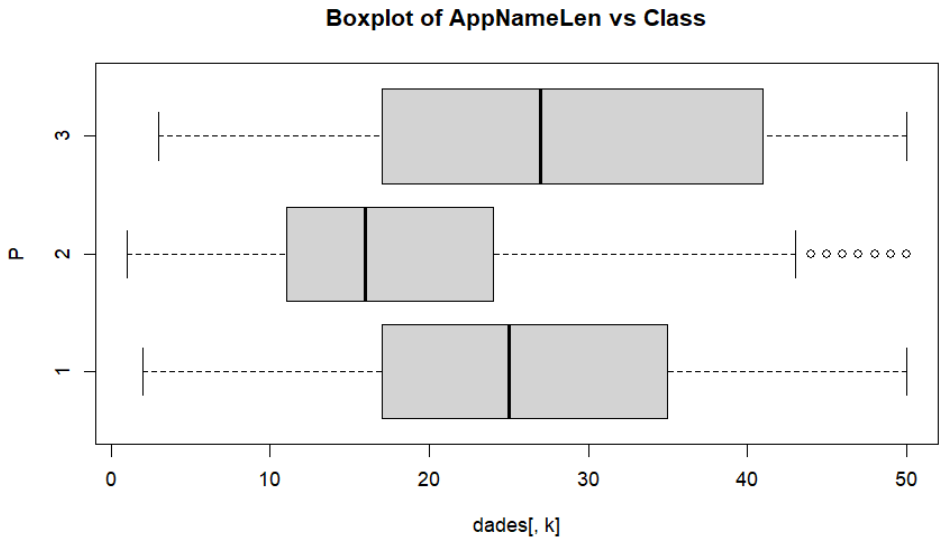


Figure 150: Boxplots of AppNameLen for each cluster

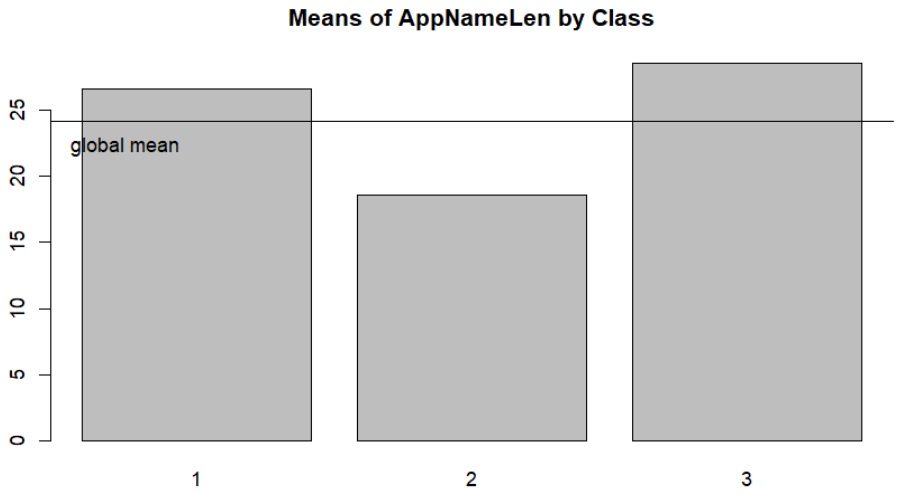


Figure 151: Means of AppNameLen for each cluster

Looking at Figures 150 and 151 we can say that for the feature `AppNameLen` there is not much difference for the cluster 1 and 3, but the cluster 2 is below the global mean.

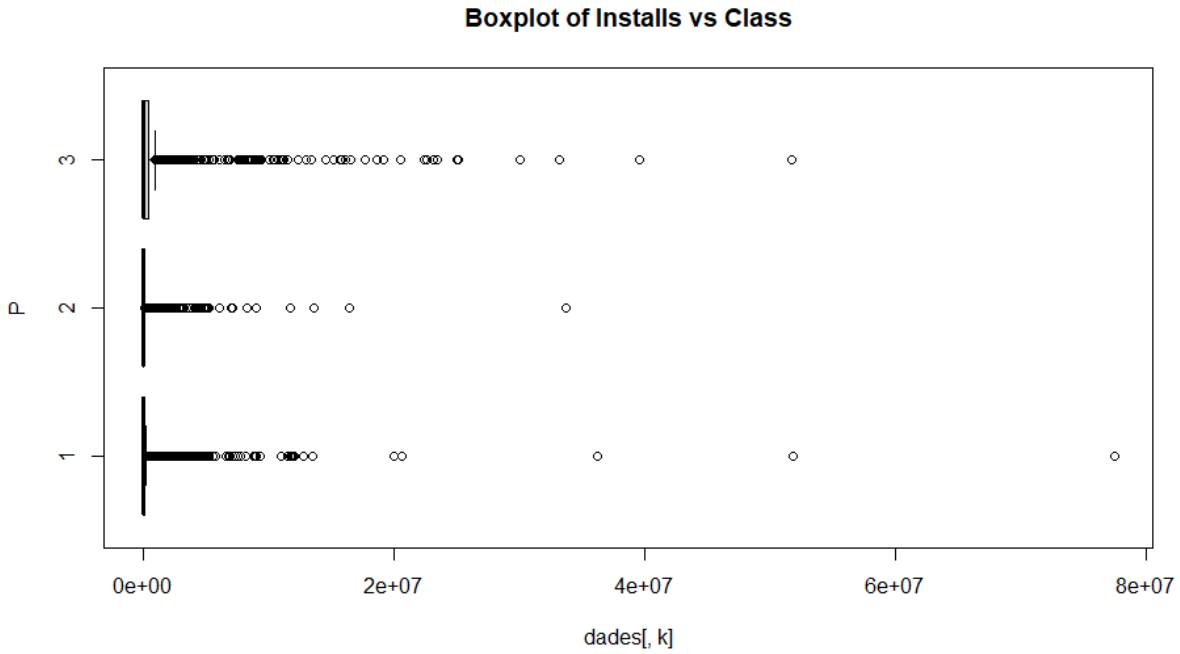


Figure 152: Boxplots of Installs for each cluster

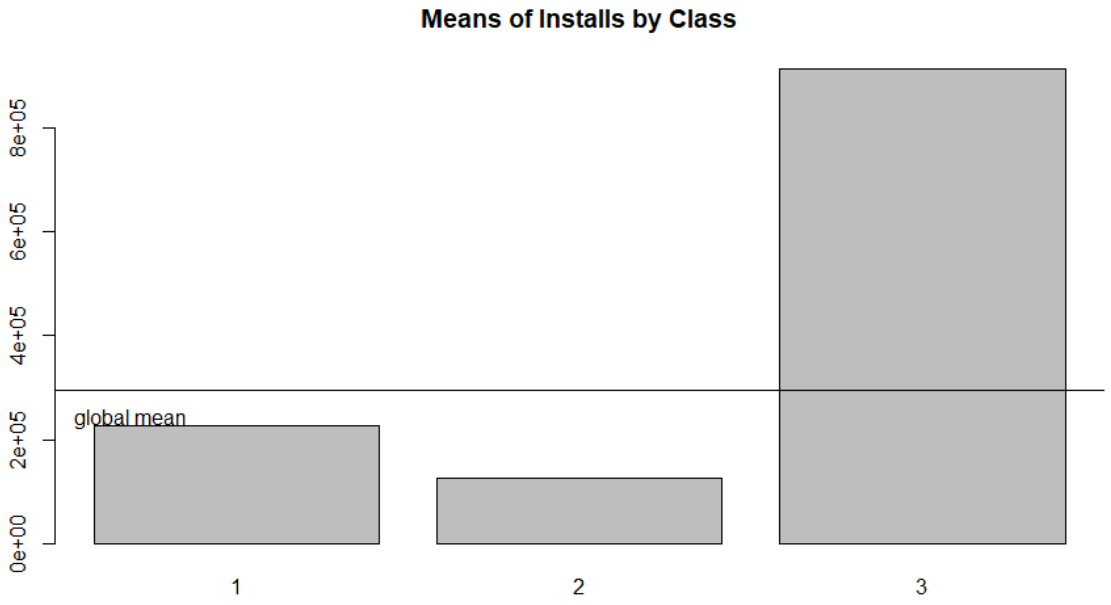


Figure 153: Means of Installs for each cluster

Looking at Figures 152 and 153 we can say that for the feature `Installs` there is a huge difference for cluster 3 around three times the global mean, cluster 1 is below the global mean and cluster 2 is halve the global mean.

12.2 Selecting relevant variables

As we have a really large dataset, the tests are not useful because they use the variance divided by the square root of samples and for this reason a small change in the mean succeeds in rejecting the null hypothesis. For this reason, we look at the previous plots to decide whether a feature is significant or not.

At first glance, `Category` does not look significant, but in the end we decided to include it, because there are more apps for some category types in the clusters (like a lot of gaming apps in cluster 3).

`Rating.Count` and `Size` are significant because for cluster 3, the mean is much higher than the others. `Ad.Supported`, `In.App.Purchases` and `Content.Rating` are significant, because for some clusters the proportion of a group is higher than the other groups.
`DaysLastUpdate`, `ReleasedDays`, `AppNameLen`, `Installs` are significant because the mean of some of the clusters are way different than the others.

`Rating` clearly is not significant, because the boxplots and the means of each cluster are practically the same. `Minimum.Android` is not significant, because there are the same proportions for all the clusters.

12.3 Final profiling of clusters

As a reminder, all our apps are free and with more than 20 votes (`Rating.Count`). With all the information from the previous sections, we can profile the clusters in the following way:

- **Cluster 1**: Lifestyle, entertainment and educational apps, low `Rating.Count`, low `Size`, for everyone, with ads, without in-app purchases, low frequency of updates.
  In this cluster there are well-established apps that everyone uses and they do not really need updates because their functionality is really basic.

- **Cluster 2**: Lifestyle and educational apps, low `Rating.Count`, medium `Size`, for everyone, without ads, without in-app purchases, medium frequency of updates, old apps, short names, low number of `Installs`.
  In this cluster there are outdated apps with low popularity, with short names in order to appear more in the search results.

- **Cluster 3**: Gaming apps, high `Rating.Count`, big `Size`, for teens, with ads, with in-app purchases, high frequency of updates, high number of `Installs`.

In this cluster there are apps for teens that want to play games. Because of the fact that the apps are free, they have ads and in-app purchases in order to have the budget to keep up with the high frequency of updates.

12.4 Comparison between Hierarchical Clustering & HCPC

After analyzing the outputs given by HCPC (Section 11.6), we got the following insights:

- The principal components most associated with the clusters are the dimensions 1, 2 and 3, which makes sense because there is no big difference as for the proportion of variance explained by each dimension.

- The variables that describe the most each cluster are the following:

```
Description of each cluster by quantitative variables
======================================================
$`1`
                  v.test Mean in category Overall mean sd in category   Overall sd       p.value
ReleasedDays    73.596564     2.060540e+03 1.287030e+03   7.133413e+02 8.498703e+02  0.000000e+00
DaysLastUpdate  69.978508     1.033596e+03 5.345455e+02   6.532147e+02 5.766658e+02  0.000000e+00
Installs        -5.577024     1.753002e+05 2.941383e+05   5.939561e+05 1.723049e+06  2.446675e-08
Rating.Count    -5.640556     1.233035e+03 2.153584e+03   4.648310e+03 1.319681e+04  1.695024e-08
AppNameLen     -34.593548     1.882857e+01 2.417051e+01   8.557906e+00 1.248672e+01 3.158848e-262
Size           -43.176854     2.104012e+00 2.633932e+00   8.748707e-01 9.924396e-01  0.000000e+00

$`2`
                  v.test Mean in category Overall mean sd in category   Overall sd       p.value
Size            41.899654     2.924178e+00 2.633932e+00   9.286120e-01 9.924396e-01  0.000000e+00
AppNameLen      34.016050     2.713523e+01 2.417051e+01   1.334068e+01 1.248672e+01 1.290108e-253
Installs        -6.906398     2.110769e+05 2.941383e+05   7.842938e+05 1.723049e+06  4.971157e-12
Rating.Count    -7.812243     1.433977e+03 2.153584e+03   5.802241e+03 1.319681e+04  5.617912e-15
DaysLastUpdate -68.927992     2.571048e+02 5.345455e+02   2.526087e+02 5.766658e+02  0.000000e+00
ReleasedDays   -74.351848     8.459731e+02 1.287030e+03   5.559459e+02 8.498703e+02  0.000000e+00

$`3`
                  v.test Mean in category Overall mean sd in category   Overall sd      p.value
Rating.Count   82.775735     1.305014e+05 2.153584e+03   8.408569e+04 1.319681e+04 0.000000e+00
Installs       76.803305     1.584284e+07 2.941383e+05   1.250784e+07 1.723049e+06 0.000000e+00
Size            6.887666     3.437075e+00 2.633932e+00   8.637587e-01 9.924396e-01 5.671521e-12
ReleasedDays    6.330414     1.919153e+03 1.287030e+03   7.448276e+02 8.498703e+02 2.445035e-10
AppNameLen      2.771147     2.823611e+01 2.417051e+01   9.901589e+00 1.248672e+01 5.585920e-03
DaysLastUpdate -4.880393     2.038750e+02 5.345455e+02   3.092975e+02 5.766658e+02 1.058748e-06
```

Figure 154: Description of each cluster by quantitative variables

The column Mean in category gives us the average of the variable in the cluster and the column Overall mean is the average of the variable for the whole data set. The p-value holds the following hypothesis: “the mean of the category is equal to the overall mean”.
A value of the v.test greater than 1.96 means that we reject the null hypothesis, the mean of the category is not equal to the overall mean and the sign of the v.test tells us if the mean of is lower or greater than the overall mean.

From the above figure, it can be seen that in cluster 1, the most associated variables are `ReleasedDays`, `DaysLastUpdate`, the mean value of these variables is greater than the overall mean across the clusters, so the conclusion here is that

cluster 1 is characterized by apps with a low value of `ReleasedDays`, `DaysLastUpdate`. On the contrary, cluster 2 is characterized by apps with a low value of `ReleasedDays`, `DaysLastUpdate`. For cluster 3, the most associated features are `Rating.Count` and `Installs`.

Finally, from the above most representative individual clusters information and the figure 154 we made the following tables that compare the similarities between clusters generated by HC (Hierarchical Clustering) and HCPC (Hierarchical Clustering on Principal Components).

Top 5 closest individual in cluster 1:

	Category	Rating	Rating.Count	Size	Minimum.Android	Content.Rating	Ad.Supported	In.App.Purchases	DaysLastUpdate	ReleasedDays	AppNameLen	Installs
10577	Educational	3.9	32	1.871802	4	Everyone	True	False	609	2671	22	4715
2830	Educational	2.1	278	1.871802	4	Everyone	False	False	628	2694	19	97393
674	Lifestyle	4.4	34	1.887070	4	Everyone	True	False	184	3150	22	26592
9696	Productivity	3.5	51	2.240710	4	Everyone	False	False	992	2217	20	5506
3252	Lifestyle	4.4	921	2.116256	4	Everyone	True	False	996	2232	22	206489

	HC	HCPC
Category	Lifestyle,entertainment and educational	Lifestyle and educational
Rating.Count	Low	Medium
Size	Low	Low
Minimum.Android	-	-
Content.Rating	Everyone	Everyone
Ad.Supported	Yes	Yes in general
In.App.Purchases	No	No in general
DaysLastUpdate	High	High
ReleasedDays	-	High
AppNameLen	-	Medium
Installs	medium	medium in general

Top 5 closest individual in cluster 2:

	Category	Rating	Rating.Count	Size	Minimum.Android	Content.Rating	Ad.Supported	In.App.Purchases	DaysLastUpdate	ReleasedDays	AppNameLen	Installs
6238	Lifestyle	4.6	319	2.944439	4	Everyone	False	False	111	1120	28	7327
3737	Educational	4.4	119	2.944439	5	Everyone	False	False	20	1269	29	17007
6223	Lifestyle	4.6	51	2.944439	4	Everyone	True	False	55	977	28	4071
4571	Lifestyle	4.6	521	2.944439	4	Everyone	True	False	298	997	28	58070
9100	Educational	4.1	2029	2.639057	4	Everyone	False	True	61	1135	29	91535

	HC	HCPC

Category	Lifestyle and educational	Lifestyle and educational
Rating.Count	Low	Medium
Size	Medium	Medium
Minimum.Android	-	-
Content.Rating	Everyone	Everyone
Ad.Supported	No	No
In.App.Purchases	No	No
DaysLastUpdate	Low	Low
ReleasedDays	High	Low
AppNameLen	Short	medium
Installs	Low	medium

Top 5 closest individual in cluster 3:

```
          Category Rating Rating.Count     Size Minimum.Android Content.Rating Ad.Supported In.App.Purchases DaysLastUpdate ReleasedDays AppNameLen Installs
2729          Game    3.9       142697 4.060443               4       Everyone         True             True             48          2484          30 15747146
2130          Game    4.0       185459 4.564348               7           Teen        False             True             26          1637          28  8277773
233           Game    4.1       146998 3.258097               4       Everyone         True            False            327          3050          34 11050132
9827 Entertainment    4.3       132217 3.637586               5       Everyone         True            False             53          2574          30 11575818
9538  Productivity    3.5        84542 2.079442               4       Everyone         True             True              3          2570          22 19223375
> |
```

	HC	HCPC
Category	Gaming	Gaming
Rating.Count	High	Medium
Size	Large	Large
Minimum.Android	-	-
Content.Rating	For teens	Everyone
Ad.Supported	Yes	Yes
In.App.Purchases	Yes	Yes
DaysLastUpdate	Low	Low
ReleasedDays	-	-
AppNameLen	Short	Long
Installs	High	High

To sum up, in general the clusters that we obtained by performing either HC or HCPC share some common characteristics like the category of the app, its size, whether it's ad supported, if there are in-app purchases, the frequency of update, the lifetime, etc.

13. Decisions tree

13.1 Parameter selection

For the selection of variables we choose `Rating` as the response variable. Because the data was heavily distributed to the values of `Rating` between 3-5 we decided to include more data for rating below 3 from the original dataset after processing it to have a more balanced distribution.

For the explanatory variables we selected all of them.

13.2 Models comparison and selection

13.2.1 Decision Tree for Regression

The response variable `Rating` is a continuous variable that ranges from 1 to 5, so the first thing we checked was if our target is nor normally distributed nor exponential. As we can see from Figure 155, the data is heavily right skewed, most of the data are between 3.5-5.

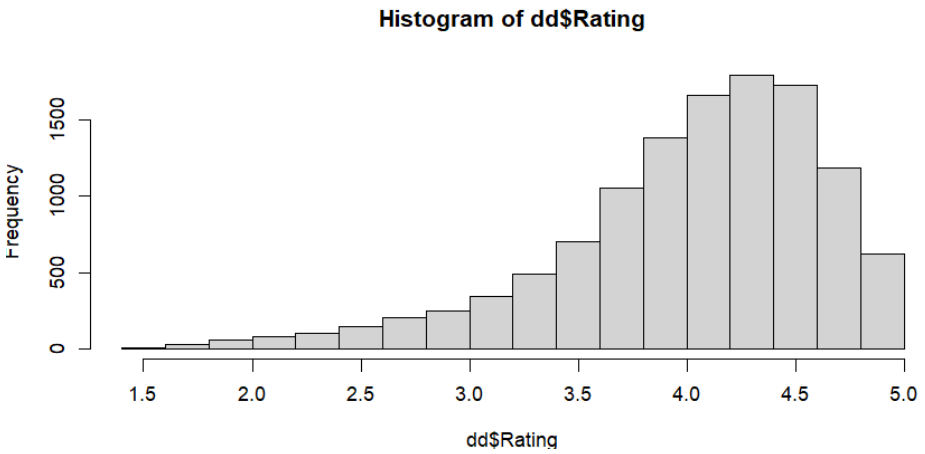


Figure 155: Histogram of `Rating`

This distribution of data would lead to the model output having a bias in the same direction. We can see from Figure 156 that with this target distribution the trained model only has five output predicted values, all ranging from 3.8 to 4.6.

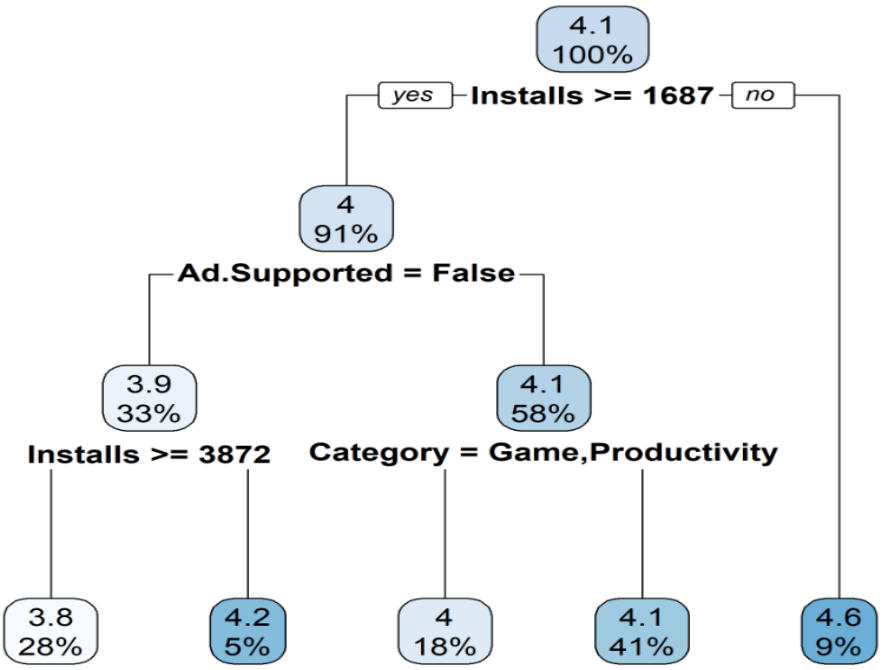


Figure 156: Model generated when target is not normally distributed

In order to solve this problem (heavily right skewed distribution target) we found that techniques like SMOGN and SMOTER are available to be used. However, as our current dataset is extracted from a large dataset of 2M rows, we decided to just bring more data from the original dataset in order to have more examples for the lower rating range.

After that, from Figure 157 we can see the tree plot of our basic model without applying any techniques of improvement. Now the predicted output values range from 2.2 to 4.6.

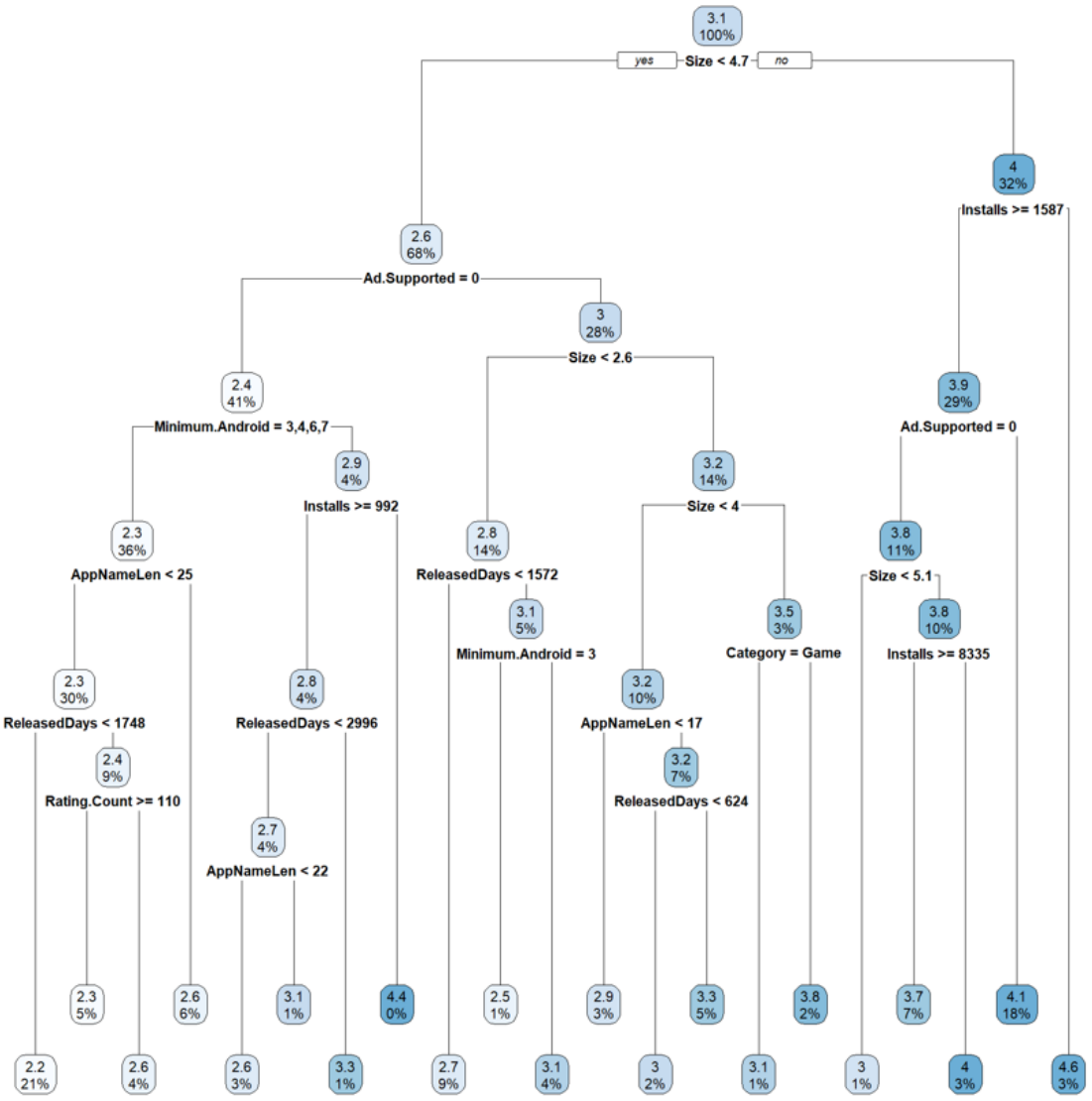


Figure 157: Basic decision tree model

We used the metrics RMSE, MAE and R2 to evaluate the performance of this basic model and the results are the following:

RMSE	MAE	R2
0.8360	0.6476	0.4307

All these values indicate that the model has a really poor performance, in order to improve it, we tried to apply the following strategies:

- **New feature generation** by looking at the basic tree plot, for example, from the figure x we can see that one of the most important variables used to split

the tree is Size, so for that variable we generated a new factor which indicates if the size value of apps equals to the split value.

- **Dimensionality reduction using PCA**
  Before fitting our data to the model, we tried to apply PCA to reduce the number of features that Decision Tree would use to build the model. Our goal here is by doing so, obtain a better performing model and increase the probability of finding the most discriminative features for the model.
  After performing the decision tree on PCA outputs, we didn't see any improvement from the results obtained:

	RMSE	MAE	R2
Decision Tree basic	0.799	0.660	0.458
Decision Tree after PCA	0.8362	0.6751	0.34

  The conclusion is that dimensionality reduction technique does not have any effect on the performance of the decision tree.

- **Transforming skewed independent variables using log**
  Taking into account that one of the disadvantages of Decision Tree is that a small variation in the data might lead to generating a completely different tree.
  We tried to transform variables Installs, Size and Rating.Count using log as they have a non normal distribution. The aim is to find some hidden patterns after changing the scale. However, the results indicate that there is no improvement in the model performance.

- **Normalization of data**
  We also tried to normalize all the numerical variables to ensure that features with large scale are not prioritized over the rest. But then we found out that neither Decision Tree nor Random Forest are distance-based models as K-means, so these algorithms are insensitive to the scaled features. And a feature does not have influence over other features.

- **We did not apply methods to prevent the tree from overfitting** like post pruning or cost complexity pruning because the model performance is so poor that it never overfitted.

- **Applying Random Forest**
  This algorithm uses multiple trees to do the prediction or classification, and in general, it has better performance than a single decision tree because it can reduce bias and variance present in decision trees.

We performed random forest to fit our data with the following default configuration:
number of trees = 500
maxim number of features = 3

And the results are shown below:

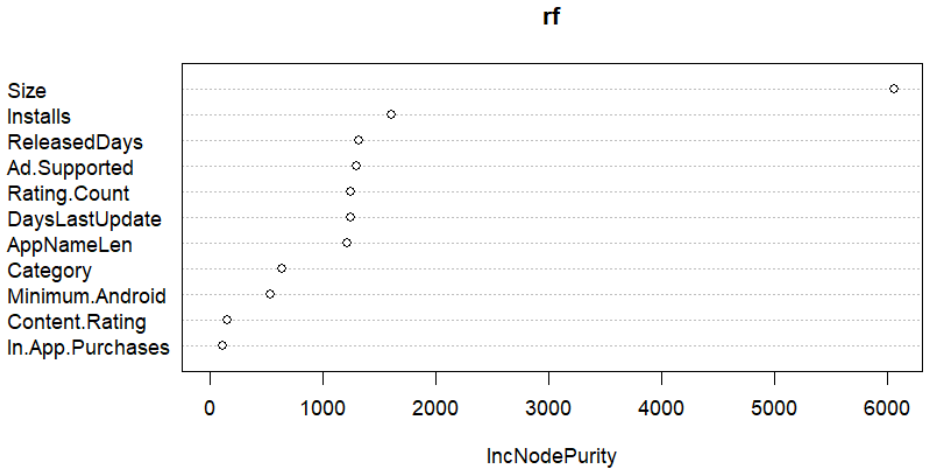


Figure 158: Mean decrease Gini (IncNodePurity) of each variable

From the above Figure 158 we can see that the most important variables here are the ones that achieved higher increase in IncNodePurity, in this case, we have `Size`, `Installs`, `ReleasedDays`, `Ad.Supported`, `Rating.Count`, `DaysLastUpdate` and `AppNameLen`. What the model does when splitting is find the variable with higher inter node variance and small intra node variance.

	RMSE	MAE	R2
Decision Tree	0.799	0.660	0.458
Random Forest basic model	0.769	0.622	0.51

From the above table we can see that the performance is slightly better with this model than with a single tree, but it's still not good enough. For this reason, in order to improve it we tried the technique **Hyper Parameters tuning.**

Firstly, among all the random forest tuning parameters we chose the most important ones that are the number of trees (n_trees), maximum nodes of the tree (max_nodes) and then we use the GridSearchCV method to find the best parameters configuration by passing our model, the parameter we want to tune and the number of cross-validation iteration, in this case we choose 5.
The results we obtained show that it doesn't help to improve the performance of the model in this case.

13.2.2 Decision Tree for Classification

After performing all above strategies the model still did not fit the data well, so we decided to convert this regression model to a classification one, by creating a new feature f.rating that discretizes the `Rating` values into intervals (eg: from 1 to 2), as Rating ranges from 1 to 5.

- **Approach 1: 4 Classes Classification**

To begin with, we created a new feature f.rating by dividing our continuous target to a factor one with 4 levels (1-2, 2-3, 3-4 and 4-5). Then we applied classification on our data, the results are the following:

```
Confusion Matrix and Statistics

          Reference
Prediction  1-2  2-3  3-4  4-5
       1-2 1234  708  411   91
       2-3   95  145   68   14
       3-4  235  307  552  135
       4-5   38  191  436 1231

Overall Statistics

               Accuracy : 0.5368
                 95% CI : (0.5239, 0.5495)
    No Information Rate : 0.2719
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3759

 Mcnemar's Test P-Value : < 2.2e-16
```

Figure 159: Confusion matrix for 4 classes classification

We can see from Figure 159 that having four balanced classes (almost 5000 rows per each class), the accuracy in the testing set is about 53.68%. The probability of labeling an observation of class 2-3 as 1-2 is pretty high and the same happens to the classes 2-3 and 3-4.

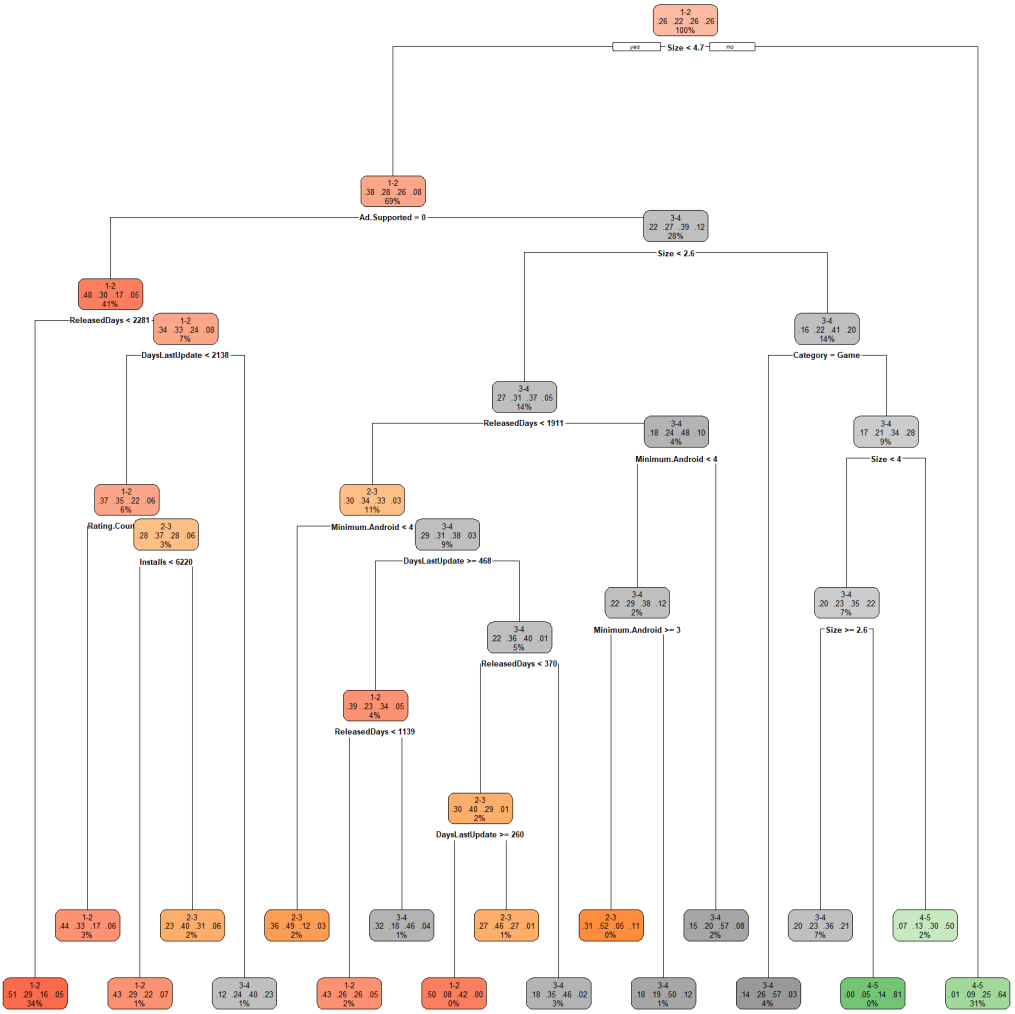


Figure 160: 4 classes classification model

From the above figure we can see that the vast majority of the observations of the class 1-2 has a mean Size value less than 4.7 and they are also not ad supported. In most of the cases we find classes 1-2 and 2-3 having the same father node(coming from the same path),this behavior may explain the incapability of the model in distinguishing between these two classes. The same happens to the classes 2-3 and 3-4, but with less frequency.

We also tried to perform a random forest to check if there could be any improvement and we got the same results, the accuracy increased from 53.68% to 55.86%. We believe that this little improvement is due to the fact that when training the model, the vast majority of the trees built have the same structure as they use the same split nodes because the decision tree algorithm is based on the algorithm greedy, which tend to always find the local optimal solution(in this case the same split node is being chosen by the trees) and the global optimal is never reached.

- **Approach 2: 3 Classes Classification**

After analyzing the results obtained in the previous 4 classes classification and taking into account the problems found, we decided to reduce the number of classes by merging the classes 1-2 and 2-3, as they share some common characteristics.

```
Confusion Matrix and Statistics

          Reference
Prediction  1-3  3-4  4-5
       1-3 1599  702  179
       3-4  442  779  140
       4-5  243  828 2056

Overall Statistics

               Accuracy : 0.6363
                 95% CI : (0.6249, 0.6476)
    No Information Rate : 0.3408
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4537

 Mcnemar's Test P-Value : < 2.2e-16
```

Figure 161: Confusion matrix for 3 classes classification

From Figure 161 it can be seen that now the problem is between classes 1-3 and 3-4, the probability being labeled incorrectly as the other class is also high for both cases. As for class 4-5, there is a probability of 22% of being labeled as class 3-4. And the accuracy increased from 53.68% to 63.63%.

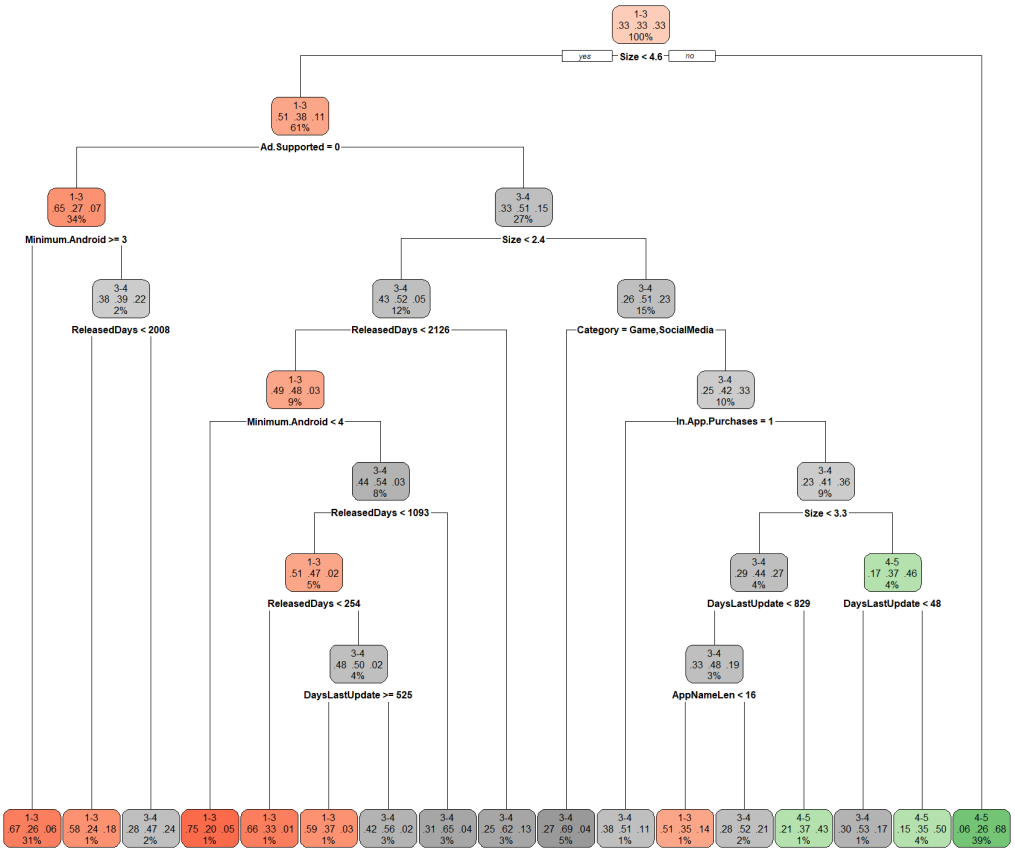


Figure 162: 3 classes classification model

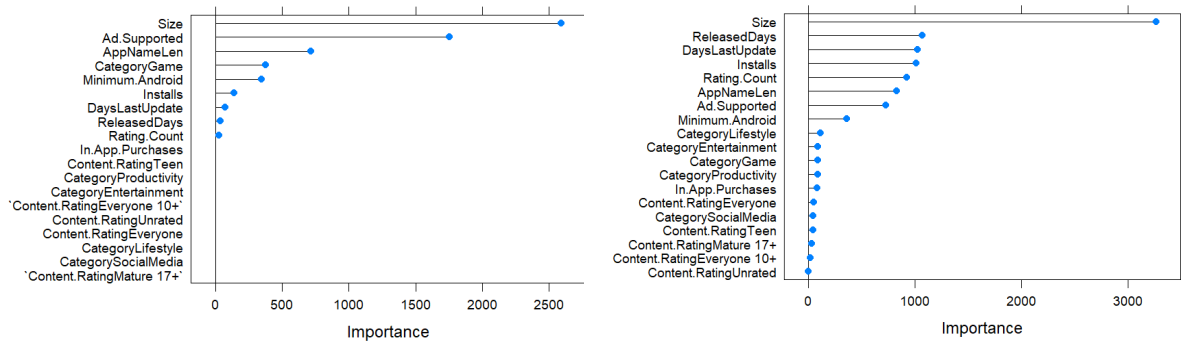
From Figure 162 we can see that most of the leaves of classes 1-3 and 3-4 have the same ancestor node, which explains why the model tends to label these two classes incorrectly.

- **Approach 3: 2 Classes Classification**

Finally, we decided to reduce the number of classes to 2 by merging the classes 3-4 and 4-5. By doing so, the accuracy obtained is the highest (about 77%) compared to the previous ones and after all the strategies applied in order to increase the model performance, we believe that this accuracy is the best we can obtain by performing the algorithm decision tree. In the case of our data, the random forest algorithm cannot help to achieve a higher accuracy.
To sum up, in the following sections, we will explain more in detail this final model with 2 classes.

13.3 Most important parameters selection

In order to select the most important variables, we plot the importance value of each feature returned by a Decision Tree (DT) and a Random Forest (RF) with 500 trees. In the figure below, we show the results obtained with each method:



Decision tree Random Forest

Figure 163: Importance value of each feature in Decision tree and in random forest

Noticing these results, the following explanatory variables are selected to retrain the models: `Installs`, `Size`, `Ad.Supported`, `AppNameLen`, `Category`, `ReleasedDays`, `Rating.Count`, `Minimum.Android`, `DaysLastUpdated`.

We can see that the most important numerical variables obtained using the DT and RF are the same ones obtained as when we performed PCA.

After doing some testing, we did not remove any feature from the training data since the accuracy decreased. Thus, at the end, we ended up working with all the 11 explanatory features of our data.

13.4 Tree plot and interpretation

The resulting DT obtained can be seen in Figure 164.

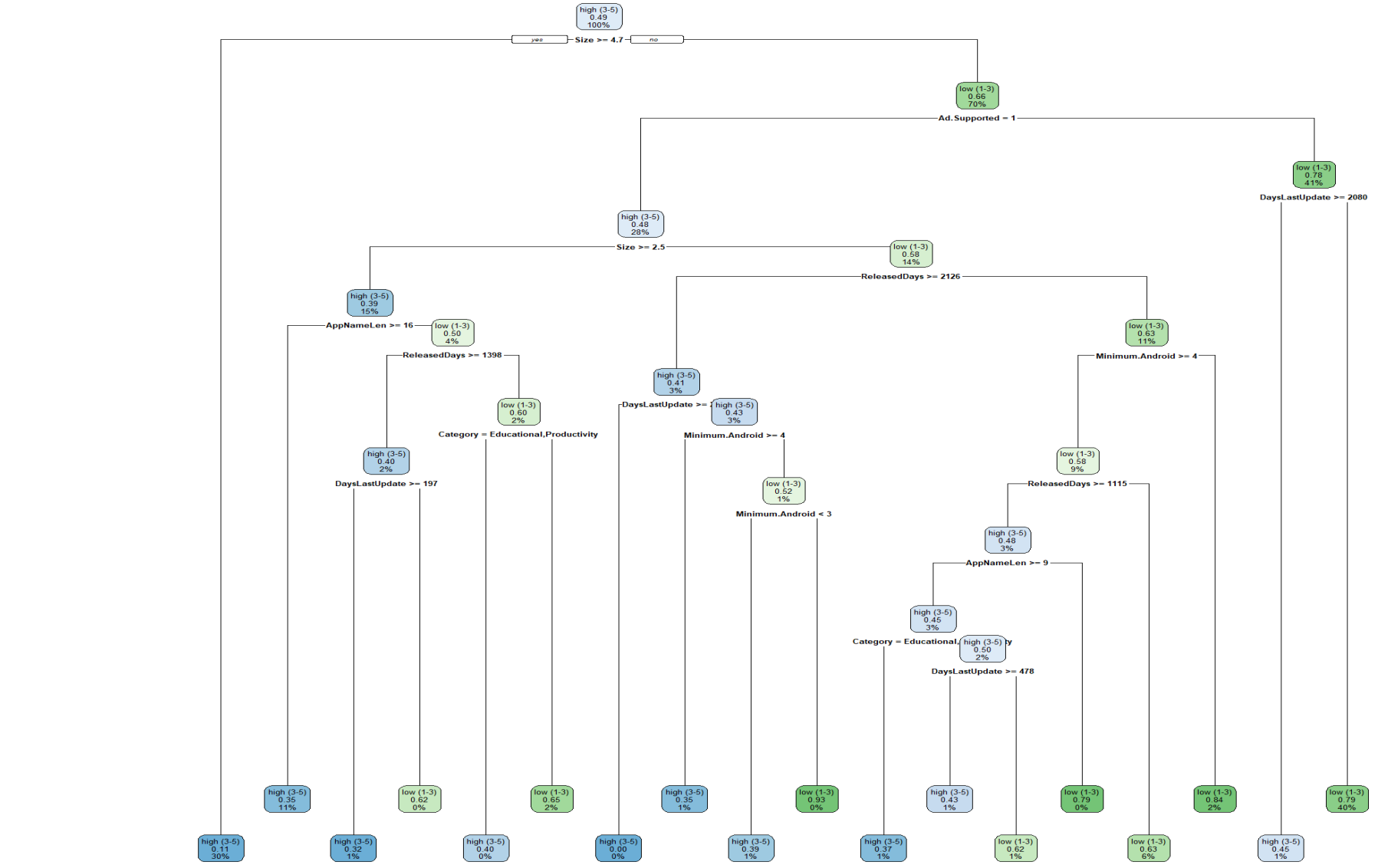


Figure 164: Decision Tree model

The DT classifies the 81% of the training data with the following three main rules:

- Rule 1:
  IF **size >= 4.7**
  THEN **rating = high**

- Rule 2:
  IF **size < 4.7** && **ad.Supported != 1** && **DaysLastUpdate < 2080**
  THEN **rating = low**
  ELSE **rating = high**

- Rule 3:
  IF **size < 4.7** && **ad.Supported = 1** && **Size >= 2.5** && **AppNameLen >= 16**
  THEN **rating = high**

From the tree plot, we can extract the following rules:

- The apps with a size higher or equal to 4.7 tend to have high rating. **30 %** of the data satisfy this rule, with a 0.11 probability of having a lower rating.

- The apps with a size higher than 2.6, that include ads and that have a name length higher than 16 tend to have a high rating with a 0.35 of probability of having low rating. This rule is satisfied by **11%** of the data.

- The apps that have a size smaller than 4.7, do not support ads and have less than 5.5 years of antiquity, tend to have a lower rating (with a 0.79 of probability of having a high rating). This split includes **40%** of the data.

- **6%** of the data are classified as apps with low rating since they have a size higher than 2.5, include ads, have android version lower than 4 and have less than 1115 released days. These apps have a 0.63 probability of having a high rating.

As a conclusion, we can see the following general patterns from the tree plot:

- Almost all the data with a relatively large size will have a high rating.

- The apps that do include ads usually have high rating, while the ones that do not have lower probability of high rating.

- In general, the apps with a low minimum android version have high rating, while for the other ones it is less common for them to have high rating.

- In general, the older the app is (the higher the number of released days), the rating is high.

- There is a higher probability of obtaining high rating than low rating.

- The leafs with the worst accuracy are also the ones not containing much data.

13.5 Model validation

In order to evaluate our model, we first split our data into training (with 70% of the total of observations) and testing data (with the 30% remaining data). After that, we performed five metrics to evaluate the model: confusion matrix, accuracy, recall, precision and ROC function.

CONFUSION MATRIX & ACCURACY

We checked if the accuracy of the model obtained with the training set is the same as with the test model:

```
            Reference
Prediction   high (3-5) low (1-3)
  high (3-5)       2288       641
  low (1-3)         711      2274

               Accuracy : 0.7714
                 95% CI : (0.7605, 0.782)
    No Information Rate : 0.5071
    P-Value [Acc > NIR] : < 2e-16
```

Figure 165: Concussion Matrix in Test

```
            Reference
Prediction   high (3-5) low (1-3)
  high (3-5)       5323      1409
  low (1-3)        1678      5359

               Accuracy : 0.7758
                 95% CI : (0.7687, 0.7827)
    No Information Rate : 0.5085
    P-Value [Acc > NIR] : < 2.2e-16
```

Figure 166: Concussion Matrix in Training

We can see from the results that the accuracy is acceptable, the model classified 77% of the observations correctly. Furthermore, we get an accuracy of 77% in both training and testing, which means that our model does not have neither overfitting nor underfitting.

RECALL

We calculated the recall of the model predicting the test data, and we got a value of **0.74**. That means that the model identified 74% of the actual positive cases, while **26%** of positive cases were not identified.

PRECISION

We also calculated the precision of the model detecting the positive cases in the testing data, as a result the model has a **79,5** precision. That means that, from the total cases that were identified as positive cases, **79,5** of them were identified correctly.

ROC

The ROC plot shows the rate of true positive versus the false positive in each threshold. The more the line is closer to the top-left, the better is the performance of

the model. From the ROC figure, we can see that our model keeps increasing positively but then it stops increasing when it reaches the 0.7 in the y-axis.

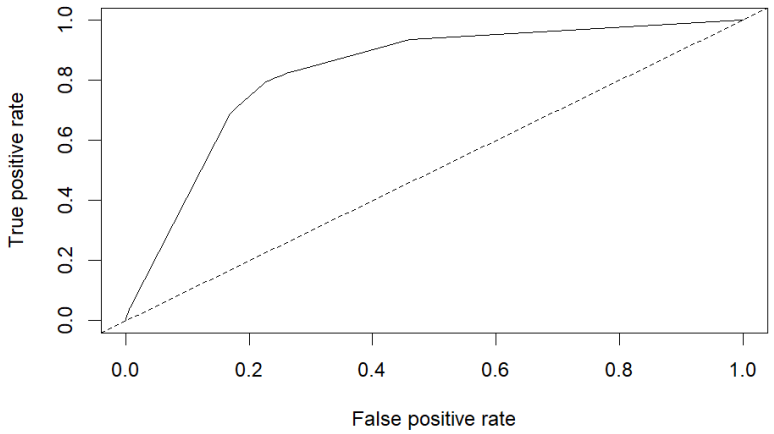


Figure 167: Roc Function

13.6 Model predictive power

In order to check the predictive power of the model we take a look at the confusion matrix:

```
Confusion Matrix and Statistics

            Reference
Prediction   high (3-5) low (1-3)
  high (3-5)       5323      1409
  low (1-3)        1678      5359

               Accuracy : 0.7758
                 95% CI : (0.7687, 0.7827)
    No Information Rate : 0.5085
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5518
```

Figure 168: Confusion Matrix

As it can be observed the model has an accuracy of 0.77, making more wrong predictions when the value is high than when it is low, but not by much. The problems with these model predictions would be with the precision of the prediction, because we have only two levels that take lots of values, and that even with these restrictions the accuracy we have makes it so that we have a considerable number of false positives.

13.7 Conclusions

In conclusion, from all the models we build, the decision tree as a classificator with two levels was the model with the better performance. Our target variable, `Rating`,

includes numeric values from 1 to 5. The first approach was building a regression DT model using rating as numeric, but the performance of the model was very poor, even after rounding the values and trying with a regressor RF.

Our second approach was to transform our problem into a classification problem by converting the Rating column to factor. The performance of the model was also poor, so we tried to discretize the values from [1,2,3,4,5] to [1-3,3-4,4-5]. In this case, the model was misclassifying the classes 3-4 with 4-5 since these observations have similar features. Since these classes have similar properties, we joined them in order to improve the model performance.

Thus, after the transformations we end up dealing with a binary classification problem, where the target has the following levels:

- **Low**: means that the rating is between 1 and 3.
- **High**: means the rating is between 3 and 5.

With this new target variable, we improved the DT model accuracy from 0.5 to 0.77. We also built a RF model, but we got the same results as the DT, thus, we ended up choosing the DT as our final model. Moreover, our DT model has neither overfitting nor underfitting, since we got the same accuracy by predicting both training and testing data.

From the results obtained from the decision tree we can see that high rating values are the most common, being slightly less probable that they appear for apps with no ads of small size and being generally more common for older apps, apps of relatively big size and apps with a low minimum version of android required to have a high rating.

The model will predict in most cases that the rating is high, having an accuracy of 0.77 means that most of the time the prediction will be correct, but there will still be a considerable number of false positives that we will need to take into account. The problem with the predictions of this model is the precision, that it is only able to predict if the rating is high or low, not making it possible to know how high or low the rating truly is.

14. Linear discriminant analysis

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that reduces dimensions based on a supervised algorithm. The goal of LDA is to find the feature subspace that optimizes class separability.

14.1 Data preparation

In order to perform LDA we need a categorical output and the predictors should be numerical. This is why the first thing we did was to transform our numerical target, `Rating`, to categorical. Rating ranges from 1 to 5 so we decided to have two levels when transforming it, one called `low` for individuals with rating less than 3 and another one called `high` for individuals with rating greater than or equal to 3.

We applied a logarithmic transformation to `Installs`, `Size`, `DaysLastUpdated` and `AppNameLength`. Then, we transformed the variables `Ad.Supported` and `In.App.Purchases` to numeric.

Taking into consideration PCA results as well as decision tree results we decided to use the following numeric variables as predictors: `Size`, `DaysLastUpdated`, `Minimum.Android`, `AppNameLength`, `Ad.Supported` and `Installs`. We do not use `Rating.Count` and `ReleasedDays` because they have huge correlations with `Installs` and `DaysLastUpdated` respectively (no multicollinearity).

14.2 Process description

In order to perform LDA, once data preparation was done, we splitted the data into training and test sets (70-30 %).

In LDA there is no need to standardize the data, but we did it because by doing so the results obtained are usually easier to interpret.

Figure 168 shows the results after applying LDA. As we only have 2 levels, there is only one discriminant function (LD1).

```
Prior probabilities of groups:
high (3-5)  low (1-3)
 0.5052257  0.4947743

Group means:
                 Size DaysLastUpdate Minimum.Android AppNameLen Ad.Supported    Installs
high (3-5)  0.4754529    -0.01861634       0.0993590  0.2410608    0.3694829  0.07428783
low (1-3)  -0.4854962     0.01900959      -0.1014578 -0.2461529   -0.3772877 -0.07585706

Coefficients of linear discriminants:
                        LD1
Size            -0.769638265
DaysLastUpdate  -0.102544775
Minimum.Android -0.072125478
AppNameLen      -0.195592740
Ad.Supported    -0.529668511
Installs         0.001333657
```

Figure 169: LDA results

As it can be seen in Figure 170, when plotting the stacked histograms of the LDA values we got a complete overlap for the values of low rating, which is not good.

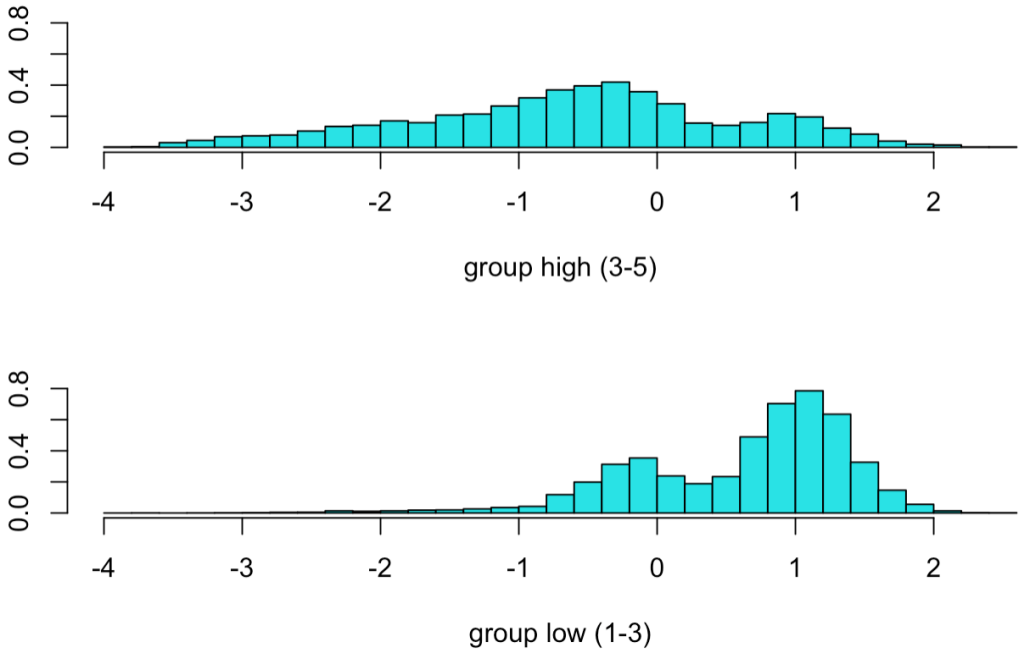


Figure 170: Histogram of LDA values of each group

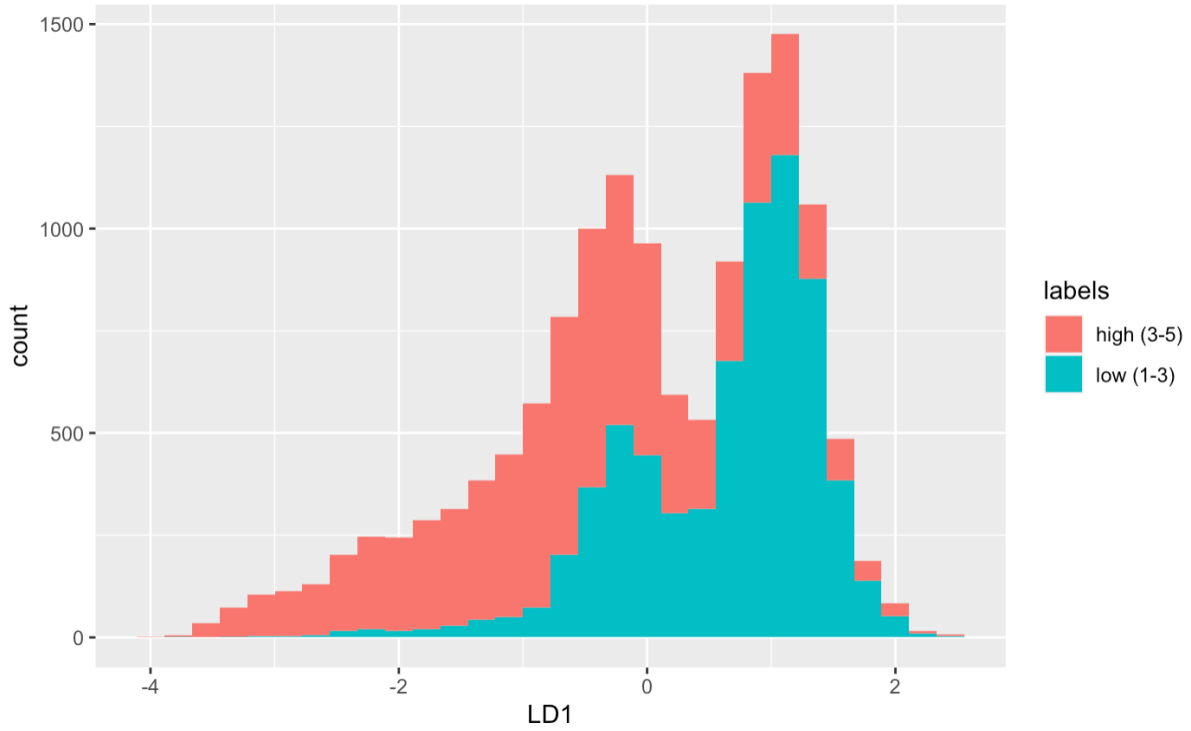


Figure 171: Stacked histogram for discriminant function values

We then calculated the confusion matrix, as depicted in Figure 172.

	observed	
predicted	high (3-5)	low (1-3)
high (3-5)	5022	1659
low (1-3)	1939	5158

Figure 172: Confusion matrix training data

We got a training accuracy of 74% and a training misclassification rate of 26%.

After this, we proceeded to apply the discriminant function obtained over the test data.

We obtained the confusion matrix depicted in Figure 173. From it we can see that from the 2852 real individuals with rating high, 2148 have been correctly assigned and 704 have been wrongly assigned. From 3053 real individuals with rating low, 2162 have been correctly assigned while 891 have been wrongly assigned.

	observed	
predicted	high (3-5)	low (1-3)
high (3-5)	2148	704
low (1-3)	891	2162

Figure 173: Confusion matrix test data

We got a test data accuracy of 73% and a misclassification rate of 27%.

14.3 Conclusions

As we did not get good results after applying LDA, we tried transforming `Rating` to a categorical variable with 3 levels, which gave us two discriminant functions but the results were also bad. We also tried to apply LDA on PCA outputs but the performance did not improve.

We believe that categorical variables are very important, consequently using just the numeric variables is not enough. As we are dealing with apps, the category to which an app belongs is important.

15. Discussion and conclusions

About the dataset

This dataset is a complex dataset due to the fact that there are a lot of apps. First, there are 48 types of apps categories that are unbalanced. Second, we deal with apps with different time ranges and this app changes along the time. Third, the descriptors of the apps are very general in order to fit all the diversity of apps, this means that the data set does not contain specific values that could give us important insights of specific categories.

Initial analysis

When we started to study the data, we discovered important insights about our data like the apps with low `Rating.Count` but having higher `Rating` do not follow the same distribution as the rest of the apps. And the apps with a high number of `Installs` have a `Rating` around 4. Moreover, there is a huge correlation between `Rating.Count` and `Installs`. And also there is a huge difference between paying apps and free apps. So we decided to just study the apps with more than 20 in `Rating.Count` and free apps.

Factorial analysis

From the PCA, we found out that the most important variables in explaining the variability of the dataset are:`Rating.Count`, `Installs` and `DaysLastUpdated`. As `Rating.Count` and `Installs` are highly correlated, in terms of total variance explained by the components, the deletion of one of them does not change the results. Also we found out that a higher number of installs does not mean that the app will have a higher rating, and vice-versa.
Finally, the `Gaming` and `Lifestyle` categories are the ones with the highest number of installs.

From the MCA we noticed the following:

- A level that could be **entertainment**, with gaming apps on the positive side and productivity apps on the negative.
- A level that could be **procrastination** with social media apps on the positive side and educational and entertainment apps on the negative.
- A level that could be **companionship** with gaming apps on the positive side and social media on the negative side.
- A level that could be **longevity** having on the positive side old apps and on the negative side new apps.
- A level that could be **helpfulness in a person's lifestyle**, having lifestyle apps on the positive side and educational ones on the negative.

From the MFA we noticed the following:

- There is correlation between `Rating.count` and `Installs`, `Rating` is not dependent on the popularity of the app and older apps are sometimes more popular than new ones.
- There are no clear clusters of individuals in the data.

The clusters

After doing the hierarchical clustering, we decided that the ideal number of clusters is three. The profiling of the three clusters was done by looking at the profiling boxplots and barplots. The plots of the means of the numerical features were extremely useful to see if the feature was significant to discriminate the clusters. Regarding the categorical features, we looked at the proportions of each cluster to see if there were differences between them.

In the end, we obtained three clusters that have different profiles of apps. In the first cluster there are well-established apps that everyone uses and they do not really need updating. In the second cluster there are outdated apps with low popularity and with short names. In the third cluster there are apps for teens that want to play games, with a high frequency of updates, ads and in-app purchases.

Rating Predictions

From the initial results obtained in Decision Tree and Discriminant Analysis, it was hard to build models able to make accurate predictions for our target variable `Rating`, especially when the ground truth value is around 4 as the vast majority of the target values range from 3.8 to 4.5.

To solve this, we decide to convert a prediction problem to classification one by creating two new groups of Rating: low rating(1-3) and high rating(3-5). By doing so, we improved the accuracy of the classification.

The most important variables used to build the decision tree model are the size of the app, whether the app has ads, the minimum version of android the app supports and, the antiquity of the app. In general, apps with bigger size, with ads, older and require a lower version of minimum android tend to have higher rating.

Final Conclusions

We have discovered valuable knowledge about apps as we explained in the previous discussions, the most important ones relates to the target variable are the following:

- Due to the high variability of the apps it is difficult to make an accurate prediction of `Rating`. The model is accurate only for the extreme values.
- `Rating` is not dependent on the number of `Installs`, so we could have a popular app with low rating and vice-versa.
- When classifying an app between the classes high and low, `Size`, `Ad.supported`, `Minimum.Android`, the antiquity of the app are the more relevant variables.

We could have obtained more refined knowledge if we have focused on more specific apps like games, productivity or social media. Moreover, we could limit the time range of apps in order to reduce the variability. Also, we could have studied the number of `Installs` instead of the `Rating` that could be more interesting for the business decisions.

16. Initial and final working plan

16.1 Diagram of Gantt

You can find the original Gantt diagram attached at the following link: https://drive.google.com/file/d/1StXyfpV_H5_BvNLa_W5yKcX2WzL_rEJz/view?usp=sharing

The final Gantt diagram can be found at the following link: https://drive.google.com/file/d/18e1RIoxBqleSOcI3ddo_scBHYnb78huF/view?usp=share_link

16.2 Final division of tasks (assignment grid)

After completing the project the final division of tasks is the following:

Tasks/Participants	Victor	Ange	Eloi	Fatima	Zhongkai	Ximena
Preprocessing	x	x	x	x	x	x
Basic initial univariate descriptive statistics of preprocessed variables	x					x
Dataset description according to the main conclusions of the univariate and bivariate statistics		x		x		
PCA analysis for numerical variables	x	x		x		x
ACM analysis of multiple qualitative variables			x		x	
Multiple Factorial Analysis		x		x		
Association rules mining analysis	x		x			
First part D3 report writing	x	x	x	x	x	x
Prepare for the D3 presentation	x	x	x	x	x	x

Task created to resolve unforeseen incidents		x			x	
Hierarchical Clustering on original data			x		x	x
Profiling of clusters			x		x	x
Decisions tree	x	x		x		x
Discriminant analysis	x	x	x	x	x	x
Discussion and conclusions	x	x	x	x	x	x
Second part D4 report writing	x	x	x	x	x	x
Task created to resolve unforeseen incidents	x	x	x	x	x	x
Discussion, conclusions, comparison of results among several methods	x	x	x	x	x	x
Update the initial Working plan	x	x	x	x	x	x
Write the final report	x	x	x	x	x	x
Prepare for the final presentation	x	x	x	x	x	x

16.3 Deviances and risk during the project

In the following table appears the risk contingency plan we had for the project:

Risk	**How to prevent**	**How to manage**
Member does not work	Regular reports	Communicating via Discord
Files lost	Online backups	Shared folder in Google Drive
Poor communication	anything should be clearly clarified	Ask frequently if someone missed
Team member quitting	Have more than one team member in a task. Assign a substitute for each task	Reassign work
Some member of the team gets stuck with some task	Constant communication and asking each other if someone	Communicating via Discord. Helping each other

	has any problem with the task assigned to him/her	
Conflict between members	Respect and acknowledge other's ideas	Resolve the conflict in a meeting
Errors in some of the tasks	Assign more than one team member in a task.	Everyone in the team should check the tasks done by other members in order to find ways of improving it and/or find mistakes.
Lack of communication within the team	Having regular meetings	Using discord with for regular meetups
Team members with different backgrounds and different culture	Constant communication and trying to understand each other	Regular meetups, discord
Another pandemic	Using online tools	Using discord and Google Drive to keep communication
Low data quality	Correct data treatment	Use the correct tools to improve the data quality
Some member of the team does not feel comfortable with the tasks he/she has been assigned	Make a plan and assign tasks taking into account everyone's opinion.	Communication
Initial plan errors	Make an initial plan taking into account that it can be changed or updated.	Constantly monitor that the plan is being followed and adjust it accordingly.
Lack of experience by some team members	Making pairs in which one of the team members has more experience	During the task distribution, make sure the pairs are distributed correctly
Missing tasks in the initial task distribution	Check of the initial list of tasks by all team members	Use the delivery file to check that all the task are included

Thanks to the contingency plan we had in place we manage to perform the tasks of the project correctly with no conflicts between team members by scheduling regular meetups and keeping everyone updated in how the project was advancing, by working with the same files thanks to the online backups and by making sure of the correctness of each part by having more than one team member revise and accomplish that task.

In terms of unplanned risks the only one we can mention, but an important one, is the compagination of this project with the other projects of the master, making it so that we had to change the distribution of the activities we had assigned at the beginning of the project, having to reduce the duration of most activities and change the people assigned to some of them, specially during the second half of the semester.

Another thing that made us change the planned schedule was the underestimation of the duration of some task. For instance, the preprocessing step, where because of the importance of this step we decided to assign more resources and increase its duration, but thanks to the fluid communication between the team we manage to keep the increase of the duration of the tasks to a minimum.