Is Update-frequency and content connected top App success?

An Analysis of productivity Apps in Google's Play Store

Applied Project
Minor Digital Science
Digital Science Center Innsbruck

Tim Jonathan Rupp
Summer term 2021
21.07.2021

Introduction

Almost everyone owns a cell phone today. According to Statista (2021), there are over 6 billion active cell phones worldwide in 2021.
Based on a study by Google (2016), the average user had 35 apps installed on their smartphone in 2016. It can be assumed that this number has increased significantly to date.
The Google Play Store offers just under 2.9 million apps in July 2021, as per Statista (2021).
Users of Android-Apps have the option to rate those apps on a star scale of 1-5 and can optionally write a review to explain what they based their rating on. On the one hand, the reviews serve as a reference point for other users to assess how good an app is or what doesn't work so well. On the other hand, app developers can learn from the reviews where their app's weak points are and what users like so that they can deduce how the app should be developed further.
App developers have the opportunity to enhance their apps through updates and to implement them in new versions. In this project, I will investigate whether updating apps more frequently leads to better ratings, more downloads and more positive sentiment in reviews. In addition, Topic Modelling will be used to investigate whether app developers who respond to negative topics in the reviews and take them into account in app updates can achieve higher ratings in the app store than developers who respond less to user feedback.

Methods

The following section describes data, the collection and preparation of the data, and data analysis.

Data Description

Reviews of apps in Google Play can be downloaded via an API and the Python package Google Play Store Scraper. The reviews are stored in JSON format. In addition to the actual reviews, the user name of the creator, the date of creation and the star rating are included. In addition, the Google Play Store Scraper can also retrieve information about the respective apps in JSON format. This includes the release date, whether the app supports advertising, wether the app is free, whether the app is marked as "Editor's Choice", number of installations, information about in-app purchases, number of reviews and ratings, as well as the average rating in the store.

In addition, the patch notes of the apps were copied out from appannie.com, since there is no direct way to get historical data about patches and updates via the Google Play Store.

The study examined 40 apps in the American app store, randomly selected from the appbrain.com website's top lists in the productivity section of both free and paid apps.
Reviews and updates from the last five years were examined, starting from 5/17/2016 to 5/17/2021. A total of 3,185,528 reviews were retrieved and saved in that process.

Data Collection/-preparation/-processing

JSON files of the reviews and information of the 40 apps were downloaded using the Google Play Scraper, then tabulated with the R package 'jsonlite' and stored in a SQLite database.
Subsequently, the patch notes for each app were copied out manually from the website appannie.com, since the site is commercially operated, prohibits scraping and in this way logging into the website with e.g. Selenium could be avoided. The patch-notes for each app were stored separately in a .txt file and the information was extracted with regex and the R package 'stringr' and stored in the database.
After that, the sentiment of the reviews was determined using the R package 'sentimentr'. This package uses a dictionary-based approach and assigns a value between -1 and 1 to each sentence, which is averaged for multiple sentences in a text.
The age of the apps in days and the update frequency in days (the average duration between updates in days per app) were then determined, using base R's 'difftime' function.
Finally, the mean rating of reviews per app and the mean sentiment of reviews per app were calculated and also stored in the SQLite database and the collected data was merged.

For topic modelling the reviews were processed in Python 3.9. Emojis were removed using regular expressions. The reviews were then tokenised with the function ‘simple_pre-process' of the Python package 'gensim', lowercased and punctuation was removed.
Subsequently, the tokenised words were lemmatised and POS-tagged (given the part of speech) with the package 'spacy' and, in order to increase the information content, only nouns, adjectives, verbs, and adverbs were retained in the further course.
After that, English stop-words were removed with the NLTK package and Bigrams (words that occur together) that appeared with a frequency greater than five were included in further analysis.
The corpus was formed and a first LDA model was created and visualised with 'pyLDAvis', perplexity and coherence values (u_mass) were calculated. Since in 'gensim' and mostly in LDAs in general, the number of topics must be determined before the model is computed, several models and their coherence were then computed and

visualised to determine the optimal number of topics. A u_mass value close to zero and is an acceptable value (Röder et al., 2015). The optimal model was then selected and the ten top-words of the topics and the loads of the reviews on the individual topics were stored in the SQLite database.

Data Analysis

To analyse the correlations of update frequency with mean rating, number of downloads, and mean sentiment per app, the necessary data were retrieved from the database. Uninteresting variables were removed and factors were formed for non-numerical variables, if this made sense. Subsequently, the summary statistics were viewed and the distribution of the dependent variables was examined. Three apps (Dragonanywhere, Google Analytics & Podio) were excluded from the analysis as they had very high update frequency values (many days between updates) and standard deviation.

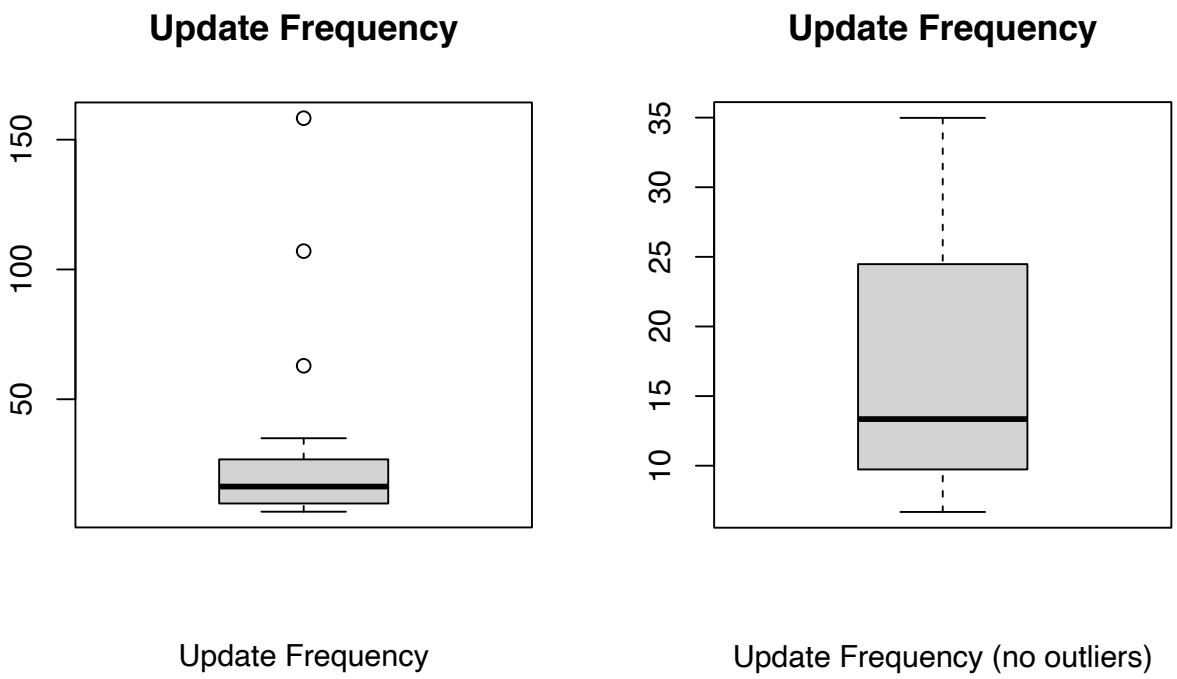

Figure 1. Boxplots for displaying distribution of update frequency.

An intercorrelation graph using the Spearman correlation was created for the numerical variables of the data set, since ordinally scaled data are also included in the data set. No significant correlations seem to exist between update frequency and rating and update frequency and sentiment, but there are between update frequency and the number of downloads in the Google Play Store.
Subsequently, this relationship was tested with Siegel nonparametric regression. This is distribution-free and robust for non-normally distributed data. For each point, the median

of all lines through this point is calculated. Then the median of these averages is calculated to estimate the regression line.

To evaluate the topics of the reviews, the calculated topics were merged with the reviews and intercorrelations with sentiment and score were calculated to find out which topics are more related to positive reviews and which are more related to negative reviews.
Then, for each topic of reviews and patch notes, the 10 most loading texts on that topic were extracted to get a sense of the subjects of each topic.
Finally, I averaged the values of the Topics per app, both for patch notes and reviews, to see if they were related to review rating, store rating, or app sentiment. This was again done using intercorrelation graphs.

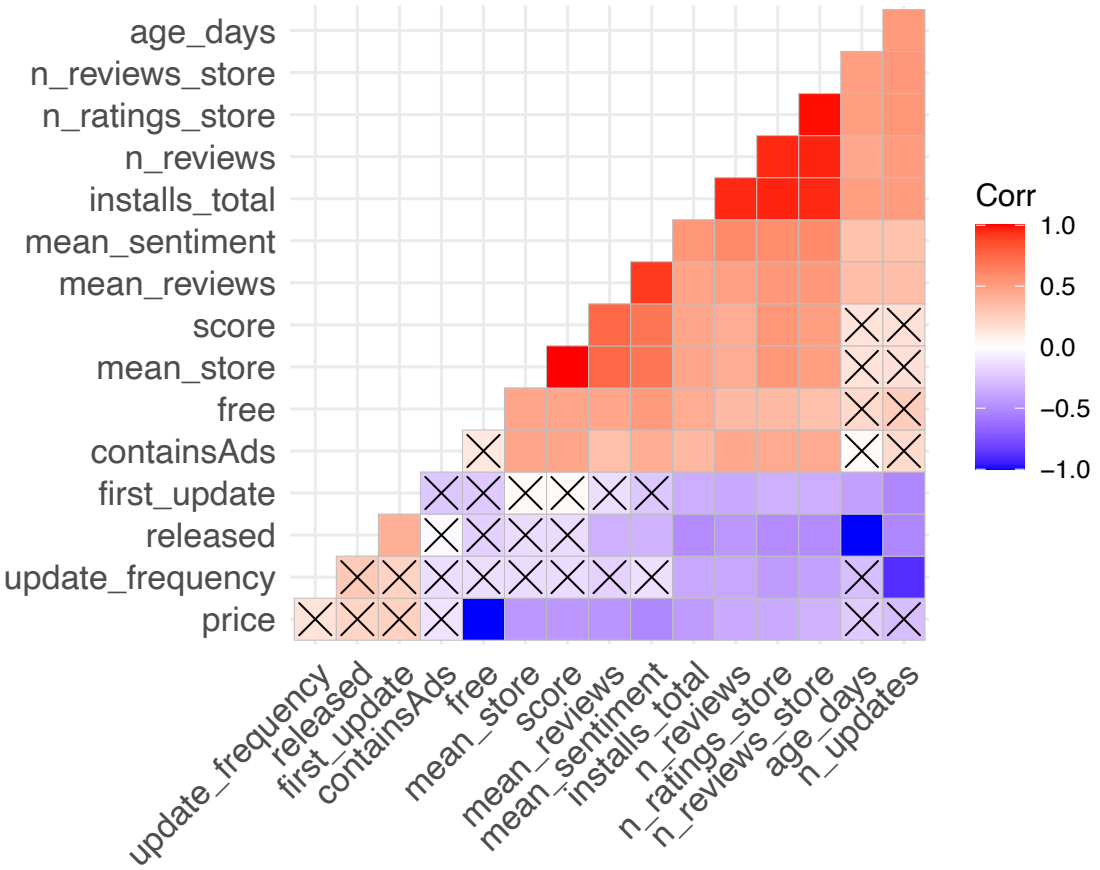

Figure 2. Intercorrelation of app metrics (crossed out quadrants show now significant p-value).

Results

In the following section, the results of the analysis are presented and visually illustrated.

Statistical Analysis

Both the average of the review ratings and the average of the ratings of the entire store are not significantly related to update frequency (r_s = -.14, p = .25 for reviews; r_s = -.08, p = .51 for store). In contrast, such a significant correlation was found for update frequency and the factor of installations (r_s = -.38, p = .019). This can be interpreted as meaning that the more frequently an app is updated (lower number of days between updates), the more frequently it is installed. However, no causal relationship can be identified that way.
To examine the relationship in more detail, a Siegel nonparametric regression (Mangiafico, 2016) was then performed, which was able to confirm the significant relationship even with the absolute numbers of installations (*Intercept = 7529153, slope* = -116229, p <.001).

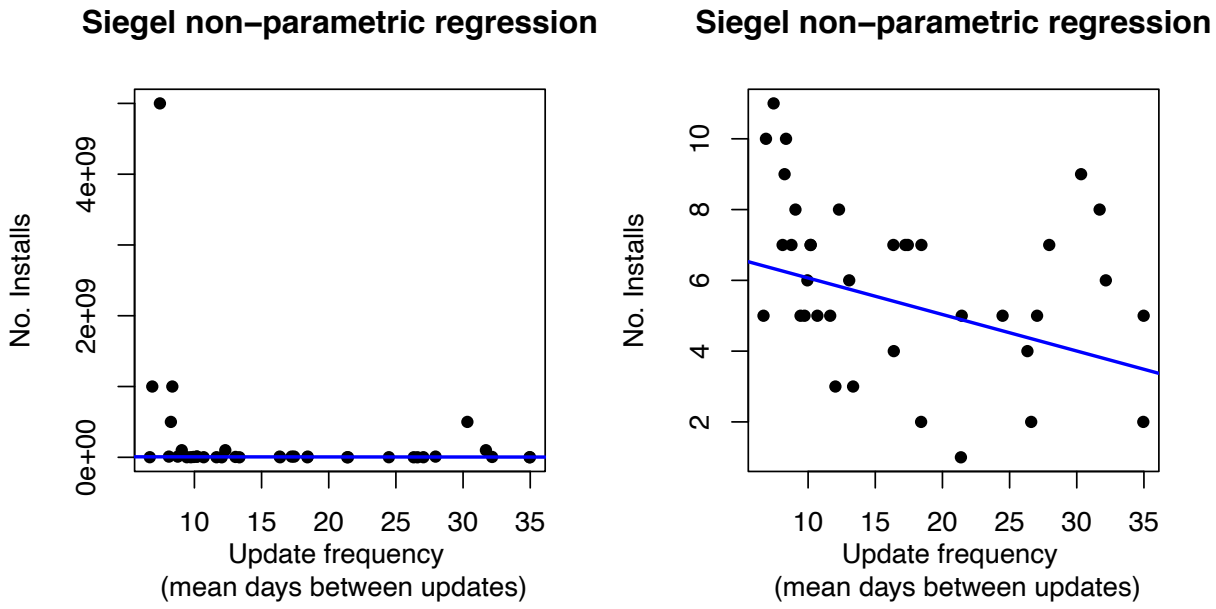

Figure 3. Siegel non-parametric Regression. Left shows absolute installs, right shows ranks.

As an exploratory part of the analysis, the difference between the average score of the reviews and the total ratings in the store was examined. Again, a nonparametric test was used, in this case the one sided Wilcoxon-Test for linked samples. This test could show that the ratings of the reviews are significantly lower than the overall app ratings in the store (z = -2.17, p < *.001,* n = 40).

To examine the correlations of the topics of the LDA with app success, correlations of the review topics with rating and sentiment of the reviews were first examined. This was done with an intercorrelation matrix using Pearson's r. Topic 4 was positively correlated with sentiment and score (r(3185526) = .42, p < .001; r(3185526) = .19, p < .001) while Topic 1

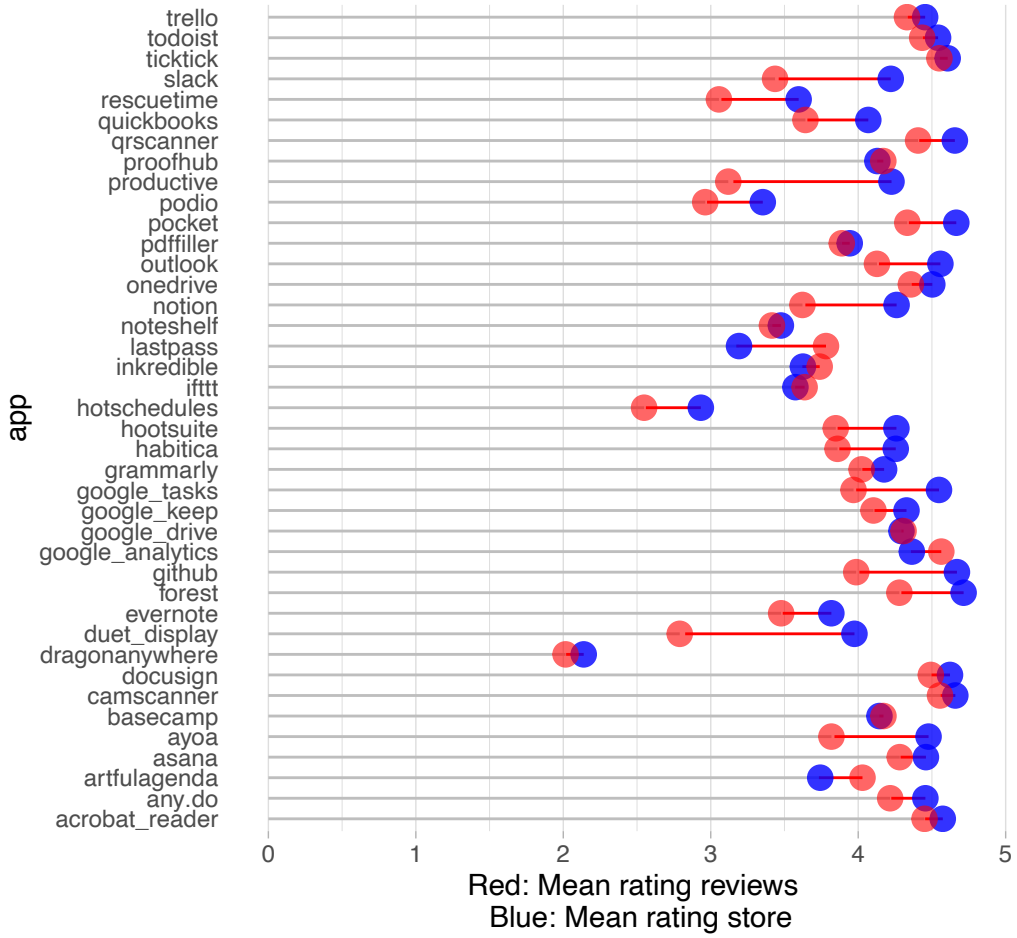

Figure 4. Differences in ratings of reviews and overall rating in store.

was negatively correlated with sentiment and score ($r(3185526) = -.37$, $p < .001$; $r(3185526) = -.31$, $p < .001$). Topic 1, when taking a closer look at the top words and representative reviews, seems to deal mainly with missing features, too high costs, and unexpected billings. On the other hand, if one looks at Topic 4, the 500 most representative reviews consist mostly of praise ('very good', 'nice app good working', or just 'good good good' etc.), but this seems like spam in parts, or ist not very informative at least. In this respect, the negative reviews seem to be much more informative for app developers and other users.

When examining correlations of topic affiliation and app success at the app level of the patch notes, no significant correlations of topics with ratings and sentiment were found. Among other things, this can be due to the fact that app developers often publish quite generic texts about the updates (e.g. a simple bug fix) or even repeat them almost identically with every update. In this case, the patch notes were therefore unfortunately not very informative.

Data Visualisation

For data visualisation and analysis, base R was used in part, but primarily the R package ‘ggplot2’. For visualisation of the coherence values of the LDAs the Python package

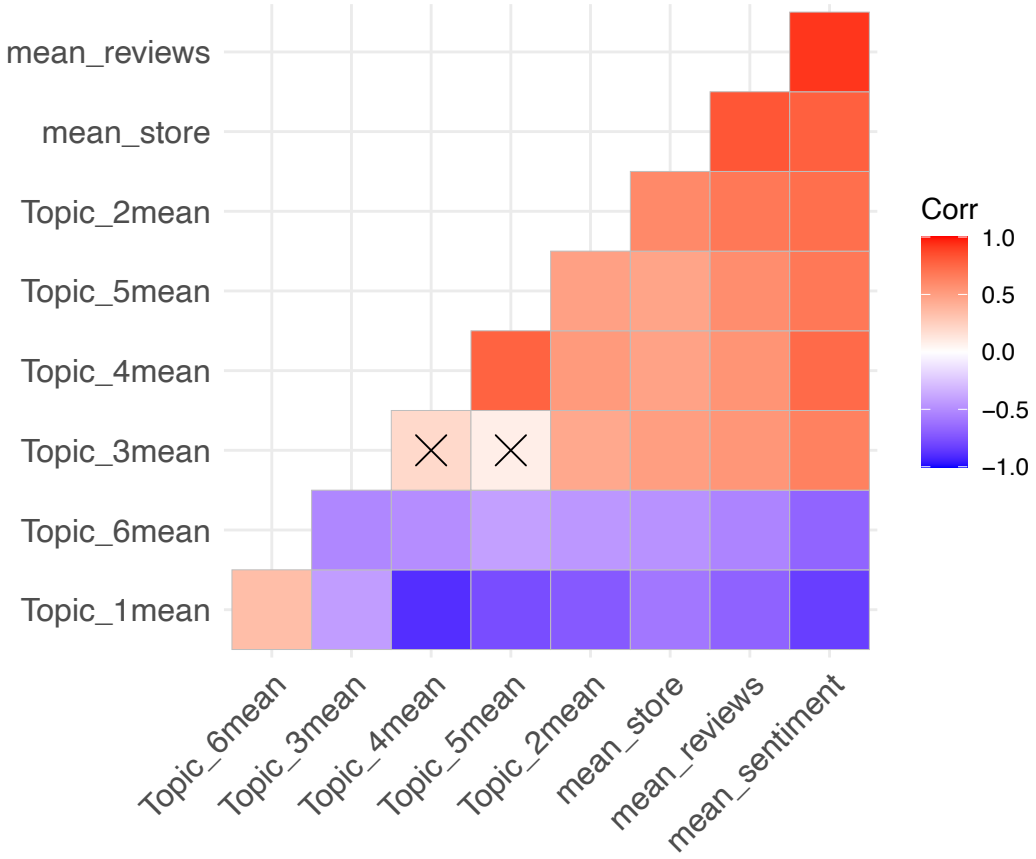

Figure 5. Intercorrelation of review topics, sentiment and ratings. Crossed out quadrants show no significant p-value.

'matplotlib' was used, for visualisation of the topic distributions of the LDAs the Python package 'pyLDAvis'.

Discussion

No significant correlation could be found between ratings and update frequency, but one could be found between update frequency and the number of installations of an app. Of course, this could mean that successful apps update more often because they have more resources at their disposal and can publish regular routine updates without any problems. However, frequent updates could also lead to a better usability of the app and thus appeal to a larger number of users.
It could also be that download numbers is a more objective way to measure app success. On the one hand, the average rating for an app "hides" its distribution and the variance in more controversial apps is not included.
On the other hand, it is also questionable whether app reviews should be trusted in principle or not. On the one hand, there are relatively easy ways to buy good reviews, on the other hand, apps could also be "bombarded" with negative reviews by competitors (e.g. Hill, 2018).

In any case, when performing topic modelling on the reviews, it could be shown that certain topics had very positive correlations with sentiment and also rating, but were not very informative in terms of content. Especially with topics that are negatively correlated with sentiment and rating, it is easy to find out where an app might be lacking and what users would like to see. This is useful for other users, of course, but also for app developers who want to learn how they can improve their app.

Getting the patch notes is not necessarily easy and they are more likely to be reliably stored at paid providers like appannie.com. At the beginning I was unsure how to get the patch notes in a good way, because appannie.com does not allow scraping. I was then advised that it is okay to copy out the data as part of the research, which I then did.

To obtain more informative results overall through topic modelling, future research could use a text corpus of reviews and patch notes to better find correlations between reviews and patch notes. However, since patch notes are not written in a very "research-friendly" way, it is not entirely clear whether this approach would lead to better results.

Appendix

Script name	Purpose	Module
0_requirements.R	Installs and loads needed packages in R	M1
1_PlayStore_reviews.py	Scraping of reviews and saving as JSON	M1/M2
2_PlayStore_apps.py	Scraping of app details and saving as JSON	M1/M2
3_save_reviews.R	Reading reviews from JSON-format and save in SQLite database	M2
4_save_app_details.R	Reading app details from JSON-format and save in SQLite database	M2
5_save_patchnotes.R	Reading unstructured patch notes from .txt-files with regex and save in SQLite database	M2/M3
6_save_names.R	Read JSON files and extract names from their title with regex	M2/M3
7_review_sentiment.R	Get sentiment for each review and store in SQLite database	M2
8_app_age.R	Compute app age in days and store in database	M1/M2
9_update_frequency.R	Compute update frequency (mean days between updates) and store in database	M1/M2
10_mean_ratings.R	Compute mean ratings of reviews for each app and store in database	M1/M2
11_mean_sentiment.R	Compute mean sentiment for each app and store in database	M1/M2

Script name	Purpose	Module
12_combine_tables.R	Merge tables for better overview	M1
13_Analysis.R	Analysis of the relationships of update frequency	M1/M3
14_LDA_reviews.py	Topic modelling (LDA) of reviews	M3
15_LDA_patchnotes.py	Topic modelling (LDA) of patch notes	M3
16_Analysis_LDA	Analysis and Visualisation of LDA results	M3

References

Hill, S. (2018, 3. März). *Can you really trust app store ratings? We asked the experts*. Digital Trends. https://www.digitaltrends.com/android/can-you-really-trust-app-store-ratings/

Mangiafico, S. S. (2016). *R Handbook: Nonparametric Regression*. Rcompagnion.org. https://rcompanion.org/handbook/F_12.html

Röder, M., Both, A., & Hinneburg, A. (2015, February). Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining* (pp. 399-408).

Statista. (2021a, 7. Juli). *Smartphone subscriptions worldwide 2016–2026*. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

Statista. (2021b, Juli 13). *Google Play: number of available apps 2009–2021*. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

ThinkwithGoogle. (2020, 3. September). *Smartphone Users 22*. Think with Google. https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/average-number-of-apps-on-smartphones/