Mobile App Monitor

Sapienza University of Rome

Visual Analytics

Silvio Dei Giudici, Marco Morella

July 2, 2020

Figure 1: *Visualization with dark mode*

Abstract

The ever-growing amount of apps have a correlated increase of utility and variety to our mobile phones' uses, but it also creates a multitude of challenges for both users and developer alike. Our visual analytics tool has the main focus of helping both categories and aid them in doing easier and more thoughtful decisions. We started from a web-scraped dataset of the Google play store and managed to realize a comprehensive tool where in all the useful features are displayed to give an accurate overview, highlight patterns, find outliers and more.

I. Introduction

Applications are a huge chunk of the time spent by the average user on his mobile phone. Due to a number of factor which won't be the focus of the present paper, we have seen a huge increase in the spread of technology and especially in the spread of mobile phones possession. This has created an influx of developers to this market which started making an enormous amount of apps, indeed the latest Google's statistic marks that there are more than 3 millions of applications on the play store.

The sheer amount of variety between the type of applications, their content and their targets makes it extremely hard to paint a clear picture without some aid, in particular multiple visualization tools are needed to digest this kind of input.
We designed the tool in order to help mainly two categories of users:

- **Developers**, in understanding how the market fares with respect to the number of installs and average review for applications with different parameters, mainly the category, the target and the

price.

- **Users**, to recognize how trustworthy are some parameters when checking an app out, what to expect when downloading a certain category and so on.

We believe that the overall project has more use and more users possible than the ones listed and anyone having the need for a clear visualization of this dataset can find what he is looking for in our tool.
One note to make before going on, the dataset contains ten thousand applications, which is only a fragment of all apps available to download, due to technical limitation and the project specifics the whole dataset couldn't be analyzed for this project.

II. Dataset

The dataset was made by scraping the web version of Google Play store meaning that only data available to the general user has been inserted in the dataset.
In the CSV we can find data for 10 000 apps, each row containing 13 features for a given app. In the following we're going to describe each of the ones we ended up using:

- **Application name**.
- **Category**: The category that the app belongs to. The most frequent are family, games, tool and medical.
- **Rating**: Average of ratings received by the app, on a scale from 1.0 to 5.0.
- **Reviews**: Total number of reviews, rounded down to the thousands by the Play Store.
- **Size**: size in MB of the apps, given that it is a small dataset albeit recent we found strange that there are no apps in the GB range.
- **Installs**: number of installs, also rounded down to the thousands.
- **Type**: true or false depending on whether the app has to be paid before installation, it does not consider in app purchase.
- **Price**: how much the app costs, also does not include in-app purchase.
- **Content rating**: age group the app is targeted for, it is divided into Teen, Adults only, Mature, Everyone 10+, Everyone, as expected the most inclusive rating is the most frequent.
- **Last updated**: date of last update on the play store for the app. Here we found a correlation between outdated apps and their low appeal to users.
- **Android version**: minimum version of Android required to run the application, some of them have different requirement depending on the device of installation.

There were two more features that we scrapped because we didn't find them useful to the tool:

- **Current Version**: since every developer gives different names to the various release this information was not useful to either give patterns or outliers because two apps being in version 2.0 could be at very different stages of growth.
- **Genres**: Other categories aside the main one which an app can belong to. Applications and medias have the tendency of giving themselves as many tags as possible to appeal to a wider range of customers and fooling the search engines in showing them to more people, thus making this feature impossible to use properly in our case.

A. Dataset preprocessing

Since the dataset comes with a lot of arguments, we decided to do some preprocessing in order to improve the analysis and simplify the final application from the exception cases.

First of all, we filtered the "dirty" entries of the dataset. The main goal of this phase is to transform every features' entry in a string that can be easily converted to float afterward. Indeed the original dataset is full of entry with special characters (like points, commas, dollars, and so on), probably added for readability purposes, and also dimensions (like the size of the application in Kb or Mb) that need to be converted. Another activity done in this phase is the elimination of NaN values. This happens in particular with the **Rating** and **Size** features. In place of the missing values, we decided to put the mean of all the other entries, in order to not alter the analysis.

Then the other task performed in the preprocessing is dimensionality reduction. In general, PCA is used for this kind of analysis but in our case we have both continuous and categorical data so we can't adopt it. Instead we opt for MDS, which gives the possibility to put a custom distance function, specially tailored for our problem, providing to the algorithm the matrix NxN with all the distance couples (called dissimilarity matrix). So what we did in the first place is to collect all the continuous features (i.e. **Rating**, **Reviews**, **Size**, **Price**) and build the a dissimilarity matrix using the euclidean distance between each couple of entries using only the continuous features. Afterwards we did the same using the categorical dimensions (i.e. **Category**, **Installs**, **Type**, **ContentRating**, **AndroidVer**) but this time we change the function using the Jaccard distance. The latter is very useful to define a distance between sets (so also categories).

Finally, we merge the two matrices simply by adding them. Of course we also perform a normalization of the two matrices, in order to avoid that some

features (in particular the continuous one, that have high absolute value with respect the Jaccard distance) prevail to others. So we perform MDS with 100 iterations, using the merged matrix as dissimilarity matrix, and the result is written on the dataset as two additional features for every entry. This is done in order to run MDS only one time (since it's heavy from a computational point of view) and quickly build the 2d scatterplot of the app.

III. Visualization

We used multiple different views to create the most complete analysis without needing much prior information and without overloading the user. We will now analyze each view individually.

A. App List

The app list is not an advanced tool but we deemed it useful for the categories of users we had in mind when making the blueprint and initial decisions for the project.
It is important since it lists 20 applications respecting the brushing made in the parallel coordinates plot, if the selection includes more than 20 applications only the ones with the twenty most installs are shown in the list. The list and the rankings are computed at run time and will thus be recomputed each time the user changes the brushings.
One function that we think would be a good addition in future version is making the list clickable, where clicking on one application will direct the user to the page of said application. This will require one more scraping of the Google Play Store and a join of the new table containing the URLs with the one we already have.

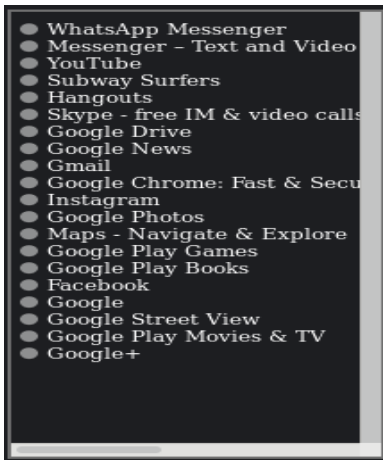

Figure 2: *App List*

B. Parallel Coordinates Plot

This graph has a focusing position in the page since this is the main element of the tool, indeed it easily reveals patterns, their absence and some outliers. Furthermore the nature of this graph encourages the user to play with it and will bring him to make interesting discoveries.
In order we have these axes: Category, Rating, Reviews, Size, Installs, Price, Content Rating, Android Version. Those were the most interesting and useful to brush and display their relations as paths in the graph.
This visualization, as we said, is the focus of the whole and as such brushing one axis will constraint the dominion for all other visualizations as a filter, thus interacting with all of them and updating as the user brushes, this is done through a shared class which holds the brushed data and gives it to other visualizations when a brush event happens.
We thought about implementing reviews and installs as logarithmic scales since most of the data is found at the minimum of the dominions but after implementing it we found it incredibly straining for efficiency and didn't bring as much visual improvement as expected.

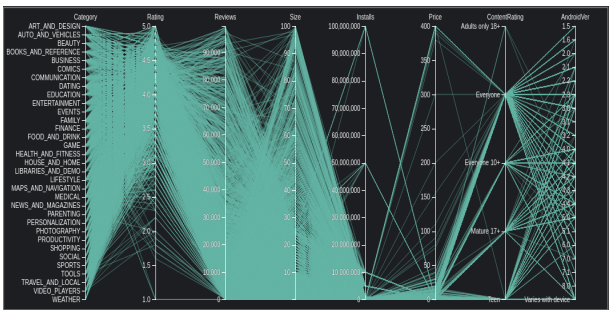

Figure 3: *Parallel Coordinates Plot*

C. Scatterplot

We applied MDS on the dataset and the result shown in the scatterplot is the distribution of elements(applications) over the two principal components found by the algorithm.
The difference in color given by the rating can help the user distinguish clear similarity in the clusters highlighted and the presence of outliers.
The user can brush over the scatterplot which will highlight in red the path of those elements over the parallel coordinates plot further making it possible to find clusters and relations.

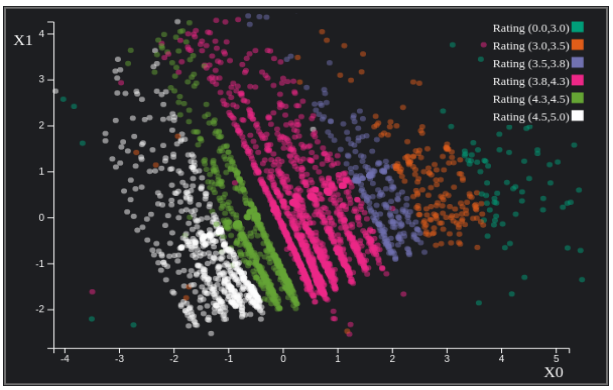

Figure 4: *Scatterplot*

D. Bar Charts

The following views are different bar charts, all of them require runtime count of the elements still in the dataset each time we brush to get the bins.

D.1 Content Rating Bar Chart

This bar chart shows the user how many apps there are for each content rating.
As the other bar charts, this graph is updated while the user brushes, to avoid confusion the order of content ratings in the bottom axis stays the same even when the rankings in amount of apps are changed by the brushing.
There was no need to filter the number of content ratings in this view since there are only 4 allowed in the play store and the view is very digestible.

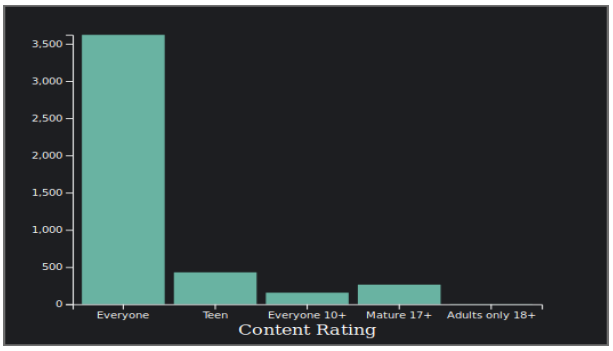

Figure 5: *Content Rating Bar chart*

D.2 Android Version Bar Chart

The second vertical bar chart is a stacked kind, indeed in this visualization we wanted to show the different android versions required to run the apps while keeping the subversions on the same bar. This put a good constraint on the space since a standard bar chart would be quite large due to the number of subversions.
Each version on the axes has bars on top of it where each part is linked to a different subversion, in order.
This view is extremely useful both for developers and users since the first ones need to know which versions their apps should support to reach a wider range of users and the second ones can pick a phone able to run at least that version to be able to use most apps they are interested to.
As expected the version 4.x is the most used, indeed most mobile phones still being used are not new enough to run 5.x or more and thus most applications will restrict themselves to the most used android version.

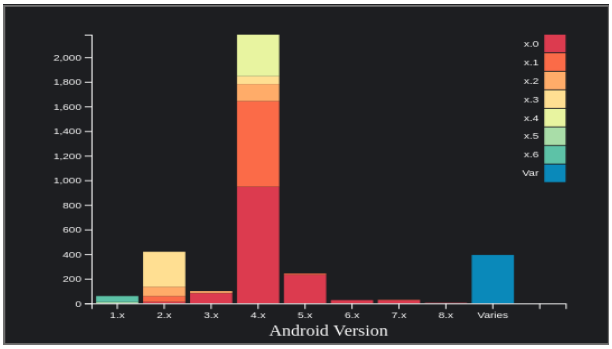

Figure 6: *Android Version Bar Chart*

D.3 Category Horizontal Bar Chart

While most of the users will find useful to brush over category in the parallel plot, we decided to implement an horizontal sorted bar chart to show how many apps in each category are in the dataset. This is useful to find some patterns when brushing over the other axes, for example by brushing over size we find out that most of the apps with heavy size are related to gaming and family and this is understandable since gaming normally have more content and media than other kind of apps and family apps are the most frequent in the dataset.

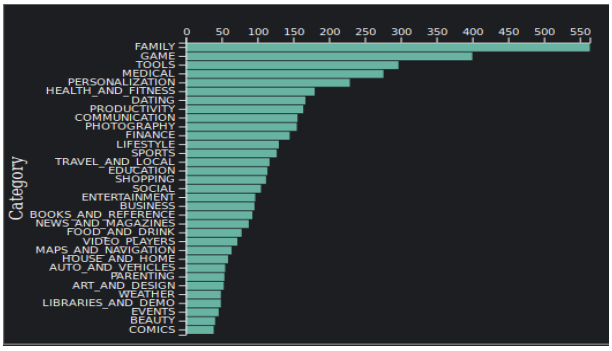

Figure 7: *Category Bar Chart*

E. Box plots

We implemented three boxplot, respectively for Average Rating, Size and number of Reviews.
The boxplots help the user understand the distribution of those features for the app filtered thanks to the division in quarters of the distribution over the axis.
The initial Reviews boxplot is so close to the beginning of the axis due to heavy outliers that it's almost impossible to see but we believe that using a logarithmic scale for this one would make it way more confusing. By using the tool and brushing that boxplot becomes more visible the more the user brushes.
At runtime we need to compute the minimum, maximum and quantiles each time there's a new brushing to update the boxplots.

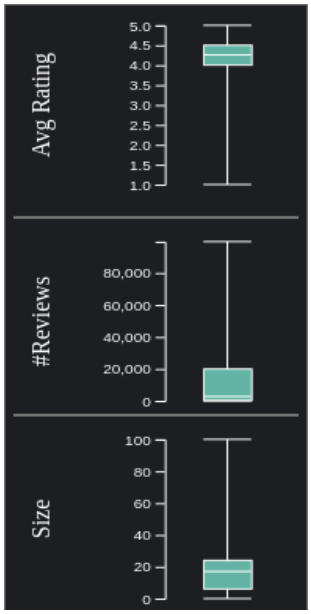

Figure 8: *Rating,Reviews and Size boxplots, since it's without brushings, Reviews is flattened to the bottom*

F. Header

The upper part of the page shows the possibility of picking only apps that are free, only paid apps or both(default).
The user can select a range of time where only apps which were updated in that time frame will be considered in the visualizations.
Furthermore the user can swap the colors of the page in order for it to be accessible during the night. The related button in the header will swap the color palettes of texts and visualizations so that the user won't have to strain hiss eyes to look at the tool. When choosing the images to show in the paper we picked this view because the details are clearer and easier to notice.

Figure 9: *Paid/free checkboxes of header*

Figure 10: *Time range and dark mode selection part of the header*

IV. Interactions and analytics

The project includes multiple possibilities of interactions initiated by the user.
This section will explain each interaction the user can have with our visualization tool, in the specific we will talk about what it's possible and its side effects on the views.
The interactions are:

- **Parallel Coordinates Plot**: this standard tool allows the user to do a selection on each axis by brushing, to filter out all entries not respecting the new constraints.
- **Scatterplot**: the user can brush over the items and those will be highlighted on the parallel coordinates chart.
- **Header**: will filter over the time range picked by the user, whether to consider paid, free or both kind apps and will let user decide whether to see the standard color palette or dark mode.

Now we will analyze the interactions that modify the data shown by other visualizations:

- **Header**: In the header we find two kind of this interaction. Indeed when the user selects a time range, a filtering over the dataset is activated and all other visualizations will change based on the new data. Similarly the paid and free checkboxes will filter out all the data not correct with respect to the selection.
- **Parallel Plot**: All other views will reflect the filtering done on the brushed data and change. In the parallel plot itself only the paths corresponding to elements still in the dataset will retain the original color.
- **Scatterplot**: As we said the user can brush on the scatterplot. Since the elements on the scatterplot are the result of data reduction it wouldn't make sense to let the user filter other visualization out and indeed we only highlight the selected elements' path over the parallel plot.

Since parallel and scatter will interact with each other

we have a two way coordination between the two of them. Regarding the analytics, the following are the run time computations needed by the tool:

- **Most installed apps list**: After each brushing we need to update the list of apps conforming to the filtering. We analyze the brushed data and compute a sorted list of all apps, sorted by their number of installs and display this list in the bottom left part.
- **Axis of boxplots and bar charts**: Since we decided to have the axis change based on the current data we need to recompute them each time there's a brushing and indeed the user will see this transition happening.
- **Box plots features**: At run-time each time the data changes we need to re-compute the quantiles, minimum and maximum to update the features displayed by the boxplots.
- **Bar Chart features**: As expected the recomputations of the bar bins will be triggered each time we need to update the visualizations, furthermore, due to the nature of stacked bar charts we need to recompute in the Version bar chart the height of the elements based on the sum of the precedents each time the data is changed.

V. Considerations

The view we decided to implement were the ones we felt would give the most information without the need of knowing too much about the dataset.
The parallel plot was an obvious choice as the main focus since it is the most useful tool when the interaction over multiple features is a need. In particular we felt this was an optimal addition since it will encourage the user to fiddle with it to discover patterns in the data.
The checkboxes will divide the data in a clear way and the users will most likely want to analyze the difference in paid or free data.
With respect to clusters and outliers the scatter plot in conjuction with the parallel plot let the users see them at an eyes' glance.
The bar charts and box plots will show the users the distributions of features over the data space.
Overall the movement gives a visual feedback of the data changing with the brushing and selections done by the user and indeed the transitions over the bar charts are the most noticeable in conjuction to the parallel plot.

VI. Conclusion

We are satisfied of the end results since it excels in giving all the information and filtering possibilities we wanted to accomplish and we found it to be a good tool for the analysis of the dataset.
Indeed we worked over the different selections and it was quite easy to find the answer to any questions the data or combinations of it could answer.
All the visualizations implemented bring something different to the tool and we believe that the general overview they gave is comprehensive of the dataset.
While working on the project we had different ideas on the future development on the tool, in particular it would be interesting:

- Add history: meaning that we could see the statistics for each app in given time periods but this would need months of data collecting over the public website or a private dataset given by Google.
- Add review content: where we preprocess the content of the reviews of each app and classify them. This is a big data computing problem and combining it with visualizations may be a complicated problem by an efficiency point of view.
- Add more data: the current dataset and the specifics limited us extensively since the play store apps are way more than those analyzed by us, ideally we would need to retrieve a dataset for all applications.

References

[1] Play Store dataset - `https://www.kaggle.com/lava18/google-play-store-apps`

[2] Material of Visual Analytics course, 2019-2020
M. Angelini, G. Santucci

[3] ColorBrewer2 - `http://colorbrewer2.org/`