Updates Management in Mobile Applications. iTunes vs Google Play.[*]

Stefano Comino[†], Fabio M. Manenti[‡] and Franco Mariuzzo[§]

May 18, 2015

Abstract

In September 2014, more than 1.3 million apps were available in Apple iTunes and Android Google Play stores. Very low entry barriers and an extremely high degree of competition characterize these markets. In such environment one of the critical issues is how to attract the attention of users. In this paper we focus on a specific strategy that app developers may use to stimulate demand for their products: versioning management. Practitioners and developers recognize that managing app updates (i.e., releasing new versions of an existing app) is critical to increase app visibility and to keep users engaged, disguising a hidden strategy to stimulate downloads. We develop a stylized theoretical model to describe why and when updates should be released. We then use an unbalanced panel with characteristics on the top 1,000 apps on iTunes and Google Play stores for five European countries to empirically test our theoretical predictions. Our results confirm that updates boost downloads and are more likely to be released when the app is experiencing a poor performance. We interpret this finding as evidence that app developers use updates as a "bet for resurrection" strategy.

Keywords: mobile applications, updates, downloads, iTunes, Google Play, visibility, minor updates, major updates.

JEL Codes: L10, L63, M31.

[*]PRELIMINARY AND INCOMPLETE

[†]Dipartimento di Scienze Economiche e Statistiche, Università di Udine, Udine (Italy).

[‡]Dipartimento di Scienze Economiche ed Aziendali "M. Fanno", Università di Padova, Padova (Italy). Email: fabio.manenti@unipd.it.

[§]School of Economics & CCP, University of East Anglia, Norwich (UK).

1 Introduction

The market for mobile applications is one of the most dynamic segments in today's economy. In December 2014, more than 3 Million apps were available on the various stores which include Apple's iTunes, Android's Google Play and Microsoft's Windows Store. Looking at iTunes only, in 2014 the number of apps has grown nearly 60%, from 890,000 available apps on 1/1/2014 to over 1.42M on 12/31/2014;[1] The growth in the number of available apps has been accompanied by an exponential increase in downloads. According to Statista.com, the cumulative number of apps downloaded from iTunes from July 2008 to October 2014 was about 85 billions. Also in terms of the number of developers and publishers involved in the app market, figures are quite impressive: according to Priori, in February 2014 more than 600 thousand developers published at least one app on iTunes or Google Play, with an increase of nearly 10% with respect to the previous month. Indeed, producing and distributing a mobile application for a developer is relatively easy as it requires a small fraction of investment to produce a traditional computer software.

The market for mobile apps is therefore characterized on the one side by a fast growing demand from users and by low entry costs, large number of apps and developers, high turnover rates on the other. For these reasons the app market has been defined as an "hyper-competitive" marketplace where developers struggle to attract adopters (see Datta and Sangaralingam, 2013). In this dynamic context, with several millions of apps available for download, one of the most challenging problems faced by developers is to catch the attention of users (see Bresnahan et al., 2014).

In order to improve app visibility, stores created the so-called "top-ranked" apps charts, listing the most popular applications. Several scholars have shown that such charts favor the matching between users and developers with top-ranked apps enjoying a remarkable boost in downloads (Carare, 2012; Ghose and Han, 2014; Ifrach and Johari, 2014; Garg and Telang, 2014). In turn, top ranked-charts exacerbate another feature characterizing app markets that is the skewness of the distribution of downloads. According to www.appbrain.com, on Google Play about 1 million apps out of 1.4 millions have less than one thousand downloads each while, on the contrary, just thirteen thousand apps have more than one million downloads. Being in the top positions guarantees success. Therefore, the distribution of downloads is extremely skewed to the right with only a small fraction of applications capturing most of the market.

To climb the top-ranked charts, publishers and developers look for any possible strategy to improve the visibility of their apps. App promotion, marketing and pric-

[1] Data taken from www.adjust.com.

ing are among the typical strategies that developers use to get noticed by customers. In this article we focus on another strategy that developers may exploit to attract users' attention, that is, the release of frequent updates.

Our dataset confirms that both in iTunes as well as in Google Play developers release updates with an extremely high frequency: in Google Play apps are updated on average every 28 days, while in iTunes this occurs every 59 days. It is widely recognized that by frequently releasing new software versions, developers are able to increase user engagement. By releasing updates developers stimulate user interest, thus improving app visibility. On top of this, developers usually promote the new versions of their products on blogs, social networks or simply in the "What's new" section of the app store. Again, the update may represent an effective tool to stimulate app visibility and, via this channel, sales.

We analyze the strategy of releasing frequent updates by exploiting the differences across the two main app stores, iTunes and Google Play. Interestingly, the two stores follow different policies regarding the publication of apps and updates. The iTunes "App store review guidelines" explicitly sets a strict screening of apps quality. For example, apps that exhibit bugs or that are in a beta/trial version are going to be rejected by the store. Similarly, applications that are considered not very useful or not providing any lasting entertainment value to users are not published. On top of this, apps that include undocumented or hidden features inconsistent with the description of the app are rejected as well. By contrast, publication on Google Play does not go through a similar quality check; updates can be published instantaneously by developers with a "simple click of the mouse". The absence of a formal screening has led several commentators to criticize Google Play for the poor quality of the apps available in the store. This issue is so critical to Google that it has stepped up its efforts to improve apps quality. For instance, in February 2013, it has removed from its stores 60,000 spammy and low quality apps at once. Similarly, in October 2014 it launched a new feature that allows users to filter out all apps that are not rated at least 4 stars.[2]

In this paper, we present a stylized theoretical model investigating the developer's decision about whether to update her mobile application. We then use an unbalanced panel with characteristics on the top 1,000 apps on iTunes and Google Play stores for five European countries to empirically test the predictions we derive from the theoretical model. Interestingly, we find that the release of an update positively affects downloads in iTunes while it has no significant impact in Google Play. We interpret this finding in terms of the institutional differences characterizing the two stores. As argued above, only in iTunes there is a strict quality check for apps and

[2] www.appbrain.com estimates that nearly 15% of Google Play apps are of low quality.

updates. Therefore, the absence of a significant impact of updates in Google Play might be due to the fact that in this store both high and low-quality updates get published so that the overall effect of downloads is not significant. Another interesting prediction of our model that is confirmed by the empirical data relates to the conditions under which the developers finds it profitable to release an update. As we show in the paper, an update is more likely to be released when the developer observes a worsening of its performances. Hence, in order to stimulate the attention of potential users, and "revive" the app, developers are induced to release a new version of the software. Given that the release of a new version might be risky, we interpret this finding as a sort of "bet for resurrection" strategy that developers employ when observing their apps performing poorly. We conclude our analysis by distinguishing in iTunes between major (significant changes in app functionalities) and minor updates (bug fixing and minor changes); our empirical investigation suggests that the latter are more likely to be employed by developers as a strategic tool to improve app performances.

The paper is organized as follows. In Section 2 we discuss the relevant literature on mobile apps. In Section 3, we present the theoretical model and we derive the testable predictions. Sections 4 and 5 are devoted to presenting the data and the empirical strategy we employ in the estimations. The results of the empirical investigation are discussed in Sections 6 and 7. Section 8 concludes.

2 Literature review

The astonishing growth of mobile ecosystems has attracted the attention of several scholars. Despite this is a recent phenomenon, the literature that studies the characteristics and the functioning app markets is already quite developed. As we have pointed out above, the number and the type of apps available to users is extremely large. This raises the issue of how to attract the attention of users in order to emerge from the mass of hundreds of thousands of applications. According to Bresnahan et al. (2014) this is the crucial issue for app developers. The authors argue that app stores play a dual role: they lower the technical costs of developing and distributing applications, but set up very high marketing costs for developers. In app stores, competition is pervasive and it does not emerge only among apps performing the same or similar tasks. Each app competes for consumer attention with all the other applications available in the store. For this reason, how to become visible to consumers is the most interesting issue issues discussed in the literature.

The first paper studying the demand for mobile applications is Ghose and Han (2014). The authors quantify the consumer preferences for different app characteristics based on a structural model that combines a multinomial nested logit demand model with the pricing equations of software developers. The analysis is based on daily information on the top-400 free apps and the top-400 paid apps in Apple iTunes and Google Play platforms. The authors estimate downloads by means of a calibration exercize, relating apps ranking with the number of downloads. They authors find that demand is larger when app description is more accurate (measured in terms of description length and number of screenshots); when the app has the in-app purchase option, and when it is older (they measure both the age of the app and of the version). They also find that the demand increases with the number of apps available from the same developer, and when the app is available on multiple platforms (iTunes and Google Play). Users' reviews are also found to play a significant role as their volume and valence stimulates downloads. A key result which is relevant to our investigation is the finding that demand is boosted by the number of previous versions of the same app.

Interestingly, Ghose and Han find that cross-charting (namely, app appearing both in the top-free as well as top-paid charts) has a positive impact on app demand. This suggests that being in the list of top-ranked apps may have a valuable effect in terms of stimulating additional downloads. Carare (2012) investigate in details the role top-ranked charts in stimulating future app demand. The work is based on the top-100 paid apps available in the US iTunes store. The author shows that the bestseller status of the top-ranked apps is a very important determinant of consumer willingness to pay, and that the effect of rank declines very steeply for the top 10 apps and becomes negligible for apps ranked higher than 50.

The strategy of releasing frequent updates is not only specific to mobile applications but it is also commonly observed in the "traditional" desktop computer software (see Greenbaum, 2005). This occurs especially when the software package has reached a high level of penetration so that little revenues can be collected from new customers. Software firms are therefore induced to release upgraded versions of their packages in the attempt to re-sell the software to their installed base of users.[3] It is worth noticing that this strategy cannot explain the large number of updates featured by mobile developers. As a matter of fact, a common rule in app stores is that updates must be made available free of charge to anyone who has previously

[3]This strategy may give rise to a classical Coasian commitment problem in durable goods which damages producers. Sankaranarayanan (2007) suggests that software vendors may overcome the Coasian problem contractually by entitling customers to any update they are going to release during a predetermined period of time. Within this period, vendors are unable to collect revenues from their installed base of users, a constraint that decreases substantially the temptation to release updated versions.

downloaded the app. As a consequence, developers cannot exploit their installed base of users trying to re-sell upgrades of their software.

In the next section we sketch a simple model to explain the strategic role of version updates.

3 A theoretical framework: The decision to update

In the market for apps, developers compete for consumer attention. With the huge mass of software available for downloads, success depends heavily on how "visible" an app is. Visibility might be related to the ranking the app achieves within the top-ranked charts or it might be related to what we call the "buzz" surrounding the app, i.e. how much blogs, social networks or specialized magazines and websites "talk" about the app.

The ability of a developer to attract the attention of potential users depends on a number of elements. The intrinsic quality of the software code written by the developer and the ability to meet consumer needs/tastes are certainly crucial for app success. The prestige of the developer and/or the recognition of the brand are also very important factors; however, in order to emerge from the mass of available apps, the ability of the developer to attract the interest of bloggers and journalists of specialized magazines aimed at stimulating the buzz is also essential to succeed.

In the following pages, we present a very stylized model that studies the choice of a developer about whether or not to release a new version of her app. Following our previous discussion, the key features of the model are: *i*) downloads depend both on the intrinsic quality of the software and on the buzz surrounding it, *ii*) demand/downloads are right skewed and *iii*) the release of an updated version stimulates the buzz around the app.

3.1 The model

Consider a developer who has already published two apps in the store. The developer has to decide whether to update one of their apps (that we indicate as the "focal" app). In taking this decision the developer aims at maximizing downloads, net of further possible costs to update the software. Let v denote the visibility of the focal app perceived by potential users. We model visibility as the sum of two components: the intrinsic/true quality of the software, q, and the impact of what we call the "buzz" around the app, b. Both software quality and the "buzz" depend on the released version. Formally, we express it as $v(u) = q(u) + b(u)$, where $u = 0$ for the current version of the app, and $u = 1$ in the case the developer releases an update. An app can be highly visible if its intrinsic quality is high and/or it is surrounded by a positive buzz (i.e. good users reviews, positive discussions on dedicated blogs, etc.).

The crucial assumption of the model is that the decision to release an update stimulates the buzz surrounding the app and this buzz can either be positive or negative. These assumptions are taken on practical grounds; updates tend to stimulate discussions or comments in dedicated blogs/magazines/websites or on social networks but these discussions can go either way: bloggers, journalists, but also regular users might positively or negatively welcome the new features added through the update. In other words, the increased buzz may improve or worsen app visibility. Formally, we assume that an update makes app visibility more uncertain.

In what follows, b is assumed as a realization of a random variable while q, for the sake of simplicity, is assumed to be deterministic.

According to the empirical evidence outlined in the introduction, we assume that the number of downloads is highly skewed on the right. Formally, we assume that there exists a threshold level τ for app visibility such that:

$$\begin{array}{ll} \text{if } v \geq \tau & \text{downloads are } \overline{D} = \overline{d} + \rho\overline{d}, \\ \text{if } v < \tau & \text{downloads are } \underline{D} = \underline{d} + \rho\underline{d}, \end{array}$$

where $\overline{d} > \underline{d} \geq 0$. $\overline{d}$ and $\underline{d}$ represent the amount of downloads of the focal app while $\rho\,\overline{d}$ and $\rho\,\underline{d}$ measure the impact of downloads of this app on the other app distributed by the developer. In other words, $\rho\,d$ indicates the increase/decrease in the other application downloads due to the performance of the focal app. The two apps can be either complements ($\rho > 0$) or substitutes ($\rho < 0$). Therefore, an increase in downloads of the focal app can either contribute to stimulate downloads of the other app or it can cannibalize this latter. Complementarity may emerge from a "branding effect" or from cross advertising between the two apps. On the opposite, substitutability may occur when the two apps address the same or similar user needs: a crowding out effect.

The current version of the focal app has visibility $v(0) = q(0) + b(0)$, where $b(0)$ is the realization of a random variable uniformly distributed over the segment $(B - \varepsilon, B + \varepsilon)$, with density function $f(b(0)) = 1/(2\varepsilon)$. Total expected downloads/payoff generated by the current version of the focal app amount to:[4]

$$\left(\int_{B-\varepsilon}^{\tau-q(0)} \frac{1}{2\varepsilon} db(0)\right) \underline{D} + \left(\int_{\tau-q(0)}^{B+\varepsilon} \frac{1}{2\varepsilon} db(0)\right) \overline{D} = \frac{\left(\overline{D}-\underline{D}\right)\left(B-\tau+q(0)\right)}{2\varepsilon} + \frac{\overline{D}+\underline{D}}{2}. \tag{1}$$

In the case the focal app is updated, the developer incurs the (development) cost ϕ; the new version of the software has visibility $v(1) = q(1) + b(1)$, where:

- $q(1) = q(0) + \Delta$, with Δ either positive or negative. In other words, the new version of the software may have a higher or smaller intrinsic quality, $\Delta > 0$ and $\Delta < 0$ respectively;[5]
- $b(1)$ is the realization of a random variable uniformly distributed over the segment $(B - \gamma\varepsilon, B + \gamma\varepsilon)$, with $\gamma \geq 1$, according to the density function $f(b(1)) = 1/(2\gamma\varepsilon)$. Following the above discussion, we assume that the release of the update stimulates the buzz around the app, thus increasing the uncertainty about its visibility ($\gamma \geq 1$).

Based on the above assumptions, the expected payoff associated with the decision to release an update is:[6]

$$\left(\int_{B-\gamma\varepsilon}^{\tau-q(0)-\Delta} \frac{1}{2\gamma\varepsilon} db(1)\right) \underline{D} + \left(\int_{\tau-q(0)-\Delta}^{B+\gamma\varepsilon} \frac{1}{2\gamma\varepsilon} db(1)\right) \overline{D} - \phi = \frac{\left(\overline{D}-\underline{D}\right)\left(B-\tau+q(0)+\Delta\right)}{2\gamma\varepsilon} + \frac{\overline{D}+\underline{D}}{2} - \phi. \tag{2}$$

In this case, the payoff is composed of two elements, the expected total downloads generated by the release of the new version of the software, and ϕ, the cost of developing the update.

By comparing (1) with (2) one can easily determine the condition under which the developer chooses to release an update:

[4]We focus on the most interesting case where both $v(0) \geq \tau$ and $v(0) < \tau$ occur with positive probability. Therefore, parameter t satisfies the condition $q(0) + B - \varepsilon < \tau < q(0) + B + \varepsilon$.

[5]The case $\Delta < 0$ might occur when the new version comes with bugs or when it includes new features that users do not appreciate or that worsen the usability of the software.

[6]We assume that also when the update is released both $v(1) \geq \tau$ and $v(1) < \tau$ occur with positive probability. Therefore, parameter τ satisfies the condition $q(0)+B+\Delta-\gamma\varepsilon < \tau < q(0)+B+\Delta+\gamma\varepsilon$.

Proposition 1 *The developer releases a new version of the focal application when the increase in total expected downloads is higher than the cost of developing the update; formally,* $u = 1$ *when:*

$$(\overline{D} - \underline{D})[(\tau - q(0) - B)(\gamma - 1) + \Delta] > 2\gamma\varepsilon\phi. \tag{3}$$

A necessary condition for the release of a new version of the software is that the update generates an increase in total expected downloads. Formally, this occurs when the term into the square brackets of expression (3) is positive. A simple inspection of (3) reveals that this is more likely to happen when the expected visibility of the current version of the focal app $(q(0) + B)$ is small relative to the threshold level τ.

In other words, when the developer expects a poor performance of the current version of her app, she might be induced to release an update in the hope of stimulating positive buzz around it and, via this channel, downloads. This is a risky strategy as app visibility becomes more uncertain. This is why we reinterpret the decision to release the update as a sort of "bet for resurrection" strategy. Clearly, this decision is profitable provided that the development cost ϕ is small compared to the increase in expected downloads.

Looking more closely to expression (3), it is possible to verify that an update is more likely to be published whenever its intrinsic quality is large (Δ is large) and when the impact on downloads, $\overline{D} - \underline{D}$, is sizeable. This latter condition is more (less) likely to occur when the apps of the developer are complements (substitutes), that is, when $\rho > 0$ ($\rho < 0$).[7]

An interesting implication of Proposition 1 is the following:

Corollary 1 *The developer may decide to release an update even if it does improve the intrinsic quality of the app,* $\Delta \leq 0$.

Proof. Suppose that $\Delta = 0$. A necessary condition for inequality (3) to be satisfied is $\tau > q(0) + B$. Notice that with $\Delta = 0$, the model is defined for $\tau \in (q(0) + B - \varepsilon, q(0) + B + \varepsilon)$. Therefore, when $\tau \in (q(0) + B, q(0) + B + \varepsilon)$, condition (3) holds provided that $\overline{D} - \underline{D}$ large enough/ϕ small enough. By continuity, it follows that updates with lower intrinsic quality ($\Delta < 0$) might also be profitable.■

[7] A larger γ (greater variance of the visibility of the update) may induce the developer to release the update more or less often depending on the value of the threshold. When $\tau > q(0) + B + \Delta$, then a larger γ makes condition (3) more likely to be satisfied. In other words, when the expected perceived quality of the app is very low with respect to the threshold τ, then an increase in γ makes the update more profitable. By contrast, when τ is smaller than $q(0) + B + \Delta$, then a larger γ makes condition (3) less likely to be satisfied.

3.2 Testable predictions

The theoretical model can be used to derive some testable predictions. The first prediction follows from Proposition 1 according to which the developer releases an update whenever the expected visibility of the current version is low compared with the threshold. It is reasonable to assume that the developer forms expectations on the visibility of the current version of her app by looking at its past performance. In this case, Proposition 1 suggests that:

Conjecture 1 *Developers are more likely to release an update when they observe a worsening of the app performance.*

As discussed above, condition (3) is more likely to hold when the developer distributes more than one complementary application. On the contrary, when apps are substitutes, developers are less prone to update. Based on these arguments, we predict that:

Conjecture 2 *Developers distributing more than one application are more (less) likely to release updates when apps are complements (substitutes).*

According to Corollary 1, developers may also decide to release qualitatively worse updates, hoping to stimulate downloads via the effect on buzz. This corollary suggests an interesting prediction related to the institutional differences between the iTunes and Google Play stores. As we argued in the introduction, while iTunes has a formal quality check for publishing apps and related updates, such a check is not implemented in Google Play. Therefore, in principle, quality-worsening updates can be released in Google Play but they cannot in iTunes. This amounts to say that Corollary 1 may apply only to Google Play apps; based on this reasoning we expect that:

Conjecture 3 *The effect of the release of an update on downloads is stronger in iTunes than in Google Play.*

The theoretical model assumes that developers base their choice about whether to release an update just looking at future downloads (and development costs). Therefore, in the model, we implicitly assume that developers only profit from new users of their apps. This is a well taken assumption given that developers must make new versions of the software available free of charge to earlier adopters of the app. However, a common commercial strategy used by developers is to include so-called in-app purchases, that is the option to pay in order to access improved functionalities of the software (or, in the case the app is a game, access to more challenging parts of the story). When apps come with the in-app option, developers profit not only from new consumers but also from the installed base of users. In this case, it is even more compelling for developers to adopt strategies that increase users' engagement. In terms of updating strategies, when the app has the in-app purchase option, this translates into stronger incentives to release new versions. Therefore, we expect that:

Conjecture 4 *Developers of applications with the in-app purchase option are more likely to update their apps.*

4 The data

We obtained monthly data on the top 1,000 most downloaded apps in iTunes and Google Play from the consulting analytics Priori. Data refer to five European countries (Germany, France, Italy, UK and Spain) for the period September 2013, February 2014. According to Priori (2014), the top 1,000 apps represents around 60% of the market in each country (e.g. in October 2013 our data cover 55.3% of downloads in iTunes in UK and 62.09% in Italy). We then complemented these data with additional information publicly available on AppAnnie.com, an analytics consulting specialized in mobile apps.

For each app, the Priori dataset provides the following information: name of the app, name of the publisher, app monthly and overall number of country downloads, the worldwide average customers rating of the app (in a scale from 1 to 5), the number of users' ratings, the date when the app has been seen the first time and the overall number of updates released at the end of the month, the price of the app, when suitable, and whether the app has in-app purchases.

All the information gathered by Priori is taken directly from the app stores except for the number of downloads which is obtained combining publicly available information (financial statements and other reputable or verified press sources) with Priori proprietary metrics establishing a relationship between downloads and user ratings, ranking (i.e. position in the app stores top-ranked charts) and number of reviews. For a sample of apps, Priori then cross-checked these estimates by using real downloads data provided by partner developers.[8] It is important to stress that

[8]According to Priori's statements, this internal validation study based on 2,000 Android apps returned mean absolute error of +/- 24.6%. It is important to stress that in our regression analysis we employ the growth rate of downloads rather than the absolute level and this should mitigate concerns related to their under/overestimation.

only first-time installations are considered as downloads; in other words, downloads of updates of already installed applications are not counted.

For iTunes apps, we complemented these with additional information taken from the AppAnnie web site; specifically, for each app we collected the following information: the type of compatibility (app for iPhone/iPod touch, for iPad only, for iPhone only or Universal), the size of the code (data collected in May 2014), the age restriction (4+, 9+, 12+, 17+) and the country language, that is whether the app is available in the language of the country or not. On top of this, we identified in the sample the so-called corporate apps, namely apps used by companies as an additional distribution channel (e.g. airlines or banking apps, newspaper or TV network apps etc.) or to provide information about their services. We also collected additional information on the monetization strategy followed by the publisher; for each free app we checked whether the publisher also distributes a pro version or, in case of a paid app, a free version (usually with few functionalities or with in-app advertising).

Finally, for each iTunes app we also gathered information regarding the type of the update. Conventionally, software developers keep track of the different versions of their products by means of a three-digit sequence, where the first digit identifies major updates and the second and third digits minor updates of decreasing significance. So, for example, the current version of a given app can be 2.12.4 meaning that there have been two major updates and sixteen minor ones (12+4). Developers may choose to jump multiple minor versions at a time to indicate that significant features have been added, but these features are not enough to warrant incrementing the major version number; for example an app may jump directly from 3.1.2 to 3.6.0. It is customarily to consider minor updates new versions of the software aimed at fixing bugs (e.g. crashing) or at adding minor features while major updates are aimed at distributing software with significant jumps in functionalities.

For each iTunes app in the sample we are able to distinguish between minor and major updates and for each major update when it was published in the store. It is important to note that, once published, updates are available worldwide, namely they are not country-specific.

4.1 Descriptive statistics

Table 4.1 provides summary statistics of the top 1,000 most downloaded apps in the two stores in Germany in September 2013; for the sake of brevity, we omit to show the evidence for the other countries and months as it is qualitatively very similar. From this table it follows that:

Table 1: Statistics Germany, Sept. 2013

	Google Play		iTunes	
	mean	std. dev.	mean	std. dev.
free	0.998	0.045	0.894	0.308
price	2.212	1.322	3.423	0.594
apps with in-app	0.390	0.488	0.609	0.488
age (in months)	16.365	13.996	18.772	15.046
updates	42.485	72.404	10.576	9.836
major versions			1.950	1.877
size			83.472	188.110
apps same dev.	11.289	21.184	14.343	25.366
monthly downloads	101,998.2	263,446.1	32,069.75	53,361.21
local	0.228	0.4120	0.248	0.432

1. in Google Play nearly all the top 1,000 apps are free, with only two paid apps; also in iTunes free apps are prevalent although paid apps are more frequent than in Google Play (10.6% of the observations have a positive price);

2. there is a significant difference between the two stores in terms of the number of apps with in-app purchases: 39% of the Google Play sample has in-app purchases while in iTunes the same occurs in 60.9% of apps;

3. iTunes apps are on average older than apps in Google Play, thus suggesting a higher turnover rate in the top 1,000 apps in the latter store;

4. on both stores, apps are updated quite frequently with Google Play apps that are updated about four times more than the iTunes ones. On average in Google Play an app is updated about 42 times since its publication while this figure reduces to about 11 in iTunes. This difference may be due to the aforementioned divergent policies towards the publication of apps and updates implemented by the two stores; the absence of a strict quality check may partially explain why we observe so many more updates in Google Play. A possible additional explanation to these figures is related to the fact that Google Play apps run on the Android mobile operating system which can be installed on several different devices; this may require a closer management of updates by developers;

5. downloads are much higher in Google Play than in iTunes, with Google Play apps that are on average downloaded three times more than iTunes apps;

6. roughly, between 20 and 25% of apps are local; an app is named as local in a given country when at least 40% of its all time downloads occurs in that country.

These figures provide a picture of our data-set in September 2013. As regard the dynamics of our sample during the period September 2013 - February 2014, some interesting observations emerge.[9] The first one relates to the number of times an app is updated. On average, apps which keep staying in the top 1,000 most downloaded list all through six-month period of observation are updated 3.5 times in iTunes and about 23 times in Google Play, thus confirming that the number of updates is substantial in both stores, and particularly large in Google Play. When focussing on the type of updates, we find that about 1 out of 5 apps in iTunes has released a major update during the period of observation. Hence, as one might expect, most of the updates are minor ones and major changes to the software code represent a small share of the updates.

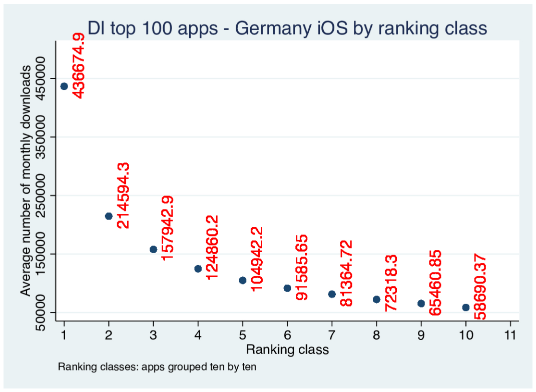
Our data confirms a couple of features that have already been found in the literature (see, among others, Bresnahan et al., 2014): *i*) downloads exhibit an extremely skewed distribution, with top apps accounting for a large fraction of total downloads, and *ii*) large turnover/churn, with few applications which succeed in staying in the top 1,000 list in all the six months of observation. As regard feature *i*), Figure 1 shows the distribution of downloads for the top 100 apps in Germany for iTunes and Google Play; the two diagrams show the average number of downloads for each decile of the distribution. In the case of Google Play, for instance, the average monthly downloads of apps of the first decile is 436,674.9, which is twice the average number of monthly downloads of apps in the second decile (214,594.3) and about eight times more of the downloads of tenth decile (58,690.337).

As regard feature *ii*), Table 4.1 shows the high level of turnover that characterizes iTunes. The overall number of different apps that we observe during the six-month period in the five countries is 10,986;[10] 18.24% of these apps is observed every month (indicated in the table with "All months"). However, a substantial percentage (about 44%) of apps appears only in one month. The level of turnover is even larger in Google Play (table not reported). For the Android store, the overall number of applications that we observe increases to 13,034 and the share of apps that appears only one month is about 50%.

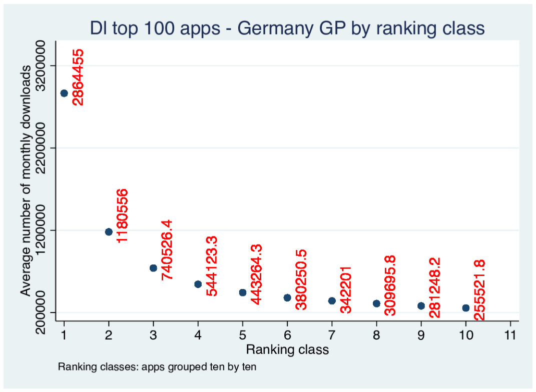
[9]Again, we present data on one country only (Germany); the evidence for the other countries is similar.

[10]The minimum number of apps we could have observed in our sample is 1,000 (the top 1,000 apps are the same in the five countries during the whole period) while the maximum is 30,000 (the top 1,000 apps change every month and are different in the five countries under observation).

Figure 1: Distribution of downloads



(a) iTunes



(b) GP

Figure 2 shows the kernel density of the growth in downloads in the two stores distinguishing between updated and non updated apps.[11] Interestingly, in Google Play the two density functions nearly perfectly overlap; this suggests that updated and not updated apps perform very similarly in terms of downloads. On the contrary, in iTunes the density function of non updated apps is more asymmetric to the left and concentrated on negative values of the growth rate of downloads. This is a very preliminary evidence towards a different role played by updates in the two stores. The aim of our theoretical model and of the econometric exercize is to shed some light on this evidence.

5 The econometric model

We deal with a longitudinal dataset which has a large cross-section of mobile applications (apps), of size J, and limited number of periods and countries, of size T and C, respectively. Ensuing that asymptotics relies on the apps dimension.

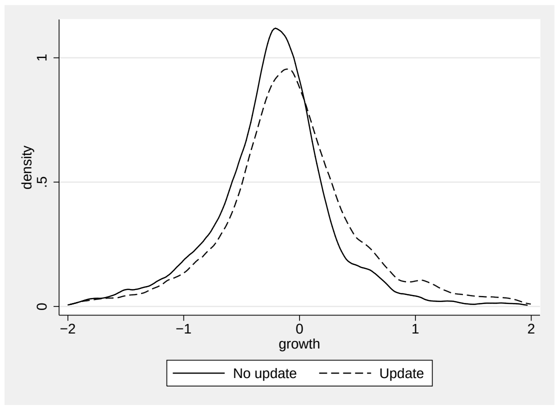
For each store separately, we study whether new updates, captured by a binary variable $\{0, 1\}$, affect the downloads - specifically, the download growth between period t and $t-1$ -. Also, we are interested in unravelling the determinants of the release of updates. We deal with the possible simultaneity between download growth and change in versions (updates). We denote with g_{jct} the download growth rate for

[11] Growth rates are expressed in logarithms and refer to the overall amout of downloads each app obtains in the five countries.

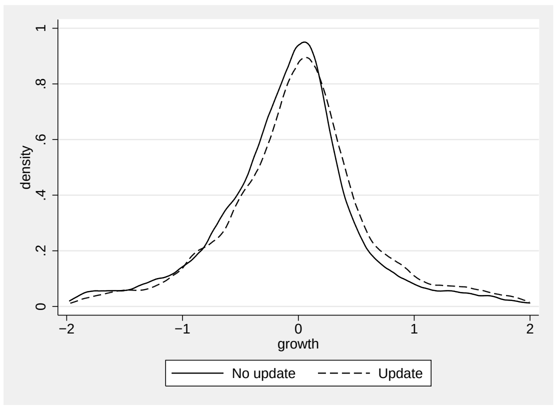
Table 2: Pattern of apps, iTunes, all periods & countries

Freq.	%	Cum.	Pattern
2004	18.24	18.24	All months
1098	9.99	28.24	Sept only
1025	9.33	37.57	Feb only
762	6.94	44.50	Oct only
755	6.87	51.37	Nov only
637	5.80	57.17	Jan only
592	5.39	62.56	Dec only
423	3.85	66.41	Sept & Oct
354	3.22	69.63	Dec,Jan & Feb
3336	30.37	100.00	(other patterns)
10986	100.00		N. of different apps

Figure 2: Download growth density function



(a) iTunes



(b) Google Play

a mobile application $j \in \mathcal{J}_t$ distributed in country $c \in \mathcal{C}$ in the period t. We model the growth rate empirically as as a linear dynamic model made of observable and unobservable mobile application characteristics. One key observable characteristic we focus our analysis on, is the apps versioning update. Updates can either be imposed by the platform, as an attempt to warrant users a minimum quality standard, and/or chosen by developers - possibly as strategy to boost the number of downloads -. Updates are released simultaneously in all countries and, apart for possible language translations are homogeneous. Because updates are country invariant. Hence, we obtain an aggregated measure of the downloads for the list of countries we have data on. Growth for the app j in period t is computed as $g_{jt} = \frac{\sum_{c \in \mathcal{C}}(Q_{jct} - Q_{jc,t-1})}{\sum_{c \in \mathcal{C}} Q_{jc,t-1}}$, treating the number of downloads (contemporaneous and lagged: Q_{jct} and $Q_{jc,t-1}$) as zero if the app is not present in top 1000 ranking in the country c in the relevant period.

We specify both the growth and the update versioning equation as linear autoregressive distributed lag model of order 1. In addition to a set of controls we complement the download growth equation with the contemporaneous impact of versioning update u_{jt} and supplement the update versioning equation with lagged growth, yielding the system of linear equations

$$\begin{aligned} g_{jt} =& \phi_{11} g_{j,t-1} + \phi_{12} u_{jt} + h_{1t} + \mathbf{x}_{1jt}\boldsymbol{\beta}_1 + \alpha_{1j} + \varepsilon_{1jt} \\ u_{jt} =& \phi_{21} g_{j,t-1} + \phi_{22} u_{j,t-1} + h_{2t} + \mathbf{x}_{2jt}\boldsymbol{\beta}_2 + \alpha_{2j} + \varepsilon_{2jt}, \quad t = 2, \cdots, T. \end{aligned} \tag{4}$$

For short panels, as it is the case here, it is common to let the time effect be fixed and hence exclude it from the composite error term, ε_{kjt}, which is made for each equation $k = \{1, 2\}$ by the sum of the unobserved heterogeneity, α_{kj}, and the pure idiosyncratic error term, ζ_{kjt}. We assume the idiosyncratic error to be uncorrelated both over time and apps. We account for the time effect on growth with the function h_{kt}. We include a set of controls in the vector $\mathbf{x}_{kjt}$. Versioning update is endogenous and the unobserved heterogeneity brings in other source of endogeneity.

As both the lagged dependent variable and the current value of the update are expected to be correlated with the unobserved heterogeneity, we follow the dynamic linear panel model literature and take the first difference of both sides of the system of equations (4) to remove inconsistencies of the parameters driven by such correlation, leading to the first differences econometric system of equations of interest

$$\begin{aligned} \Delta g_{jt} =& \phi_{11} \Delta g_{j,t-1} + \phi_{12} \Delta u_{jt} + \tau_{1t} + \Delta \mathbf{x}_{1jt}\boldsymbol{\beta}_1 + \Delta\varepsilon_{1jt} \\ \Delta u_{jt} =& \phi_{21} \Delta g_{j,t-1} + \phi_{22} \Delta u_{j,t-1} + \tau_{2t} + \Delta \mathbf{x}_{2jt}\boldsymbol{\beta}_2 + \Delta\varepsilon_{2jt}, \quad t = 3, \cdots, T. \end{aligned} \tag{5}$$

Though in the structural system of equations (5) we have eliminated the unobserved heterogeneity and the aforementioned issue of correlations, still the system cannot be estimated consistently by OLS. We deal with other sources of correlation. In the first equation we cope with the correlation between $g_{j,t-1}$ and $\varepsilon_{1j,t-1}$ and that between the update variable and growth, which causes $E(\Delta u_{jt}, \Delta\varepsilon_{1jt}) \neq 0$. Similarly in the second equation we tackle the endogeneity between $\Delta\varepsilon_{2jt}$ and $\Delta u_{j,t-1}$, in addition to the endogeneity between $\Delta\varepsilon_{2jt}$ and $\Delta g_{j,t-1}$.

In the next subsection we discuss the instruments that we correct for such multifaceted endogeneity.

5.1 Instruments

To identify the download growth dynamics and the effect of an update on growth, along with the update dynamics and the impact of lagged growth on the update - first and second equation in (5) - we select a set of instruments that are *strong*, i.e. explain the endogenous variables in each equation, but do not explain the dependent variables directly, if not via the other explanatory variables that determine the dependent variable, that is, are *valid*. To select the instruments that satisfy validity and strength we exploit the time series dimension of the panel and the multiproduct position of a sizeable number of developers.

Underneath we describe the main successful instruments and refer the complete list, supported by relevant tests on the validity and strength of the instruments, to the footnote of the results table.

- We instrument $\Delta g_{j,t-1}$ with $g_{j,t-2}$, as suggested in Anderson and Hsiao (1981), and with the first difference of average lagged growth of the other $J_{f,t-1} - 1$ apps distributed in top positions (top-1000 apps) and in at least one of the countries of investigation by the developer f. The instrument for the for the first difference of lagged growth of app j is calculated as $\Delta \frac{\sum_{l \in (\mathcal{J}_{f,t-1} - j)} g_{l,t-1}}{J_{f,t-1} - 1}$. For cases where the developer has marketed only one app in top positions ($J_{f,t-1} = 1$), the instrument takes value zero.

- We instrument the update variable, Δu_{jt}, with the second difference of average update (as the first difference raised issues of validity) of the other $J_{ft} - 1$ apps distributed in top positions (top-1000 apps) and in at least one of the countries of investigation by the developer f. The formula for this instrument is $\Delta^2 \frac{\sum_{l \in (\mathcal{J}_{ft} - j)} u_{lt}}{J_{ft} - 1}$. Here, the variable takes the mid-point value between zero and one, which is one-half, if the app is the only top-app distributed by the developer in one of the relevant countries ($J_{ft} = 1$). Additional instruments that we use are: the first difference of the lagged age of the version, the first difference of lagged number of versions, and the first difference of the average age of the version.

This logic of instrumentation is extended to the endogenous variables of the update equation. As stated earlier we document the full list of instruments in the footnote of the results table.

In the next section we discuss the main results.

6 Results

Columns (1) and (2) of Table 3 show the estimates of the first equation of the system (5), both for iTunes and Google Play respectively. For iTunes, the third column highlights the estimates when distinguishing between major and minor updates. From the table it follows that:

1. the past performance of the app, measured in terms of the growth rate of downloads during the previous month, has a negative and statistically significant effect on the current rate of growth. This result seems to suggest that downloads follow a cyclical pattern with alternating periods of growth and downturn;

2. consistently with Conjecture 3, we find that the release of an update has quite different effects in iTunes and Google Play. In iTunes the growth rate of downloads is positively affected by the variable *Update*, meaning that the release of an update boosts downloads; quantitatively we find that having released an update during the previous period increases the rate of growth of downloads by about 36%. By contrast, the publication of an update in Google Play does not have a significant impact. This evidence might seem quite surprising but it can be interpreted on the basis of our theoretical model. Corollary 1 suggests that developers might be willing to release low quality updates in order to stimulate the buzz surrounding the app; this strategy can be implemented only in Google Play, where developers are not subject to any screening concerning the quality of their applications. From these considerations the non significance of the *update* coefficient in Google Play might be due to the fact that developers release both high and low quality updates and, on average, they do not significantly impact on downloads. On the contrary, the strict quality check in iTunes ensures that only high quality updates get to be published in the store, thus explaining the positive impact of the variable *Update*;

3. both in iTunes as well as in Google Play, the growth rate of downloads increases with the number of other applications of the same developer listed in the top 1,000; the number of other applications is measured at the country, month and store level. This result reveals the presence of complementarities among apps that developers might exploit through to "branding effect" or cross-advertising;

4. downloads of Google Play apps are negatively affected by the presence of the in-app purchase option while they are not affected in iTunes. Possibly, the first result can be interpreted as the classical negative effect of price on demand.

This observation is reinforced by the fact that almost all Google Play apps in our sample are free and the in-app purchase option represents the only form of pricing;

5. in iTunes, free apps are characterized by a larger growth rate of downloads.[12]

The estimates of the second equation of the system (5) about the determinants of updates are in columns (4) and (5) of Table 3. The theoretical model developed in Section 3 suggests that the main determinant of the developer's decision to release a new version of her app is the past performance, here measured in terms of growth in downloads, gdl_{t-1}.

We find that the past performance does affect the decision to release an update only in iTunes. In this platform, the coefficient of the variable gdl_{t-1} is negative and statistically significant, meaning that, other things equal, a poor performance of the app in the previous period makes more likely for the developer to release an update. This amounts to say that our data supports Conjecture 1 with app update being an effective strategy to react to downturns in downloads. The same does not occur when considering Google Play where past performance does not have an impact on the decision to release an update. This difference between iTunes and Google Play can, again, be reinterpreted on the basis of their different way to govern the release of updates. As mentioned, iTunes apps must pass a quite severe quality control and this limits the freedom of developers to publish updates; as a consequence, developers who have written an update and are ready to publish it on the store may decide to delay publication until when they really need it, that is when the performance of the app is poor and triggers an action to counter the drop in downloads. On the opposite, in Google Play developers can publish apps any time they want; at the very moment an update is ready, they can make it available for download with a simple click of the mouse. In this environment, updates are continuously published thus diluting the impact of past performance on the decision to update.

From columns (4) and (5) other, minor, observations follow:

1. in both platforms, apps that have been updated in the previous period are more likely to be updated today. This suggests a form of persistency in the decision to release new versions of the apps;

2. apps with in-apps purchases are more likely to be updated. This evidence is consistent with Conjecture 4 and occurs in both platforms. As discussed

[12]The dummy variable $free$ is omitted for Google Play, provided that nearly all apps are free of charge.

above, we interpret this result on the basis that developers of apps with in-app purchases have more to gain from stimulating buzz and improving visibility.

3. the number of other apps of the same developer listed in the top 1,000 during the same month and in the same country positively affects the likelihood to update apps in iTunes while it is not statistically significant in Google Play. This partially confirms Conjecture 2. As discussed above, estimates of the growth equation of system (5) suggest the presence of complementarities among apps, both in iTunes and Google Play. According to Conjecture 2 this implies that the probability of updating an app should increase with the number of other apps of the same developer.[13]

Concluding this section, it is important to discuss an implicit assumption we have made when computing the estimates of the second equation in (5). In testing Conjecture 1, we have considered the past performance experienced by the app in the five countries included in our sample (measured in terms of the growth rate during the previous months); hence, we have implicitly assumed that those five are the reference countries for the developers' decisions. However, it might be the case that developers base their strategies by looking at the performance in a wider set of countries or in a set of countries different from that composing our sample.[14] As a matter of fact, when published in the store an update becomes available in all countries worldwide; hence, the decision to release a new version of the software might be based on the world-wide performance of the app. Alternatively, a developer might decide to update the app on the basis of the growth rate experienced in a set of countries different from our ones (e.g. a US-based developer might take her decisions by looking at the performance in the USA or in North America). In both cases, by considering France, Germany, Italy, Spain and the UK we would not control for the right set of countries determining the decision about whether to make an update.

In order to tackle this issue, we employ a useful information provided in the Priori dataset, namely whether an app is local or not. An app is defined as local in a given country when at least 40% of its all time downloads occurs in that country. For local apps the performance in the five countries composing our sample is very likely to represent the basis developers use in order to take their decisions. Therefore, we re-estimated the second equation of the system (5) by restricting the sample only to

[13]The theoretical model only takes into account complementarity/subsitititutabilty looking at the demand side, but complementarity/substitutability can also emerge on the production side. For example, a developer may re-use the code written for another app (complementarity).

[14]By contrast, it is quite natural to estimate the growth equation using the total downloads, as the aim is to estimate their determinants.

local apps. The results of this estimation are given in Table 4 and largely confirm our previous findings.

7 Zooming on iTunes. Major *vs* minor updates

For iTunes apps we collected additional information about the type of updates by distinguishing major updates (i.e. significant changes in functionalities) from minor ones (i.e. bugs fixing and minor changes to the software code). This allows us to investigate more closely the versioning strategy followed by developers. We have re-estimated the two equations of the system (5) by considering *Update major* and *Update minor* separately.[15] The results are in columns (3), (6-7) of Table 3.

Estimates of the growth equation confirm the results obtained in the general estimation where we have not distinguished between major and minor updates (column (1) of Table 3). Interestingly, major updates have stronger impact on the growth rate of downloads than minor ones (the coefficient of *Update major* is nearly twice as large as the coefficient of *Update minor*). This evidence suggests that adding new and significant functionalities to an application is more effective in stimulating downloads than bug fixing and minor adjustments.

As far as the second equation of the system regarding updates is concerned, the results obtained in the general estimation are confirmed particularly with respect to minor updates. Although statistically significant, the coefficient of the lagged growth rate has an extremely small magnitude in the regression on the determinants of major updates (columns 6). On top of this, the coefficient of the variable *Number of apps by developer* is not significant anymore. These findings reveal that the strategic use of updates described by the theoretical model is particularly suited to minor updates. This result can be easily interpreted if one considers the different nature of major and minor updates; while minor updates can be developed with a certain ease, major ones require much more development effort and time. This implies that only minor updates can be used strategically in reaction to poor performances or to exploit cross-app effects. On the contrary, major updates are inherently less suitable for these purposes.

8 Conclusion

To be written.

[15]Notice that the second equation of the system must be estimated separately for minor and major updates.

References

Bresnahan, T., Davis, J., and Ying, P.-L. (2014). Economic value creation in mobile applications. Forthcoming in The Changing Frointier: Rethinking Science and Innovation policy, Jaffe A. and B. Jones eds, University of Chicago Press.

Carare, O. (2012). The impact of bestseller rank on demand: Evidence from the app market. *International Economic Review*, 53(3):717–742.

Datta, D. and Sangaralingam, K. (2013). Do App Launch Times Impact their Subsequent Commercial Success? 2013 International Conference on Cloud Computing and Big Data.

Garg, R. and Telang, R. (2014). Estimating App Demand From Publicly Available Data. School of Information Systems and Management, Heinz College Carnegie Mellon University.

Ghose, A. and Han, S. P. (2014). Estimanting demand for mobile applications in the new economy. Forthcoming in Management Science.

Greenbaum, J. (2005). This is just in: Consumers hate their software vendors. *Intelligent Enterprise, San Mateo*, 8(10) 14.

Ifrach, B. and Johari, R. (2014). The Impact of Visibility on Demand in the Market for Mobile Apps. Available at SSRN: http://ssrn.com/abstract=2444542.

Priori (2014). Global Inisghts Report - Android Platform & iOS Platform. February 2014.

Sankaranarayanan, R. (2007). Innovation and the durable goods monopolist: The optimality of frequent new-version releases. *Marketing Science*, 26(6):774–791.

Table 3: FD growth and update equations[†]

	Growth equation			Update equation			
	iTunes	GP	iTunes maj-min	iTunes	GP	iTunes major	iTunes minor
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Lag growth (gdl_{t-1})	-0.026^a	-0.009^a	-0.028^a	-0.042^a	0.0004	-0.007^c	-0.032^b
	(0.008)	(0.001)	(0.008)	(0.015)	0.0003	(0.004)	(0.015)
Update (u_t)[††]	0.362^a	0.281		0.056^b	0.219^a		
	(0.059)	(0.186)		(0.027)	(0.037)		
Major update (u_{1t})[††]			0.621^b			0.048^b	-0.347^a
			(0.266)			(0.023)	(0.070)
Minor update (u_{2t})[††]			0.316^a			-0.0002	0.110^a
			(0.065)			(0.0099)	(0.030)
In-app	-0.462	-0.657^a	-0.455	0.698^a	0.317^a	0.234^b	0.525^a
	(0.287)	(0.286)	(0.285)	(0.089)	(0.125)	(0.110)	(0.163)
Free	1.687^a		1.681^a	-0.034		-0.008	-0.0002
	(0.497)		(0.504)	(0.124)		(0.023)	(0.113)
Number of apps by developer	0.078^a	0.031^b	0.083^a	0.011^a	-0.002	-0.001	0.012^a
	(0.014)	(0.014)	(0.014)	(0.003)	(0.002)	(0.001)	(0.003)
Lag age (minor) version				0.134^a	0.236^a	-0.006^b	0.166^a
				(0.012)	(0.017)	(0.003)	(0.015)
Lag age major version						0.046^a	-0.028^a
						(0.004)	(0.004)
Lag average age (minor) version				-1.282^a	-0.008	-0.105^b	-1.224^a
				(0.155)	(0.007)	(0.046)	(0.161)
Lag average age major version						-0.248^a	-0.682^a
						(0.070)	(0.176)
Tests of hypothesis							
Strength IV:	F(7,1483)	F(7,1372)	F(10,1432)	F(4,1483)	F(5,1372)	F(6,1423)	F(6,1423)
Lagged growth (gdl_{t-1})	2075.31	1.4e+05	1686.14	30.99	1.8e+05	22.73	22.73
p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Update (u_t)[††]	177.45	48.57		598.09	377.85		
p-value	0.000	0.000		0.000	0.000		
Major update (u_{1t})[††]			6.45			483.50	483.50
p-value			0.000			0.000	0.000
Minor update (u_{2t})[††]			114.64			426.74	426.74
p-value			0.000			0.000	0.000
Under identif. p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Weak identif. p-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Over identif. p-value	0.579	0.245	0.199	0.444	0.177	0.728	0.295
Observations	3,660	2,956	3,556	3,660	2,956	3,530	3,530
Uncentered R-squared	0.058	0.019	0.064	0.139	0.066	0.266	0.140

[†]All regressions are in first difference and include period dummies. Standard errors are in parenthesis and clustered by app. Superscripts a, b, c indicate significance at 1%, 5% and 10% respectively. [††]The variable in the update equation is lagged.

Table 4: Update equation for local apps[†]

	Update equation			
	iTunes	GP	iTunes major	iTunes minor
	(1)	(2)	(3)	(4)
Lag growth (gdl_{t-1})	-0.035^b	-0.009	0.004	-0.037^b
	(0.015)	0.017	(0.005)	(0.015)
Update (u_{t-1})	0.107^b	0.224^a		
	(0.044)	(0.056)		
Major update ($u_{1,t-1}$)			0.043	-0.519^a
			(0.045)	(0.123)
Minor update ($u_{2,t-1}$)			0.003	0.200^a
			(0.014)	(0.047)
In-app	0.785^a	0.445	0.122	0.570^b
	(0.109)	(0.275)	(0.139)	(0.270)
Number of apps by developer	0.021	-0.008	0.004	0.025^c
	(0.014)	(0.011)	(0.005)	(0.015)
Lag age major version			0.037^a	-0.031^a
			(0.004)	(0.006)
Lag age (minor) version	0.108^a	0.208^a	-0.005	0.158^a
	(0.015)	(0.026)	(0.005)	(0.021)
Lag average age major version			-0.145	-1.261^a
			(0.128)	(0.330)
Lag average age (minor) version	-1.781^a	-0.026	-0.005	-1.631^a
	(0.302)	(0.026)	(0.005)	(0.295)
Tests of hypothesis				
Strength IV:	F(4,497)	F(4,512)	F(6,480)	F(6,480)
Lagged growth (gdl_{t-1})	907.88	183.03	607.68	607.68
p-value	0.000	0.000	0.000	0.000
Update (u_{t-1})	274.13	187.89		
p-value	0.000	0.000		
Major update ($u_{1,t-1}$)			228.48	228.48
p-value			0.000	0.000
Minor update ($u_{2,t-1}$)			177.11	177.11
p-value			0.000	0.000
Under identif. p-value	0.000	0.000	0.000	0.000
Weak identif. p-value	0.000	0.000	0.000	0.000
Over identif. p-value	0.089	0.478	0.189	0.789
Observations	1,233	1,068	1,194	1,194
Uncentered R-squared	0.139	0.045	0.255	0.106

[†] All regressions are in first difference and include time period dummies. Standard errors are in parenthesis and clustered by app. Superscripts a, b, c indicate significance at 1%, 5% and 10% level, respectively. Instruments: 1) Multi-app developers instruments (first difference of the lagged variables: gdl (columns 1-4), Dver (columns 1 and 2), and Dminver and Dmajver (columns 3 and 4)); 2) Two period lags for gdl (columns 1-4), Dver (columns 1 and 2), and Dminver and Dmajver (columns 3 and 4).