A Longitudinal Study of the Google App Market

Bogdan Carbunar
Florida International University
Miami, FL
Email: carbunar@cs.fiu.edu

Rahul Potharaju
Cloud and Information Services Lab, Microsoft
Redmond, WA
Email: rapoth@microsoft.com

***Abstract*—Recently emerged app markets provide a centralized paradigm for software distribution in smartphones. The difficulty of massively collecting app data has led to a lack a good understanding of app market dynamics. In this paper we seek to address this problem, through a detailed temporal analysis of Google Play, Google's app market. We perform the analysis on data that we collected daily from 160,000 apps, over a period of six months in 2012. We report often surprising results. For instance, at most 50% of the apps are updated in all categories, which significantly impacts the median price. The average price does not exhibit seasonal monthly trends and a changing price does not show any observable correlation with the download count. In addition, productive developers are not creating many popular apps, but a few developers control apps which dominate the total number of downloads. We discuss the research implications of such analytics on improving developer and user experiences, and detecting emerging threat vectors.**

I. Introduction

The revolution in mobile device technology and the emergence of "app markets", have enabled regular users to evolve from technology consumers to enablers of novel mobile experiences. App markets such as Google Play provide new mechanisms for software distribution, collecting software written by developers and making it available to smartphone users. This centralized approach to software distribution contrasts the desktop paradigm, where users typically obtain their software directly from developers.

With more than one million apps, the Google Play Store has evolved significantly since its initial AppStore (Apple) inspired model. By making revenue-sharing transparent for developers in paid downloads (70-to-30 cut), Google offers financial incentives for contribution to app development.

Despite the popularity of app markets, due to their recency and to the intricacies involved in obtaining time series data from them, little is known about their internal dynamics. Developers and users play key roles in determining the impact that market interactions can have on future technology and economics. However, new developers currently lack an understanding of how their actions can impact the success of their apps, and users lack guidance when choosing among apps claiming similar functionality.

In this paper, we perform one of the first characteristic studies of Google Play using real-world time series data, that we collected from more than 160,000 apps [1] (out of a total of around 700,000 apps in 2012), over a period of six months in 2012. With a focus on developers and their apps, we seek to answer several questions:

- Q1 Is the app market stale or are the apps updated frequently? What are the characteristics of an app update? Are they bandwidth intensive?
- Q2 Are developers pricing their apps appropriately?
- Q3 How many developers control the app supply? Do they adjust app prices? Do developer actions impact the popularity of their apps?
- Q4 How do top apps and top app lists evolve in time?

Our Findings. Our analysis reveals several surprising facts:

1) Market inactivity has a significant impact on the price distribution. Relying on statistics computed on the entire population (as opposed to only active apps) may mislead developers, e.g., to undersell their apps (§IV).
2) Typical app update cycles are bi-weekly or monthly. More frequently updated apps (under beta-testing or unstable) can impose substantial bandwidth overhead and expose themselves to negative reviews (§IV).
3) With every subsequent software update, a developer is more likely to decrease the price. However, contrary to popular belief, changing the price does not show an observable correlation with the app's download count(§V).
4) Developers that create many applications are not creating many popular applications. However, a few elite developers are responsible for applications which dominate the total number of downloads (§V).
5) A majority of apps in top-k app lists follows a "birth-growth-decline-death" process: they enter/exit from the bottom part of a list. Apps that attain higher ranks have better stability in top-k lists than apps that are at lower ranks (§VI).

Research Implications. We describe threat vectors that are likely to grow in importance, including scam apps, i.e., paid apps that attract downloads through misleading names and unfounded claims, and update-based misbehavior such as DoS attacks through frequent and large updates, as well as Jekyll-Hyde apps that use updates to transform from popular and benign into malware.

In addition, we argue that a detailed time series study of app markets can improve developer and user experiences. For instance, by integrating predictions of the impact that price, permissions and code changes will have on the app's popularity, as well as insights extracted from user reviews, app development tools can help developers optimize the success of their apps. Visualizations of conclusions and analytics similar

[1]We have collected data from more than 470,000 apps, but we have complete daily data for 160,000 (see §III).

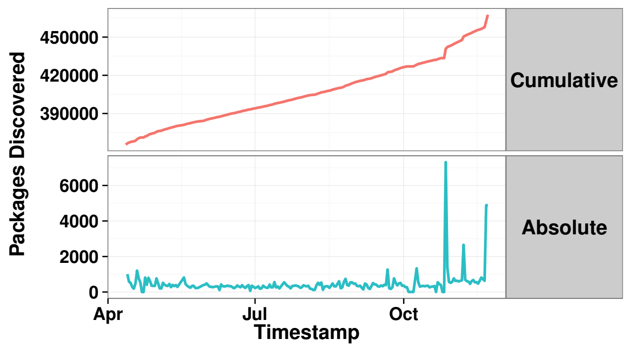


Fig. 1. Number of packages discovered per day, using 700 machines. At the end of the process, we had collected data from more than 470,000 apps.

to the ones performed in this paper can help users choose among apps with similar claimed functionality. We include a detailed discussion of the applicability and future research directions in app market analytics in §VII.

Limitations. While the Google Play market represents a large body of other third-party markets and their environments, we do not generalize our conclusions. We aim to study economic aspects such as network effects, bidding wars and price elasticity, system aspects such as permission evolution and app size properties in future work.

II. Related Work

Petsas et al. [18] are the first to explore mobile app markets in the context of 4 providers, that do not include Google Play. They show that the distribution of app popularity deviates from Zipf, due in part to a strong temporal afnity of user downloads to app categories. They show that on the markets they studied, paid apps follow a different popularity distribution than free apps. In contrast, our work exclusively analyzes Google Play, the most popular Android app market. In addition, we focus on different dimensions: (i) market staleness and its effect on app pricing, (ii) app updates and their effect resource consumption, (iii) the control of the market and the effect of developer actions on the popularity of their apps and (iv) the evolution in time of top apps and top-k app lists.

Xu et al. [23] use IP-level traces from a tier-1 cellular network provider to understand the behavior of mobile apps. They provide an orthogonal analysis of spatial and temporal locality, geographic coverage, and daily usage patterns.

Security is has been a theme in the large scale collection of mobile apps. Previous work includes malware detection [25], malware analysis [24], malicious ad libraries [14], vulnerability assessment [11], overprivilege identication [12] and detection of privacy leaks [10]. While in this paper we focus on the different problem of understanding the dynamics of Google Play, we also introduce novel mobile app attacks.

III. Data & Methodology

We collected our data from *Google Play* [3], an app distribution channel hosted by Google for its open source software stack Android. In order to submit apps to Google Play, a developer first needs to obtain a publisher account for a one-time fee of $25 [13]. Google allows developers to freely submit any number of apps without imposing an app

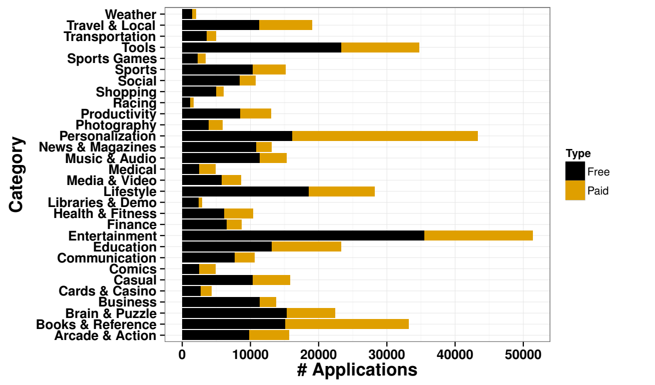


Fig. 2. Distribution of free vs. paid apps, by category. While the number of free apps exceeds the number of paid ones, this property does not hold for several popular app categories.

review process. Developers can sell their apps for a price of their choice, or distribute them for free. Google Play lists apps according to several categories, ranging from "Arcarde & Action" to "Weather". Users download and install apps of interest, which they can then review. A review has a rating ranging from 1 to 5. Each app has an *aggregate rating*, an average over all the user ratings received.

Methodology. We have built a crawler to collect and process metadata of Google Play apps using a total of 700 machines for a period of 7.5 months (February 2012 - November 2012) which we call the *crawl* period. As the performance of a crawler depends heavily on the initial seed list — a bad seed list will either send the crawler in a loop (links pointing to each other) or will make it hit a dead-end (no more links to crawl) — we relied on our own app discovery process that relies on the "Similar Apps" portion of each new HTML page that is fetched to discover new apps. Due to this, the number of apps we discover increases with time. Figure 1 shows the app discovery of our crawler during the last six months of the crawl period (April 2012 - November 2012), which we call the *observation* period. We consider the first 1.5 months as the warm up period during which roughly 240,000 apps were discovered and do not consider data collected during this period for subsequent analysis. At the end of the crawl period, we had collected data from more than 470,000 apps.

To avoid overloading the provider, we limited our crawler to perform 20 parallel crawls with a two second sleep period. A daily crawl takes ≈7-14 hours, a time window safe enough to prevent roll-over to the next day. At the end of each day, we archive the raw HTML pages (≈14 GB compressed/day) to support any additional data analytics (e.g., analyzing HTML source code complexity). Our six months of archived raw files consume ≈7 TB of storage and the processed data (described below) consumes ≈400 GB including relevant indexes. At the end of the crawl period, 160K apps (out of the total 470K apps) form a good set – we had daily meta-data for these apps for six months (the observation period).

Dataset. Our 160K apps include both free and paid apps. As mentioned earlier, for each app, we have taken daily snapshots of the application-related meta-information consisting of developer name, app category, downloads (as a range i.e.,

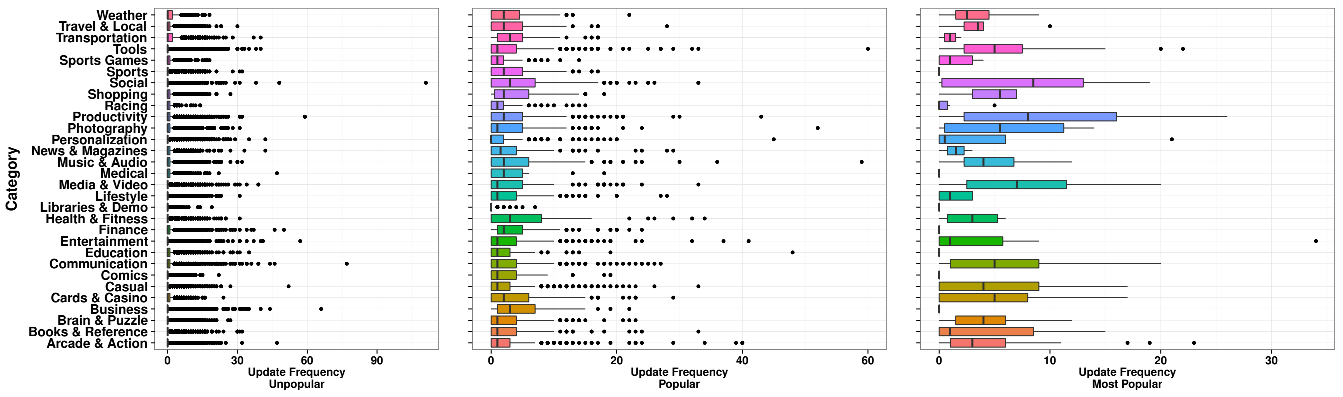


Fig. 4. The distribution of update frequency, i.e., the update count for each app per category. As expected, unpopular apps receive few or no updates. Popular apps however received more updates than most-popular apps. This may be due to most-popular apps being more stable, created by developers with well established development and testing processes.

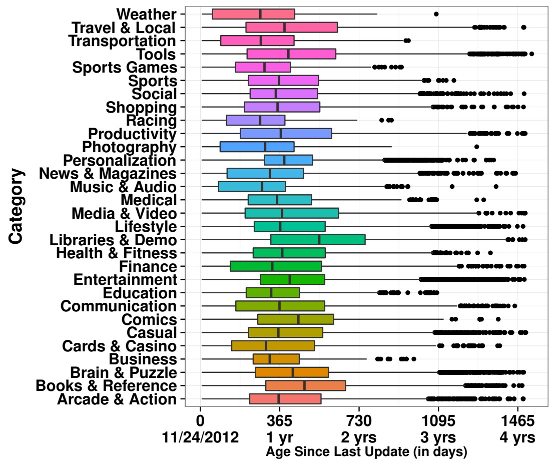


Fig. 3. Box-Whisker plot of the time distribution from the last update, by app category. At most 50% of the apps in each category have received an update within a year.

Class	Review counts	Percent Apps
Unpopular	$0 - 10^3$	74.14
Popular	$10^3 - 10^5$	24.1
Most-Popular	$> 10^5$	0.7

TABLE I
POPULARITY CLASSES OF APPS, ALONG WITH THEIR DISTRIBUTION. NOTE THAT MOST APPS ARE UNPOPULAR, AND ONLY 0.7% OF THE APPS HAVE MORE THAN 100,000 DOWNLOADS.

10-100, 1K-5K etc.), app aggregate rating (on a 1-5 scale), review count (absolute number of user ratings), last updated timestamp, software version, OS supported, file size, price, url and the set of permissions that the app requests. Figure 2 shows the per-category distribution of paid vs. free Google Play apps. While overall, the number of free apps exceed the number of paid apps, several popular categories such as "Personalization" and "Books & References" are dominated by paid apps.

IV. Popularity and Staleness

In this section, we classify apps into classes based on their popularity towards gaining an understanding of market staleness i.e., the fraction of apps that are active, and the implications it can have on app pricing. Next, we study in detail the frequency of app updates for apps from various classes and the implications they can have on end-users.

Market Staleness: We classify apps according to their popularity into three classes (shown in Table I) based on their review count. We say an app is *stale* if it has not been updated within the last year from the observation period. We say that the market is stale if 50% of its apps are stale.

Figure 3 shows the Box-Whisker plot [6] of the per-app time since the last update, by app category. At most 50% of the apps in each category have received an update within an year from our observation period. For instance, most apps in *Libraries & Demo* have not been updated within the last 1.5 years. Some categories such as *Arcade & Action*, *Casual*, *Entertainment*, *Books & Reference*, *Tools* contain apps that are older than three years. By randomly sampling these apps, we found the following to be the main reasons for their staleness: (1) they are either stable or classic (time-insensitive apps that are not expected to change) and hence do not require an update (e.g., "Divide and Conquer" in *Casual*), (2) developers have abandoned them (e.g., "Amazed" in *Casual*), and (3) they do not require an update (e.g., e-books, wallpapers, libraries).

Note that the presence of a significant percentage of stale apps, and of popularity classes, may mislead developers in their process of determining a listing price for their app. The median price in our dataset is $0.99 when all apps are considered and $1.31 when considering only active apps. This indicates that developers that set their price based on the former value may be selling their apps at lower profits.

App Updates: We now focus on the update frequency of apps in our dataset, during the observation period (April-November 2012). We seek to understand whether app developers prefer seamless updating i.e., do they push out releases within short time periods? In our dataset, only 24% apps have received at least one update in the observation period.

Figure 4 plots the distribution of update frequency of apps across categories based on their popularity. As expected, *Unpopular* apps receive few or no updates. We observed that this is due to the app being new or abandoned by its developer. For instance, "RoboShock" from "DevWilliams" in *Arcade & Action* with good reviews from 4 users has received only one update on September 28, 2012 since its release in August 2011 (inferred from its first comment).

Outliers (e.g., "Ctalk" in the *Social* category) push out large number of updates (111). Popular apps are updated more frequently: 75% in each category receive 10 or less updates, while some apps average around 10-60 updates during our observation period. User comments associated with these apps indicate that the developer pushes out an update when the app attracts a negative review (e.g., "not working on my device!"). In the *Most-Popular* category, the population differs significantly. While some apps seldom push any updates, apps like "Facebook" (*Social*) have been updated 17 times. The lower number of updates of most popular apps may be due to testing: Companies that create very popular apps are more likely to enforce strict testing and hence may not need as many updates as other apps.

To identify how frequently developers push these updates, we computed the average update interval (AUI) per app measured in days (figure not shown). In *Popular* and *Unpopular* classes, 50% of apps receive at least one update within 100 days. The most interesting set is a class of Unpopular apps that receive an update in less than a week. For instance, the developer of "Ctalk" pushed, on average, one update per day totaling 111 updates in six months indicating development stage (it had only 50-100 downloads) or instability of the app. On the other hand, *Most-Popular* apps receive an update within 20 to 60 days.

Updates, bandwidth and reputation. A high update frequency is a likely indicator of an on-going beta test of a feature or an unstable application. Such apps have the potential to consume large amounts of bandwidth. For instance, a music player "Player Dreams", with 500K-1M downloads, pushed out 91 updates in the last six months as part of its beta testing phase (inferred from app description). With the application size being around 1.8 MB, this app has pushed out ≈164 MB to each of its users. Given its download count of 500K-1M, each update utilizes ≈0.87-1.71 TB of bandwidth. We note that frequent updates, especially when the app is unstable, often attract negative reviews. For instance, "Terremoti Italia" that pushed out 34 updates in the observation interval, often received negative reviews of updates disrupting the workflow.

Findings 1: (1) Similar to desktop ecosystem, developers in mobile ecosystem prefer seamless updating with typical update cycles being bi-weekly or monthly. (2) Apps that are unstable or under beta-testing can impose substantial bandwidth overhead and expose themselves to negative reviews. (3) Google Play is stale with at most 50% of the apps being updated in all categories. (4) Market staleness has a significant impact on the median price (difference of ≈$0.32). Thus, computing aggregate statistics on the entire population may result in counter-intuitive findings, and may mislead developers to undersell their apps.

V. Developer Impact

In this section, we are interested in understanding what fraction of popular apps are being controlled by an elite set of developers and if there is a power-law effect in-place. Next, we analyze the impact that developer actions (e.g., changing the price, permissions etc.) can have on the app popularity.

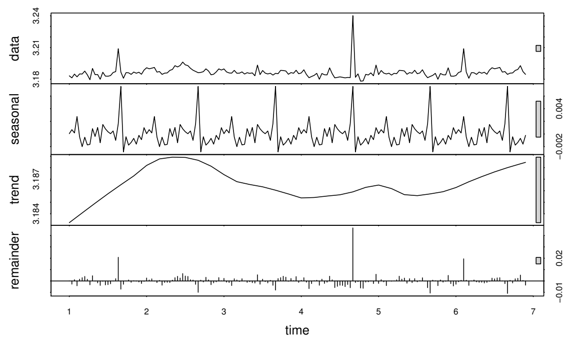


Fig. 6. Monthly trend for average price. The average price does not exhibit seasonal monthly trends.

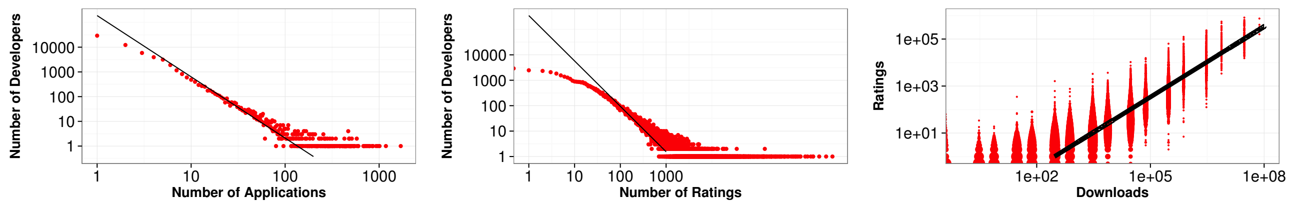
A. Market Control

To understand the impact that developers have on the market, we observe their number of apps, downloads, and review count. Figure 5 plots these distributions, all showing behavior consistent with a power-law distribution [17]. We display the maximum likelihood fit of a power-law distribution for each scatter plot as well [16], [8]. The α coefficients for the number of applications, reviews and downloads are 2.48, 1.79, and 1.78, respectively. The Kolmogorov-Smirnov (a goodness-of-fit indicator) fit values are 0.045, 0.032, and 0.031, respectively. The coefficients determine the steepness of the power-law curve. They show that the number of apps per developer has the highest power-law distribution: a few developers have a large number of apps while many developers have few apps. However, the developers that post the most apps do not have the most popular apps in terms of reviews and download counts. Instead, Figure 5(b) shows that a few developers control apps that attract most of the reviews. Since Figure 5(c) shows an almost linear relation between review and download counts (1 review for each 300 downloads), this implies that those apps are also popular.

B. Price Dispersion

Price dispersion is the spread between the highest and lowest prices in the market. In our dataset, we used the *coefficient of variation* (COV) [22], the ratio of standard deviation to the mean, to measure price dispersion. COV= 1 indicates a dispersal consistent with a Poisson process i.e., uniformly at random; COV> 1 indicates greater variability than would be expected with a Poisson process; and COV< 1 indicates less variation. In our dataset, we observed an average COV (computed for all apps) to be 2.45 indicating a non-negligible price dispersion, in agreement with results in the context of other electronic markets [7].

Figure 6 shows the STL decomposition [9] of the average price timeseries in the observation interval, for a periodicity of one month. The grey-bar on the seasonal panel (see Figure 6) is only slightly larger than that on the data panel indicating that the seasonal signal is large relative to the variation in the data. In the trend panel, the grey box is much larger



(a) (b) (c)

Fig. 5. (a) Distribution of apps per developer. (b) Distribution of total reviews per developer. (c) Scatter plot of downloads vs. ratings in Google Play. Both axes are log-scaled. A linear curve was fitted with a slope of 0.00341 indicating that an application on this market is generally rated once for about every 300 downloads.

	D ↑	P ↓	P ↑	RC ↑	SV ↑	TP ↓	TP ↑
D ↑		0.18	-0.02	0.13	0.34	0.09	0.21
P ↓	0.18		-1.00	0.09	0.89	0.89	0.93
P ↑	-0.02	-1.00		-0.23	0.72	0.51	0.76
RC ↑	0.13	0.09	-0.23		0.73	0.65	0.70
SV ↑	0.34	0.89	0.72	0.73		0.99	1.00
TP ↓	0.09	0.89	0.51	0.65	0.99		-1.00
TP ↑	0.21	0.94	0.76	0.70	1.00	-1.00	

TABLE II
YULE ASSOCIATION MEASURE (EQUATION 1) FOR ALL PAIRS OF ATTRIBUTES. D DENOTES NUMBER OF DOWNLOADS, P IS PRICE, RC IS REVIEW COUNT, SV IS SOFTWARE VERSION NUMBER AND TP IS THE TOTAL NUMBER OF PERMISSIONS. (↑) DENOTES AN INCREASING ATTRIBUTE AND (↓) DENOTES A DECREASING ONE. THE DARKER THE SHADING OF A CELL THE STRONGER THE (POSITIVE) CORRELATION.

than either of the ones on the data/seasonal panels, indicating the variation attributed to the trend is much smaller than the seasonal component and consequently only a small part of the variation in the data series. The variation attributed to the trend is considerably smaller than the stochastic component (the remainders). We deduce that in our six month observation period these data do not exhibit a trend.

Findings 2: (1) Price dispersion is significant, with a COV of 2.45, an observation similar to the results obtained in the context of other electronic markets [7]. (2) The average price does not exhibit seasonal monthly trends.

C. Impact of Developer Actions

Developers have control over several attributes they can leverage to increase the popularity of their apps. This includes pricing, the number of permissions requested from users and the frequency of updates. In this section we investigate the relation between such levers and their impact on app popularity. For instance, common-sense dictates that a price reduction should increase the number of downloads an app receives. Reducing permissions should also increase app popularity thereby impact the download count. We note that permission changes occur when the developer updates the app.

We are interested in verifying whether this is indeed the case and if so, how weakly or strongly one attribute change causes a change in the other. We study the association between app attribute changes. We define a random variable for increase or decrease of each attribute, and measure the association among pairs of variables. For example, let X be a variable for price increase. For each $\langle$ day, app $\rangle$ tuple, we let X be a set of all of the app and day tuples where the app increased its price that day (relative to the previous day's value). For this analysis we consider 160K apps which actually have changes throughout our observation period, and we discard the remaining apps. We use the Yule measure of association[21] to quantify the association between two attributes, A and B:

$$\frac{|A \cap B| * |\overline{A} \cap \overline{B}| - |A \cap \overline{B}| * |\overline{A} \cap B|}{|A \cap B| * |\overline{A} \cap \overline{B}| + |A \cap \overline{B}| * |\overline{A} \cap B|} \tag{1}$$

$\overline{A}$ is the complement of A, i.e., each $\langle$ day, app $\rangle$ tuple where the attribute does not occur, and $|A|$ denote the cardinality of a set (in this case A). This association measure captures the association between the two attributes: zero indicates independence, +1 indicates perfectly positive correlation, and -1 perfectly negative correlation. Table II shows the association measure values for all pairs of download count (D), price (P), review count (RC) and the total number of permission (TP) attributes.

Table II shows that a price decrease has a high association with changes in software version and permissions. However, similarly high associations are not observed with a price increase. Thus, when a developer is updating software or permissions they are more likely to decrease the price than increase the price of an app.

Contrary to popular belief, changing price does not show significant correlation with the download or review counts. We randomly sampled 50 apps where this is happening and observe the following to be the main reasons. First, apps are initially promoted as free and a paid version is released if they ever become popular. However, in some cases, the feature additions are not significant (e.g., ads vs. no ads) and hence do not cause enough motivation for users to switch to the paid version. Second, with app markets offering paid apps for free as part of special offers (e.g., Thanksgiving deals), users may expect the app to be given out for free rather than take a discount of a few cents.

In addition, software version and total permissions are highly positively correlated. Such association agrees with intuition since a developer changing permissions generally is making a change to the software.

Findings 3: (1) Productive developers are not creating many popular apps. (2) Metrics indicating popularity of a developer follow a power-law distribution: a few developers control apps which dominate the total number of downloads. (3) Price decrease has a high association with software version and permission usage, and (4) Contrary to popular belief, changing price does not show any observable correlation with the download count.

VI. Top-K Dynamics

Google publishes a variety of lists including *Free* (most popular free applications), *Paid* (most popular paid applications), *New (Free)* (newly released free applications), *New (Paid)* (newly released paid applications) and *Gross* (highly grossing applications). These lists are typically updated based on application arrival and the schedule of Google's own ranking algorithms. We took hourly snapshots of five top-k lists ($\approx$ 3000 apps) between Jul-Nov, 2012 ($\approx$2880 hours worth of data).

The ranking algorithms are kept secret. We believe this is done to avoid their misuse by developers who want to be placed high on specific lists. However, in this section we seek answers to several fundamental questions: How long will an app remain on a top-k list? Will an app's rank increase any further than its current rank? How long will it take for an app's rank to stabilize? Developers can leverage this knowledge to better prepare for sudden incoming traffic (in terms of comments, ratings etc.) and apply financial forecasting (for paid apps in terms of app-revenue; for free apps in terms of ad-revenue).

A. Top-K App Evolution

We seek here to understand and characterize the life of an application on the top-k list. Specifically, we would like to investigate whether apps follow the "birth-growth-decline-death" process (inverted bathtub curve [15]). We summarize the evolution of apps on the top-k lists using three measures, DEBUT and EXIT, denoting the rank on the top-k list when the app joins/leaves the list, and TOTL.HRS denoting the total number of hours spent by an app in a top-k list.

Figure 7 shows the histograms for the 3 measures for the 3000 apps we monitored. For DEBUT and EXIT, measuring rank, smaller numbers indicate better performance. For TOTL.HRS measured in hours, higher values are better. Figures 7(a) and 7(b) show that most apps entered and exited from the bottom part of the list (indicated by the high debut and exit ranks). This is consistent with the lifetime metaphor discussed earlier. Notable exceptions include the "ROM Manager" app, that entered at #1 on August 14, 2012, and exited at rank #20 on October 6, 2012, occupying seven different ranks during its lifetime on the list.

Figure 7(c) shows that *New (Free)* and *New (Paid)* apps do not stay on the list for more than 500 hours ($\approx$ 20 days) indicating that these lists may be taking into account all those applications which were last updated in the last 20 days. We have confirmed this hypothesis also by verifying that indeed the "last updated" field of these apps is within the last 20 days. From the same figure, for other lists, we also emphasize the presence of a long tail of apps that have been present for thousands of hours. We conclude that:

Findings 4: (1) A majority of apps follows a "birth-growth-decline-death" process, as they enter/exit from the bottom part of a list. (2) Most of the apps with modest DEBUT and EXIT values have a short, eventful life occupying many positions quickly, and (3) The *New (Free)* and *New (Paid)* lists choose among apps that were updated within the last 20 days.

B. Top-K List Variation

We now characterize the changes in the rankings of the top-k items from the five lists over time. Changes over time can be explained not only by the dynamic nature of the app uploading process but also by changes in the ranking algorithm.

We use the *Inverse Rank Measure* to assess the changes over time in each of the rankings. This measure gives more weight to identical or near identical rankings among the top ranking items. This measure tries to capture the intuition that identical or near identical rankings among the top items indicate greater similarity between the rankings. Let us assume the following: k_n is the list of top-k apps at time t_n, $\sigma_n(i)$ is the rank of app i in k_n, Z is the set of items common to k_{n-1} and k_n, S is the set of items in k_{n-1} but not in k_n, T is the set of items in k_n but not in k_{n-1}. Then, the inverse rank measure is [5] defined as follows:

$$M^{(k_{n-1},k_n)} = 1 - \frac{N^{(k_{n-1},k_n)}}{Nmax^{(k_{n-1},k_n)}} \tag{2}$$

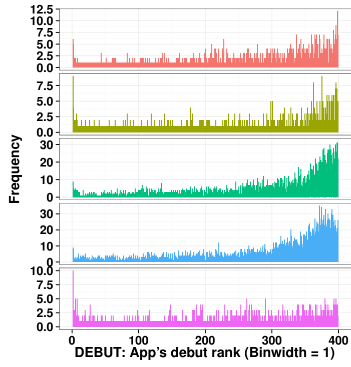
where $N^{(k_{n-1},k_n)} = \sum_{i\in Z} |\frac{1}{\sigma_{n-1}(i)} - \frac{1}{\sigma_n(i)}| + \sum_{i\in S} |\frac{1}{\sigma_{n-1}(i)} - \frac{1}{(|k_n|+1)}| + \sum_{i\in T} |\frac{1}{\sigma_n(i)} - \frac{1}{(|k_{n-1}|+1)}|$, and

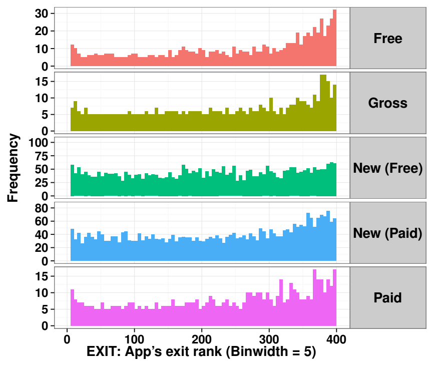
$$Nmax^{(k_{n-1},k_n)} = \sum_{i=1}^{|k_{n-1}|} |\frac{1}{i} - \frac{1}{(|k_n|+1)}| + \sum_{i=1}^{|k_n|} |\frac{1}{i} - \frac{1}{(|k_{n-1}|+1)}|$$

Figure 8(a) shows the variation of $M^{k_{t_1},k_{t_2}}$ for consecutive days in the month of September. Note that values above 0.7 indicate high similarity [5]. We observe that the lists are similar from day to day for *Free* list but this is not the case for *Paid* and *Gross*. Intuitively, this indicates that the effort to displace a free app seems to be higher than that of a paid app or the frequency with which the ranking algorithm is run on *Free* list is less than that of the *Paid* list.

Figure 8(b) shows the number of apps that occupy a rank position in 5 different list-types over our observation period. Note that a *lower rank is preferred*. For example, the 300^{th} rank position in the *New (Free)* list is occupied by 441 applications. With the increase in rank, the rate of applications being swapped is increasing for each category indicating an increased churn – it is easier for apps to occupy as well as get displaced on high ranks. For *Paid*, *Gross*, and *Free*, the number of apps varies from 34 to 142, 30 to 173, and 43 to 163, respectively, from the 1^{st} to the 400^{th} rank.

Figure 8(c) shows the distribution of the lifetime of applications that occupy a specific rank position. To evaluate the variation in the distributions we choose the 1^{st}, 50^{th}, 100^{th}, 200^{th}, and 400^{th} rank positions. For each category, the average lifetime is longer for higher ranking apps then for lower ranking apps. We can clearly observe this phenomenon in the case of *New (Free)* and *New (Paid)*. In both the cases, the lifetime of the apps at the lowest rank (*i.e.,* 400^{th}) is the


(a)


(b)

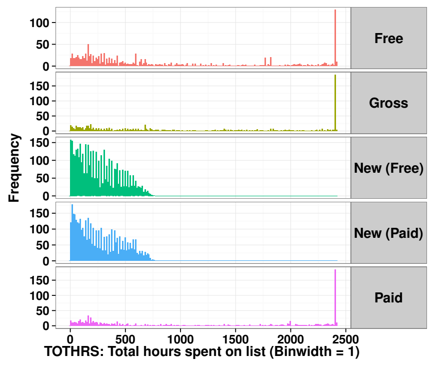
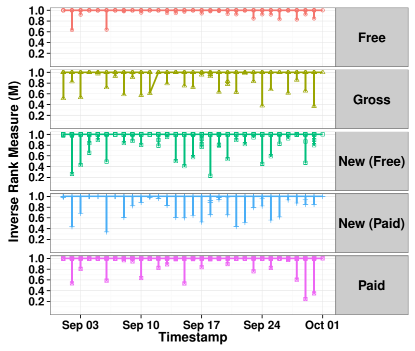
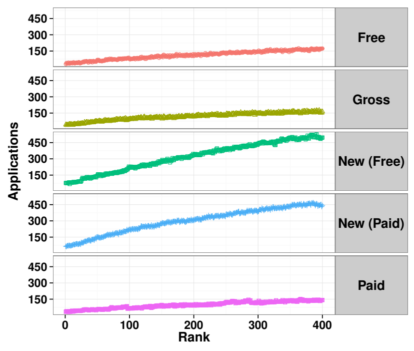

(c)

Fig. 7. Distributions of (a) DEBUT, (b) EXIT and (c) TOTL.HRS measures. The y axis displays the number of apps whose DEBUT and EXIT ranks, as well as the TOTL.HRS correspond to the values displayed on the x axis. Most apps entered and exited from the bottom part of the top-k list. The *New (Free)* and *New (Paid)* lists choose among apps that were updated within the last 20 days.


(a)


(b)

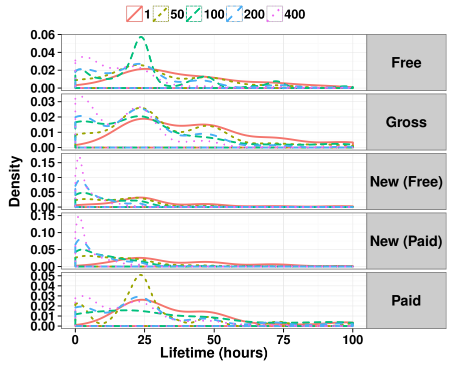

(c)

Fig. 8. (a) The Inverse Rank Measure vs. Timestamp. The *Free* list varies little from day to day, which is not the case for *Paid* and *Gross*. (b) Number of apps vs. ranks. (c) Lifetime of apps at various ranks. The average top-k list lifetime is longer for higher ranking than for lower ranking apps.

lowest, *i.e.*, ≈6 hours and it starts increasing with the increase in the ranks. We can attribute this effect to the frequently changing list of new apps and the relatively easier competitions to be on the top-400 lists. However, in case of *Free* apps, the average lifetime of apps on the 1^{st} rank is 94.2 hours and decreases to 16.7 hours for the 400^{th} rank. For *Paid* and *Gross* categories, the lifetime changes from 81.6 to 20.6 hours and 65.7 to 17.9 hours, respectively, for the rank $1 \rightarrow 400$. We attribute these effects to the stability of the apps in these lists.

Findings 5: Apps that attain higher ranks (e.g., 1-10) have better top-k list "stability" than apps that are at lower ranks (e.g., 300-400). That is, they are allowed to stay longer by a factor of between 3.6 and 5.5.

VII. Research Implications

We discuss now the implications of our work on future directions for security and systems research in app markets.

A. Emerging Threat Vectors

Our analysis has shown that several developers upload many unpopular apps (see §V), while others tend to push frequent updates (§IV). In the following, we expose new vulnerabilities related to such behaviors.

Scam Apps. We have identified several "productive" developers, that upload many similar apps. Among them, we have observed several thousands of premium applications (priced around \$1.99) that are slight variations of each other and have almost no observable functionality. Such apps rely on their names and description to scam users into paying for them, then fail to deliver. Each such app receives ≈500-1000 downloads, bringing its developer a profit of \$1000-2000. A promising research direction is to identify relevant features, including reviews, update frequency, price, and use machine learning tools to detect suspicious scam apps.

Update Attacks. While updates enable developers to fix bugs and push new functionality in a seamless manner, we have identified two attack vectors that rely on them. The attack vectors can be exploited both by malicious developers and by attackers that infiltrate developer accounts.

• **Jekyll-Hyde apps.** A motivated attacker develops and uploads a benign app. Once it gains popularity, the attacker pushes malware as an update. For instance, the attacker can ramp up the permissions required by the app, exploiting the observation that a user is more likely to accept them, then to uninstall the app. One direction to explore is to monitor the evolution of an app, as related to uploads. This includes detecting review and download spikes, fraudulent reviews and sudden rating variations (e.g., see [20]), as well as strong negative reviewer sentiments.

• **DoS attacks.** In §IV we discus the case of apps such as "Player Dreams", whose each update utilizes ≈0.87-1.71 TB of bandwidth (assuming automatic updates on the user side). Amazon EC2 [1] charges \$12K for a bandwidth of 10 TB/month. Attackers can utilize this channel to launch bandwidth and even battery exhaustion attacks, by pushing frequent updates to a large user base.

B. App Market Ecosystem

Analytics-driven Application Development. We envision a development model where insights derived from raw market-level data is integrated into the application development. Such a model is already adopted by websites such as Priceline [4] through their "Name Your Own Price" scheme where the interface provides users with hints on setting an optimal price towards a successful bid. We propose the extension of development tools like Google's Android Studio [2] with market-level analytics, including:
• Median Price: In §IV, we showed that developers may be settling down for lower profits. Provide them with hints on the optimal price for their app based on, say, #features, price of active apps in the same category etc.
• Application Risk: Provide predictions on the impact of permissions and updates on reviews and download count.
• App Insights: Present actionable insights extracted from user reviews (e.g., using solutions like NetSieve [19]), including most requested feature, list of buggy features, features that crash the app.
Enriching User Experience. We believe data-driven insights will be indispensable to enhance the end user experience:
• Analytics Based App Choice: Visualize app price, update overhead, required permissions, reviewer sentiment to enhance the user experience when choosing among apps with similar claimed functionality. For instance, develop scores for individual features, and even an overall "sorting" score based on user preferences. Scam apps (see §VII-A) should appear at the bottom of the score based sorted app list.
• Analytics Based Update Quarantine: We envision a quarantine based approach to defend against Jekyll-Hyde apps. An update installation is postponed until analytics of variation in app features indicates the update is stable and benign. To avoid a situation where all users defer installation, we propose a probabilistic quarantine. Each user can update the app after a personalized random interval after its release.

VIII. Conclusion

In spite of the widespread interest in smartphone app markets, little has been published that reveals the nature of applications, or the developers. This paper is a first attempt to capture temporal patterns in Google Play, an influential app market. Using data collected from more than 160,000 apps daily over a six month period, we examined market trends, application characteristics and developer behavior in real-world market settings. Our work provides insights into the impact of developer levers (e.g., price, permissions requested, update frequency) on app popularity. We proposed future directions for integrating analytics insights into developer and user experiences. We introduced novel attack vectors on app markets and discussed future detection directions.

IX. Acknowledgments

This research was supported in part by DoD grant W911NF-13-1-0142 and NSF grant EAGER-1450619.

References

[1] Amazon EC2. http://goo.gl/4ipMy.
[2] Getting started with Android Studio. http://goo.gl/wgeUok.
[3] Google Play Market. http://play.android.com.
[4] Priceline. http://priceline.com.
[5] J. Bar-Ilan, M. Levene, and A. Lin. Some measures for comparing citation databases. *Journal of Informetrics*, 1(1):26–34, 2007.
[6] Y. Benjamini. Opening the box of a boxplot. *The American Statistician*, 42(4):257–262, 1988.
[7] E. Brynjolfsson and M. Smith. Frictionless commerce? a comparison of internet and conventional retailers. *Management Science*, 2000.
[8] A. Clauset, C. Shalizi, and M. Newman. Power-law distributions in empirical data. *Arxiv preprint arxiv:0706.1062*, 2007.
[9] R. Cleveland, W. Cleveland, J. McRae, and I. Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.
[10] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Procs. of OSDI*, 2010.
[11] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of android application security. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 21–21, Berkeley, CA, USA, 2011. USENIX Association.
[12] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 627–638, New York, NY, USA, 2011. ACM.
[13] Google. Developer Registration. *http://goo.gl/wIwpa*, 2012.
[14] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112, New York, NY, USA, 2012. ACM.
[15] R. Jiang, P. Ji, and X. Xiao. Aging property of unimodal failure rate models. *Reliability Engineering & System Safety*, 79(1):113–116, 2003.
[16] R. Johnson and D. Wichern. *Applied multivariate statistical analysis*. Prentice hall, 2002.
[17] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet mathematics*, 2004.
[18] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. P. Markatos, and T. Karagiannis. Rise of the planet of the apps: A systematic study of the mobile app ecosystem. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 277–290, New York, NY, USA, 2013. ACM.
[19] R. Potharaju, N. Jain, and C. Nita-Rotaru. Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In *Proceedings of USENIX NSDI*, 2013.
[20] M. Rahman, B. Carbunar, J. Ballesteros, G. Burri, and D. H. P. Chau. Turning the tide: Curbing deceptive yelp behaviors. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2014.
[21] M. J. Warrens. On association coefficients for 2x2 tables and properties that do not depend on the marginal distributions. volume 73, 2008.
[22] E. Weber, S. Shafir, and A. Blais. Predicting risk sensitivity in humans and lower animals: risk as variance or coefficient of variation. *Psychological Review; Psychological Review*, 111(2):430, 2004.
[23] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 329–344, New York, NY, USA, 2011. ACM.
[24] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 95–109, Washington, DC, USA, 2012. IEEE Computer Society.
[25] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *Proceedings of the 19th Network and Distributed System Security Symposium (NDSS)*, 2012.