Version Update Effects on Rankings in Mobile App Markets

Scott Woods, Computer Science Department, Boston University
Huy Le, Computer Science Department, Boston University

ABSTRACT
Worldwide mobile app revenues are predicted to exceed $45 billion in 2015. In the Apple iOS app store alone, there are over 1.5 million apps [1], [2]. In such a saturated market, what makes an app successful? From the perspective of an app developer, understanding the behavior of app consumers is crucial to building an app that can reach and stay on the top charts. While prior studies have examined the life cycle and survivability of an app within the top charts [3], [4], we focus specifically on the role played by versioning. Most app developers periodically release updates to their apps, a crucial operation in maintaining the security, stability, compatibility, and interest in the application. Updates generally introduce additional features and patch issues with the previous version. But, the impact of introducing an updated version is not always positive. For instance, a poorly executed update could introduce bugs, security flaws, and potentially unwanted features, and damage the reputation of a thriving app. By isolating apps that have released an update rated significantly higher or lower than their previous version, we are able to quantify the impact on an app's ranking in the app store. We discuss our findings in the context of a publicly available Apple App Store dataset [5], which allows us to approach the analysis using only data that is available to the informed consumer. Our results show that although a great new update has little effect on the short term ranking of an app, a poor update can have immediate negative impacts in as little as a day.

Categories and Subject Descriptors: J.4 [**Social and Behavioral Sciences**]: Economics
General Terms: Economics, Measurement
Additional Key Words and Phrases: Apple, app store, ranking, rating, update

1. INTRODUCTION

Since its inception in 2008, the Apple iPhone app store has experienced massive growth, largely fueled by increasing adoption of smartphones worldwide. The store contains over 1.5 million apps in a variety of categories including business, games, lifestyle, productivity, and many more. In the early stages of Apple's app market which initially boasted just 552 apps [6], the casual developer could stumble upon tremendous success. The app selection was small at first with just 552 apps, and the barriers to entry as a developer were non-existent. Today, most successful contenders in the market are large businesses with well-funded engineering teams, marketing, sales, and a variety of other resources. So, as mobile developers ourselves, we were highly interested in gaining a deeper understanding of the mechanics of the market. As a casual developer with limited resources, what variables can you control that will help drive the success of your apps? What insights about consumer behavior can be used to the developer's advantage?

While there may be countless methods of driving app growth, we chose to focus our analysis on the perspective of the consumer shopping in an app store. We ignore variables like brand recognition and marketing, which are hard to quantify, and isolate our study to the

impact of information displayed to every consumer within the app store. While many different app stores exist, the Apple App Store is unique because it has an exclusive relationship with the iPhone -- apps in the store can only be downloaded using an iPhone, and an iPhone can only download apps from the app store. So, every app available to consumers is listed in the store, and every consumer is exposed to the exact same store environment. This ensures there are no apps missing from the dataset, but even more importantly, allows us to assume that all informed consumers are making decisions based on a uniform dataset and presentation.

The Apple App Store is an application that comes preinstalled on every iPhone, and is also accessible through iTunes. There are several sections of the store, but our study is interested mostly in the Top Charts sections. This page displays the top 100 ranked apps at the present time for three categories: Free, Paid, and Grossing. For each app in the chart, consumers are shown a rank from 1 to 100, the apps name, a star rating from 1 to 5, the category, an icon image, and the price (if it's a paid app). Aside from the rank, this star rating is the consumer's first exposure to any information about the quality of an app, intuitively leading us to propose that the rating could be highly correlated with success. Upon further inspection, we discovered that the star rating shown in the Top Charts represented only the star rating average for the current version of the app. In other words, every time an app releases an updated version, the star rating data that is displayed on the storefront is reset.

This feature of the app store motivated the focus of our model. If an app's star rating gets reset to zero every time it updates, how big of an impact can it make when an update drastically changes the rating? We investigate this question by classifying updated versions either as good or bad and tracking the overall trend in ranking after the update.

While our model produces a good visualization of trajectory after an update, it also has some weaknesses. We are focusing only on a single variable, so its is challenging to make any assumptions about causation. Instead, we demonstrate that there is definitely some correlation between the quality of an update and its short term ranking trajectory in the following days, especially for the *bad* updates.

2. RELATED WORK

Due to the popularity of the Apple app store, much research has been done to understand and study the determinants of app download rate and success. Since we focus on the contribution of good and bad updates to app rankings on top charts, our present work relates to many lines of existing research. First, there is research on the app store's overall determinants of app demand, whereas our particular focus dives deeper into the effects of current version ratings in an app's short term ranking. Second, Apple's ranking algorithm for its top apps is private, but research has been done to hypothesize the type of algorithms and variables that come to play when determining the top apps of the store. That research is relevant because our work pays attention to fluctuations of ranking when new updates hit the market. Third, other online papers make observations of app demand based on overall ratings. However, to our knowledge, no research paper has specifically analyzed the individual effects of version updates on ranking. Because we build on top of research involved with app store success determinants, this section provides useful context and information about the app store market.

The first subject that we observed in related works is the determination of app store demand. Because of the app store's large consumer base, determining app store provides opportunities for analyzing strategies for obtaining the largest percentage of users. Since July 2015, there have been 100 billion downloads in the Apple app store alone [7]. The key detriment to understanding the app store market is lack of demand data. Although the Apple app store provides ranks of the most downloaded apps, without proper demand data, it is hard to access the value of a certain rank. Using publicly available data, research has found a direct relationship between the ranks on the Apple app stores and sales of apps [8]. As expected, more downloads increase ranking, and it is expected that an increase in ranking insinuates more downloads.

However, papers have also observed that the Apple app store does not simply have a 1:1 relationship between downloads and rankings. Although we acknowledge that no single determinant influences ranking completely, the analysis of the specific influence of version updates can help us understand the app store in general. Papers regarding simulations of the app store have been created in an attempt to discover the private algorithms that Apple uses in determining rank of a specific app. For instance, AppEco, an Artificial Life model of mobile app ecosystems, simulates the Apple app store and demonstrates the complexity involved in app store ranking algorithms [9]. Observing the overall trends with user downloads and simulating the entire app store ecosystem provides better understanding of user behavior and archetypes [10]. For Apple's ranking algorithm, if success to determine top ranks is measured poorly, then undesirable apps linger, and downloads will fall.

We observe that the app store's top ranked charts tend to have a bunch of apps that have stayed on the top charts for months, and for the most popular few, up to years. If the app store ranking algorithm is based solely on downloads, then an app like Messenger, currently #1 in top free iOS iPhone apps as of the whole month of Dec. 2015, must maintain a greater number of downloads than the next in rank. The more likely reason for its long reign on the top is that the ranking algorithm also strongly considers user updates. In simulations of app ecosystems, updates may also be considered as downloads in the ranking algorithm, and the simulations also propose that keyword search for an app also contributes to its rank [9], [10].

There has been much interest in ratings and reviews and their relationships with app demand and ranking. Through the collection of user reviews on specific apps, research has found statistically significant positive impact of app downloads with high overall ratings and reviews [11]. To take this analysis on user ratings and reviews deeper and understand the behavior of the app store ranking algorithm, the logical next step is to look at latest updates in particular and find whether a single update can make a difference in statistically significant change in short term ranking.

3. DATASETS

Our data was collected from a website called AppAnnie.com. App Annie is a business intelligence company that provides mobile app market reports including the Apple app store's. Although App Annie contains data from the main mobile app markets such as Android, Amazon, and Apple, we focused our dataset specifically on iPhone data. App Annie contains data as early as Feb. 8, 2010, but we only managed to scrape iPhone data from Jan. 7, 2014 to

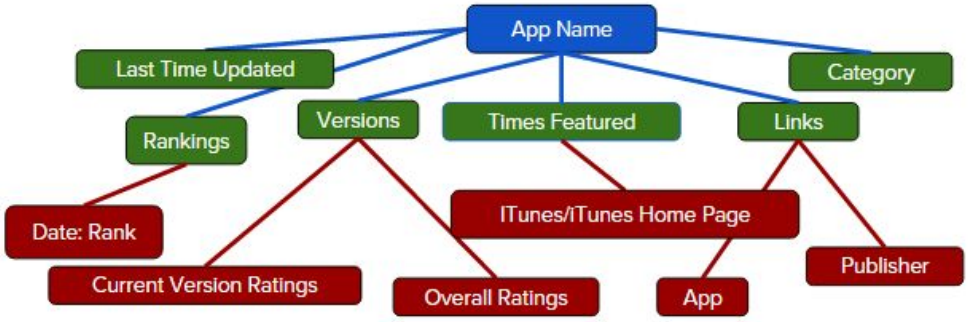

Fig. 1: We collected the above tree hierarchy data for every app in the top charts stored in a JSON file.

Dec. 9, 2015. For every one of those days, we were limited in extracting the top 100 on the Free and Paid charts. The extracted dataset contained a total of 3,592 unique apps that secured a spot on the top 100 Free and Paid charts on the iPhone at least once. From those 3,592 apps, 2,563 were free, and 1,516 were paid.

To collect App Annie data, we started checking what data was available on App Annie's publicly available Store Stats page. From there, we could find the top 500 charts for the popular app stores, but we chose to look at iOS iPhone data that was separated into three columns for free, paid, and grossing. App Annie has a unique page for every day of the top charts. They also provide more specific charts for country and category, but we chose to focus on the United States country and Overall category only. These unique pages based on day contain the *app name, publisher, rank position, and whether the app is free, paid, or grossing*, An important note is that one must be logged into an App Annie account to see any store data not from the current day.

For every unique page on the day's top charts data, every app also contained a unique link with additional information on the app. App Annie's app-specific links contained metadata including but not limited to description, number of times featured, and overall ratings. From this set of data, we chose to extract information on the *last date updated, latest version ratings, overall version ratings, number of times featured, and category of the app*. Like the data-specific pages, one must be logged into an App Annie account for these app-specific pages to see the app's detailed metadata.

Because of the layout of the date-specific top charts pages, we chose to scrape the top 100 free and paid from the list of 500 for each day. And from these date-specific pages, we scraped an app's specific metadata page if we had not seen the app before in the top charts. Even though we scraped a total of 3,592 apps that had entered the top charts at least once, not all of the apps fit our main criteria: having a previous version. From this 3,592 apps, only 2,504 of these apps had undergone an observable update. The Apple app store only contains data on the last version update ratings and overall cumulative ratings.

3.1. SCRAPING

App Annie was by far the best source of data on app markets from what we observed of competitors. Apple does not provide an easy means of collecting all the data that we required easily, so we chose to collect data from App Annie. Compared to other app market analysis sites, App Annie provided the most appropriate data pertaining to our research on version updates.

To scrape for a specific dataset like iPhone data from Jan. 7, 2014 to Dec. 9, 2015 on App Annie's website was a difficult feat because of App Annie's defense protocols against scraping. To obtain even such a limited dataset size, we had to use popular data mining techniques such as IP proxy, multiple scripting, account switching, intervals between calls [12], [13]. By outlining our methods of obtaining our app store top charts dataset, we can present options and insight for others to obtain theirs.

Our data collection process was divided into two steps. First, we used a script to scrape the date-specific pages where we could collect app data in the top 100 of free and paid for every day. We ran this particular script to scrape in intervals of six months of top chart data. Each portion of the data was stored inside a separate dictionary to create a tree/list of dictionaries demonstrated in Figure 1 for every app. Then, we would write the dictionary, containing all the apps inside a JSON file. Secondly, we used another script to visit the date-specific pages again but scrape from every app-specific page. The second script ingested the JSON file that was produced from the first script. We only needed to add app-specific pages for a particular app once, so for the second script, we checked whether we had added the app's specific information before and if we had not, then we added the data. We ran this second script for the same interval of time, and after adding all the app-specific metadata to the first script's data, we wrote a second JSON file with the accumulation of the two's data scraping.

App Annie set up many defenses to deter users from scraping their website. One of the main issues regarding the scraping that made the process much more difficult was that date-specific pages, not from today's top charts, and app-specific pages both required an account logged into the session. Though creating the account and login session is simple, if a user logs into App Annie too many times from the same IP, App Annie implements a captcha to stop scripts from web scraping. With our work, we were not able to programmatically circumvent the submission of the captcha, so we had to chose to assign our machine with a new IP in the face of receiving a page that asked for a captcha.

There were two major issues that caused much of our data collection problems. First, once logged in, if a user requests too many calls from his or her machine, then App Annie will request a captcha submission on all pages. On one hand, we could change our IP address when App Annie requested captcha submission on our specific IP, but since App Annie would also require captcha submission on our account, we found no way to circumvent this issue unless we manually submitted the captcha. We found that a rough estimate of a mere 250 calls to App Annie would cause a captcha request to verify that we were not a robot. 250 calls would only allow us to collect data for about 20 days of app-specific pages at a time. Since App Annie would request captcha submission for our IP address along with for our specific user account, we would have to submit the captcha once for every 20 days of data before letting the script run

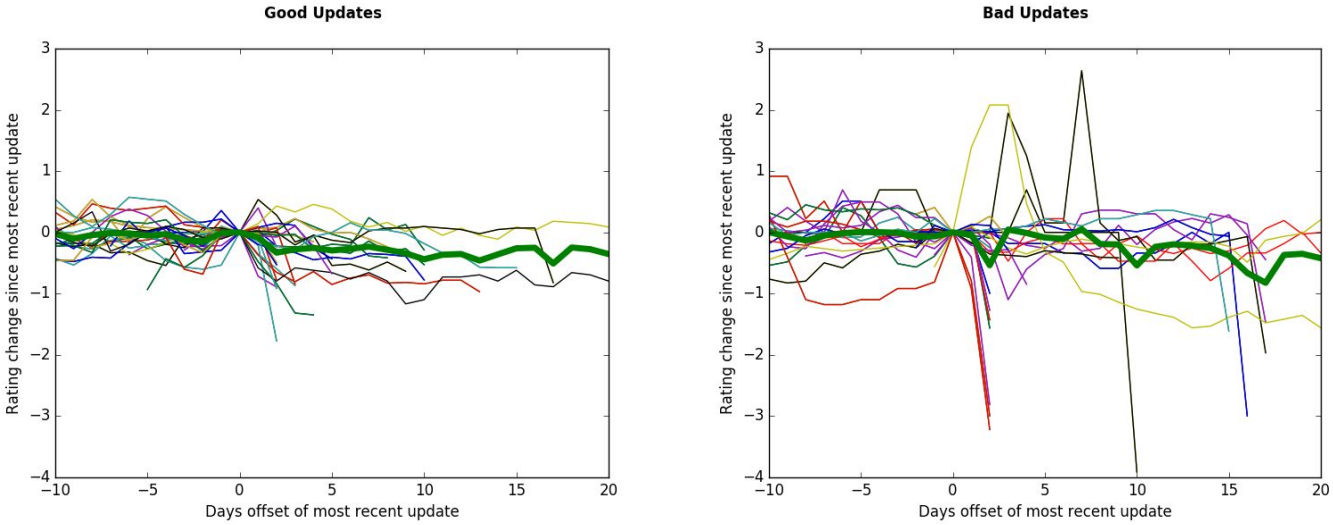

Fig. 2: Ranking change based on good updates and bad updates, centered on date when app received the latest update.

again. If faced with a captcha page, our scripts would write the current dictionary contents to the JSON file and notify us. We logged the date of the page from when we received the captcha so that we could continue from that day and onward, accumulating data bit by bit.

The second major issue involved with complete bans of our IP addresses or user accounts. App Annie would occasionally ban our IP address and user account once we requested approximately 1000 calls. For the IP address, we could programmatically change it, but the banning of user accounts was a huge problem. After about a month and a half of data collection, our user account would be banned, and we would receive a 403 when requesting any page. At first, we would request App Annie support to unban our user account for the continuation of our academic endeavors, but the turnaround time would be roughly a day. We decided that we would save more time by creating another free account whenever one of our accounts were banned. From what we could scrape in a timely duration and overcoming the captcha and banning hurdles, our work was limited to collection of iPhone top charts data from Jan. 7, 2014 to Dec. 9, 2015 [14].

3.2. TERMINOLOGY

We will clarify the main terminology from our collected dataset in the paper. A review on the Apple app store consists of a star rating format, which takes a value from {1, 2, 3, 4, 5}. Our dataset contains the number of current version ratings in this 1-5 set as well as the cumulative overall number of total ratings. We use the term *current version* and *latest update* interchangeable. Our work analyzes the current version ratings as well as separates them from the overall ratings. We call this separated overall ratings as the *before update* ratings.

Table I: Summary of short term ranking change before and after good update

Days offset from update date	-5	1	3	5	10
log(Average ranking change)	0.002	-0.015	-0.113	-0.182	-0.208
Standard deviation	0.281	0.399	0.406	0.428	0.430

Table II: Summary of short term ranking change before and after bad update

Days offset from update date	-5	1	3	5	10
log(Average ranking change)	0.004	-0.423	-0.215	-0.116	-0.347
Standard deviation	0.529	0.488	0.611	0.245	0.629

Throughout our analysis, we use the terms, *good* and *bad* updates. To define the difference between a good and bad update, we calculate the average change from before update ratings and current version ratings. From the average, we calculate the standard deviation in rating change from before update and current version. In our work, we consider a good update as an update that resulted with at least 1 standard deviation above the average rating change. We consider a bad update as an update that resulted with at least 1 standard deviation below the average rating change. Although not perfect, classifying rating change as good and bad provides ourselves a means of qualifying updates for our work.

4. METHODOLOGY

Our analysis of the data focuses on the ranking trajectory in the twenty days following *good* and *bad* updates. We chose to track the apps only for twenty days because at this point, our dataset started to decrease rapidly in quantity. Our dataset likely included many apps whose last update was only days before we stopped collecting data, so we could only map their trajectory for a few days. One issue we faced with this approach was normalizing the dataset. Dropping in rank from 1 to 5 is a much more significant drop than from 70 to 75, and we wanted our data to reflect this. To do so, we took the log of all the rankings.

In order to effectively identify a trend in ranking change after the version update, we aligned the apps by converting the date of the ranking to the day offset from the apps most recent update. We also aligned the ranks at the day of the update using the rank at that date as a baseline, such that for any app, its change in ranking on the update date is zero. This way, when we plot the ranking trajectories on a graph, we are able to measure ranking change in comparison to the baseline value on the update date.

Another challenge we faced was figuring out how to represent apps that dropped out of the rankings. If we just stopped including the data from these apps, the plot would not accurately represent that a drop off in ranking occurred. To try to represent this, if an app dropped out of the top 100, we added an additional ranking at that day of 100. In other words,

we know the app dropped to a rank greater than 100, so we know that ranking it as 100 is a safe estimate for its actual ranking which is likely worse.

Figure 2 below shows our plots for both the *good* updates and *bad* updates. From these visualizations, it is clear to see that apps with bad updates are dropping out of the rankings much more frequently. We see the a large negative spike in the first few days after the update, indicating that the average ranking change moved sharply in the negative direction. This supports our initial hypothesis. On the other hand, *good* updates do not seem to have too strong an impact-- certainly not significantly positive.

Tables 1 and 2 verify the observations that we see for good and bad update ranking effects respectively in Figure 2. Like our observation in Figure 2, the numbers reinforce that good updates result with maintaining the app's position in rank whereas bad updates result with a drop in ranking around the value of 1 standard deviation of ranking change after one day. We consider that if average change in ranking meets the value of 1 standard deviation, then the change in ranking is statistically significant. With bad updates, we see clearly that after one day, a bad update results with the average change in ranking at the value of roughly 1 standard deviation. Although the Table 1 provides numbers that show that good updates result with an app staying close to its original rank, Table 2 supports our hypothesis that bad updates hurt short term ranking.

5. CONCLUSION

In our paper. we hypothesized that good updates and bad updates have short term effects on rankings. Our findings resulted with statistically significant effects to short term ranking with bad updates, but for good updates, the ranking maintained neutral position. While we still do not know whether updates have a long term influence to ranking, we have found statistically significant changes in ranking when an app has bad update. Though only a part of our hypothesis as good updates only maintained the rank, a bad update resulted in a significant drop in ranking.

Our findings provide greater analysis and insight to the ranking algorithms of the app store by focusing specifically on version updates and their effects. Examining the short term ranking changes caused by updates gives future work in analyzing other specific determinants of app rank and demand more closely. Though we found results that supported half of our hypothesis, the next step in our analysis would be to provide more detailed models as well as track a series of updates to determine long term effects in addition to the short term. We would also like to see if more detailed models would result with a correction of our findings for good updates. Our work supports that app developers should care about updates enough to avoid bad ratings while their app is on the top charts. Since ranking correlates to number of downloads and our findings reinforce that positive updates generally increase ranking, we would advise app developers to examine updates closely enough for the most user satisfaction before uploading.

REFERENCES

[1] Apple App Store. URL: *https://itunes.apple.com/us/genre/ios/id36?mt=8*

[2] "Number of apps in available in leading app stores as of July 2015." URL: *http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores*

[3] G. Lee, T. S. Raghu. "Determinants of Mobile Apps Success: Evidence from App Store Market." *Journal of Management Information Systems, Forthcoming*. Nov. 2014. Pdf.

[4] X. Liu, H. Jia, C. Guo. "Smartphone and Tablet Application (App) Life Cycle Characterization via Apple App Store Rank." Department of Information and Library Science, School of Informatics and Computing, Indiana University Bloomington. Bloomington, IN, USA. Pdf.

[5] AppAnnie. URL: *https://www.appannie.com*

[6] J. Snell, P. Cohen. "Apple opens iTunes App Store." July 2008. URL: *http://www.macworld.com/article/1134380/app_store.html*

[7] "Cumulative number of apps downloaded from the Apple App Store from July 2008 to June 2015 (in billions)." URL: *http://www.statista.com/statistics/263794/number-of-downloads-from-the-apple-app-store*

[8] R. Telang, R. Garg. "Estimating App Demand From Publicly Available Data." Carnegie Mellon University. Sept. 2011. Pdf.

[9] S. Lim, P. Bentley. "Investigating App Store Ranking Algorithms using a Simulation of Mobile App Ecosystems." United Kingdom. Pdf.

[10] L. Cocco, K. Mannaro, G. Concas, M. Marchesi. "Simulation of the Best Ranking Algorithms for an App Store." Mobile Web Information Systems. 233-247. Pdf.

[11] D. Eric, R. Bacik, I. Fedorko. "Rating Decision: Analysis Based on IOS App Store Data." Nov. 2014. Pdf.

[12] M. Harman, Y. Jia, Y. Zhang. "App Store Mining and Analysis: MSR for App Stores." University College London. London, UK . Pdf.

[13] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, Y. Zhang. "App Store Analysis: Mining App Stores for Relationships between Customer, Business and Technical Characteristics." UCL Department of Computer Science. Sept. 2014. Pdf.

[14] App Annie scraper. URL: *https://github.com/huyle333/appannie-scraper*