Timely versus Quality Innovation:

The Case of Mobile Applications on iTunes and Google Play

Yongdong Liu*, Denis Nekipelov* and Minjung Park*

[Very incomplete. Please do not circulate]

This version: December 2013

Abstract

iTunes and *Google play* are dominant platforms where users of portable electronic devices with Android and Apple operating systems can purchase and download applications for those devices. The applications ('apps') are developed and brought to the platforms by a large number of independently operating developers. It is a highly competitive dynamic marketplace where it is essential for the developers to keep innovating by both upgrading their existing apps and introducing new apps in order to generate revenues. In this paper we use a unique and comprehensive dataset containing information regarding apps on iTunes and Android platforms. Using a combination of techniques from the Computer Science literature, we were able to identify and validate the complete set of developers that operate on both platforms as well as the same apps that were introduced on both platforms. Using this matched dataset we study how the threat of competitors' entry influences the timing and quality of app entry. In particular, we find that the threat of competitors' entry can have a sizeable negative impact on the quality of an app under development by forcing the developer to introduce the app prematurely before it has been properly tested and debugged. Our reduced form analysis demonstrates varying effects of this phenomenon depending on both the size of the developer and its competitors' and the sparcity of the product space on a given platform. Then we develop and estimate a structural strategic model of timing and quality decisions of the cross-platform app introduction. We use novel techniques from the Machine Learning literature to model the beliefs of developers in our semiparametric two-step estimator. The estimated structural model is then used to analyze the effects of various counterfactual changes, such as an increase in demand for certain app categories, transfers from the platform to the developers as well as 'A+B'-type contracts between the platform and the developers, on the resulting app quality.

Keywords: innovation; competing platforms; strategic entry

*University of California, Berkeley. Generous supports from the National Science Foundation (SES 1024504) and the University of California at Berkeley are acknowledged. The usual disclaimer applies.

1 Introduction

The market for 'apps' – programs that can be run under the operating systems that are installed on portable computer devices such as smartphones and tablets – has become extremely popular over the past few years, and it represents a dramatic showcase of tradeoffs that arise for innovative firms in a highly competitive marketplace. The app market is represented by two dominant platforms that distribute apps for two competing operating systems: Android and Apple. The apps are not directly portable between the two platforms. Without a substantial change in the source code an app developed for Apple cannot run on an Android device.

In this paper we study the decisions of developers to port apps that were successful on one platform to another platform, which we call cross-platform entry. In a competitive marketplace such a decision can be complex. If there are multiple competing developers who have apps that serve similar functions, the effort to introduce a quality app can be undermined by competitors' entry which can divert the user base and reduce the potential profits from entry. In this case, a developer making such a decision faces a tradeoff between finishing the beta-testing to create a quality app at the risk of losing demand to competitors and introducing an app quickly to preempt competitors without adequate testing at the risk of receiving poor user ratings as well as incurring an increased cost of fixing a running app. We provide evidence that the timing and quality decision of entry in the app market is strategic and that the firms making decisions about porting their apps to another platform take into account the actions of their competitors.

In addition to providing a contribution to the literature on the impact of competition on the quality and timing of innovations, our paper also makes contributions to methodological literature on structural estimation of strategic competition models. Our structural model is based on the Bayes-Nash equilibrium where the developers make inferences regarding the app development processes of their competitors. Then, on the basis of this prediction they make entry decision that trades off the quality of the app that they want to introduce on the new platform and the duration of the development process. To provide a statistical procedure that imitates the prediction process for the developers and generate an accurate and parsimonious predictor, we

use a range of model selection tools from the Machine Learning literature to create a statistical model that incorporates all information available to developers in an efficient way. The same tools are also used to model the process by which the developers make predictions regarding the performance of their own apps on a new platform based on their observed performance on the platform of the "first entry". It allows us to detect which characteristics of app performance on one platform are facilitating the prediction of its performance on another platform. We then invoke a pseudo-likelihood approach to estimate the parameters of the structural model based on the first-stage estimates from the described prediction models.

Our analysis sheds a new empirical light on the process of innovation that has been long looked at from the theoretical viewpoint, going back to Schumpeter (1934, 1942). Schumpeter suggests that market power might increase the rents of innovations and thus firm's incentive to innovate, called "Schumpeterian effect" by Dasgupta and Stiglitz (1980). Besides Schumpeterian effect, several other theories also try to explain the relationship between innovation and competition. For instance, the replacement effect by Arrow (1962) posits that innovation would replace monopoly profits with the profits from innovation so that the monopolist's gain from the innovation is only the incremental profits. Therefore, a firm in a monopoly position has less incentive to innovate than a firm in a competitive industry. Gilbert and Newbery (1982) and, later, Aghion et al. (2009) show that the incentive to preempt may be stronger than Arrow's replacement effect. Even the well-established market power of a monopolist may still be challenged by the introduction of new products or cost saving innovation by its competitors. Then the monopolist would have an incentive to innovate in order to forestall competition.

Besides the monotonic relationship between competition and innovation, an inverted-U curve relationship also has a long history (Scherer (1967)). Aghion et al. (2005) show an inverted-U curve relationship between market competition and innovation theoretically and empirically. If a larger equilibrium fraction of sectors involves neck-and-neck competing incumbents, competition tends to induce more innovation because of the dominant escape-competition effect. However, in sectors where innovations are made by laggard firms with low initial profits, competition reduces innovation incentive.

While theories on the relationship between competition and innovation are rich and well developed, empirical findings in this literature are frequently weak. Early empirical work on innovation and market structure mostly uses industry-level data (Levin et al. (1985), Geroski (1995), Nickell (1996), Blundell et al. (1999)). Recent papers start to use micro level data, such as Aghion et al. (2005) and Aghion et al. (2009), but few datasets offer clean empirical evidence regarding innovation. This is because construction of sufficiently large and comprehensive datasets is hard, if not impossible in many cases, partly due to the long cycle of innovation. Our rich and unique dataset allows us to examine how competition affects timing and quality of innovation in the app market, helping this paper fill the gap in the literature.

This paper is organized as follows. In Section 2, we describe the raw data and the methodology for matching datasets from the two platforms. In Section 2, we also provide reduced-form evidence linking the quality of app, the duration between entries into the two platforms, and the intensity of competition. In Section 4.1, we use a class of model selection techniques from the Machine Learning literature to construct a flexible model for expectations of app developers. In Section 3, we estimate a structural model of competing entry of multiple developers across the platforms, using the estimated expectations of developers as an input to the structural model. Section 5 concludes.

2 Data

2.1 Data description

Our data come from AppMonsta Inc., company specializing in collection and analysis of data from the app stores and providing consulting services to new and existing developers. The data comprise all iTunes apps and Android apps from October 2011 to October 2012. They were scraped from iTunes App Store and Google Play App Store on a daily basis, resulting in a daily panel dataset of all iTunes and Android apps. Table 1 lists the variables in our dataset and brief descriptions. App characteristics available on Apple Store and Google Play are slightly different and Table 1 marks the variables unavailable on each platform. Some variables do not change

much over time, such as app name and developer name. Other variables are updated with every new version. App size, version description and required OS are such variables. Also, there are some variables that would change every day, for instance, app price and the number of ratings.

[Table 1 about here.]

Although Google Play and iTunes provide developer names, name cannot be used as the unique developer ID. Instead, information on apps by the same developers enables us to create a unique developer ID for each platform.

Since our goal is to analyze cross-platform entry behavior, it is crucial to identify the set of apps that are present on both platforms. Unfortunately, the same apps could be launched with slightly different names on the two platforms, making the data matching task difficult. In general, the task of re-identifying observations from separate datasets is a routine necessity in many practical applications. In many cases there do not exist perfect cross-database identifiers of individual entries. There could be multiple reasons why that is the case. For instance, there could be errors in data entry and processing, wrong variable formatting, and duplicate data entry. The Computer Science literature has developed an extensive machinery for re-identification of such observations. The idea that has arisen in Newcombe, Kennedy, Axford, and James (1959) and was later formalized in Fellegi and Sunter (1969) was to treat the record linkage problem as a problem of classification of record subsets into matches, non-matches and uncertain cases. This classification is based on defining the similarity metric between two records. Then given the similarity metric one can compute the probability of a particular pair of records being a match or non-match. The classification of pairs is then performed by fixing the probability of erroneous identification of a non-matched pair of records as a match and a matched pair of records as a non-match by minimizing the total proportion of pairs that are uncertain. This matching technique is based on the underlying assumption of randomness of records being broken. As a result, using the sample of perfectly matched records one can recover the distribution of the similarity metric for the matched and unmatched pairs of records.

A large fraction of the further literature was devoted to, on one hand, development of classes of similarity metrics that accommodate non-numeric data and, on the other hand, development of fast and scalable record classification algorithms. For obvious reasons, measuring the similarity of string data turns out to be the most challenging. Edit distance (see, Gusfield (1997) for instance) is a metric that can be used to measure the string similarity. The distance between the two strings is determined as the minimum number of insert, delete and replace operations required to transform one string into another. Another measure developed in Jaro (1989) and elaborated in Winkler (1999) is based on the length of matched strings, the number of common characters and their positions within the string. In its modification it also allows for the prefixes in the names and is mainly intended to link relatively short strings such as individual names. Alternative metrics are based on splitting strings into individual "tokens" that are substrings of a particular length and then analyzing the power of sets of overlapping and non-overlapping tokens. For instance, Jaccard coefficient is based on the relative number of overlapping and overall tokens in two strings. More advanced metrics include the "TF/IDF" metric that is based on the term frequency, or the number of times the term (or token) appears in the document (or string), and the inverse document frequency, or the number of documents containing the given term. The structure of the TF/IDF-based metric construction is outlined in Salton and Harman (2003). The distance measures may include combination of the edit distance and the TF/IDF distance such as a fuzzy match similarity metric described in Chaudhuri, Ganjam, Ganti, and Motwani (2003).

The matching procedure implemented in this paper is as follows.

Step 1 Remove special characters, symbols and punctuations from app names and developer names. Strip off redundant whitespaces and standardize all characters to lower cases, if applicable.

Step 2

Four similarity metrics are calculated for any pair of app names and any pair of developer names, respectively. These four similarity measures are exact match, Levenshtein distance, partial string

Exact match simply tests whether two strings are identical. Levenshtein distance is one of the most common edit distances used for text matching. Partial string similarity measures the local similarity of two strings. For example, the Levenshtein distance between "accurate weather" and "accurate weather app" is 0.88. However, the string "accurate weather" could be perfectly matched with the substring "accurate weather" extracted from "accurate weather app", so partial string similarity between these two strings is 1.00. Token sort score is an improved or relaxed partial string similarity. This method first tokenizes two strings, sorts these tokens and then implements partial string similarity. The partial string similarity between "angry birds" and "angry cute birds" is only 0.54, but the token sort score returns 1.00.

Exact match is the strictest criterion and token sort score is the weakest one. The weaker the matching criterion is, the higher matching score it returns for a given pair. Usually token sort set gives the highest matching score. Applying these four similarity metrics all together ensures a very low probability of failing to match a pair which is supposed to be matched, but meanwhile mismatching becomes more likely. Also a high matching score under weak criterion cannot overturn a very low score under stricter criterion, since in real data this case usually implies a mismatch. Furthermore, computation is more burdensome for more flexible and weaker matching procedures. Since almost one trillion pairs need to be compared in our dataset, we need to avoid unnecessary computations in practical implementation. Therefore, the easiest exact match is calculated first. If a pair is matched exactly, other distances are skipped. Otherwise, this pair is passed for Levenshtein distance. If the Levenshtein is lower than certain threshold, other two similarities are skipped too. The most computational burdensome partial string similarity and token sort set are calculated only if a pair fails to pass the first two algorithms.

Step 3

Pick out pairs of apps whose names as well as their developer names from the two platforms are exactly matched. They are identified as cross-platform apps. The remaining sample is passed to Step 4.

Separate the remaining sample into three groups: (1) only developer names exactly matched, (2) only app names exactly matched, (3) neither exactly matched. To identify cross-platform apps, three different criteria on similarity scores are set for these three groups.

Suppose that type I error is that two apps are the same, but the algorithm fails to match them and type II error is that two apps are not the same, but are matched. Obviously reducing type I error will increase the occurrence of type II error. The screening criteria were tuned on a small training sample to balance these two errors. In the end, several testing samples are randomly drawn from the final outcome and the matching quality is checked manually. Type I error is less than 20% and type II error is less than 5%.

These four steps are graphically summarized in Figure 1.

[Figure 1 about here.]

We provide some summary statistics on the raw data as well as the matched sample. Except for those that involve dynamic changes, all other summary statistics are snapshots of data as of October 2012. Table 2 and Table 3 show the app distribution across genres in Android and and iTunes respectively. "Utilities & Tools", "Games" and "Entertainment" account for more than 40% of Android apps. In iTunes app store, there are fewer "Utilities & Tools" apps, but more "Books & Reference", "Education" and "Lifestyle" apps. The third column in Table 2 and Table 3 reports the proportion of cross-platform apps in each genre. 11.4% Android apps are cross-platform apps and that number is only 7.3% in iTunes. Using informaton on app launch date, we could identify the cross-platform entry order. The release date of some Android apps is missing in our data, and entry orders cannot be identified for those cases. In the sample with accurately identified release date, the number of cross-platform apps that were launched on Android first is twice the number of cross-platform apps launched on iTunes first. In terms of absolute number, "Game", "Entertainment", "Business", "Lifestyle" and "Education" have more cross-platform apps than other genres, but "Business", "Finance", "Social Networking",

"Medical" and "Travel" have higher proportion of cross-platform apps.

[Table 2 about here.]

[Table 3 about here.]

Table 4 and Table 5 report four major app characteristics in each genre. The number of ratings refers to the total number of ratings received by an app by the end of data period. Without controlling for app tenure, app quality measured by this variable is misleading, since older apps tend to accumulate more ratings. Therefore, the weekly incremental number of ratings is provided as a standardized measure. Weekly incremental number of ratings reflects how many ratings are received by an app on weekly average. Higher quality apps usually get more ratings and therefore they should have more weekly incremental ratings. We also report monthly major update frequency and monthly minor update frequency as other dimensions of app quality. Major update is usually related to some major changes or improvements, such as addition of another function or new interface. Minor update is mainly bug fixing. It is plausible to expect that apps with higher quality are less likely to be updated or fixed. Therefore, we interpret more frequent updates, major or minor, as an indication of lower quality. In general, "Game", "Social Networking" and "Entertainment" apps harvest more ratings and accumulate ratings more quickly. However, apps in these genres do not necessarily update more often than apps in other genres.

[Table 4 about here.]

[Table 5 about here.]

The competition level in app market is measured by Herfindahl index. Since the number of downloads is unobservable, the cumulative number of ratings is used to approximate the number of downloads. An app's market share is the ratio of the number of its ratings to the total number

of ratings in that genre on the chosen platform. Each genre is treated as an independent market and the Herfindahl index is calculated by each genre. Figure 2 shows the Herfindahl index in two platforms during our data period. The blue line represents Android and the red one is HHI of iTunes. In most genres, the competition level of Android app store decreases over time. On the contrary, the competition level of iTunes app store remains relatively stable over timel, although HHIs in some genres are bumpier than others.

[Figure 2 about here.]

2.2 Reduced-form evidence

In this section, we provide some preliminary evidence that developers port their apps to the second platform hastily at the expense of app quality when their competitors enter the market. In particular, we focus on cross-platform entries following cross-platform entry by top developers, where top developers are defined as those whose number of apps is among top 50 or 100 in the particular genre. Suppose that an Android app by a top 50 developer enters iTunes at date t. Then the occurrence of entering iTunes by other apps that were previously offered only on Android is recorded. We separate these followers into top developers and non-top developers. We plot the frequency of cross-platform entries within a week, two weeks, and four weeks after cross-platform entry of an app by a top developer.

[Figure 3 about here.]

[Figure 4 about here.]

Figure 3 shows the kernel density of cross-platform entry following cross-platform entries by top Android developers. Figure 4 shows a similar kernel density of iTunes apps. Top panels show the kernel density of entries of non-top developers and bottom panels show the kernel density of entries of top developers. The distribution of the number of followers is very skewed to the right. The distribution of following entries by non-top developers is slightly more even and the descending trend from the peak to the right tail gets less steep.

[Table 6 about here.]

[Table 7 about here.]

To compare the quality of followers, we calculate the deviation of mean and median of followers from the genre mean and median. As before, app quality is measured on three aspects, weekly incremental number of ratings, monthly major updates and monthly minor updates. Table 6 and Table 7 list the difference between mean (median) of followers and genre mean (median). Columns I, II and III are weekly incremental number of ratings, major updates frequency and minor updates frequency, respectively. Regardless of whether followers are apps by other top developers or apps by non-top developers, their quality is inferior to the genre average and median level.

[Figure 5 about here.]

[Figure 6 about here.]

[Figure 7 about here.]

[Figure 8 about here.]

[Figure 9 about here.]

[Figure 10 about here.]

[Figure 11 about here.]

[Figure 12 about here.]

[Figure 13 about here.]

[Figure 14 about here.]

[Figure 15 about here.]

[Figure 16 about here.]

Next we want to see the relationship between app quality and entry duration. First, we run the regression below

$$d_{ij} = c + a_1 HHI_{itunes} + a_2 HHI_{android} + a_3 q_{ij} + \zeta_j + \epsilon_{ij}$$

Subscript i represents app and j refers to developer. d_{ij} is the duration (in days) between entries of top developers and entries of followers. HHI_{itunes} and $HHI_{android}$ are genre specific Herfindahl Indices in iTunes and Android, respectively. q_{ij} is the quality measure of app i. In the regression above, we also control for the developer fixed effect, ζ_j. We estimate this equation on each genre and calculate

$$\tilde{d}_{ij} = \hat{c} + \hat{\zeta}_j$$

$\tilde{d}_{ij}$ is the duration controlling for the Herfindahl index and the app quality.

Figure 5 to Figure 16 plot the relationship between entry duration and app quality. Figure 5 to Figure 10 are for Android apps and Figure 11 to Figure 16 plot iTunes apps. For each genre, there are two graphs. The first graph shows the relationship between duration and app quality from the raw data. The third graph shows the same relationship where duration is calculated from the regression as discussed above. The graphs with "uncontrolled" duration show no clear pattern, but graphs using "controlled" duration show that followers entering at a later time have higher qualities than hasty followers.

3.1 Theoretical setup

We now consider a structural model of the cross-platform entry behavior of developers. According to our reduced-form evidence, we assume that the developers have a type space $\mathcal{T} = \{L, S\}$ corresponding to the large and small developers. We assume that the entire set of developers is fixed with $d^L = 1, \ldots, D^L$ indexing developers of type L and $d^S = 1, \ldots, D^S$ indexing the developers of type S. This entire set of players is considered to be common knowledge to all participating developers. The platforms are indexed $\mathcal{P} = \{A, G\}$ where A stands for Apple iTunes and G stands for Google play. We assume that there is an exogenous two-dimensional stationary Poisson process $(\xi_A^{d^\tau,\tau}(t), \xi_G^{d^\tau,\tau}(t))$ with values in $\{0,1\} \times \{0,1\}$ (where 1 stands for entry) which characterizes the "first entry" of developer of type τ on one of the platforms. To emphasize that this is the process of the first entry, we assume that, first:

$$P\left(\xi_P^{d^\tau,\tau}(t+\Delta) = 0 \,|\, \xi_A^{d^\tau,\tau}(t) + \xi_G^{d^\tau,\tau}(t) > 0\right) = 0, \quad P \in \{A, G\}$$

for any $\Delta > 0$. We also assume that the instantaneous Markov kernel for the transition of this process is fixed, meaning that

$$\lim_{\Delta \to 0} \frac{1}{\Delta} P\left(\xi_P^{d^\tau,\tau}(t+\Delta) = 1 \,|\, \xi_A^{d^\tau,\tau}(t) + \xi_G^{d^\tau,\tau}(t) = 0\right) = \delta_P^\tau, \quad P \in \{A, G\}.$$

This characterization also allows for the first entry to be simultaneous to both platforms. The process $(\xi_A^{d^\tau,\tau}(t), \xi_G^{d^\tau,\tau}(t))$ can be considered as a reduced-form interpretation of the "first entry" decisions by the developers. The full structural characterization of this process can be found in Liu (2014).

Consider a continuous-time game of cross-platform entry at time t. Now we focus on the decision of each developer at the instant $t + \Delta$, assuming that Δ is infinitesimal. Note that a developer will be eligible to enter into the new platform at time $t + \Delta$ if (i) this developer already entered into one of the platforms by $t + \Delta$; (ii) this developer has not entered into both platforms at the

Now consider the structure of the state space. We assume that each developer observes the set of her competitors (along with their individual characteristics and preferences) and their entry status in both platforms $\{z_P^{d^\tau,\tau}\}_{d^\tau=1}^{D^\tau}$ with $\tau \in \{L, S\}$ and $P \in \{A, G\}$ such that $z_P^{d^\tau,\tau} = 1$ means that the developer d^τ of type τ has entered on the platform P. We use shorthand notation $z_{P,t} = (z_P^{d^\tau,\tau},\, d^\tau = 1, \ldots, D^\tau)$ and $z_t = (z_{A,t}, z_{G,t})$. There is also a commonly observed real-valued state variable s_t with values in the set $\mathcal{S} \subset \mathbb{R}^k$ driven by the stochastic differential equation

$$s_{t+\Delta} - s_t = \int_t^{t+\Delta} \left(\bar{s}\, dr + \Sigma\, dB_s(r)\right),$$

where $\bar{s}$ is a k-dimensional vector corresponding to the drift in s_t and $B_s(\cdot)$ is a k-dimensional Brownian motion.

Each developer also has a random cost of development. The total development cost from time t to time $t + \Delta$ (provided that the developer did not enter to the second platform) is determined by two components. The first component $c^{d^\tau,\tau}(s_t)$ is a deterministic function of the state, and a stochastic component determined by a Brownian motion process $B^{d^\tau,\tau}$ which is independent from such a process for other developers. This process reflects the stochastic transition of development cost for each developer over time. Then if $C^{d^\tau,\tau}(s_t, t)$ is the total development cost since the first entry, then we can define

$$C^{d^\tau,\tau}(s_{t+\Delta}, t+\Delta) - C^{d^\tau,\tau}(s_t, t) = \int_t^{t+\Delta} e^{-\rho\, r} \left(c^{d^\tau,\tau}(s_r)\, dr + \sigma^{d^\tau,\tau}\, dB^{d^\tau,\tau}(r)\right),$$

where ρ is the discount factor. We assume that $C^{d^\tau,\tau}(s_t, t)$ is commonly observed among developers, while $B^{d^\tau,\tau}(t)$ is private information of developer (d^τ, τ).

We impose a restriction on the action space of developers. In particular, we assume that those developers who are eligible to enter on a platform instead of making a deterministic decision to enter, choose the intensity of the Poisson process whose first occurence would correspond to the entry of a given developer. In particular then for each developer d^τ such that $\xi_A^{d^\tau,\tau}(t') + \xi_G^{d^\tau,\tau}(t') =$

0 for $t \leq t' < t + \Delta$. The action space is $\mathcal{A}^{d^\tau}(t + \Delta) = \emptyset$, meaning that the developers who have not done the first entry cannot do the cross-platform entry. For each developer d^τ such that $\xi_A^{d^\tau,\tau}(t') + \xi_G^{d^\tau,\tau}(t') = 2$ for $t \leq t < t + \Delta$, the action space is $\mathcal{A}^{d^\tau}(t + \Delta) = \emptyset$, meaning that the developers who have entered into both markets upon the first entry cannot do the cross-platform entry. Finally, for each developer d^τ such that $\xi_A^{d^\tau,\tau}(t') + \xi_G^{d^\tau,\tau}(t') = 1$, the action space is a set $\Lambda^{d^\tau,\tau}$ of functions (corresponding to the frequency of the Poisson entry process) $\lambda^{d^\tau,\tau} : \{0,1\}^{D^L \times D^S} \times \mathcal{S} \times \mathbb{R}_+ \mapsto \mathbb{R}_+$ such that $\Lambda^{d^\tau,\tau}$ are differentiable in $\mathcal{S}$ and in time t, strictly bounded by a universal constant $\bar{\Lambda}$ and strictly decreasing in t.

Without loss of generality, consider developer (d_τ, τ) on platform A. The discounted cost of testing the app is determined by $C^{d^\tau,\tau}(s_t, t)$. The payoff from the cross-platform entry is characterized by the potential profit $\Pi^{d^\tau,\tau}(s_t, T^*)$ where T^* is the interval of time between the first entry and the entry on another platform. We assume that $\Pi^{d^\tau,\tau}(s_t, \cdot)$ is a strictly increasing function. This assumption reflects the empirical fact that the apps that are thoroughly developed require less "on the fly" maintainance. The future payoff also has a component $\gamma^{d^\tau,\tau} \sum_{\tau' \in \{L,S\}} \sum_{d_{\tau'}=1}^{D_{\tau'}} z_{At}^{d^{\tau'},\tau'}$ which reflecs the loss of profit due to demand diversion to the products introduced by competing developers.

3.2 The structure of best responses

Suppose that developer d^τ enters the first platform (A) at time 0. Suppose that the developer does not enter in the time interval $[t, t+\Delta)$. Then at $t+\Delta$ this developer accumulates the utility

$$U_{NE}^{d^\tau,\tau}(\Delta, s_t, t) = -(C^{d^\tau,\tau}(s_{t+\Delta}, t + \Delta) - C^{d^\tau,\tau}(s_t, t)).$$

If then an entry occurs at instant $t + \Delta$, the corresponding flow payoff of the developer is

$$V_E^{d^\tau,\tau}(z_{t+\Delta}, s_{t+\Delta}, t + \Delta) = \Pi^{d^\tau,\tau}(s_{t+\Delta}, t + \Delta) - \gamma^{d^\tau,\tau} \sum_{\tau' \in \{L,S\}} \sum_{d_{\tau'}=1}^{D_{\tau'}} z_{A,t+\Delta}^{d^{\tau'},\tau'}.$$

Each developer then forms beliefs regarding the choice of entry intensity by her eligible competitors denote these beliefs $\lambda_*^{d^\tau,\tau}(z,s,t)$.

Now the best response of the developer characterizes the intensity of the process determining the entry into a platform, given the current state variable and the belief of this developer regarding the intensity choices of other developers.

To characterize the best response, consider instant t where the information set of developer (d^τ,τ) who entered on the Apple platform at time 0 is characterized by vector $z_{A,t}$ of the entry statuses of all her competitors and the state variable s_t. We note that provided that the time interval Δ is infinitesimal, in the interval $[t,t+\Delta)$ with probability proportional to Δ (up to a smaller order term) there will be only one more entrant. To see this, note that provided that entry is driven by a Poisson process, the probability of one entry is proportional to the intensity of this process times Δ. The probability of two entries will then be proportional to Δ^2 which is of smaller order of magnitude.

Suppose that developer $(d^{\tau'},\tau')$ is such new entrant. If this entrant has not entered before on either platform, meaning that $z_{P,t}^{d^{\tau'},\tau'}=0$ for $P\in\{A,G\}$, then the probability that this developer will enter on Apple platform is $\delta_A^{\tau'}\Delta$. If this developer is already on both platforms, then $z_{P,t}^{d^{\tau'},\tau'}=1$ and entry will not occur. If this developer has andered Google platform but not Android, meaning that $z_{G,t}^{d^{\tau'},\tau'}=1$ and $z_{A,t}^{d^{\tau'},\tau'}=0$, then this developer will enter with probability $\lambda_*^{d^{\tau'},\tau'}(z_t,s_t,t)\Delta$. Thus, the combined entry probability can be expressed as

$$(1-z_{A,t}^{d^{\tau'},\tau'})(1-z_{G,t}^{d^{\tau'},\tau'})\delta_A^{\tau'}\Delta+z_{G,t}^{d^{\tau'},\tau'}(1-z_{A,t}^{d^{\tau'},\tau'})\lambda_*^{d^{\tau'},\tau'}(z_t,s_t,t)\Delta.$$

By $V_{NE}^{d^\tau,\tau}(z_t,s_t,t)$ denote the continuation value from not entering into the platform. Now we propose a heuristic argument for derivation of the analog of the Bellman equation for this case (usually referred to as the Hamilton-Jacobi-Bellman equation).

Suppose that Δ is small so that $\Delta^2\ll\Delta$. In this case with probability approximately $1-\lambda^{d^\tau,\tau}(z_t,s_t,t)\Delta$ developer (d^τ,τ) does not enter and thus obtains overall payoff equal to the sum

of utility $U_{NE}^{d^\tau,\tau}(\Delta, s_t, t)$ plus the discounted value from the future payoffs

$$(1-\rho\,\Delta)V_{NE}^{d^\tau,\tau}(z_{t+\Delta}, s_{t+\Delta}, t+\Delta)$$

We then represent the component of expected payoff from not entering as

$$\begin{aligned}
&(1-\lambda^{d^\tau,\tau}(z_t,s_t,t)\Delta)\Big(U_{NE}^{d^\tau,\tau}(\Delta,s_t,t)+(1-\rho\,\Delta)V_{NE}^{d^\tau,\tau}(z_{t+\Delta},s_{t+\Delta},t+\Delta)\Big)\\
&=c^{d^\tau,\tau}(s_t)\,\Delta+\sigma^{d^\tau,\tau}\left(B_s(t+\Delta)-B_s(t)\right)+(1-\rho\,\Delta-\lambda^{d^\tau,\tau}(z_t,s_t,t)\Delta)\Big(V_{NE}^{d^\tau,\tau}(z_t,s_t,t)\\
&+\sum_{\tau\in\{L,S\}}\sum_{d^\tau=1}^{D^\tau}(1-z_{A,t}^{d^{\tau'},\tau'})\left[(1-z_{G,t}^{d^{\tau'},\tau'})\delta_A^{\tau'}+z_{G,t}^{d^{\tau'},\tau'}\lambda_*^{d^{\tau'},\tau'}(z_t,s_t,t)\right]\\
&\quad\times\Big(V_{NE}^{d^\tau,\tau}(z_t+e(d^{\tau'},\tau'),s_t,t)-V_{NE}^{d^\tau,\tau}(z_t,s_t,t)\Big)\,\Delta\\
&+\frac{\partial V_{NE}^{d^\tau,\tau}(z_t,s_t,t)}{\partial s}\left(\bar{s}\Delta+\Sigma\left(B_s(t+\Delta)-B_s(t)\right)\right)\\
&+\frac{1}{2}(B_s(t+\Delta)-B_s(t))'\Sigma'\frac{\partial^2 V_{NE}^{d^\tau,\tau}(z_t,s_t,t)}{\partial s\partial s'}\Sigma(B_s(t+\Delta)-B_s(t))\\
&+\frac{\partial V_{NE}^{d^\tau,\tau}(z_t,s_t,t)}{\partial t}\Delta,
\end{aligned}$$

where $e(d^{\tau'},\tau')$ is a $(D^L+D^S)\times 1$ vector whose entires are zero except for $(d^{\tau'},\tau')$ which is equal to one. Now we take expectation with respect to the information at time t, and eliminate the terms of order of magnitude lower than Δ. We also recall that by the property of the Brownian motion, $E[(B(t+\Delta)-B(t))^2|\mathcal{B}_t]=\Delta^2$. This leaves the expression

$$\begin{aligned}
&V_{NE}^{d^\tau,\tau}(z_t,s_t,t)+\Delta\Bigg(\frac{\partial V_{NE}^{d^\tau,\tau}(z_t,s_t,t)}{\partial t}+\frac{\partial V_{NE}^{d^\tau,\tau}(z_t,s_t,t)}{\partial s}\bar{s}+\ell'\Sigma'\frac{\partial^2 V_{NE}^{d^\tau,\tau}(z_t,s_t,t)}{\partial s\partial s'}\Sigma\ell\\
&+\sum_{\tau\in\{L,S\}}\sum_{d^\tau=1}^{D^\tau}(1-z_{A,t}^{d^{\tau'},\tau'})\left[(1-z_{G,t}^{d^{\tau'},\tau'})\delta_A^{\tau'}+z_{G,t}^{d^{\tau'},\tau'}\lambda_*^{d^{\tau'},\tau'}(z_t,s_t,t)\right]\\
&\quad\times\Big(V_{NE}^{d^\tau,\tau}(z_t+e(d^{\tau'},\tau'),s_t,t)-V_{NE}^{d^\tau,\tau}(z_t,s_t,t)\Big)\\
&-(\rho+\lambda^{d^\tau,\tau}(z_t,s_t,t))V_{NE}^{d^\tau,\tau}(z_t,s_t,t)+c^{d^\tau,\tau}(s_t)\Bigg).
\end{aligned}$$

Next, considering the situation where the cross-platform entry occurs in the interval $[t, t+\Delta)$, we note that up to the terms of the smaller order of magnitude, the expected value can be computed as

$$\lambda^{d^\tau,\tau}(z_t, s_t, t)\Delta V_E^{d^\tau,\tau}(z_t, s_t, t).$$

Now recall that the Bellman equation implies that

$$V_{NE}^{d^\tau,\tau}(z_t, s_t, t) = \sup_{\lambda^{d^\tau,\tau}(\cdot,\cdot,\cdot)} E\Bigg[(1-\lambda^{d^\tau,\tau}(z_t, s_t, t)\Delta)\Bigg(U_{NE}^{d^\tau,\tau}(\Delta, s_t, t) + (1-\rho\,\Delta)V_{NE}^{d^\tau,\tau}(z_{t+\Delta}, s_{t+\Delta}, t+\Delta)\Bigg) + \lambda^{d^\tau,\tau}(z_t, s_t, t)\Delta V_E^{d^\tau,\tau}(z_{t+\Delta}, s_{t+\Delta}, t+\Delta) \,\Big|\, z_t, s_t\Bigg].$$

Combining with our previous results, we can provide the following theorem.

THEOREM 1. *The best response of the developer in the continuous-time cross-entry model is characterized by the law of motion of the value function is characterized by the following partial differential equation*

$$\begin{aligned}
&\frac{\partial V_{NE}^{d^\tau,\tau}(z_t, s_t, t)}{\partial t} + \frac{\partial V_{NE}^{d^\tau,\tau}(z_t, s_t, t)}{\partial s}\bar{s} + \frac{1}{2}\ell'\Sigma'\frac{\partial^2 V_{NE}^{d^\tau,\tau}(z_t, s_t, t)}{\partial s \partial s'}\Sigma\ell \\
&+ \sum_{\tau\in\{L,S\}}\sum_{d^\tau=1}^{D^\tau}(1 - z_{A,t}^{d^{\tau'},\tau'})\left[(1 - z_{G,t}^{d^{\tau'},\tau'})\delta_A^{\tau'} + z_{G,t}^{d^{\tau'},\tau'}\lambda_*^{d^{\tau'},\tau'}(z_t, s_t, t)\right] \\
&\quad\times\left(V_{NE}^{d^\tau,\tau}(z_t + e(d^{\tau'}, \tau'), s_t, t) - V_{NE}^{d^\tau,\tau}(z_t, s_t, t)\right) \\
&- \rho\, V_{NE}^{d^\tau,\tau}(z_t, s_t, t) + \lambda^{d^\tau,\tau}(z_t, s_t, t)(V_{NE}^{d^\tau,\tau}(z_t, s_t, t) - V_{NE}^{d^\tau,\tau}(z_t, s_t, t)) + c^{d^\tau,\tau}(s_t) = 0.
\end{aligned}$$

Equations of the form displayed in the preceeding theorem are called heat equations with delay. In general, they don't have a closed form solution and needs to be solved numerically. Its form, however, leads to the following simple structure of the optimal choice of intensity process $\lambda^{d^\tau,\tau}(z_t, s_t, t)$:

$$\lambda^{d^\tau,\tau}(z_t, s_t, t) = \begin{cases} 0, \text{ if } V_{NE}^{d^\tau,\tau}(z_t, s_t, t) < V_{NE}^{d^\tau,\tau}(z_t, s_t, t), \\ \bar{\Lambda}, \text{ if } V_{NE}^{d^\tau,\tau}(z_t, s_t, t) \geq V_{NE}^{d^\tau,\tau}(z_t, s_t, t). \end{cases}$$

Thus, if $\bar{\Lambda}$ is sufficiently large, then the duration between the first entry on the Apple platform

and the second entry into the Google platform is determined by the minimunm time t when inequality

$$V_{NE}^{d^\tau,\tau}(z_t, s_t, t) \geq V_{NE}^{d^\tau,\tau}(z_t, s_t, t)$$

is valid. To implement this inequality, we recall that function $V_{NE}^{d^\tau,\tau}(z_t, s_t, t)$ depends on (i) the own cost structure; (ii) the beliefs of the developer regarding other developers. We can implement this structural model by considering a two-step strategy where in the first step we estimate the belief of each developer regarding the entry of her competitors. Then we can use those beliefs as an input into the duration model defined by the optimal decision of the developer.

4 Estimation of the structural model

4.1 First step: Modeling developers' expectations with high-dimensional data

When the market is large, then the beliefs of developers are defined over a very high-dimensional object (z_t, s_t). Provided that even with as large a dataset as ours, it becomes impossible to manualy select the model that will best approximate the true beliefs of developers. To do that, we employ the methods for reducing dimensionality of the estimated object.

This section ais at providing the results allowing us to consistently estimate the beliefs of each developer (d_τ, τ), $\tau \in \{L, S\}$ from the observed state variables as well as the observed durations between the entires in two platforms. Provided that, according to our model the developer will use a cutoff strategy for entry, we will able to fully capture the function $\lambda_*^{(d_\tau,\tau)}(z_t, s_t, t)$ from observed durations.

Developers make inferences regarding the app development processes of their competitors, and in order to model how developers' cross-platform entry decision is influenced by their expectations about competitors, we need to recover developers' beliefs. One practical challenge for a researcher in doing so is the high-dimensionality of data. With a very large number of variables that could be potentially used in forming beliefs, it becomes crucial to make use of techniques that allow the researcher to deal with the high dimensional data in a computatioanlly tractable way. This

The essence of model selection is to select a parsimonious set of variables which explain data variation best. The more variables a model includes, the better fitting performance one could get. If fitting power is the only goal, then the model should include all possible variables. Usually overfitting occurs in this case, which means the model fits data patterns as well as data noises. Obviously, an overfitted model has bad performance on prediction. Also, in many cases a researcher needs to know the importance of variables, so model selection should balance the model fitting power and model size. Some statistical criteria were designed to address this issue by penalizing the number of variables. The most common one is adjusted R-square. Akaike information criterion (AIC), Bayesian information criterion (BIC) and Mallows's C_p are also widely used in model selection. Given one of these criteria, another issue in model selection is how to find the best model. For example, the number of any possible subsets of 20 variables is $2^{20} - 1$, which is 1048575. Computationally, it is a huge burden to estimate and assess such a large number of model specifications. Some searching algorithms were developed to address this problem. Backward elimination and forward selection are two most common selection algorithms. Branch & Bound selection implements an exhaustive search on all subsets.

Some up-to-date model selection techniques include regression tree, random forest and Lasso. Regression tree is similar to a decision tree which partitions data into several sub-areas. Each sub-area predicts one outcome. Random forest grows many regression trees randomly and this randomness guarantees good statistical properties of random forest. Lasso places constraints on the L^1-norm of parameter vector, which is an alternative regularized version of OLS.

Selection Criteria

The error sum of squares SSR always decreases when another variable is added. Thus, the adjusted R-square penalizes SSR and total sum of square SST by dividing by their degrees of freedom.

$$R^2_{adj} = 1 - \frac{SST/(n-K)}{SSR/(n-1)}$$

AIC stems from the information entropy. Suppose the true data generating process is $f(x|\theta)$, which is unknown, and a MLE $f(x|\hat{\theta})$ is built to approximate the true data generating process. The information loss is evaluated by Kullback-Leibler divergence $D_{KL}(f||g) = E_{x|\theta}(-2lnf(f(x|\hat{\theta}))- E_{x|\theta}(-2lnf(f(x|\theta))$. The better approximation of $f(x|\theta)$ is, the smaller this divergence is. However, $f(x|\theta)$ is unknown and calculation of Kullback-Leibler divergence is not feasible. $E_{x|\theta}(-2lnf(f(x|\theta))$ is not a function of $\hat{\theta}$, so $D_{KL}(f||g)$ is solely determined by $E_{x|\theta}(-2lnf(f(x|\hat{\theta}))$. Still, since the true data generating process $f(x|\theta)$ is not known, it is unfeasible to calculate and compare this expectation directly. Akaike showed that $AIC = -2L(x|\hat{\theta}) + 2k$ is an asymptotically unbiased estimator of $E_{x|\theta}(-2lnf(f(x|\hat{\theta}))$. The formula of AIC shows that this criterion rewards model's fitting power on the likelihood and penalizes the number of variables.

Bayesian information criterion has a similar equation to AIC, $BIC = -2L(x|\hat{\theta}) + kln(n)$. Compared to AIC, BIC penalizes model size more stringently. As a result, BIC tends to select a smaller model. BIC could be explained from the perspective of Bayesian statistics. Suppose there are several candidate models and a discrete prior over these models. It is easy to get the posterior of these models and the most likely model should have the largest posterior. It has been proven that maximizing the posterior is equivalent to minimizing BIC.

Mallows's C_p is mainly used for the model selection of ordinary least squares. The C_p statistic is defined as $C_p = \frac{SSE_p}{S^2} - N + 2p$. Suppose there are k variables in total and s^2 is the mean square error of the full model. SSE_p is the error sum of squares for the subset model containing p variables including intercept. If the subset model is unbiased, it is not hard to prove that in OLS $\frac{SSE_p}{S^2} \approx N - p$ and $C_p \approx p$. In practice, C_p is usually plotted against p for various subset models and the model whose C_p is closest to p is selected.

Search Algorithms

Forward selection starts from an empty subset and each step only selects one variable based

on some criterion. Forward selection stops when none of remaining variables could improve the selection criterion. For a model with k candidate variables, forward selection only needs to evaluate at most $k(k+1)/2$ models and the computation cost is trivial. Forward selection cannot find the global best model, unless candidate variables are all orthogonal to each other.

Backward selection is a reverse of forward selection. Starting with a model including all variables, backward selection removes one variable at each step and the deletion stops when removal of any remaining variables cannot result in a better selection criterion. Backward selection shares some similar properties with forward selection. They both are computationally tractable and neither guarantees the global best model. However, when variables are not independent with each other, backward selection is more favorable. Forward selection usually fails to select the set of variables that fit model better together, but backward selection would keep such variables. When the number of variables is close to the number of observations, selection criterion becomes sensitive to model changes and backward selection may not be suitable for such data.

Stepwise selection adds one variable at each step and meanwhile checks if any variable could be removed. Stepwise selection stops when no removal or addition could improve the selection criterion. The computation cost of stepwise selection is slightly higher than forward selection and backward selection.

Branch & Bound algorithm is a top-down search algorithm which guarantees the global optimal selection. However, the computation is costly especially for high dimensional problems. Branch & Bound search depends on a dynamic search tree. The search starts from the root which includes all variables. Branch & Bound grows the search tree by removing one variable and reaches a node. At each node, the selection criterion is evaluated and compared with the bound which is the best selection criterion achieved so far. If the selection criterion of this branch is no better than the bound, this branch is pruned. Otherwise, the bound is updated. Search stops after a complete research-tree is grown.

Regression Tree and Random Forest

Regression tree is a data prediction model characterized by a decision tree. The root of the

tree represents the entire data and each node shows how to partition data. The partition at the node continues until a leaf is reached and each leaf represents a certain outcome. Therefore, this recursive partition along the regression tree separates the data into several sub-regions, which corresponds to some predicted outcome. The regression tree involves two important questions. First, how to partition data at each node. Second, when to stop growing the regression tree. At each node, the partition variable and the split value are found through an exhaustive search according to some measures on node impurity. The regression tree splits first by the variable that reduces the node impurity most. Therefore, the importance of variables can also be revealed by the hierarchical relations in regression tree.

The regression tree stops splitting when the node impurity reduced by an extra node is lower than some threshold γ. For instance, it is convenient to measure the node impurity by $I = \sum_{l \in T} n_l V_c$. T is the regression tree and l is a leaf in tree T. n_l is the number of observations in leaf l and V_l is the outcome variance within leaf l. However, this stopping rule may result in a smaller tree than what it is supposed to. This is because splitting on some variable may not reduce I very much, but such splitting may reduce I a lot at subsequent nodes. Therefore, a more popular treatment is to grow the largest tree using training data and then prune this tree by cross-validation on the testing data.

Random forest is a classification and prediction framework based on the regression tree. The randomness injected into the ordinary regression tree includes the random sample to train the tree and the variables randomly selected to split the tree. The random forest grows many different regression trees using bootstrapped samples and randomly selected variables. Then all of the data are pushed down these trees and the tree with the most votes is selected. Although many trees are built, random forest will not overfit data due to the way of randomly growing trees. Correlation between trees compromises the efficacy of random forest, but the strength of each tree increases the prediction power of forest. However, increasing the number of variables to grow trees will increase both the correlation and the strength. The optimal number of variables could be set based on out-of-bag error rate. The out-of-bag error rate plays a similar role as cross-validation. Around one third of bootstrapped sample is not used to grow trees. Instead

these out-of-bag data estimate the error rate by predicting the outcome. The out-of-bag error rate could also be used to rank the importance of variables.

Lasso

Consider a linear regression model

$$Y = X\beta + \epsilon$$

Dependent variable $Y = (Y_1, \ldots, Y_n)^T$ is a $n\times$ 1 vector and regressor $X = (X_1, \ldots, X_p)$ is a $n \times p$ matrix. Coefficient $\beta = (\beta_1, \ldots, \beta_p)^T$ and disturbance $\epsilon = (\epsilon_1, \ldots, \epsilon_n)^T$. The ordinary least squares is an unconstrained optimizing problem

$$min_\beta (Y - X\beta)^T (Y - X\beta)$$

Lasso puts constraints on the L^1-norm of parameter vector and the optimization becomes

$$min_\beta (Y - X\beta)^T (Y - X\beta) \;\text{ subject to }\; \sum_i |\beta_i| \leq s$$

Lasso is a quadratic programming problem with linear inequality constraints. This optimizing problem could be solved by forward stepwise regression or least angle regression. Tuning parameter s determines which variables would be driven to zero. If s is set too large, the constraint may not be effective and there is no difference between Lasso and OLS. Usually the scale of s is determined by cross-validation.

Table 8 and Table 9 summarize the model selection results for our data on Android and iTunes apps. Forward, backward and stepwise search return the same results under AIC or BIC criterion. We also try running Branch & Bound search with different penalties, namely AIC, BIC, Adjusted R^2 and C_p, and these criteria also return the same selection results. Random forest returns a list of index indicating the importance of each variable. This index could be calculated by the

permutation or the tree impurity measure. The distribution of these indices is clustered in our case and the cluster of variables with the lowest importance is excluded. AIC, BIC and Lasso give the broadest selection results. Branch & Bound has the most parsimonious model. Regression and the random forest lie in between. In our empirical analysis, we use the least parsimonious model to infer developers' beliefs about the app development processes of their competitors.

[Table 8 about here.]

[Table 9 about here.]

4.2 Estimation of the structural model

Estimation of our structural model proceeds in two stages. In the first stage, we estimate developers' beliefs about their competitors' entry behavior as well as their beliefs about post-entry performance of their own apps on the new platform under consideration. Since our dataset has very rich app and market characteristics resulting in high-dimensional data, techniques of model selection discussed in Section 4.1 are applied to select the "best" prediction model without overfitting. Using the estimated propensity of competitors' entry as well as estimated post-entry performance of own apps as inputs,the second stage of the structural estimation examines how a developer trades off between timing and quality of app introduction to the second platform. We use a linear probability specification for the first stage, and a Cox duration model for the second stage. The specification for the first stage is:

$$crossentry = \alpha + \beta X + \epsilon$$

crossentry is an indicator of whether an app entered the other platform. X is a collection of market and app characteristics which may have an impact on cross-platform entry, and the relevant characteristics will be chosen by the model selection techniques as discussed earlier. This linear probability model is estimated on "big" and "small" developers separately to control for the developer size heterogeneity. A developer is considered to be "big" if the number of apps

developed by the developer is in top 30% within that particular genre. Otherwise, the developer is treated as a "small" developer.

In the second stage, a Cox proportional hazard model is estimated.

$$h(t|X) = h_0(t)exp(X\beta)$$

X contains app performances before entry, performance after entry and predicted entry probability. App performances are measured by the total number of ratings and weekly incremental number of ratings. Monthly major and minor updates are also used to approximate app quality. The performance before entry is measured at the instant of actual cross-platform entries. For censored observations, they are taken at the last observed time period. Performance after entry is mainly measured by update frequency within one month after entry. Rating for new apps is not very reliable, since it takes time for new apps to diffuse and further accumulate ratings stably. For "censored" apps that do not cross-enter, these variables are unavailable. A dummy for censored apps is introduced to solve the issue. An interaction term of this dummy variable and update frequency is added to the regression. Table 10 and Table 11 report the coefficients of Cox model for Android and iTunes apps respectively.

[Table 10 about here.]

[Table 11 about here.]

5 Conclusion

References

Chaudhuri, S., K. Ganjam, V. Ganti, and R. Motwani (2003): "Robust and efficient fuzzy match for online data cleaning," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 313–324. ACM.

Fellegi, I., and A. Sunter (1969): "A theory for record linkage," *Journal of the American Statistical Association*, pp. 1183–1210.

Gusfield, D. (1997): *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press.

Jaro, M. (1989): "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida," *Journal of the American Statistical Association*, pp. 414–420.

Newcombe, H., J. Kennedy, S. Axford, and A. James (1959): "Automatic linkage of vital and health records," *Science*, 130, 954–959.

Salton, G., and D. Harman (2003): *Information retrieval*. John Wiley and Sons Ltd.

Winkler, W. (1999): "The state of record linkage and current research problems," in *Statistical Research Division, US Census Bureau*. Citeseer.

Figure 1: Cross-platform Apps Matching Procedure

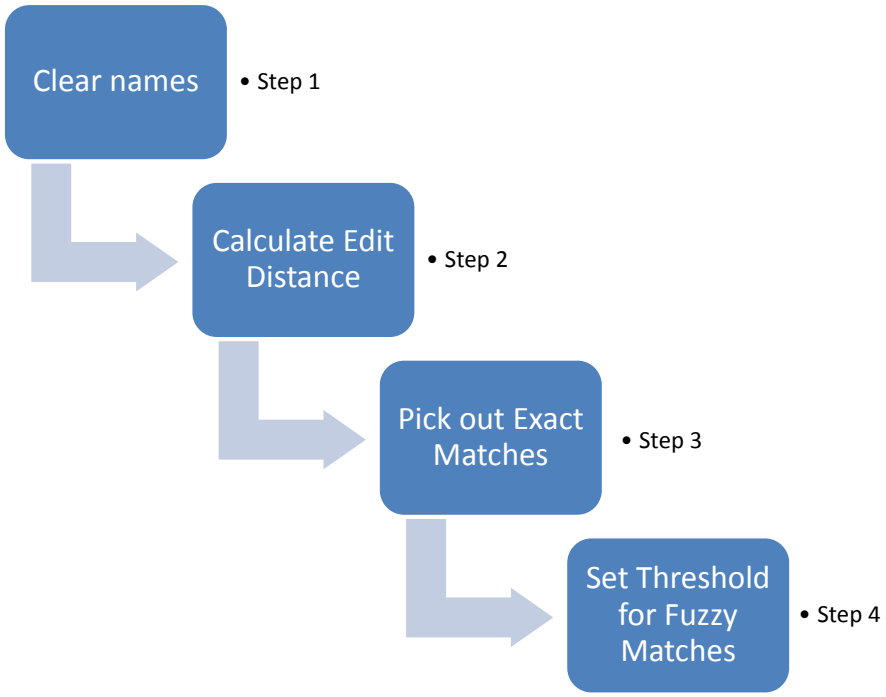

Figure 2: Herfindahl index cross Genre and Platform

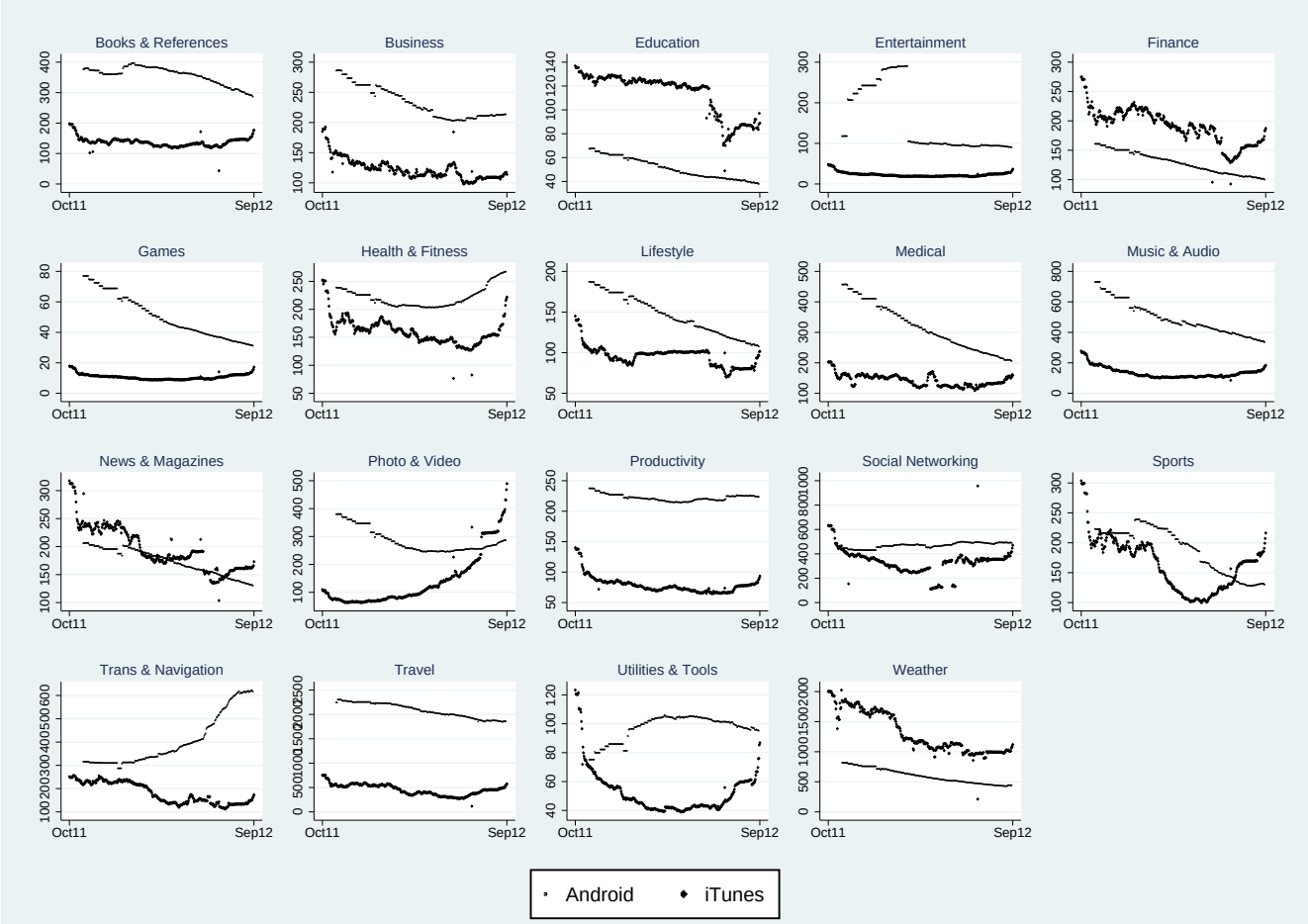

Figure 3: the frequency of entry following the entry of the top 50/100 developer on Android (Top panel is non-top follower and bottom panel is top follower)

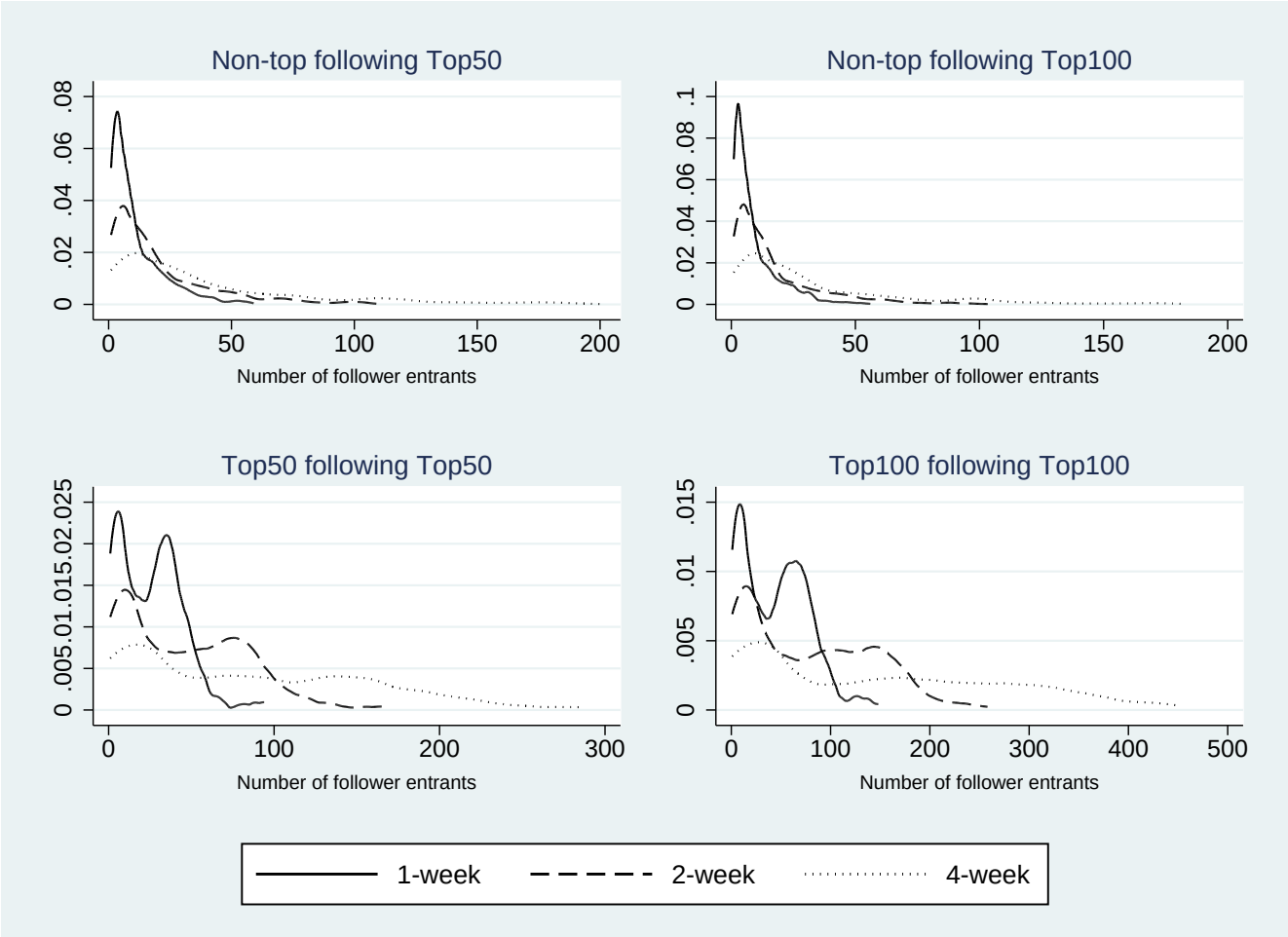

Figure 4: the frequency of entry following the entry of the top 50/100 developer on iTunes (Top panel is non-top follower and bottom panel is top follower)

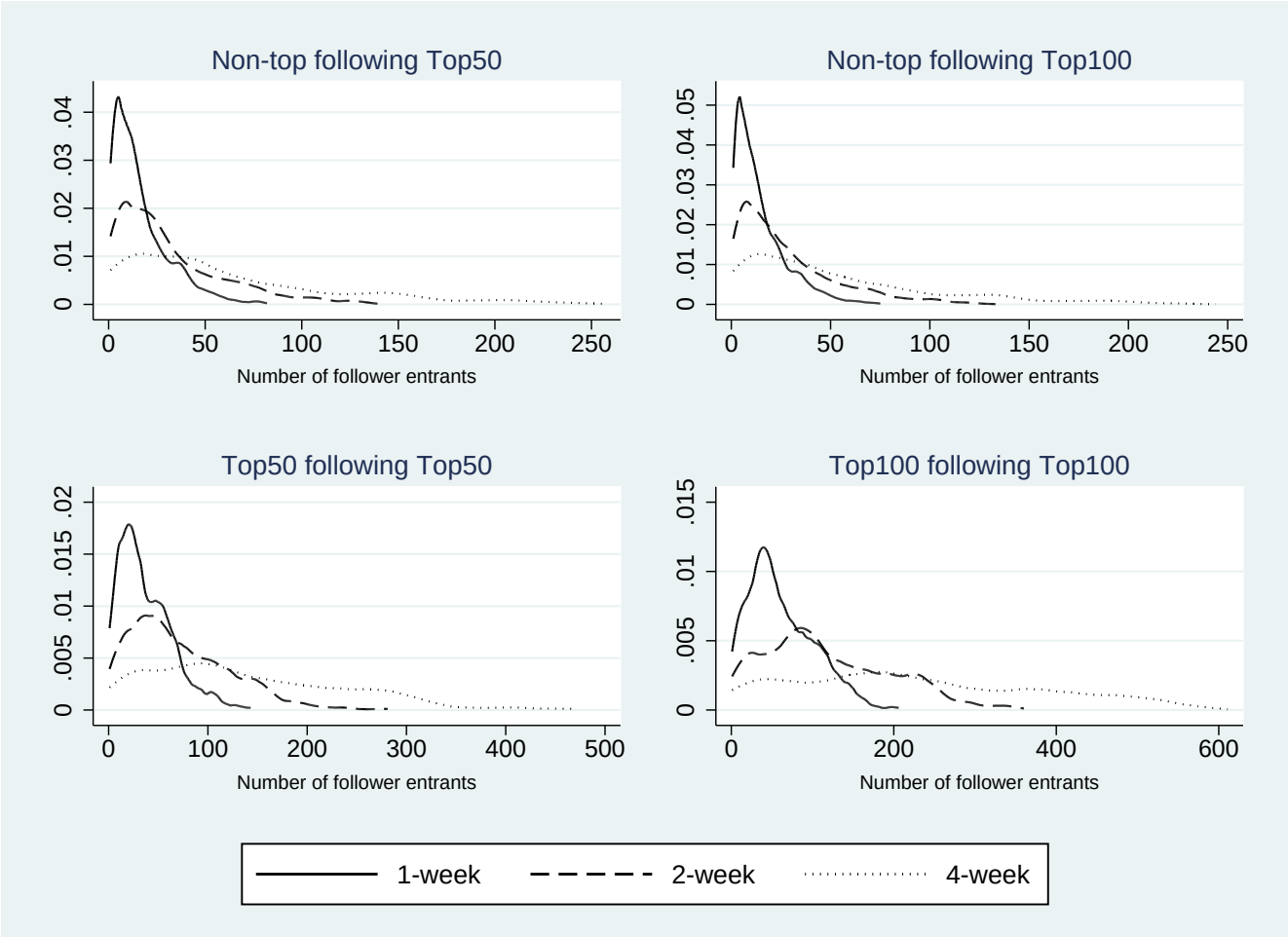

Figure 5: Weekly Incremental Ratings of Apps from Non Top50 Developers Following Top50 Developers within 2 weeks (Android)

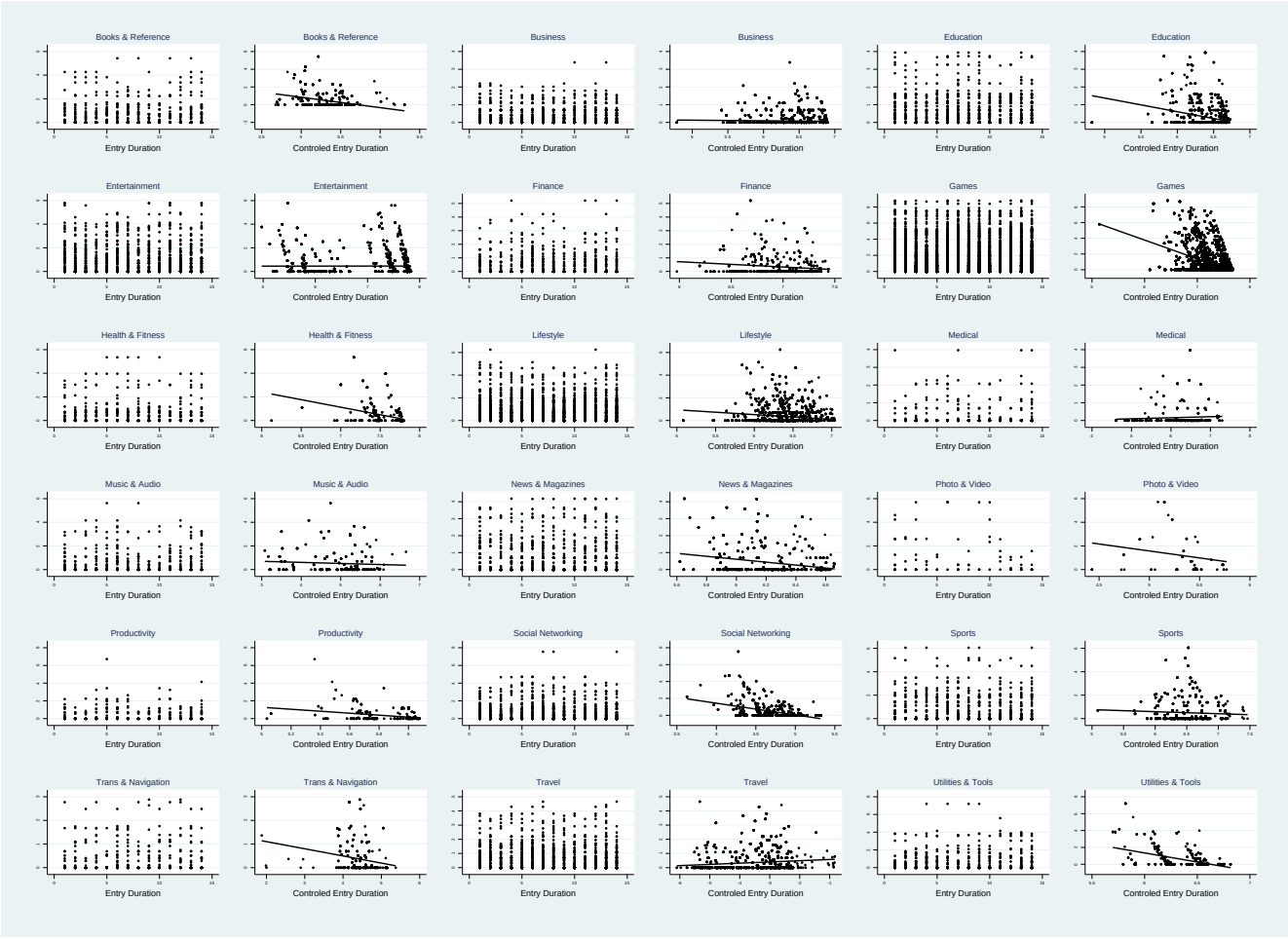

Figure 6: Monthly Major Updates of Apps from Non Top50 Developers Following Top50 Developers within 2 weeks (Android)

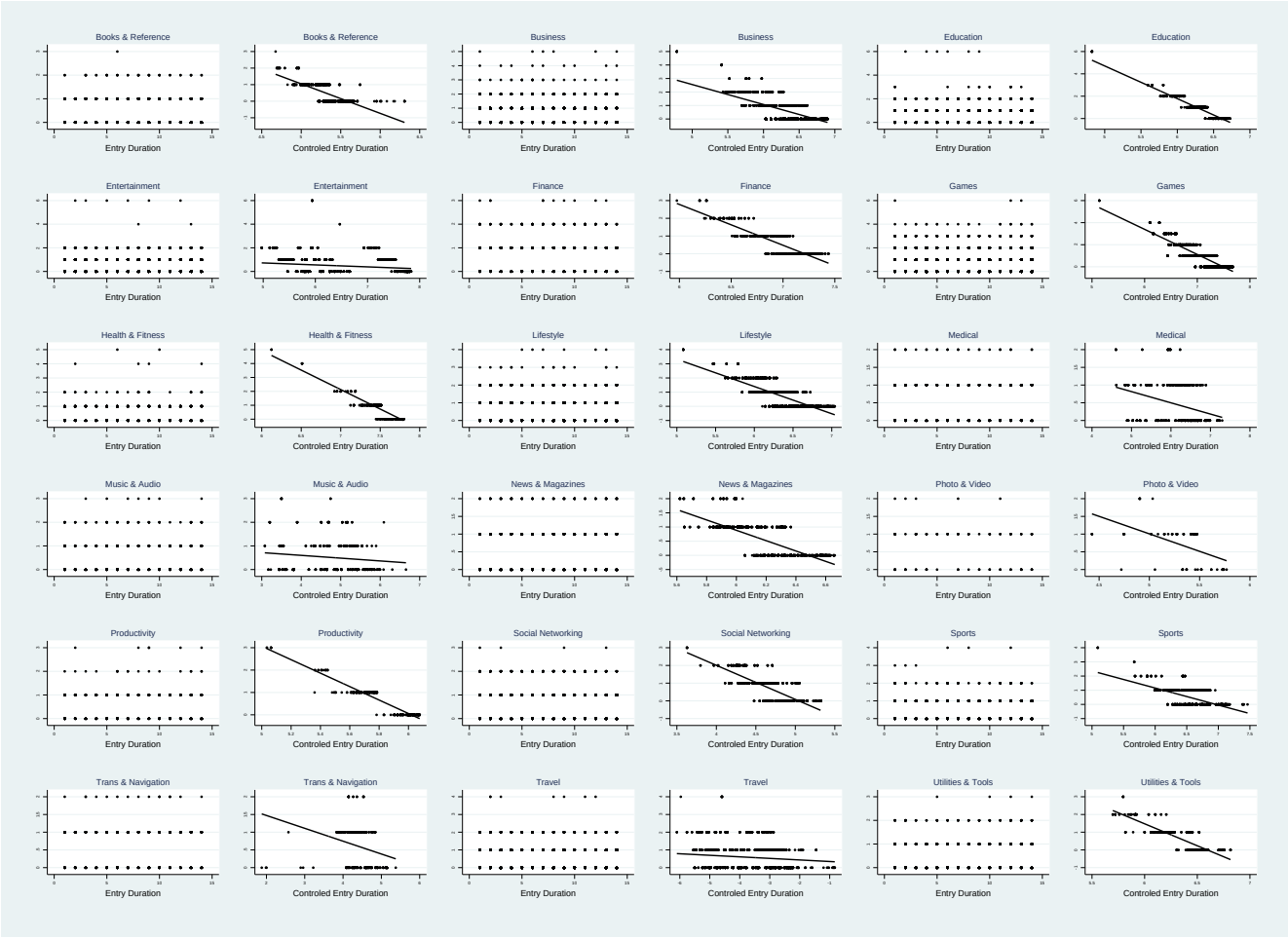

Figure 7: Monthly Minor Updates of Apps from Non Top50 Developers Following Top50 Developers within 2 weeks (Android)

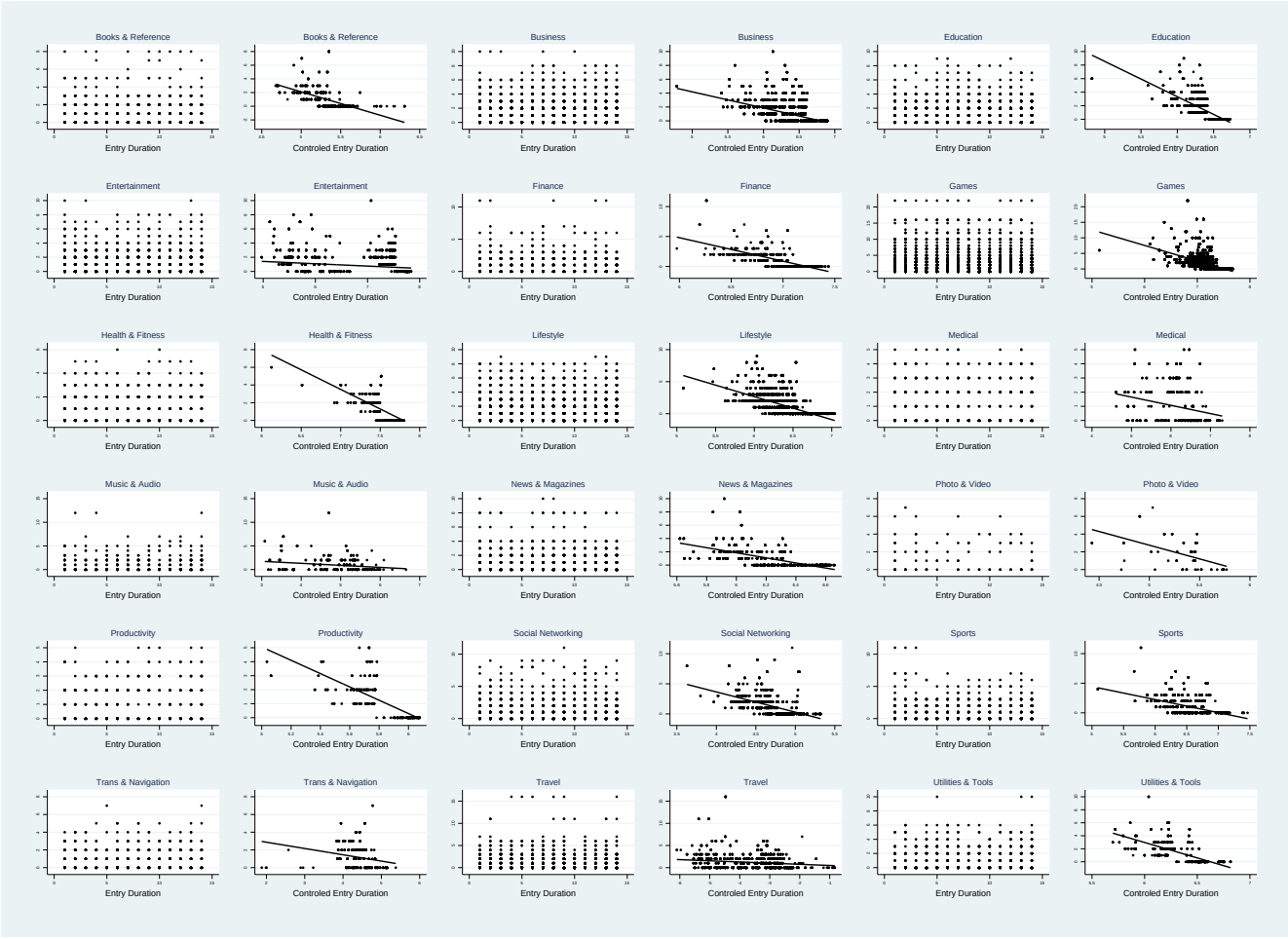

Figure 8: Weekly Incremental Ratings of Apps from Other Top50 Developers Following Top50 Developers within 2 weeks (Android)

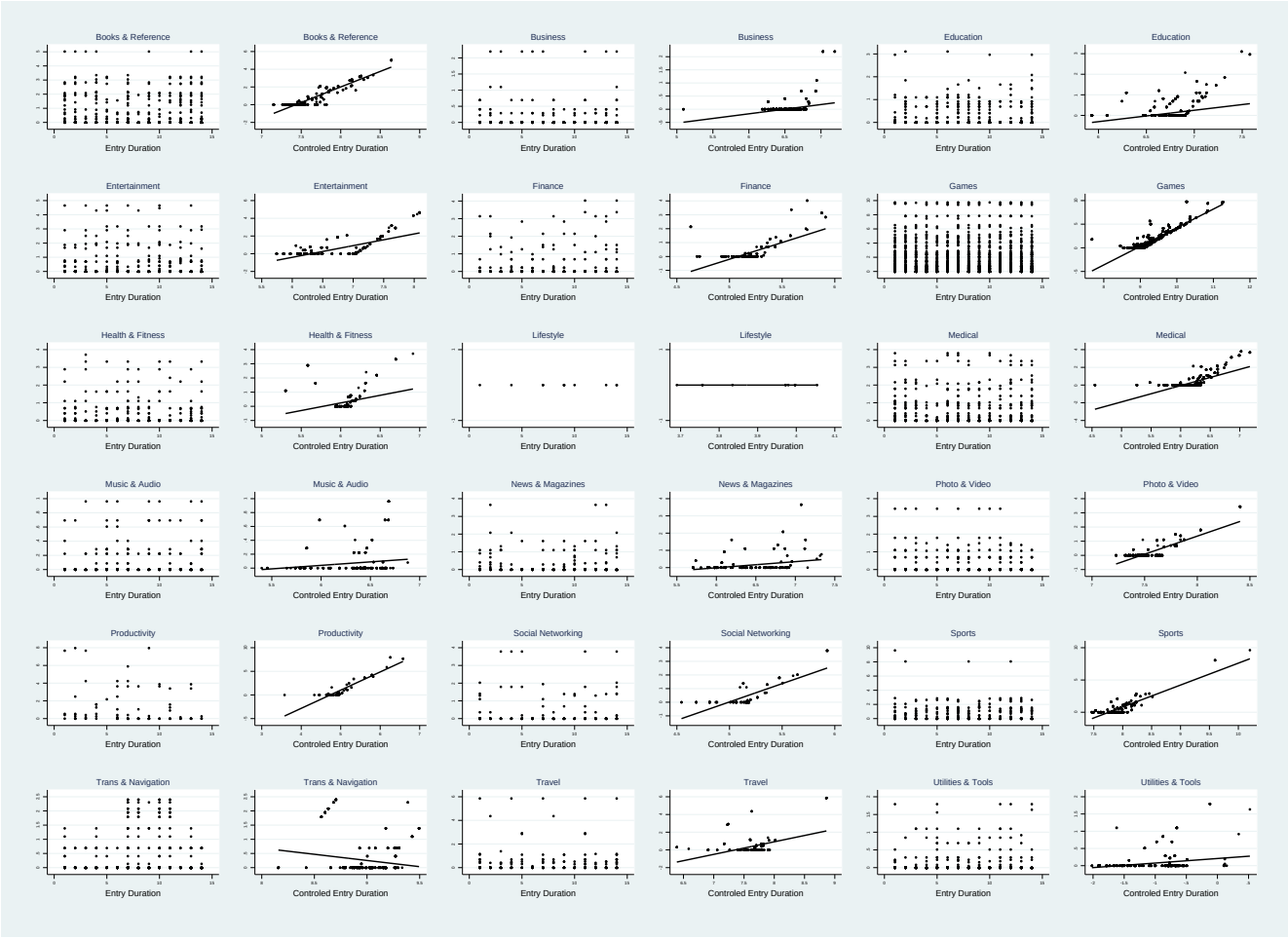

Figure 9: Monthly Major Updates of Apps from Other Top50 Developers Following Top50 Developers within 2 weeks (Android)

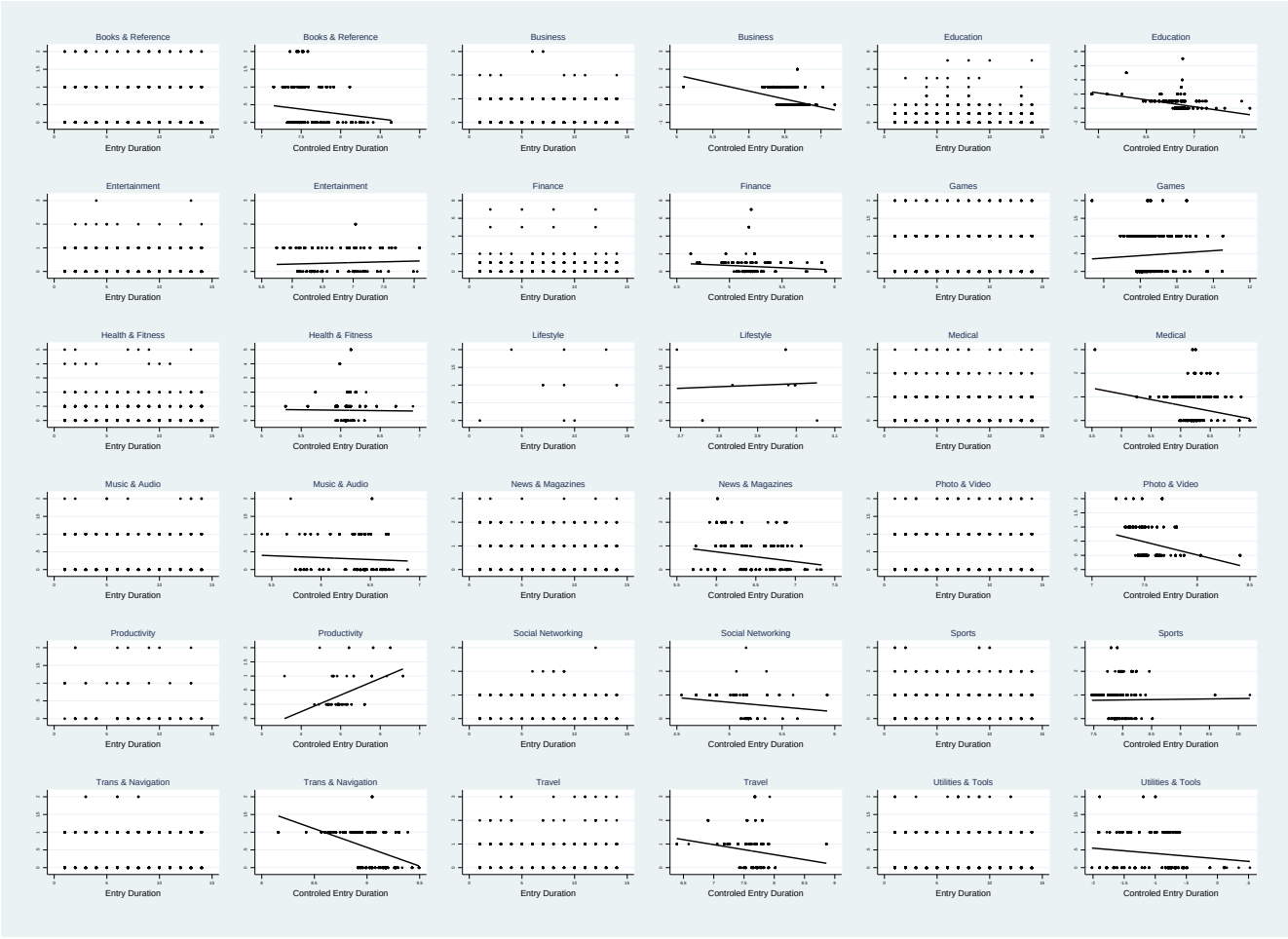

Figure 10: Monthly Minor Updates of Apps from Other Top50 Developers Following Top50 Developers within 2 weeks (Android)

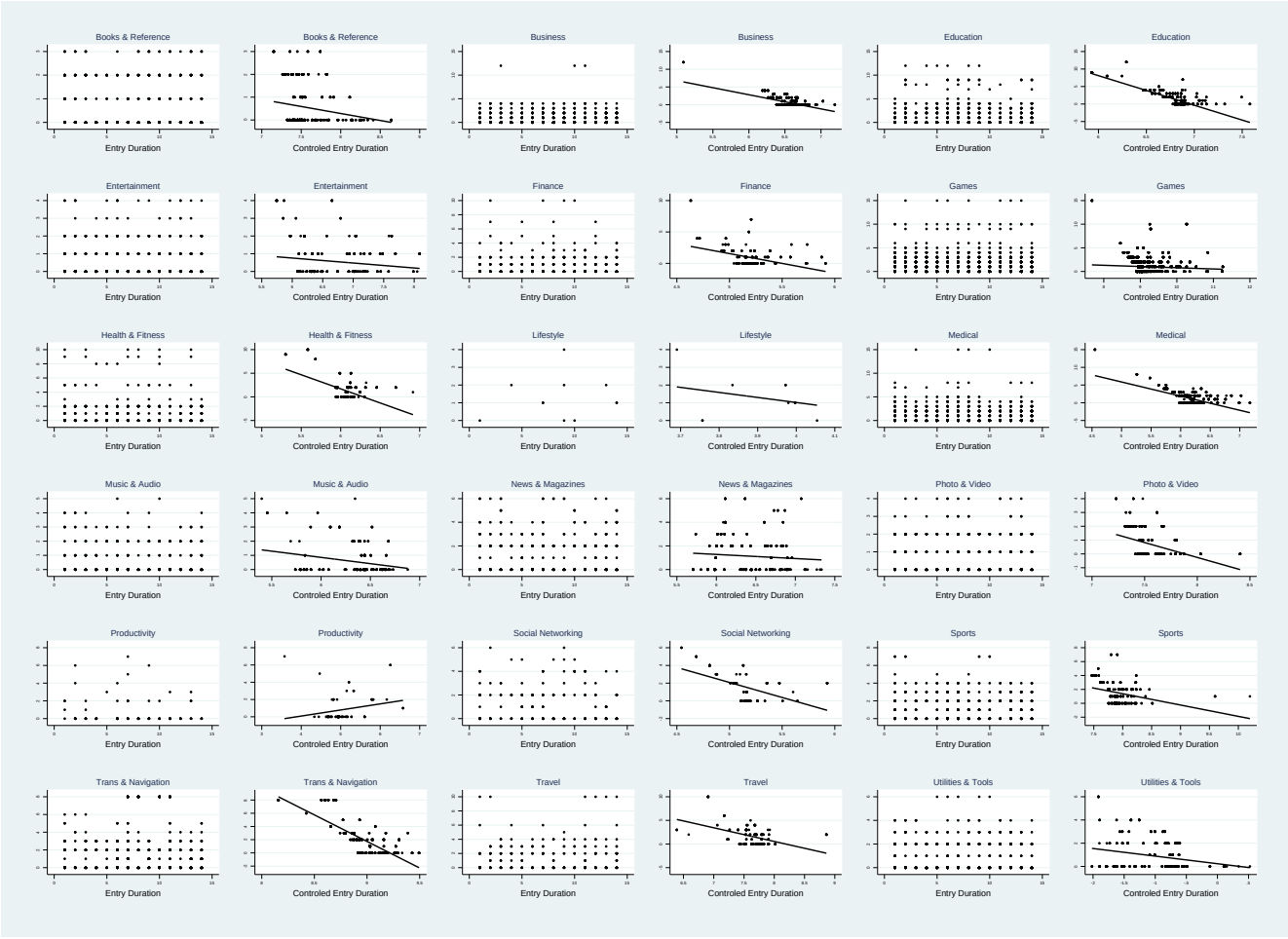

Figure 11: Weekly Incremental Ratings of Apps from Non Top50 Developers Following Top50 Developers within 2 weeks (iTunes)

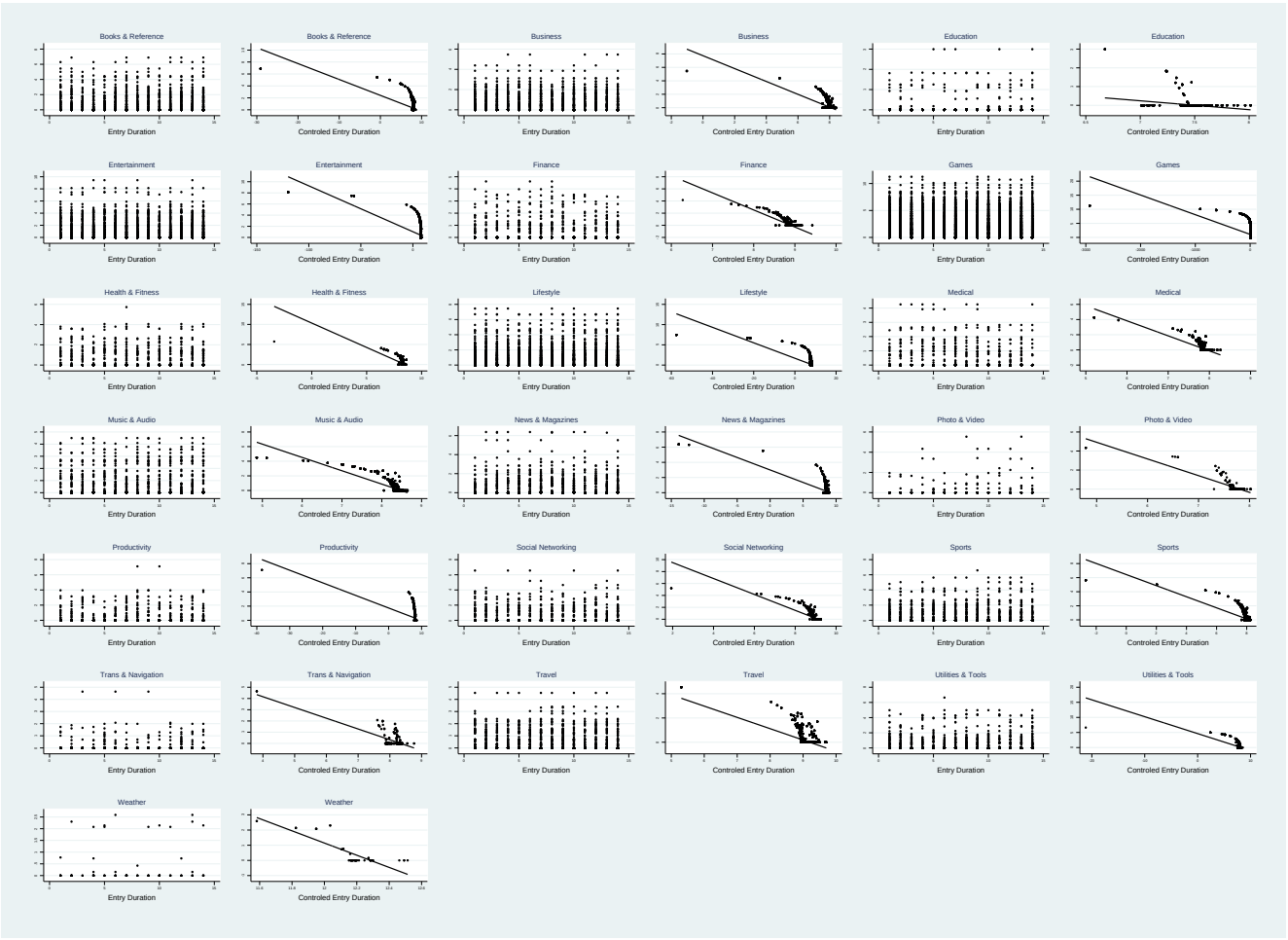

Figure 12: Monthly Major Updates of Apps from Non Top50 Developers Following Top50 Developers within 2 weeks (iTunes)

Figure 13: Monthly Minor Updates of Apps from Non Top50 Developers Following Top50 Developers within 2 weeks (iTunes)

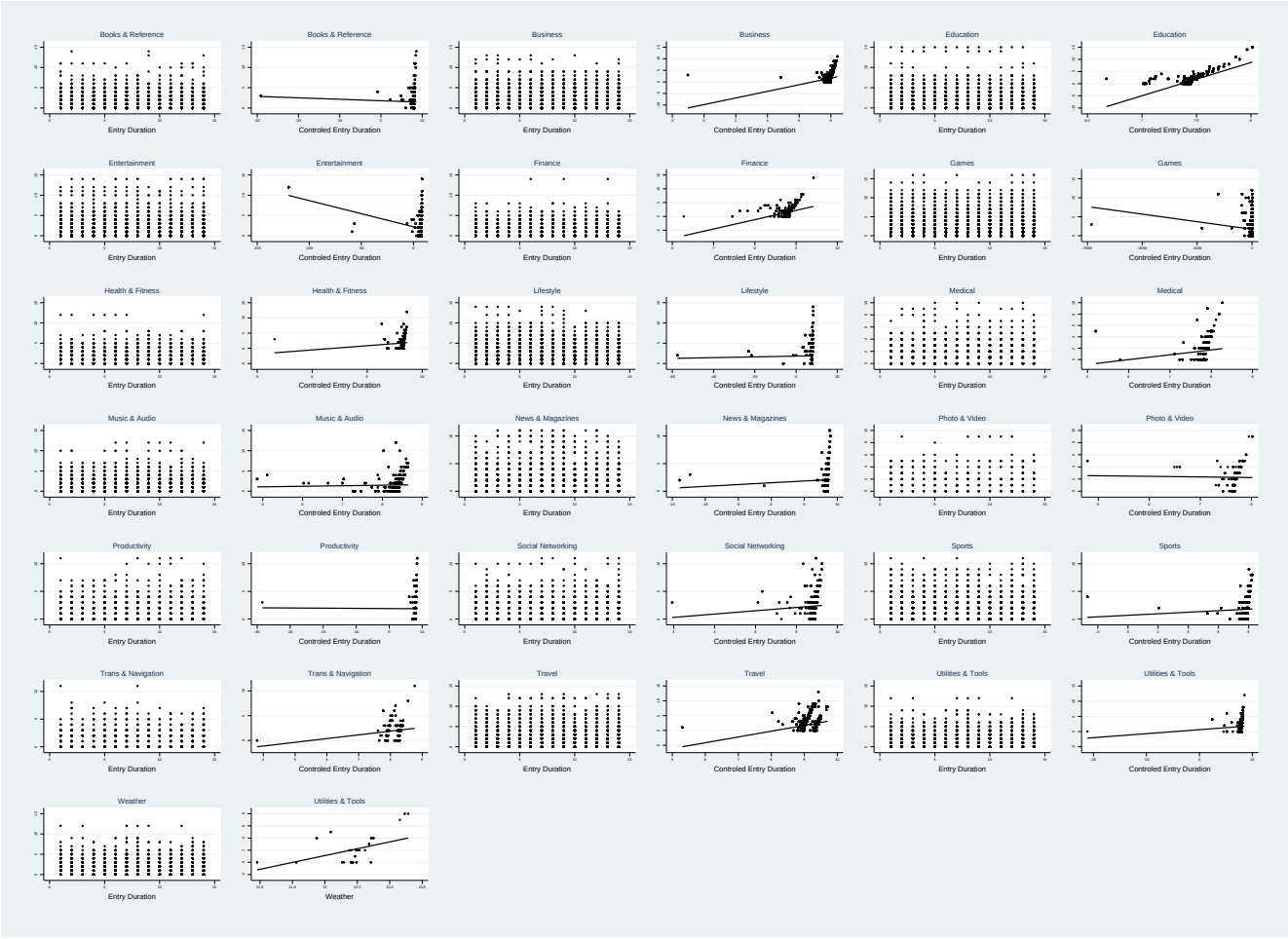

Figure 14: Weekly Incremental Ratings of Apps from Other Top50 Developers Following Top50 Developers within 2 weeks (iTunes)

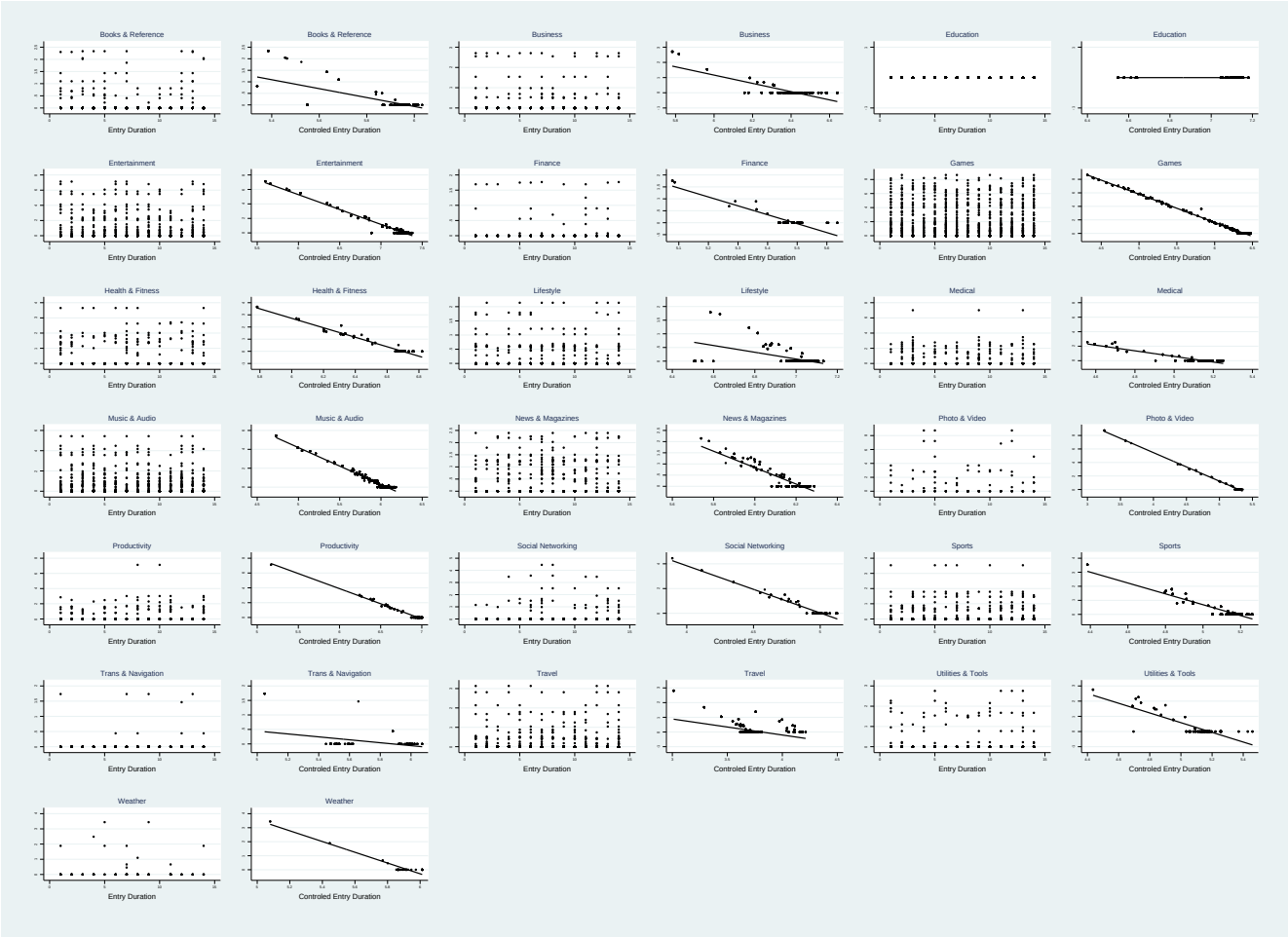

Figure 15: Monthly Major Updates of Apps from Other Top50 Developers Following Top50 Developers within 2 weeks (iTunes)

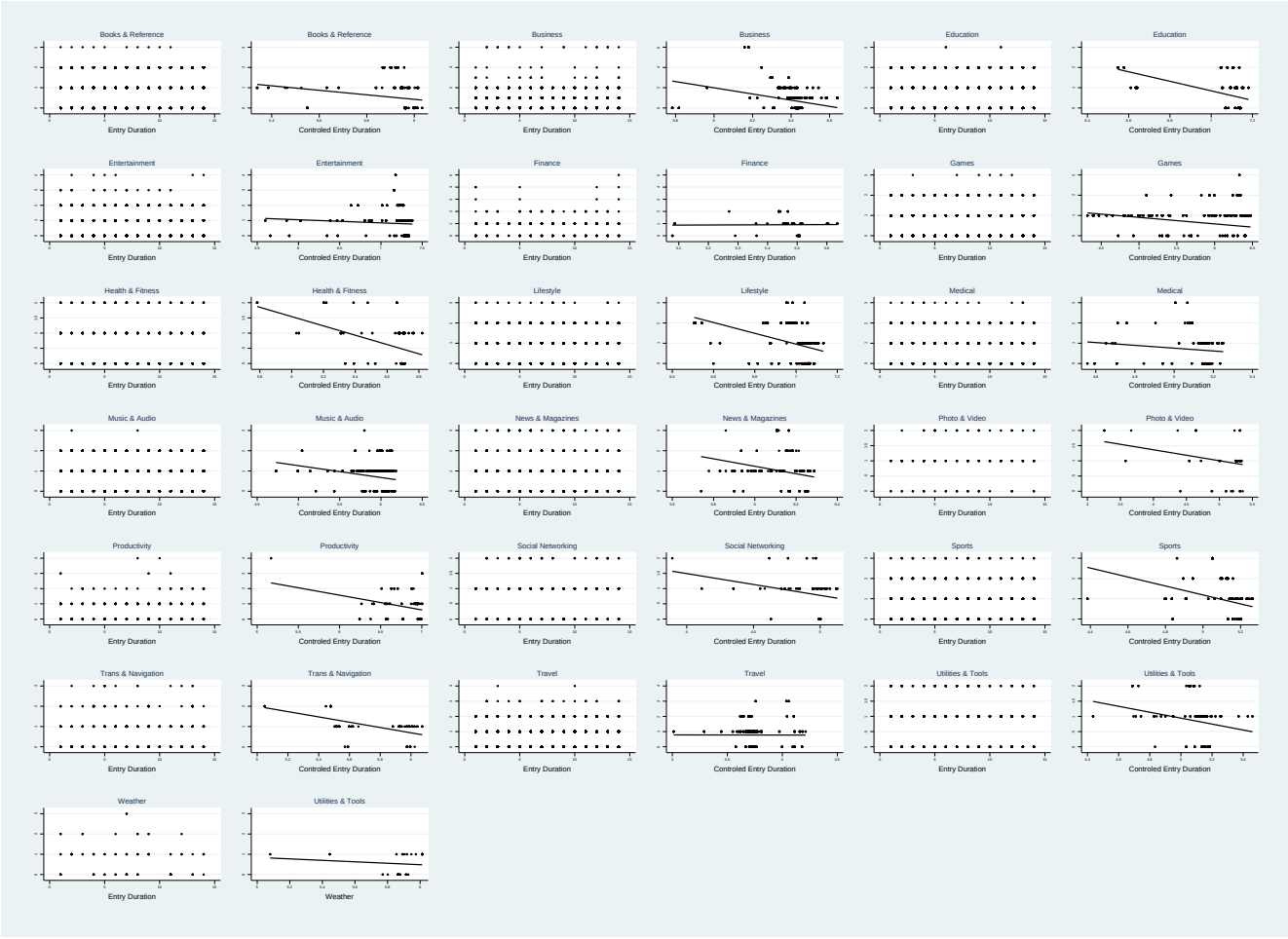

Figure 16: Monthly Minor Updates of Apps from Other Top50 Developers Following Top50 Developers within 2 weeks (iTunes)

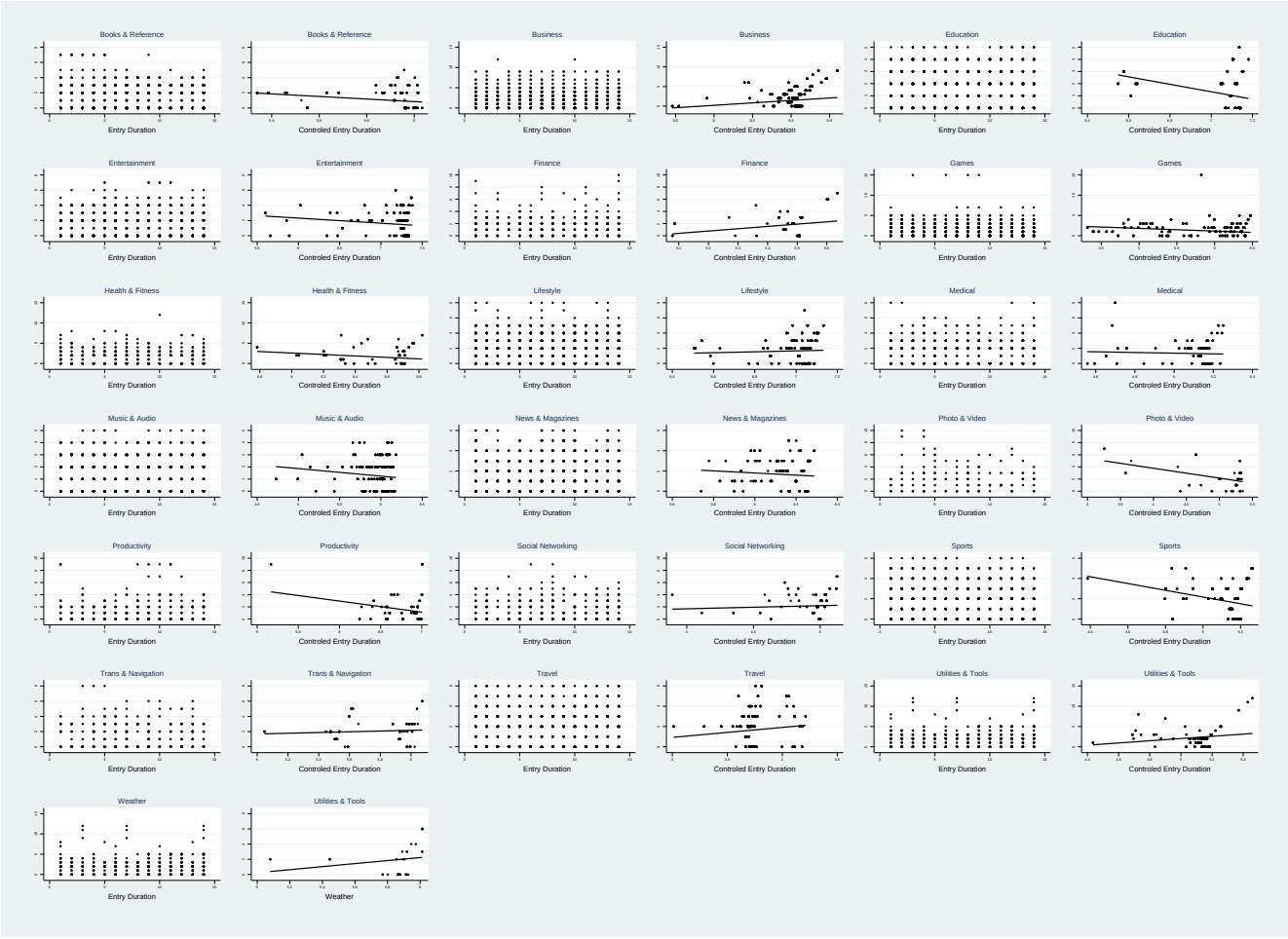

Table 1: Variable List and Description

Category	Item	Description	Unavailability	
			iOS	Android
App Identifier	App Name	App Names shown on phones		
	App ID	String of number or characters uniquely identifying apps		
Developer Info	Developer Name	Developers names shown on phones		
	Developer ID	String of number or characters uniquely identifying developers		X
	Developer URL	Developer webpage (if applicable)	X	
	Developer Email	Developer contact email (if applicable)		
App Detail	App Price	App price and currency unit		
	App Size	App size in kilobyte		
	Release Date	App launch date		X
	Genre	App category		
	App Description	One or several paragraphs of app function introduction		
	Maturity Recommendation	App maturity rating		
	App Screen Snapshot	Images of app screenshots (if applicable)		
	App Video Demo	A video clip introducing the subject app (if applicable)		
	Whether has game center	Indicator whether the subject app is multiple-player game		X
	Related Apps	Apps featured on webpage of the subject app		
Review, Rating and Download	The Number of Ratings	Both daily incremental and accumulative number of ratings received		
	Ratings	The daily and overall ratings		
	Review Contents	The texts of each review made by users		
	Download Intervals	An interval indicating accumulative downloads	X	
Version and Update	App Version/Update	Version number and update date		
	Version Description	What is updated in the new version		
Availability and Compatibility	Require OS	OS required to run the subject app		
	Compatible Devices	Devices required for the subject app		X
	Available Language	The available languages or countries		
Popularity	Ranking on Top Charts	Ranking on various top charts		

Table 2: Genre and Cross-Platform Apps Distribution(Android)

Genre	Genre Distribution		Percent of Cross-platform	Entry Order (%)			
	Number	Percent (%)	Apps (%)	Same Time	iTunes First	Android First	Unidentified
Books & Reference	44,660	7.86	8.88	1.74	17.20	59.52	21.54
Business	21,697	3.82	24.89	2.28	24.11	39.79	33.83
Education	29,951	5.27	15.80	1.58	22.59	45.20	30.62
Entertainment	72,468	12.76	7.27	1.63	21.29	47.83	29.25
Finance	12,450	2.19	22.85	1.69	19.93	40.91	37.47
Games	81,974	14.43	10.40	1.58	25.99	39.27	33.17
Health & Fitness	14,622	2.57	14.67	1.77	20.42	47.37	30.44
Lifestyle	44,460	7.83	17.26	1.33	21.04	44.57	33.06
Medical	7,472	1.32	24.20	1.71	20.91	45.80	31.58
Music & Audio	19,385	3.41	9.97	1.09	16.82	49.48	32.61
News & Magazines	17,865	3.15	19.91	1.49	16.00	45.09	37.42
Photo & Video	9,910	1.75	5.62	0.36	19.39	43.09	37.16
Productivity	15,152	2.67	10.64	1.30	19.85	45.78	33.06
Social Networking	30,159	5.31	10.68	1.27	22.10	39.35	37.27
Sports	21,293	3.75	13.60	1.90	20.66	43.32	34.13
Trans & Navigation	8,143	1.43	14.05	1.84	22.20	36.89	39.07
Travel	24,448	4.30	20.89	1.59	18.58	47.05	32.78
Utilities & Tools	88,552	15.59	2.34	1.16	21.87	47.32	29.65
Weather	3,244	0.57	11.53	0.80	14.44	36.63	48.13
Total	567,905	100.00	11.42	1.59	21.18	44.65	32.59

Table 3: Genre and Cross-platform App Distribution(iTunes)

Genre	Genre Distribution		Percent of Cross-platform	Entry Order (%)			
	Number	Percent (%)	Apps (%)	Same Time	iTunes First	Android First	Unidentified
Books & Reference	119,164	14.48	3.77	1.93	18.21	56.77	23.08
Business	47,694	5.79	12.70	1.92	23.34	39.99	34.75
Education	86,540	10.51	6.26	1.57	24.09	45.75	28.59
Entertainment	88,581	10.76	5.65	1.60	23.00	46.36	29.04
Finance	19,542	2.37	13.84	1.63	20.16	40.61	37.61
Games	142,332	17.29	5.70	1.65	25.74	39.08	33.53
Health & Fitness	21,675	2.63	9.29	1.94	22.35	44.66	31.05
Lifestyle	82,487	10.02	9.12	1.46	20.86	44.62	33.05
Medical	17,767	2.16	10.77	1.52	19.85	48.33	30.30
Music & Audio	33,661	4.09	6.84	1.39	15.51	52.04	31.06
News & Magazines	23,616	2.87	13.24	1.47	15.99	45.36	37.17
Photo & Video	19,753	2.40	3.32	0.61	23.32	39.79	36.28
Productivity	24,539	2.98	5.29	1.23	21.03	42.91	34.82
Social Networking	18,172	2.21	11.58	1.24	19.86	39.57	39.33
Sports	27,151	3.30	11.12	1.72	20.43	43.41	34.44
Trans & Navigation	14,146	1.72	7.66	1.29	16.97	43.54	38.19
Travel	42,618	5.18	10.20	1.61	18.20	46.25	33.94
Utilities & Tools	54,717	6.65	5.97	1.29	20.01	46.54	32.17
Weather	3,748	0.46	10.38	0.77	15.94	36.76	46.53
Total	823,068	100.00	7.30	1.59	21.18	44.65	32.59

Table 4: App Quality Cross Genre(Android)

Genre	Number of Ratings	Weekly Incrememtal number of Ratings	Monthly Major Updates	Monthly Minor Updates
Books & Reference	114.48	4.97	0.31	0.60
Business	54.63	1.88	0.41	0.84
Education	39.28	1.86	0.44	0.90
Entertainment	279.11	15.00	0.35	0.75
Finance	132.18	3.51	0.55	1.20
Games	842.17	52.40	0.46	1.05
Health & Fitness	167.49	5.78	0.39	0.88
Lifestyle	139.05	4.57	0.41	0.87
Medical	31.89	1.06	0.43	0.86
Music & Audio	408.12	19.53	0.41	0.89
News & Magazines	116.45	3.03	0.46	1.02
Photo & Video	327.25	14.74	0.42	1.00
Productivity	560.75	19.08	0.50	1.22
Social Networking	956.07	31.06	0.43	1.01
Sports	85.19	3.77	0.52	1.01
Trans & Navigation	179.49	5.23	0.70	1.50
Travel	224.73	4.61	0.48	0.94
Utilities & Tools	278.60	11.22	0.36	0.83
Weather	542.78	13.74	0.65	1.48
Total	339.88	16.06	0.42	0.91

Table 5: App Quality Cross Genre(iTunes)

Genre	Number of ratings	Weekly incremental number of Ratings	Monthly major updates	Monthly minor updates
Books & Reference	50.85	1.07	0.31	0.62
Business	30.72	0.72	0.58	1.27
Education	32.36	0.39	0.49	1.04
Entertainment	275.63	4.93	0.41	0.90
Finance	93.27	1.75	0.68	1.53
Games	718.98	20.60	0.43	0.98
Health & Fitness	149.90	3.74	0.52	1.17
Lifestyle	122.27	1.88	0.54	1.22
Medical	45.87	0.78	0.52	1.14
Music & Audio	206.51	4.02	0.58	1.19
News & Magazines	115.89	2.02	0.67	1.44
Photo & Video	203.94	4.78	0.56	1.43
Productivity	171.52	3.23	0.65	1.58
Social Networking	593.02	24.63	0.64	1.56
Sports	123.43	1.73	0.61	1.33
Trans & Navigation	115.66	1.48	0.61	1.39
Travel	49.69	0.75	0.58	1.25
Utilities & Tools	193.89	3.32	0.53	1.21
Weather	327.29	5.75	0.76	1.76
Total	226.02	5.58	0.49	1.10

Table 6: Apps from Non Top50 Developers Following Top50 Developers within 2 weeks

Genre	Android						iTunes					
	Median			Mean			Median			Mean		
	I	II	III	I	II	III	I	II	III	I	II	III
Books & Reference	-0.19	0.00	0.00	-4.47	0.01	0.08	0.00	1.00	1.00	2.14	0.40	0.73
Business	-0.06	0.00	0.00	-3.39	0.01	-0.04	0.00	1.00	1.00	-0.26	0.25	0.40
Education	-0.17	0.00	0.00	-1.26	-0.03	-0.06	0.00	1.00	1.00	-0.37	0.22	0.42
Entertainment	-0.42	0.00	0.00	-11.15	0.01	-0.08	0.00	1.00	1.00	10.37	0.34	0.67
Finance	-0.20	0.00	0.00	-1.14	0.00	-0.25	0.00	0.00	1.00	-0.44	0.24	0.21
Games	-0.39	0.00	0.00	-11.55	-0.01	0.00	0.00	1.00	1.00	106.65	0.26	0.56
Health & Fitness	-0.18	0.00	0.00	-4.37	0.00	-0.16	0.00	1.00	2.00	-1.81	0.25	0.50
Lifestyle	-0.22	0.00	0.00	-4.32	0.15	0.15	0.00	1.00	2.00	1.61	0.32	0.63
Medical	-0.22	0.00	0.00	-4.51	0.00	0.01	0.00	1.00	1.00	-0.12	0.22	0.38
Music & Audio	-0.09	0.00	0.00	2.11	0.04	0.06	0.00	1.00	1.00	-1.42	0.19	0.31
News & Magazines	-0.43	0.00	0.00	-25.29	-0.01	-0.10	0.00	1.00	2.00	0.50	0.27	0.65
Photo & Video	0.25	1.00	1.00	21.21	0.19	0.64	0.00	1.00	2.00	-1.19	0.47	0.77
Productivity	-0.48	0.00	0.00	-14.29	0.07	-0.08	0.00	0.00	0.00	0.29	0.14	0.24
Social Networking	-0.27	0.00	0.00	-11.86	0.08	-0.10	0.00	1.00	2.00	-0.45	0.31	0.56
Sports	-0.26	0.00	0.00	-20.13	0.04	-0.08	0.00	1.00	2.00	0.38	0.22	0.52
Trans & Navigation	-0.20	1.00	1.00	-2.75	-0.02	-0.02	0.00	1.00	2.00	-0.49	0.27	0.50
Travel	-0.30	-1.00	-1.00	-2.64	-0.27	-0.68	0.00	1.00	2.00	-0.14	0.28	0.64
Utilities & Tools	-0.13	0.00	0.00	3.62	0.01	0.11	0.00	1.00	2.00	-1.81	0.25	0.48
Weather	-	-	-	-	-	-	0.00	0.00	1.00	-4.55	0.22	0.38

Table 7: Apps from other Top50 Developers Following Top50 Developers within 2 weeks

Genre	Android						iTunes					
	Median			Mean			Median			Mean		
	I	II	III	I	II	III	I	II	III	I	II	III
Books & Reference	-0.19	0.00	0.00	-4.03	-0.02	-0.18	0.00	0.00	0.00	-0.72	0.37	0.41
Business	-0.06	0.00	0.00	-3.46	-0.21	-0.44	0.00	1.00	1.00	-0.55	0.15	0.23
Education	-0.17	0.00	0.00	-1.79	0.06	0.16	0.00	0.00	0.00	-0.40	0.17	0.07
Entertainment	-0.42	0.00	0.00	-12.31	-0.09	-0.36	0.00	0.00	0.00	6.06	0.27	0.37
Finance	-0.20	0.00	0.00	-1.66	0.02	-0.33	0.00	0.00	1.00	-1.08	0.23	0.23
Games	-0.39	0.00	0.00	215.93	-0.05	-0.19	0.00	0.00	0.00	32.32	0.11	0.02
Health & Fitness	-0.18	0.00	0.00	-5.82	0.24	0.26	0.00	0.00	0.00	-1.87	0.08	0.22
Libraries & Demo	-0.22	1.00	1.00	-5.97	0.67	0.49	0.00	1.00	2.00	-1.51	0.39	0.59
Lifestyle	-0.22	0.00	0.00	-4.11	0.07	0.14	0.00	1.00	2.00	3.70	0.28	0.45
Medical	-0.09	0.00	0.00	-1.35	-0.14	-0.31	0.00	1.00	2.00	-0.78	0.38	0.45
Music & Audio	-0.43	0.00	0.00	-26.64	0.11	0.08	0.00	1.00	2.00	-1.49	0.29	0.55
News & Magazines	-0.25	0.00	0.00	-2.60	-0.07	-0.42	0.00	1.00	2.00	189.32	0.41	1.02
Photo & Video	-0.31	0.00	0.00	115.83	-0.11	-0.30	0.00	0.00	0.00	6.27	0.13	-0.02
Productivity	-0.27	0.00	0.00	-16.18	0.02	-0.02	0.00	1.00	2.00	-2.79	0.19	0.37
Social Networking	-0.26	1.00	1.00	39.16	0.31	0.35	0.00	1.00	2.00	-1.13	0.20	0.33
Sports	-0.20	1.00	1.00	-2.99	-0.02	0.61	0.00	1.00	2.00	-1.04	0.25	0.27
Trans & Navigation	-0.30	0.00	0.00	5.28	-0.12	-0.19	0.00	0.00	0.00	-0.33	-0.11	-0.20
Travel	-0.13	0.00	0.00	-6.01	-0.13	-0.20	0.00	1.00	2.00	-2.75	0.35	0.88
Utilities & Tools	-0.30	0.00	0.00	-11.08	0.14	0.18	0.00	0.00	1.00	-4.58	0.01	0.20
Weather	-0.60	1.00	1.00	-14.86	-0.11	-0.21	0.00	0.00	0.00	-0.72	0.37	0.41

Table 8: Summary of Model Selection Results on Android Apps

Variables	AIC	BIC	Branch & Bound	Regression Tree	Random Forest		Lasso
					premutation	impurity	
Duration between Two Entries	x	x	x	x	x	x	x
The number of Rating	x	x	x	x		x	x
Rating Star	x	x	x	x		x	x
Average Price	x	x					x
Frequency of Major Updates							
Frequency of Minor Updates	x	x			x		
Average Weekly Number of Rating	x	x	x			x	x
Variance of Weekly Rating Number	x	x					x
Average of 2-week Rating Incremental	x	x					x
Variance of 2-week Rating Incremental	x	x			x		x
Variation Coefficient of Weekly Rating Number	x	x					
Variation Coefficient of 2-week Rating Incremental	x	x					x
Thu Number of Sibling Apps	x	x	x	x	x	x	x
Max Number of Rating of Sibling Apps				x		x	
Average Number of Rating of Sibling Apps						x	
If Have Other Cross-Platform Apps	x	x	x	x	x	x	x
The Total Number of Apps in Specific iTunes Genre	x	x	x	x	x	x	x
The Total Number of Apps in iTunes	x	x	x	x	x	x	x
HHI in iTunes Genre	x	x		x	x	x	x

Table 9: Summary of Model Selection Results on iTunes Apps

Variables	AIC	BIC	Branch & Bound	Regression Tree	Random Forest		Lasso
					premutation	impurity	
Duration between Two Entries	x	x	x	x	x	x	x
The number of Rating	x	x	x		x	x	x
Rating Star	x						x
Average Price	x	x			x		x
Frequency of Major Updates	x	x			x	x	x
Frequency of Minor Updates					x	x	
Average Weekly Number of Rating	x	x	x		x	x	x
Variance of Weekly Rating Number	x						x
Average of 2-week Rating Incremental						x	
Variance of 2-week Rating Incremental	x	x					
Variation Coefficient of Daily Rating Number	x	x	x			x	x
Variation Coefficient of 2-week Rating Incremental	x	x					
Thu Number of Sibling Apps	x	x			x	x	x
Max Number of Rating of Sibling Apps	x	x					x
Average Number of Rating of Sibling Apps	x	x					x
If Have Other Cross-Platform Apps	x	x	x	x	x	x	x
The Total Number of Apps in Specific Android Genre	x	x	x	x	x	x	x
The Total Number of Apps in Android	x	x	x	x	x	x	x
HHI in Android Genre	x	x	x	x	x	x	x

Table 10: Android Regression of Cox proportional hazard model

Variable	Entire Apps	Books & Reference	Business	Education	Entertainment	Finance	Games	Health & Fitness	Lifestyle	Medical	Music & Audio	News & Magazines	Photo & Video	Productivity	Social Networking	Sports	Trans & Navigation	Travel	Utilities & Tools	Weather
Ln(number of ratings)	-0.439	-0.442	-0.66	-0.483	-0.485	-0.682	-0.36	-0.613	-0.523	-0.594	-0.601	-0.527	-0.37	-0.847	-0.483	-0.403	-0.523	-0.445	-0.416	-0.292
	(0.008)***	(0.058)***	(0.053)***	(0.040)***	(0.033)***	(0.048)***	(0.016)***	(0.073)***	(0.030)***	(0.092)***	(0.086)***	(0.049)***	(0.116)***	(0.069)***	(0.038)***	(0.046)***	(0.083)***	(0.042)***	(0.047)***	(0.129)**
Ln(weekly incremental number of ratings)	0.552	1.012	1.124	0.742	0.742	0.663	0.452	0.926	0.729	0.912	0.942	0.618	0.611	1.239	0.639	0.49	0.837	0.652	0.72	0.22
	(0.010)***	(0.088)***	(0.092)***	(0.071)***	(0.045)***	(0.064)***	(0.017)***	(0.120)***	(0.043)***	(0.167)***	(0.131)***	(0.065)***	(0.167)***	(0.103)***	(0.052)***	(0.056)***	(0.134)***	(0.067)***	(0.064)***	-0.204
Monthly Major Updates	-1.456	-3.09	-2.031	-1.145	-2.021	-1.056	-1.922	-1.213	-1.417	-2.629	-0.372	-2.929	-0.64	-1.298	-1.216	-1.887	-2.332	-1.722	-2.254	-2.56
	(0.070)***	(1.018)***	(0.291)***	(0.191)***	(0.338)***	(0.304)***	(0.184)***	(0.380)***	(0.209)***	(1.018)***	-0.346	(0.571)***	-0.781	(0.556)**	(0.275)***	(0.442)***	(0.592)***	(0.352)***	(0.545)***	(1.145)**
Monthly Minor Updates	-0.226	-0.38	-0.169	-0.178	-0.358	-0.239	-0.187	0.028	-0.09	-0.516	-0.862	0.062	-0.614	-0.455	-0.213	-0.287	-0.008	-0.244	-0.354	-0.015
	(0.017)***	(0.130)***	(0.055)***	(0.069)***	(0.095)***	(0.104)**	(0.034)***	-0.101	(0.054)*	(0.175)***	(0.136)***	-0.103	(0.234)***	(0.221)**	(0.074)***	(0.050)***	-0.154	(0.102)**	(0.128)***	-0.253
Predicted cross entry probability	4.073	4.788	3.797	3.848	4.339	3.761	3.879	4.388	3.791	3.733	4.444	3.363	5.106	3.952	3.895	3.814	3.544	3.766	5.565	4.337
	(0.016)***	(0.080)***	(0.060)***	(0.055)***	(0.059)***	(0.080)***	(0.041)***	(0.104)***	(0.050)***	(0.104)***	(0.108)***	(0.078)***	(0.202)***	(0.108)***	(0.075)***	(0.081)***	(0.109)***	(0.068)***	(0.102)***	(0.270)***
Monthly Major Updates after cross entry	2.068	4.261	2.365	1.308	3.075	1.605	2.653	1.997	2.291	2.766	0.118	4.463	2.299	0.715	2.214	3.14	2.851	2.255	2.984	4.843
	(0.079)***	(1.068)***	(0.323)***	(0.234)***	(0.408)***	(0.386)***	(0.202)***	(0.466)***	(0.262)***	(1.047)***	-0.437	(0.696)***	(1.031)**	-0.577	(0.304)***	(0.482)***	(0.655)***	(0.360)***	(0.591)***	(1.603)***
Monthly Minor Updates after cross entry	0.559	0.327	0.58	0.734	0.724	0.896	0.439	-0.092	0.261	1.08	1.759	-0.422	0.516	1.44	0.574	0.4	0.187	0.536	1.01	-0.662
	(0.032)***	-0.258	(0.104)***	(0.134)***	(0.141)***	(0.203)***	(0.062)***	-0.218	(0.116)**	(0.235)***	(0.289)***	-0.258	-0.469	(0.255)***	(0.133)***	(0.118)***	-0.236	(0.141)***	(0.211)***	-0.634
N	402,276	30,533	13,224	20,791	51,935	8,248	58,691	10,014	30,405	4,763	13,437	11,524	7,185	11,646	21,392	13,828	5,657	16,146	70,536	2,321

* p¡0.1; ** p¡0.05; *** p¡0.01

Table 11: iTunes Regression of Cox proportional hazard model

Variable	Entire Apps	Books & Reference	Business	Education	Entertainment	Finance	Games	Health & Fitness	Lifestyle	Medical	Music & Audio	News & Magazines	Photo & Video	Productivity	Social Networking	Sports	Trans & Navigation	Travel	Utilities & Tools	Weather
Ln(number of ratings)	-0.47	-0.697	-18.829	-1.275	-0.518	-25.341	-0.139	-19.466	-18.933	-18.856	-0.998	-0.996	-0.371	-0.766	-0.913	-0.84	-16.904	-19.725	-0.842	-24.524
	(0.018)***	(0.189)***	-3,679,631.01	(0.204)***	(0.065)***	-1689763747	(0.017)***	-9,108,248.63	-7,177,634.02	-6,279,357.73	(0.196)***	(0.177)***	(0.131)***	(0.140)***	(0.168)***	(0.134)***	-1,098,767.23	-9,063,743.44	(0.142)***	-1725479487
Ln(weekly incremental number of ratings)	-0.164	-0.142	-0.649	0.395	-0.168	-0.391	-0.104	-0.222	-0.307	-0.23	-0.272	-0.267	-0.109	-0.37	-0.124	-0.253	-0.672	-0.463	-0.212	0.28
	(0.008)***	(0.042)***	(0.062)***	(0.067)***	(0.026)***	(0.060)***	(0.013)***	(0.052)***	(0.094)***	(0.080)***	(0.047)***	(0.058)***	(0.059)*	(0.064)***	(0.049)**	(0.048)***	(0.103)***	(0.067)***	(0.051)***	(0.086)***
Monthly Major Updates	-1.21	-0.832	-1.423	-1.032	-1.105	-1.321	-1.515	-1.278	-1.264	-2.386	-2.001	-1.596	-0.872	-1.543	-0.949	-1.111	-1.046	-1.497	-1.564	-1
	(0.042)***	(0.172)***	(0.127)***	(0.163)***	(0.153)***	(0.205)***	(0.122)***	(0.257)***	(0.255)***	(0.376)***	(0.238)***	(0.183)***	(0.274)***	(0.242)***	(0.177)***	(0.177)***	(0.249)***	(0.150)***	(0.196)***	(0.531)*
Monthly Minor Updates	-0.159	-0.058	-0.198	-0.4	-0.106	-0.39	-0.107	-0.28	-0.203	-0.16	0.21	-0.07	-0.031	0.114	-0.186	-0.284	-0.244	-0.022	0.041	-0.572
	(0.011)***	-0.045	(0.030)***	(0.061)***	(0.041)***	(0.069)***	(0.023)***	(0.068)***	(0.085)**	(0.067)**	(0.047)***	-0.048	-0.075	(0.056)**	(0.056)***	(0.054)***	(0.070)***	-0.04	-0.038	(0.146)***
Predicted cross entry probability	4.143	5.979	3.645	3.971	4.522	3.492	4.134	3.407	3.844	3.288	4.588	3.452	5.605	3.923	3.398	3.623	4.022	4.262	4.787	3.048
	(0.013)***	(0.058)***	(0.040)***	(0.040)***	(0.049)***	(0.063)***	(0.039)***	(0.067)***	(0.069)***	(0.066)***	(0.075)***	(0.061)***	(0.164)***	(0.095)***	(0.075)***	(0.059)***	(0.111)***	(0.055)***	(0.062)***	(0.164)***
Monthly Major Updates after cross entry	1.09	0.771	1.202	0.04	0.898	1.471	1.679	1.258	1.065	3.125	2.054	1.447	0.944	2.489	1.737	1.263	1.328	1.276	1.536	-0.443
	(0.051)***	(0.231)***	(0.155)***	-0.186	(0.181)***	(0.275)***	(0.140)***	(0.298)***	(0.287)***	(0.417)***	(0.292)***	(0.233)***	(0.382)**	(0.310)***	(0.219)***	(0.232)***	(0.333)***	(0.172)***	(0.237)***	-0.671
Monthly Minor Updates after cross entry	0.703	0.506	0.746	1.469	0.649	0.711	0.642	0.867	0.857	0.28	-0.017	0.62	0.41	-0.177	0.411	0.786	0.821	0.365	0.4	2.3
	(0.023)***	(0.115)***	(0.067)***	(0.096)***	(0.077)***	(0.154)***	(0.053)***	(0.122)***	(0.142)***	(0.139)**	-0.123	(0.117)***	(0.183)**	-0.152	(0.105)***	(0.113)***	(0.169)***	(0.076)***	(0.099)***	(0.318)***
N	792,374	115,958	44,088	83,152	85,313	17,837	137,398	20,458	23,626	16,569	32,150	21,619	19,349	23,686	16,772	25,255	13,388	39,697	52,585	3,474

* p|0.1; ** p|0.05; *** p|0.01