Updates management in mobile applications: iTunes versus Google Play

Updates Management in Mobile Applications: iTunes vs Google Play.*

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

May 8, 2018

Abstract

This paper focuses on a specific strategy that developers of mobile applications may use to stimulate demand: the release of updates. We develop a theoretical analysis that shows that developers have incentives to release updates when experiencing a drop in performance. The predictions of the model are then tested using an unbalanced panel of top 1,000 apps in iTunes and Google Play for five European countries. We estimate that while in iTunes the release of an update stimulates a 26% increase in download growth, in Google Play updates play a less significant role. This difference is partly due to systematic differences in apps and in developers operating in the two stores ("selection effect"), and partly to a lack of quality control on apps and updates in Google Play ("quality check effect"). These findings highlight the crucial importance of an appropriate management of updates as well as the relevance of institutional characteristics of the app stores.

Keywords: mobile applications, updates, downloads, iTunes, Google Play, quality check, buzz, multihoming.

JEL Codes: L10, L63, M31.

*Paper presented at the 2015 annual conferences: Economics of ICTs (Telecom ParisTech), French Economic Association, Centre for Competition Policy, and at the 2016 annual conferences: Florence School of Regulation, International Industrial Organization Society, Royal Economic Society, Association of Southern European Economic Theorists, and Jornadas de Economia Industrial. The authors wish to acknowledge the financial support from "Progetto di Ateneo" - Università di Padova, 2012-14 and from the ESRC Centre for Competition Policy (funding ref: RES-578-28-0002). Authors also thank Toker Doganoglu, Joeffrey Drouard, Lisa George, Thierry Penard, and Christian Peukert for useful comments on previous versions of the paper.

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

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

[§]School of Economics & Centre for Competition Policy, University of East Anglia, Norwich (UK).

> A lot of app developers will see a large spike in downloads right at launch, and shortly after see these numbers slowly dwindle. The question I get asked in this situation is, "How do I continue growth?" The answer is [...] in order to continue growth you need to provide constant value, which [...] also means updating your app! That said, I always encourage my students to update their apps and keep iterating to get feedback, which helps boost downloads. Letting your apps collect dust is the same as letting them fail [...]
>
> Chad Mureta

1 Introduction

On November 25^{th} 2015, Wooga released version 6.3.0 of its well-known arcade game Diamond Dash. This version of the software followed the 6.2.0 update by one month and a half which, itself, was released only two weeks after version 6.1.0. From November 2011 − the time the app was launched in iTunes − to November 2015, Diamond Dash was updated nearly 60 times, on average more than once per month. This is not an exception. The frequent release of updates is a common feature among apps: in our sample, apps are updated on average every 13 days in Google Play and every 58 days in iTunes.

With millions of apps available in various stores, developers face a tremendous challenge. Not only do they struggle to catch the attention of prospective users (Bresnahan et al., 2014), but they also fiercely compete for usage. Competition among developers is so harsh that app markets are labelled as "hyper-competitive" (see, Datta and Sangaralingam, 2013). As suggested by the quote from the distinguished entrepreneur and blogger Chad Mureta, the frequent release of updates represents a natural strategy to maintain performance in such competitive environment.

Developers update their apps not only to introduce new features or functionalities, thus (potentially) increasing the quality of the software, but also to stimulate what is known as "buzz" around the app. Typically, when an app is upgraded, its new features are likely to be presented and discussed in dedicated blogs, in on-line magazines or among users of social networks. Also, developers usually promote the new versions of their apps through several channels or directly in the "What's New" section of the app stores. Generating buzz is essential to stay on top of the hyper-competitive app markets. Only those apps that are able to get noticed and to attract users' attention can survive and possibly thrive.

Besides the high degree of competition, another relevant feature characterising the app market is the skewness of the distribution of downloads and usage. According to Appbrain.com, in Google Play more than 2 million apps out of 2.8 million have less than one thousand downloads each while, by comparison, just fifty thousand apps have more than one million downloads. Similarly, for app usage Google (2015) shows that, on average, only one-out-of-four installed apps is actually used daily; a similar fraction of apps is installed but never used. In a competitive environment where either you win big or you drastically fall

down in ranking, it is even more compelling to adopt an effective management of updates to attract customer attention.

In this article we study the effectiveness of releasing updates to improve app performance, and for this purpose we employ data on apps distributed in the two most popular app stores: iTunes and Google Play. Interestingly, the two stores follow different policies regarding the release of apps and updates. The functionality section of the iTunes "App store review guidelines" explicitly sets a strict screening of app 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 that include undocumented or hidden features inconsistent with the description of the app are rejected. By contrast, publication in Google Play does not go through a similar quality check as developers with a "simple click of the mouse" can publish apps and updates instantaneously.[1] In this paper, we argue that these institutional differences have important consequences on the characteristics of updates and on how they affect the performance of apps.

We start our analysis by presenting a stylised theoretical framework that focuses on the developers' decisions about whether to update their mobile applications. We show that, in an attempt to "revive" their app, developers are more likely to release an update when they observe a worsening of the performance. Interestingly, our analysis highlights that the incentives to update might be so strong that a developer could decide to also release an update of low quality to counter a drop in performance; by stimulating the buzz surrounding their app, developers might succeed in attracting users' attention even with an update that makes little improvement to the software code. This result suggests an interesting prediction based on the institutional differences characterising the two stores. The strong incentives to release new versions of the software induce developers in Google Play to release both high and low quality updates, thus diluting the effects on downloads. This strategy cannot be employed in iTunes provided that the strict control implemented by Apple limits the ability of developers to release low quality updates. Hence, we expect that updates have a stronger impact on downloads in iTunes than in Google Play.

Figure 1 provides preliminary support to our argument. The figure shows the kernel density of the growth in downloads of the apps in our sample, distinguishing between apps that have been updated during the period of observation and apps that have not been updated. Interestingly, in Google Play the two density functions nearly perfectly overlap; this suggests that updated and non-updated apps perform very similarly in terms of downloads. On the contrary, in iTunes the density function of non-updated apps is more asymmetric and concentrated on the negative values of download growth rate. According to this figure, updates seem to have a different impact on app performance in the two stores.

[1]The absence of formal screening has led several commentators to criticise Google Play for the poor quality of the apps available in the store. This issue is so critical that Google has stepped up its efforts to improve app quality. For instance, in February 2013, it removed from its store 60,000 spammy and low quality apps at once. Similarly, in October 2014 Google launched a new feature that allows users to filter out all apps that are not rated at least 4 stars. The website Appbrain.com estimates that nearly 13% of Google Play apps are of low quality.

[Figure 1 about here.]

In order to investigate the impact of updates, we use an unbalanced panel of the top 1,000 apps distributed in iTunes and in Google Play, in five European countries. Our estimates confirm that updates have a stronger impact on downloads in iTunes than in Google Play. The analysis suggests that this result is partly related to a "selection effect" (systematic differences in the apps and developers operating in the two stores) and partly to the "quality check effect". In order to isolate the quality check effect, we also conduct a series of estimations on the sample of multihomed apps − apps that developers distribute in both stores. These estimations confirm that the quality check effect plays a key role, even though differences across stores are reduced. We obtain these findings by means of classical regression analyses, as well as through treatment effect estimations, where the treatment is the release of an update.

Our findings indicate that an appropriate management of updates is of crucial importance for developers committed to improving the performance of their apps. They also highlight that the institutional characteristics of the stores play an important role and interact with the strategies of developers in determining app success.

Consistent with the theoretical model, our estimates for iTunes apps also reveal that an update is more likely to be released after a drop in downloads. The same does not occur in Google Play where, instead, app past performance does not affect the decision to update. We interpret this difference, again, on the basis of the policies governing the publication of apps and updates for the two stores. The strict quality check implemented by Apple induces developers to release updates of high quality and only when necessary, i.e. in order to counter a decline in downloads. By contrast, in Google Play developers continuously update their apps, thus minimising the role of past performance on the decision to release a new version of the app.

The rest of the paper is organised as follows. In Section 2 we discuss the relevant literature. In Section 3, we present a stylised theoretical framework that we use to derive the testable predictions. Our dataset is presented in Section 4 while the empirical analysis is given in Section 5. More specifically, in Subsection 5.1 we analyse the impact of updates on performance while in Subsection 5.2 we focus on the determinants of developers' decision to update. In Section 6 we provide two further extensions of our analysis and, finally, Section 7 concludes.

2 Literature review

The app market is certainly one of the most dynamic segments of the software industry. Hundreds of thousands of registered developers regularly publish new apps and make them available for download in different stores. According to the most recent estimates provided by to Statista.com, the number of mobile applications published in Google Play has reached 3.5 million (December 2017), and it is more than 2.2 million in iTunes (April 2017). A recent study published by the Progressive Policy Institute using information on online job

posting estimated that, as of January 2016, the number of jobs related to the app market in the European Union plus Switzerland and Norway was about 1.64 million. The figure estimated for the United States was 1.66 million. Looking at the overall value generated by this emerging market, AppAnnie.com estimates that in 2016 consumer spending on mobile apps was valued at more than 50 billion while in-app advertising was nearly 80 billion.

In this paper, we investigate a specific aspect of the app market. We focus on the role of asymmetric information in the relationship between software publishers and users. We contribute to the literature on the effect of search costs in market transactions, which is a topic that has attracted the attention of many researchers since the seminal article by Diamond (1971). With the advent of digital marketplaces, several recent contributions have focussed on eWOM (electronic-Word-Of-Mouth), such as customer reviews or users' ratings, both seen as important sources of information for consumers. Chevalier and Mayzlin (2006) and Li and Hitt (2008) are among the first empirical papers in this field. The authors look at the effects of consumer reviews on sales at Amazon.com and Barnesandnoble.com, and find that positive reviews tend to stimulate subsequent sales. Liu (2006), Duan et al. (2008) and Duan et al. (2009) reach different conclusions on the effectiveness of reviews and ratings. For instance, Duan et al. (2009) do not find a statistically significant impact of user ratings on the adoption rate of popular software. The role of ratings as an indicator of product quality is also questioned by Lee et al. (2015), with their alternative explanations that prior ratings affect subsequent user ratings through "herding" (users tend to follow previous ratings) or "differentiation" (users tend to differentiate from previous ratings) behavior. The presence of systematic trends in user ratings raises concerns about their ability to mitigate the asymmetry of information between consumers and sellers.

Several recent articles study the role of bestseller charts in stimulating demand for mobile applications (Carare, 2012; Ghose and Han, 2014; Ifrach and Johari, 2014; Garg and Telang, 2013). For example, Carare (2012) investigates the role of ranking charts in favouring the matching between users and developers. The work is based on the top-100 paid apps available in the US iTunes store. The author finds 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 in positions higher than 50. Along similar lines, Ghose and Han (2014) find that cross-charting (namely, an app appearing both in the top-free and top-paid charts) has a positive impact on app demand. This suggests that being in the bestseller list may have a valuable effect in terms of reducing the asymmetry of information thus stimulating downloads.

Our paper contributes to this literature by looking at a specific strategy that developers can use to reduce the asymmetry of information affecting their customers. As mentioned above, in order to survive in the hypercompetitive app market, developers need to increase the visibility of their products. The frequent release of updates is a strategy developers can use in order to attract the attention of users, keeping them engaged, thus stimulating app downloads and usage. The underlying idea behind our analysis is that the release of an update makes the app more visible thanks to the eWOM it generates and, via this channel, it might increase demand.

The paper that is closest to ours is Leyden (2018). The author estimates a discrete choice logit model of demand for apps in iTunes. The model incorporates updates into consumers' utility. Combining changes in version number and a textual analysis of updates descriptions, the author classifies updates as either feature-adding or bug-fixing. Interestingly, updates are found to spur demand with approximately the same effect for both types of update. In our paper we extend this analysis in two directions: we evaluate the impact of updates also in Google Play and we discuss the determinants of the developers' decisions to update.

Two recent papers have tried to empirically estimate the main determinants of mobile app demand and success. Ghose and Han (2014) use daily information on the top-400 free apps and the top-400 paid apps in iTunes and Google Play to estimate the demand for mobile applications. The authors find that demand is larger when the app has the in-app purchase option, and lower when it has the in-app advertising option. In their analysis Ghose and Han also control for user reviews and find that higher ratings stimulate downloads. Interestingly, Lee and Raghu (2014) find that in both platforms demand is boosted by the number of previous versions of the same app. The role of updates is also investigated in Yin et al. (2014). Using data on iTunes apps during the period Sept 2010 - Aug 2011, the authors study the determinants of the probability of being in the top-300 apps ranked by gross sales. Differently from other findings in the literature, they show that the number of updates increases the likelihood of entering the top-300 chart but only for non-game apps. By contrast, they find that game apps are more likely to succeed when the developer does not release updates.

The continuous release of updates is not specific to mobile applications but, with lower frequency, is also commonly observed in "traditional" desktop computer software (see Greenbaum, 2005). Following Sankaranarayanan (2007), the release of a new version of software occurs especially when the package has reached a high level of penetration so that little revenue can be collected from new customers. Software firms are therefore induced to upgrade their packages in the attempt to "re-sell" the software to their installed base of users. This explanation for the release of frequent updates is unlikely to fit the case of mobile applications. Most of the apps are available for free; on top of this, also for paid applications, a common rule in app stores is that updates must be made available free of charge to anyone who has previously downloaded the app. As a consequence, developers cannot exploit their installed base of users by trying to re-sell upgrades of their software.[2]

3 The decision to update. A stylised dynamic model

In the hyper-competitive app environment performance is highly skewed, with a handful of mobile applications obtaining the lion's share of the market. Producing high quality software is not enough to guarantee downloads and usage. Success depends heavily on the ability of developers to stimulate consumer attention, and the release of new versions is a strategy

[2]The incentives to release an update to profit from the installed base of users can be partially restored when the app comes with the in-app purchase option.

they can use for this purpose.

Updates are likely to attract the interest of bloggers, members of social networks and journalists of specialised magazines, stimulating what we call the "buzz" surrounding an app. In this way developers can spur downloads. Similarly, updates may be useful for reviving the interest in "dormant" apps, i.e. apps that consumers have already installed but that they use very little. The availability of a new version and the buzz it generates might stir up users' curiosity and, in this way, induce them to start using the app.

In this section we present a stylised dynamic model aimed at describing the decision of a developer to update an app. This model will drive our empirical analysis. The key features of the model are: *i*) the performance of an app (either in terms of downloads or usage) depends on its quality, its characteristics, the ranking it has reached in the store, as well as the reputation of the developer, but also the buzz surrounding the app. We refer to the combination of all these aspects as the "perceived quality" of the app; *ii*) the release of an update affects app perceived quality and, in particular, it stimulates the buzz. The effect of the increased buzz on perceived quality is ex-ante uncertain, as bloggers, journalists and regular users might positively or negatively welcome the new version of the software. The implication is that the augmented buzz may improve or worsen the app perceived quality, making it more uncertain. Finally, *iii*) app performance is skewed and it can be either high or low; formally we assume that if the perceived quality is above a certain threshold, returns are high, while they are low in the opposite case.

3.1 The model

We model the developer's decision to update the app in an infinite horizon setting. At each period t, with $t = 1, 2, ..., \infty$, the developer chooses whether to update the app at a cost $\phi > 0$. This decision is taken by looking at the perceived quality of the app during the previous period, $t-1$. As we detail below, we assume that at each t the developer can be of two different types: type L if at $t-1$ the perceived quality was lower than a threshold τ, and type H if, instead, the perceived quality was higher than τ.[3] The release of an update is assumed to impact app perceived quality in two ways: it increases its expected value (e.g. when it introduces new functionalities or fixes several bugs) and it generates buzz surrounding the app; this latter effect makes the realisation of the perceived quality more uncertain.

More specifically, if at t the developer of type i =L,H does not release an update, the perceived quality of the app is the realisation of a random variable $Q_i \sim U(q_i - \eta, q_i + \eta)$, with $q_L < q_H$. If, instead, the developer releases an update, both the expected value of the perceived quality as well as its variance increase; formally, we assume that following an update the perceived quality of the app is the realisation of a random variable $\widetilde{Q}_i \sim U(q_i + \Delta - \gamma\eta, q_i + \Delta + \gamma\eta)$, where $\Delta \geq 0$ is the increase in the app expected perceived

[3]It is worth noticing that, in our framework, perceived quality is a latent variable whose realisations do not have any specific meaning and that can even take negative values. The only thing that matters is whether the perceived quality is above or below the threshold τ.

quality associated with an update and $\gamma > 1$ represents the increase in the variance due to the larger buzz. App performance is highly skewed; hence, we assume that returns at time t are $\overline{R}$ if the realisation of the perceived quality is larger than the threshold τ, while they are $\underline{R}$ otherwise, with $\overline{R} > \underline{R} \geq 0$.

Two observations are noteworthy. First of all, our set-up implies that the realisation of the perceived quality at a given period − in particular whether it is above or below the threshold τ − determines not only the current returns enjoyed by the developer, but also the probability of obtaining $\overline{R}$ or $\underline{R}$ the following period; in our jargon, it determines the type of the developer. Secondly, we assume that the effect of an update lasts for just one period; this is equivalent to say that we are modeling a stationary environment, whereby the perceived quality at any period is Q_i if the developer does not update and $\widetilde{Q}_i$ if an update is released. Importantly, the stationarity of the environment we are working with implies that a decision that is optimal at t for the developer of type i is optimal in any other period for the same type of developer.

For simplicity, in the remainder of the section, we normalise q_L and $\underline{R}$ to zero; hence q_H and $\overline{R}$ can be interpreted as the differentials in perceived quality and per-period returns.

In what follows, we determine under what conditions the developer chooses to release an update with (exogenous) characteristics $\Delta \geq 0$ and $\gamma > 1$ bearing a (exogenous) cost $\phi > 0$. As we will show, the developer has more incentives to release an update when, in the previous period, the app perceived quality was below τ, i.e. when the developer is of type L.

3.1.1 The equilibrium

For the sake of brevity, in the discussion below we focus only on the case where the developer of type L updates, and that of type H does not. The other relevant cases go along similar lines and are discussed in the Appendix. Let us indicate with L the expected discounted value of returns for the developer of type L when an update is released. Formally:

$$L = \frac{\gamma\,\eta - \Delta + \tau}{2\gamma\,\eta}\,(\delta\,L) + \left(1 - \frac{\gamma\,\eta - \Delta + \tau}{2\gamma\,\eta}\right)\left(\overline{R} + \delta\,H\right) - \phi. \tag{1}$$

With probability $(\gamma\,\eta - \Delta + \tau)/2\gamma\eta$ perceived quality falls below the threshold τ.[4] In this case, returns at time t are zero, and at $t+1$, the developer is still of type L earning δL, where $\delta \in (0,1)$ is the discount factor. With complementary probability the perceived quality is above the threshold: returns at time t are $\overline{R}$ and, at $t+1$, the developer is of type H, obtaining δH.

The expression of the expected discounted value of returns for the developer of type H who does not update is similar to (1) and it is given by:

$$H = \frac{\eta - q_H + \tau}{2\,\eta}\,(\delta\,L) + \left(1 - \frac{\eta - q_H + \tau}{2\,\eta}\right)\left(\overline{R} + \delta\,H\right), \tag{2}$$

[4] All throughout the paper we assume that the probabilities take values in the $(0,1)$ interval. This assumption imposes conditions on the parameters, as we discuss in the Appendix.

where $(\eta - q_H + \tau)/2\eta$ is the probability that the perceived quality of the app falls below τ conditional on type H not releasing an update.

The solution of the system of expressions (1) and (2) with respect to L and H determines the expected pay-offs of the low and high type, $L_{U,N}$ and $H_{U,N}$ respectively. Subscripts U, N remind us that pay-offs are conditional on the L-type always updating and the H-type never doing so:

$$L_{U,N} = \frac{(\gamma\,\eta + \Delta - \tau)\,\overline{R} + (\delta\,\eta\,\gamma + \delta\,\gamma\,q_H - \delta\,\gamma\,\tau - 2\,\gamma\,\eta)\,\phi}{(1-\delta)\,(2\,\gamma\,\eta - \delta\,\gamma\,q_H + \delta\,\gamma\,\tau + \Delta\,\delta - \delta\,\tau)},$$

$$H_{U,N} = \frac{(\gamma\,\eta - \delta\,\gamma\,q_H + \delta\,\gamma\,\tau + \Delta\,\delta - \delta\,\tau + \gamma\,qH - \gamma\,\tau)\,\overline{R} - (\delta\,\eta\,\gamma - \delta\,\gamma\,q_H + \delta\,\gamma\,\tau)\,\phi}{(1-\delta)\,(2\,\gamma\,\eta - \delta\,\gamma\,q_H + \delta\,\gamma\,\tau + \Delta\,\delta - \delta\,\tau)}.$$

The scenario (U, N) is an equilibrium provided that no type of developer has incentives to unilaterally deviate. Formally, $H_{U,N}$ has to be larger than what type H would obtain by choosing to update the app, conditional on type L updating too; $L_{U,N}$ must be greater than what the developer of type L would get by not updating the app, given that type H does not update. In the Appendix we calculate the pay-offs from unilateral deviations, as well as the pay-offs in all other relevant cases. We can therefore prove the following:

Proposition 1. *The developer's optimal behavior depends on the cost of the update. When $\phi \leq \underline{\phi}$, both types of developer choose to update; when $\phi \in \left(\underline{\phi}, \overline{\phi}\right]$, only type L updates, while neither type L nor type H update for $\phi > \overline{\phi}$.*

Proposition 1 provides a very intuitive result: the lower the development cost the higher the incentives to release updates. Interestingly for our scope, the developer of type L, that is the one whose app perceived quality was low in the previous period, has greater incentive to update. The developer of type H releases the update only when $\phi \leq \underline{\phi}$, while that of type L updates for $\phi \leq \overline{\phi}$, with $\overline{\phi} > \underline{\phi}$. Therefore, according to this proposition, a developer is more likely to release an update after experiencing a worsening in app perceived quality.

From Proposition 1 it also follows that:

Corollary 1. *The developer: i) is more likely to release an update the larger the increase in expected perceived quality, Δ. However, ii) updating the app can be profitable also when it does not increase the expected perceived quality ($\Delta = 0$).*

The intuition for part *i*) of the corollary is rather obvious: the larger the increase in the expected perceived quality (e.g. an update adding several new functionalities or fixing several bugs), the stronger the incentives to update. This is true both for the developer of type L and of type H, as they both benefit from an update with a larger Δ. A less intuitive result is shown in part *ii*) of the corollary: the developer may find it optimal to spend ϕ and release an update also in the case where it only increases the variance of the perceived quality without augmenting its expected value ($\Delta = 0$). The reason why this latter result is true is the following. When q_i, $i = H, L$, is low compared to the threshold τ then, absent any update, it is very likely that the app will generate zero returns. In this case, the developer has everything to gain from releasing an update that stimulates the buzz

and increases uncertainty about the app perceived quality. The update may turn out to generate either a very low or a very high perceived quality, depending on the realisation of $\widetilde{Q}_i$. However, only high realisations matter provided that, without the update, app perceived quality would have been, in any case, below τ with a large probability. As a consequence, when q_i is low compared to the threshold τ, even an update characterised by $\Delta = 0$ can be profitable as it increases the probability of high returns. We read the decision to release a low quality update as a sort of "bet for resurrection" strategy that the developer might find profitable to pursue.[5] Needless to say, it is the developer of type L who has stronger incentives in betting for resurrection.

3.2 Empirical implications

The theoretical model suggests some interesting implications both for the conditions under which a developer releases an update and for the likely impact of updates on app performance. We start from the discussion of this latter implication.

In part *ii*) of Corollary 1, we have shown that developers may be tempted to also release updates of low quality (formally, updates with Δ equal or close to zero). Nonetheless, low quality updates are more likely to be published in Google Play than in iTunes. This is because, as observed in the introduction, in Google Play developers are free to publish apps and updates at their will whereas in iTunes the quality check implemented by the store prevents developers from publishing updates of very low quality − that is updates that do not augment app functionalities significantly and/or come with an inconsistent description. As a consequence, we expect the average quality of iTunes updates to be larger and, more relevantly, generate a greater impact on app performance than in Google Play.

The second interesting implication derives directly from Proposition 1. All else equal, the developer is more willing to release an update when experiencing a worsening in perceived quality of the app (due to, for example, a drop in the ranking, in the number of downloads, in the developer's reputation, etc.). Therefore, our model suggests app perceived quality and the decision to update to be negatively correlated. Accordingly, the update is used strategically to revive interest in the app.

These implications will drive the empirical analysis that follows.

4 The data

We test our empirical implications using monthly data on the top 1,000 most downloaded apps in iTunes and Google Play. Data are provided by the consulting analytics Priori and refer to five European countries (Germany, France, Italy, Spain, and the UK); they cover the period September 2013-February 2014. According to Priori (2014), the top 1,000 apps cover

[5]This strategy is reminiscent of the effect of limited liability in firms' investment decisions. As shown in Gollier et al. (1997), limited liability induces firms to invest in risky projects. The authors show the existence of a lower bound on the value of the firm below which managers will "bet for resurrection", investing in risky projects.

about 60% of the market in each country (e.g. in October 2013 our dataset covers 55.3% of total downloads in iTunes in the UK and 62.09% in Italy).

For each app, Priori dataset provides the following information: name of the app, name of the publisher, the category to which the app belongs (e.g. games, utilities, ...), monthly and overall number of country downloads of the app, the worldwide average customer rating of the app (on a scale from 1 to 5), the number of user ratings, the date when the app was published in the store, the overall number of updates released, the day when the last update was published, the price of the app − when suitable −, and whether the app has the in-app purchase option. Finally, Priori also provides information about whether an app is "local" in a given country, i.e. whether at least 40% of its all-time downloads occur in that country. It is worth pointing out that except for downloads, no information is country-specific. In particular, relevant for our scope, it is important to stress that when a developer updates an app the new version becomes available worldwide.

With the exception of number of downloads, all the information gathered by Priori is taken directly from the app stores. Downloads, instead, are computed 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. Priori cross-checked these estimates for a sample of apps, by using real download data provided by partner developers.[6] Only first-time installations are counted as downloads in our data, that is users' upgrades of already installed applications are not counted as downloads.

4.1 Descriptive statistics

One of the purposes of our research is to estimate the determinants of developers' decisions to update an app. In particular, we look at how past performance affects the decision to release a new version of the software. Given that updates apply to all countries worldwide, it seems sensible to assume that this decision is based on the overall performance of the app. For this reason, we aggregate the data from the five European countries. Following this aggregation, monthly downloads are the sum of downloads in the five countries in a given month.[7] Data aggregation together with the non-balanced nature of the data, imply that from the original 30,000 observations per store (1,000 apps, during 6 months, in 5 countries) our sample drops to 15,981 observations for Google Play, and 14,759 for iTunes. This reduction in the number of observations is due to the fact that, in several cases, the same app appears in the top 1,000 ranking in more than one country in a given month.

[Table 1 about here.]

[6] According to Priori's statement, this internal validation study, based on 2,000 Android apps, returned a mean absolute error of +/- 24.6 per cent in the level of downloads. Notice that, in our regression analysis, we employ the growth rate of downloads rather than their absolute levels; in this, way we smooth out the underlying issue of nonrandom measurement error.

[7] All the other variables are not country-specific and are not affected by aggregation.

The top part of Table 1 reports the summary statistics for this sample of apps. The table also shows the same statistics for the two sub-samples that we employ in our estimations. Subsample (A) includes both multihomed and non-multihomed apps, while subsample (B) is restricted to multihomed apps only.

The main characteristics of our original sample are:

- In Google Play nearly all the apps are free. Only 17 observations represent paid apps. Free apps are also prevalent in iTunes, although paid apps are more frequent than in Google Play (8.3% of the observations have a positive price).

- There is a significant difference between the two stores in terms of the number of apps with in-app purchases: 29.7% of the Google Play sample have in-app purchases, while in iTunes this occurs in 56.1% of apps.

- 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. In iTunes the average app age is about 19 months while in Google Play it is about 15 months.

- In both stores, apps are updated frequently. On average in Google Play apps are updated about 33 times since their original publication, while this figure reduces to about 10 in iTunes (see variable *Number of updates (all time)*). This difference may be due to the aforementioned divergent regulations on the publication of apps and updates implemented by the two stores. In Google Play the absence of a strict quality check may induce developers to update their apps more frequently than in iTunes. A possible additional explanation of 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, and that may require a closer management of updates by developers.

- Downloads are much higher in Google Play than in iTunes, with Google Play apps on average being downloaded nearly five times more than iTunes apps (249,803 compared with 58,156).

- In both stores, roughly 35% of apps are local. As explained, an app is named as local in a given country when at least 40% of its all time downloads occur in that country.

- On average, in Google Play, a given developer distributes about 3 top-ranked apps (see variable *# apps same developer*). The figure for iTunes is about 3.5.

- On average, an app enters the top 1,000 ranking in about 2 out of 5 countries in both stores, 1,876 in Google Play and 2,030 in iTunes (see variable *Number of countries*).

The number of observations in subsample (A) drops significantly compared to the original sample; this is due to the fact that the instrumental variables that we employ require apps to be observed in at least four consecutive periods. As a significant number of apps enters the top 1,000 ranking for only a few months, subsample (A) is made of a fraction of the original

sample (2,956 observations for Google Play and 3,700 for iTunes). Compared with those in the original sample, the apps in subsample (A) are: older, more downloaded, distributed in a larger number of countries, and their developers distribute a larger number of applications. These apps also register a larger number of updates, which could be due to their older age. Overall, this sample is made of apps that are generally better performing than those in the original sample and that are owned by the most successful developers.

In the bottom part of the table we focus on multihomed apps (subsample (B)). We define an app to be multihomed if it appears in both stores in at least one period.[8] Subsample (B) is made of all the multihomed apps included in our instrumental variables estimations, that is the apps that we observe for at least four consecutive periods. The differences in descriptive statistics with the original sample go along similar lines of those between the original sample and subsample (A), although with a slightly larger magnitude.

On top of the evidence provided in Table 1, our data confirms a couple of features that have already been found in the literature (see, among others, Bresnahan et al., 2014). Downloads exhibit a very skewed distribution, with top apps accounting for a large fraction of total downloads; for instance, in our sample, the average number of monthly downloads for the top 10 ranked apps in Google Play in Germany is eight times larger than the average number of monthly downloads for the apps ranked between the 90^{th} and the 100^{th} position (436,674 vs 58,690).

The second feature commonly found in the literature is the large turnover/churn, with only few applications succeeding in staying persistently in the top charts. Our data confirm this feature; in iTunes, 44% of the apps enter the top 1,000 ranking for one month only, while the share of apps that we observe all through the six-month period is 18.49%. Similarly, for the Android store, the share of apps that appears only in one month in the top 1,000 list is about 50%.

5 The empirical analysis

Our empirical analysis has two main objectives. The first one is to estimate the impact of the release of an update on the rate of growth of downloads, a measure of app performance. The second one is to study the determinants of the decision to release an update. Precisely, following the conjecture of our theoretical model, we test whether or not developers update their apps in response to downturns in performance.

We run our estimations on the dataset that we obtained by aggregating information over the five European countries for which we have data on. The reason for this aggregation is that, as mentioned earlier, when a new version is released it becomes available worldwide. Hence, it is natural to represent the decision to update as being influenced by the aggregate performance of the app.

We start the empirical analysis with the evaluation of the impact of updates on the

[8] We identified multihomed apps by looking at their names in the two stores. In case of similar but not perfectly coincident names, further checks were based on the name of the developers.

rate of growth of downloads. The determinants of the decision to update are discussed in Subsection 5.2.

5.1 The impact of updates on download growth

We estimate the impact of updates on download growth by employing two alternative techniques: a standard linear panel data regression and an average treatment effect estimation.

5.1.1 The growth equation

We devote this section to estimate what we refer to as the "growth equation" by means of a classic regression analysis. In the econometric equation presented below we use subscript j to indicate the app, and subscript t to label the period. We employ an unbalanced longitudinal dataset that has a large cross-section of apps, and a limited number of periods. Asymptotics rely on the largest dimension, which is the app dimension.

We model the growth equation as a linear autoregressive distributed lag model of order 1, made of observable and unobservable app characteristics. Formally, for each store, we estimate the following equation:

$$g_{jt} = \phi_1 g_{jt-1} + \phi_2 u_{jt} + h_t + \mathbf{x}_{jt}\boldsymbol{\beta} + \varepsilon_{jt},$$

where g_{jt} is the aggregate rate of growth of downloads of app j in the five countries between time $t-1$ and time t,[9] and u_{jt} is a binary variable which takes value 1 in the case where the publisher releases an update of app j in period t, and value zero otherwise. The row vector $\mathbf{x}_{jt}$ includes a set of controls: a constant, a dummy for the category the app belongs to, one-period lags of: in-app purchase option, the number of apps distributed by the same developer in top 1,000 ranking positions, and whether an app is free or not.[10] With the term h_t we control for time dummies. The term ε_{jt} incorporates a pure error term and the impact of app unobserved heterogeneity that stems from not controlling for several determinants of app performance (e.g. the intrinsic quality of the app, its ranking, the ability and reputation of the developer, but also the buzz surrounding the app) that, in the theoretical model, we refer to as the perceived quality. Many of these determinants are likely to change with time, a fact that we try to account for in our estimations.

Importantly, our theoretical model suggests that these unobserved variables, via their effect on app perceived quality, influence the decision to update. Consequently, u_{jt}, the

[9] We treat the number of downloads as zero if the app does not appear in the top 1,000 ranking in a given country, in the relevant period. As a robustness check, we re-run our analysis by assuming downloads as random numbers drawn from a uniform distribution over the interval 0 to the number of downloads obtained by the app in the $1,000^{th}$ rank position in the country of interest. The results of these regressions do not differ in any significant respect from those presented in the paper.

[10] Regressions do not include user rating in the set of controls as, due to its cumulative nature, it varies only marginally from month to month and therefore it is ill-suited for our IV estimations based on first differences. We could have controlled for user rating in the OLS estimations; however, the variable becomes statistically not significant once we include fixed effects. For reasons of comparability with our instrumental variables estimations we have finally decided not to include user rating in the OLS regressions.

dummy variable for the release of an update, is likely to be correlated with the error term ε_{jt}. In order to account for this correlation, we first estimate the growth equation including developer fixed effects.[11] This allows us to cope with the time-invariant component of the unobserved heterogeneity. Finally, to control also for time-variant unobserved heterogeneity we conduct an instrumental variable estimation based on variables in first differences.

Before moving on to the presentation of the results, let us discuss the instruments we employ in the estimations. Notice that, on top of the potentially endogenous variable u_{jt}, in the IV estimation we also instrument the lagged dependent variable g_{jt-1}, as it is standard in dynamic panel data models.[12]

5.1.2 Instruments

In our analysis, we employ three types of instruments based on different rationales. Estimations are in first differences to account for unobserved app heterogeneity, but for ease of exposition, we present each type of instrument in "levels".

I. The first type of instrument is based on Anderson and Hsiao (1981) and is commonly used in dynamic linear panel models with a large number of cross-section units observed for a limited number of periods (Arellano and Bond, 1991; Arellano and Bover, 1995; Bond, 2002). It relies on the assumption of "sequential exogeneity", which means that in an autoregressive model of order one we can use a two-period lag value of the dependent variable to instrument its lagged value. In our case, we instrument g_{jt-1} with g_{jt-2}, because the latter is correlated with the former – as g_{jt-2} is the lagged value of g_{jt-1} – but not with the future error term ε_{jt}. To avoid correlation with ε_{jt}, we instrument the endogenous variable u_{jt} with u_{jt-2}, and not with u_{jt-1}.[13]

II. In a second type of instrument we exploit information coming from other apps produced by the same developer. The relevance of these instruments rests on the assumption that multi-app developers share common underlying resources that, via either substitutability or complementarity, determine correlated growth/update trajectories for their app portfolios. We instrument u_{jt} with the share of apps of the same developer that are updated. This instrument would have missing values for single-app developers. In order to avoid such missing values we replace them with the average share of apps that are updated in the whole sample; we also include in the set of instruments an indicator variable for the single-app developer status. Like so, g_{jt-1} is instrumented

[11] In a series of unreported OLS regressions, we include app rather than developer fixed effects. Results remain largely unchanged compared to those presented in the paper.

[12] See Bond (2002) on this point. In principle, also the set of covariates $\mathbf{x}_{jt}$ might be endogenous. Since they represent only controls in our estimates, rather than instrumenting them as our key variables, g_{jt-1}, u_{jt-1}, we have decided to account for their potential endogeneity by using lagged values.

[13] Recall that our regressions are in first differences, hence the error term is $\varepsilon_{jt} - \varepsilon_{jt-1}$, and clearly ε_{jt-1} is correlated with u_{jt-1}.

by the average growth rate of downloads of the other apps distributed by the same developer. For single app developers, missing vales are replaced as for u_{jt}.[14]

III. The last type of instruments includes exclusion restrictions, i.e. variables that are determinants of the decision to update, but are not expected to affect download growth once the update decision, along with lagged growth and app dummies, is controlled for. Specifically, the instruments we employ are the age of the version of the app, and the number of times the app has been updated.

5.1.3 Results

Table 2 shows the results of the regressions of the growth equation. Columns (1) and (2) for iTunes and (4) and (5) for Google Play display the OLS estimates, where in (2) and (5) we control for developer dummy variables. The results of the IV estimations, based on variables in first difference, are in columns (3) and (6).[15] The main variable of interest in the growth equation is *Update*, the dummy variable taking value 1 in the case the app is updated. Overall, these regressions confirm the preliminary evidence shown in Figure 1: in iTunes updates have a strong and significant impact on download growth, while in Google Play their effect is weaker, if not absent, both in magnitude and statistical significance. The coefficient of *Update* is smallest in the OLS estimations without developer dummies, it increases when we include developer dummies, and it is largest in the GMM-IV estimations. This trend is in line with our theoretical model which suggests that, not controlling for unobserved heterogeneity, the variable u_{jt} turns out to be correlated with ε_{jt}. Specifically, in Section 3 we have shown that developers tend to release updates in response to a drop in perceived quality, thus implying a negative correlation between this variable and the decision to update. When estimating the effect of updates on growth without controlling for app perceived quality, the coefficient of u_{jt} captures both the (positive) effect of the update on growth, as well as the (negative) effect induced by the drop in app perceived quality. This second effect counterbalances the former one, thus reducing the coefficient of $Update$. This is what occurs in the OLS estimations without fixed effects. By adding fixed effects we control

[14]Berry et al. (1995) and Hausman (1996) inspire this set of instruments, though there are fundamental differences from the original ones. These instruments are commonly used to tackle price endogeneity in discrete choice models of demand with differentiated goods. Secondly, prices are typically instrumented not only using own product characteristics but also the characteristics of the products of the competitors. The fact that our instruments are built only looking at the characteristics/performance of the apps produced by each developer, and not at those produced by the rivals rests on the observation that in app markets, competition among developers goes along lines different than those of oligopolistic producers on which classic Berry et al. (1995) and Hausman (1996) instruments are based; as a matter of fact, developers compete for users' attention and, therefore, competition occurs not only among developers producing close substitute products.

[15]For comparability, in Table 2 we present the OLS estimations based on the sample of apps used for the GMM-IV regressions where the computation of instruments requires apps to be observed for at least four consecutive periods (e.g. the instrument for lagged growth of downloads is the two-period lagged growth). We have also conducted OLS estimations based on the larger original sample, and results are analogous to those in the table.

for the time-invariant component of the app unobserved perceived quality. With GMM-IV estimations we also correct for the time-variant component of the unobserved heterogeneity.

The estimates in Table 2 confirm that the impact of an update on download growth is stronger in iTunes than in Google Play. Using the estimated coefficients of column (6), the autoregressive component of the model allows us to determine the dynamic impact of an update in iTunes. The estimated coefficient of the lagged dependent variable is negative; this suggests an oscillating dynamic behaviour of the impact of an update. We note that the release of a new version of the software immediately increases the rate of growth of downloads by 29.6%. This is then followed by a decrease of 3.9%, ensued by a further increase estimated at 0.5% two months later. The effect of updates on download growth then fades away, indicating that the overall increment is of about 26%.

The tests of hypotheses documented at the bottom of the table give evidence of the validity and strength of the instruments. We present the F-stats of the first-stage regressions, which show that the coefficients of the instruments are all highly significant, meaning that the instruments are strong. We complement the analysis on the strength and validity of the instruments with the χ^2 tests for underidentification – based on the Kleibergen-Paap statistic – and overidentification – based on the Hansen J statistic. The statistics provided in the table suggest that each equation is not underidentified (i.e. the null hypothesis of underidentification is rejected), and that instruments are valid (the null hypothesis of overidentification of instruments is not rejected).[16]

[Table 2 about here.]

In order to highlight the contribution of each type of instrument described in Section 5.1.2 and to convince the reader about the validity of our IV strategy, in the Appendix we re-run the GMM-IV estimations omitting one type of instrument at a time (see Table 9). Overall, these additional regressions pass the relevant tests for strength and validity of the instruments; on top of this, the coefficient of *Update* is stable all through the different regressions and this reassures us on the appropriateness of the empirical analysis.

The heterogeneous impact of the release of new versions of the software in the two stores that we have just documented in Table 2 may have different explanations. On the one hand, there may be a selection effect leading to systematic differences in apps and developers. For example, one may think that, on average, developers in iTunes are more skilled than those operating in Google Play and this would contribute to explaining why iTunes updates have a stronger impact on downloads. On the other hand, developers may behave differently in responding to institutional differences characterising the two stores. In line with our theoretical framework, the lower significance of the variable *Update* in Google Play can be interpreted as evidence that, absent any quality check, developers release both high and low quality updates which, on average, have a weaker impact on downloads. We refer to this as the store "quality check" effect.

In order to disentangle the selection from the quality check effect, we re-estimate the growth equation restricting the analysis to the apps that are distributed in both stores.

[16]First stage regressions of the estimated equations are reported in Table 10 in the Appendix.

For multihomed apps, developers and applications are the same in the two stores, and this allows us neutralise the selection effect, and to isolate the role of institutional differences characterising iTunes and Google Play.[17]

[Table 3 about here.]

The sample of multihomed apps is made of all the apps that we observe in both stores in at least one period.[18] Columns (1) and (5) of Table 3 show the OLS estimations for iTunes and Google Play, respectively; these estimations include developer dummies. GMM-IV estimations are presented in columns (3) and (7). Overall these regressions reveal that when estimations are restricted to multihomed apps, updates positively affect the growth of downloads in both stores. More specifically, for multihomed apps differences across stores are largely reduced, with the coefficient of *Update* in iTunes being only slightly greater than in Google Play. This evidence reveals that the selection effect contributes to explaining the different impact of updates in the two stores.

Nevertheless, in order to convince the reader that the quality check effect does play a role, we conduct additional estimations by exploiting an interesting feature of multihomed apps, that is, the fact that, for these apps, we observe whether a given update distributed in one store is also released in the other. In an attempt to identify the quality check effect, we classify updates for multihomed apps into two categories depending on whether they take place in one (*Update own*) or both (*Update both*) stores in the same period. The reasonable assumption behind this exercise is that when developers simultaneously release a new version of their software in both stores, the two upgrades are alike, i.e. of similar quality. As the release of updates in iTunes goes through a strict screening, this is like to say that an update in Google Play, which occurs also in iTunes, is quality checked. By contrast, an update of a multihomed app which occurs in Google Play only, is not quality checked. Columns (2) and (6) – OLS regressions with developer fixed effects – and (4) and (8) – GMM-IV regressions – of Table 3 show the estimations for the two stores when updates are classified in this way. Interestingly, and in line with our argument that the quality check plays a relevant role, in Google Play the coefficient of *Update both* (quality checked updates) has a positive and significant effect on downloads, whereas by contrast, *Update own* (not quality checked updates) does not impact on app performance. The results for iTunes (columns (2) and (4)) provide further evidence on the role of updates on app performance. Since in this store all updates are quality checked, the distinction between *Update own* and *Update both* has

[17]The focus on multihomed apps allows us to neutralise the selection that takes place in the supply-side of the market. Nonetheless, we believe that it also goes some way in neutralising differences in the demand-side, as apps that are distributed in both stores should be of interest to similar types of users.

[18]Note that in the estimations the number of observations may differ across the two stores. As a matter of fact, in order to be included in the estimates, an app not only needs to be multihomed but also to be observed in a number of consecutive periods in the store. Multihomed apps may not appear in both stores in the same period and this explains why the number of observations in columns (1) and (5) as well as in (3) and (7) do not coincide. On the contrary, observations in columns (2) and (6) as well as in (4) and (8) coincide as the definition of the variable *Update own* and *Update both* requires apps to be observed in both stores in the same periods.

no bearing on the quality of updates. Consistent with this observation, we find that both *Update own* and *Update both* have a positive and significant impact on the rate of growth of downloads in iTunes.[19]

5.1.4 Treatment effects estimation

In the previous section we employed standard regression techniques to estimate the growth equation. Alternatively, the impact of updates on app downloads can be evaluated using a propensity score method. This approach has become popular in the inference of causal analysis, and has been applied to extremely diverse fields of study. Considering updated apps as the treatment and those that are not updated as the control group, it is possible to estimate the effect of the treatment – the release of an update – on download growth, our outcome variable.

We use nearest-neighbor matching (NNM) as the matching estimator. According to this method, an observation in the treatment group is matched to an observation in the control group (or to more than one in the case of oversampling), based on some distance measure of the probability of being treated, which is calculated with the propensity score procedure. In our estimations, we employ the Mahalanobis distance, which is the default metric when employing the `Stata` add-on package `nnmatch`.

Critical for the efficiency of NNM is the selection of the control group of observations, which must be as similar as possible to the treated ones. To this end, we identify the control group of apps using a set of observable characteristics that includes: *i*) all controls we employed in the estimation of the growth equation (namely, lagged growth of downloads, age of the app, whether the app is free, whether the app has an in-app purchase option, number of other apps distributed by the same developer, period and category dummies), and *ii*) a set of dummy variables to control for the number of countries the app is in, an indicator variable for developers with more than five apps in the platform and the rating of the app. In relation to this latter control, notice that despite not being well suited as a control in the growth equation due to its negligible variability in time, app rating may represent a valid characteristic for the identification of the control group of apps. Actually, it is quite natural to believe that two apps with a similar rating are more likely to be comparable.[20]

Results of these estimations are reported in the top part of Table 4. In columns (1) and (5) estimations are conducted with only one nearest-neighbor observation for matching. In order to check for the robustness of our analysis, in columns (2)-(4) and (6)-(8) we allow for oversampling (we use 25, 50 and 100 nearest neighbors for matching). Results largely confirm our previous findings; when estimations are run on the whole sample of apps, updates do not impact the rate of growth of downloads in Google Play, while they have a positive and

[19]Notice that while the coefficients in the OLS estimations are in line with previous regressions, the estimated coefficient of *Update own* in the GMM-IV regression is of a much larger magnitude; unfortunately, we do not have a clear explanation for this latter finding.

[20]Because we are matching on several continuous variables (lagged growth of downloads, age of the app, the number of other apps distributed by the same developer, rating) we include a bias-correction term based on these covariates.

significant impact in iTunes. This occurs both without and with oversampling. Specifically, in iTunes the average treatment effect varies between 0.146 and 0.166 meaning that, all else equal, apps that are updated produce a growth in downloads which is about 15% higher than that of apps that are not updated.

[Table 4 about here.]

We have then conducted NNM estimations restricting the sample to multihomed apps, in an attempt to isolate the effect of institutional differences across the two stores. The middle panel of Table 4 reports the results of these estimations. Similarly to what we have found with the classic regression analysis, multihomed app updates positively affect download growth in both stores, but this time the average treatment effect of *Update* in iTunes is much larger than that in Google Play. NNM estimations suggest, therefore, that the release of a new version of the software has greater impact in iTunes even when we restrict to multihomed apps.

Again, in order to isolate the role of the quality check effect, for the sample of apps that are present in both stores simultaneously we estimate the impact of updates by distinguishing between updates that occur in both stores and updates released in one store only. For this selected sample it is possible to study the effect of: (i) not updating the app, (ii) updating the app in one store only (*Update own*), and (iii) updating the app in both stores (*Update both*). In this case, the treatment variable is multivalued, and takes values 0, 1, and 2. For this reason we employ the inverse probability weighted estimator to estimate the average treatment effect of treatment 1 vs 0 and of treatment 2 vs 0, and then calculate the weights using a multinomial logit model.[21] Results are in line with our previous findings. With respect to apps that are not updated, those that are updated in both stores experience a significant increase in download growth. This occurs both in iTunes and in Google Play. By contrast updates that occur in only one store generate a different impact. Consistent with the fact that such updates are quality checked only in iTunes, we find that they stimulate downloads only in this store (with almost the same effect as the *Update both* type), while they have no impact in Google Play. We interpret these findings as a further confirmation of the role played by the quality check in iTunes.

5.2 The decision to update

Let us now turn to the second objective of our empirical analysis: the determination of the decision to release an update. Our theoretical model suggests that developers take this decision by looking at the perceived quality of the app. As explained, perceived quality depends on several factors such as: the intrinsic quality of the app, its ranking in the store, the developer's ability, and the buzz surrounding the app. Unfortunately, we do not observe

[21] Following the logic of propensity-score matching, this procedure includes only observations that share the common support in all treatments. For this reason the number of apps on which estimations are calculated do not coincide in the two stores.

these factors but relying on the natural assumption that perceived quality affects downloads, we use downloads growth as a proxy for app perceived quality.

Specifically, in what follows, we estimate the impact of past download growth on the decision to update. This approach has limitations that we discuss below (see Section 5.2.1). In the same section, we also deal with another potential shortcoming of our analysis, which relates to the fact that we have information only on five European countries. Clearly, this may represent a problem if developers take their decisions looking at the worldwide number of downloads of their apps or at the downloads in a set of countries different from ours.

We represent the "update equation" as a linear autoregressive distributed lag model of order 1;[22] formally, for each store, we estimate the following equation:

$$u_{jt} = \gamma_1 g_{jt-1} + \gamma_2 u_{jt-1} + h_t + \mathbf{w}_{jt}\boldsymbol{\beta} + \epsilon_{jt}.$$

As before, u_{jt} is the update variable in period t, g_{jt-1} is the lagged aggregate rate of growth of downloads of app j, and the row vector $\mathbf{w}_{jt}$ includes a set of controls: a constant, a dummy for the category the app belongs to, the age of the app, the age of the last version of the app, a variable capturing the presence of an in-app purchase option, the number of apps distributed by the same developer in top 1,000 ranking positions, and whether an app is free or not, where the last three variables are lagged one period to avoid endogeneity due to simultaneity. With the term h_t we control for the time fixed effect.

The term ϵ_{jt} incorporates a pure error term and the impact of the unobserved heterogeneity, i.e. factors that we do not observe but that are likely to influence the decision to update, such as the ability or the propensity of the developer to release updates. In the estimates we account for the unobserved heterogeneity by controlling for developer fixed effects and using GMM-IV estimations. In the latter set of regressions, we instrument the variable g_{jt-1} to control for possible correlation between past growth and potential omitted variables affecting the decision to update; we also instrument the lagged update variable, u_{jt-1}, as it is standard in dynamic panel data models. The types of instruments used in the estimation follow the same logic as for the growth equation. Specifically, we instrument u_{jt-1} with its lagged value, u_{jt-2} (Type I instruments). We then exploit the information coming from the other apps of the same developer to instrument g_{jt-1} with the average growth rate of downloads of the other apps distributed by the same developer;[23] as with the growth equation, we include among the instruments an indicator variable for the single-app developer status (Type II instruments). Finally, we use as instrument the lagged number of countries the app is present in the top 1000 apps (Type III instruments).

Table 5 shows the estimates of the update equation. As with the growth equation, we present OLS and GMM-IV regressions. OLS estimates, with and without developer fixed

[22] We have chosen to estimate the decision to update as a linear probability model, facing the limitations of the methodology, but aware that the issues related to linear probability models are less severe than imposing a parametric assumption that may not hold, and lead to a misspecified model.

[23] Again, for single app developers missing values are replaced by the average growth rate of downloads of multi-app developers.

effects,[24] are in columns (1)-(2), for iTunes, and (5)-(6), for Google Play. The GMM-IV estimations, based on variables in first differences, are provided in columns (3) and (7), for the whole sample, and in columns (4) and (8), for the subsample of multihomed apps.

[Table 5 about here.]

The main variable of interest is the lagged growth of downloads. In all the estimations for both stores, the coefficient of g_{jt-1} is negative. However, it is never significant for Google Play, while it is significant in the OLS estimations without developer fixed effects and in both of the GMM-IV estimations for iTunes. These findings seem to confirm that developers behave differently in the two stores. In line with our theoretical predictions, we find some evidence that developers of iTunes apps use updates in reaction to a decline in perceived quality, here approximated by download growth. The same does not occur in Google Play, where, no matter the chosen specification, past performance does not have an impact on the decision to release a new version of an app. We interpret this evidence based on the different institutional features characterising iTunes and Google Play. As mentioned, the quality control implemented by Apple limits the freedom of developers to publish updates. As a consequence, developers who have written an update, and are ready to publish it in the store, may decide to delay publication until it is really necessary, i.e. when the perceived quality worsens and triggers a response to counter the drop in downloads. By contrast, in Google Play, developers can publish apps any time they want without any quality screening. At the very moment an update is ready, they can make it available for download with a simple click of the mouse; in the case they later need to counter a drop in perceived quality, they can further distribute another update of any quality. In this environment, updates are continuously published, thus diluting the impact of perceived quality on the decision to release a new version of the app.

5.2.1 On the robustness of the update equation

As mentioned above, our estimations of the update equation have a couple of important limitations. The first one concerns our approximation of the previous period app perceived quality based on the lagged growth rate of downloads. In reality, perceived quality is influenced also by factors other than downloads such as the ranking of the app in the store, the reputation of the developer, the amount of revenues the developer is able to collect, etc. Another dimension which developers may look at is certainly app usage. Which dimension is the most relevant one for the decision to update depends on the developer and on the type and the characteristics of the app.

One may wonder whether our results are still valid if developers perceive the quality of their apps along other dimensions than downloads. An attempt to verify the robustness of our findings is in columns (1)-(6) of Table 6, where we re-estimate the update equation

[24] For comparability, in Table 5 we present the OLS estimations based on the sample of apps used for the GMM-IV regressions. We have also conducted OLS estimations based on the larger original sample, and results are analogous to those in the table.

considering sub-samples of apps that differ in terms of the likely importance of usage. We run separate regressions distinguishing between apps with/without the in-app purchase option and between free/paid apps (this latter distinction is for iTunes apps only as in Google Play nearly all applications are free). Apps with in-app purchases, as well as free apps, are likely to be those for which developers care more about usage. For example, for the apps with the in-app purchase option a more intense use increases the chances of generating revenues by activating in-app purchases. Similarly, free apps are most likely to generate revenues through advertising and, therefore, usage. Columns (1) and (3) of Table 6 confirm that in iTunes past performance negatively and significantly impacts the developer's decision to update both apps in the sample with and without the in-app purchase option. As expected, the magnitude of the coefficient is lower when the option is present as for these apps usage is likely to be more relevant. The fact that for these apps the lagged growth of downloads is still significant reassures us on the validity of our previous estimates. The same holds true when distinguishing between free/paid apps. For free apps, the coefficient of the lagged download growth is smaller, but still negative and significant. As far as Google Play apps are concerned, independently from the presence of the in-app purchase option, past download growth does not impact on the decision to release an update. This result is in line with our previous findings shown in Table 5.

The second potential limitation of the estimation of the update equation relates to the fact that we have information only on five European countries. Hence, in estimating the impact of lagged download growth on the decisions to update we implicitly assume that those five are the reference countries for developers' decision. However, since updates become available worldwide as soon as they are released, it might be the case that developers base their strategies by looking at the performance in a wider set of countries or at the performance in countries other than those we consider (e.g. US-based developers might take their decisions by looking at the performance in the US). In both situations, by considering France, Germany, Italy, Spain and the UK we would not control for the right set of countries used in determining whether to make an update.

In order to tackle this concern, we employ useful information provided in the Priori dataset, namely whether or not an app is local. 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 countries composing our sample more likely represents the reference developers use in order to take their decisions. Therefore, we re-estimate the update equation by restricting the sample to local apps only.

The results of these estimations are given in columns (7) and (8) of Table 6 and largely confirm that the lagged growth rate of downloads of the app does not affect the decision to release an update in Google Play, while it does in iTunes.

[Table 6 about here.]

6 Extensions

We devote this section to presenting a couple of extensions of our analysis. In Section 6.1, we re-estimate the growth and the update equations for iTunes distinguishing between two different types of updates. In Section 6.2, we focus on the growth equation and we conduct a country-level analysis.

6.1 Focusing on iTunes: Major *vs* minor updates

Our analysis has revealed that in iTunes updates play a prominent role: they have stronger impact on downloads than in Google Play and they are more likely to be released in reaction to a decline in performance. For iTunes apps it is therefore reasonable to look at the role of updates at a greater details. We have done so by collecting additional information about the type of iTunes updates; we obtained this information from the App Annie website. 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. The distinction between the two types of updates rests on the extent to which they augment the functionalities of the app. Typically, developers consider a minor update as an upgraded version of an app aimed at fixing bugs (e.g. crashing) or at including minor additional features. Major updates are, instead, aimed at distributing software with significant jumps in functionalities. In principle, major updates are, on average, of higher quality – because of the more substantial changes in app functionalities that they introduce. However, it does not follow that minor updates are of low quality.[25] Provided that iTunes checks quality before publication, all updates, both major and minor, are above a minimum level of quality.

In Table 7 we present the results of the GMM-IV estimations for iTunes when we distinguish between two types of updates. Column (1) shows the growth equation, while columns (2) and (3) present the two update equations – one for minor and one for major updates. Estimates of the growth equation confirm to a large extent the findings of Section 5.1: updates (both major and minor) positively impact the growth rate of downloads. Even though the impact is mildly significant (the p-value is 0.068), the coefficient of the variable *Major update* is larger than that of *Minor update*, which suggests that major updates stimulate downloads more than minor ones. This result partially confirms that obtained in Leyden (2018). The author also finds evidence of a positive impact of both types of updates on the demand for applications, though no evidence is found in support of the fact that major updates impact downloads more than minor ones.

The main result obtained in Section 5.2 for the update equation is confirmed for minor but not for major updates: only for bug fixing and minor changes in the software code, does lagged download growth of the app impact the decision to update (columns (2) and (3)).

[25] Software bugs may create serious inconveniences to users and, in these cases, minor updates fixing the problem greatly improve the functioning of the app, and therefore can be considered of extremely high quality.

This evidence is consistent with what we find in part i) of Corollary 1: developers have stronger incentives to release updates that have a greater impact on the expected perceived quality of the app (updates with a larger Δ). Therefore, major updates – whose impact is expected to be larger on average – are released as soon as they are ready and no matter the perceived quality of the app, while minor updates are published strategically to counter a drop in perceived quality.

[Table 7 about here.]

6.2 The growth equation: country-level estimations

All our estimations have been conducted on the sample obtained by aggregating the amount of downloads apps received in the five countries we have information on. Aggregation is quite natural in our analysis. One of our aims is in fact to estimate the determinants of the release of a new version of the software which is a decision that developers take by looking at the overall performance of their apps. Nonetheless, the impact of updates on downloads can be also estimated at country level. In this section we check the robustness of our main findings by conducting a country-level estimation of the growth equation. This exercise is also useful as it enlarges the number of observations. With pooled data, an app that is present in, let us say, two countries in the same period is treated as two different observations, and this allows country-level estimations to be run on a significantly larger sample.

[Table 8 about here.]

In Table 8 we replicate Table 2 using country-level data: columns (1)-(2) and (4)-(5) show the OLS estimations for the two stores without and with developer dummies. Columns (3) and (6) report the results of the GMM-IV estimations. The results shown in the table reinforce our main finding: updates never – in no specification – affect downloads growth in Google Play while they always do so in iTunes. Table 8 reassures us on the validity of our empirical strategy. When estimations are conducted at country level, the magnitude of the *Update* coefficient is smallest in the OLS estimations without fixed effects, it increases when we control for developer fixed effects, and it is largest in the GMM-IV estimations, thus confirming the direction of the adjustment when we better control for the app unobserved heterogeneity.

7 Conclusions

App developers release new versions of their mobile applications with an extremely high frequency. In this paper, we have focused on app updates and we have presented a stylised model describing the determinants of the decision to release a new version of the software. Our analysis suggests that updates might be published strategically in order to revive apps and stimulate the buzz surrounding them. An interesting insight of our theoretical framework

is that when an app experiences a very poor performance, a developer might be tempted to release even a low-quality update, in the hope of reversing the trend.

Our empirical analysis − based both on classic regression techniques as well as on treatment effect estimations − has shown that updates have a stronger impact on downloads in iTunes than in Google Play. We argue that this result is partly related to a "selection effect" (systematic differences in the apps and developers operating in the two stores) and partly to a "quality check effect". This latter effect is consistent with the prediction of our theoretical analysis, which suggests that the lack of a quality control in Google Play can cause too frequent updating: developers release both high and low-quality updates, which, on average, have a smaller impact on downloads.

Overall, these findings indicate that an appropriate management of updates is of crucial importance for developers committed to improve the performance of their apps. They also highlight the relevance of the publication policies adopted by platforms. A well-designed quality check screens low-quality updates out and reduces the asymmetry of information between developers and users. An appropriate control of the quality of mobile applications by the store induces developers to release "better updates" helping them in stimulating positive buzz and improve app performance.

In addition, and still in line with the predictions of our theoretical analysis, we have found that in iTunes developers are more likely to release an update when their apps experience a decline in downloads. On the contrary, in Google Play, the past performance of the app has no impact on the developer's decision to release a new version of the software. Again, we argue that the difference between the two stores is related to the different way of regulating the release of updates.

Our paper suggests a couple of promising lines for future research. We have focused on the management of app updates, but nonetheless, the release of updates can be seen as part of a developer's overall app strategy which also includes the launch of new apps or the withdrawn of existing ones. Unfortunately, we cannot analyse such a strategy in full as, in our data, we do not observe the whole portfolio of apps of each developer. Possibly more importantly, we cannot tell how the different apps distributed by the same developer interact with each other (e.g., one app can be the evolution of a previous app, or it can represent a tool to promote another application of the same developer, etc.). Should more precise and complete data be available, one may investigate in greater details the determinants and the characteristics of developers' strategies.

Our paper has highlighted how the absence of a quality control system in Google Play may have perverse effects on app performance. Taking a broader perspective, these arguments point to the potential role of platform policies in generating value in app markets. On the one side, the absence of a strict quality screening in Google Play reduces the average performance of the apps − and, in fact, Google has recently taken actions against low-quality apps in order to limit their distribution. On the other hand, though, a looser quality control reduces the cost of development and the barriers to entry in the store − and the exponential growth in the number of applications available in Google Play stands as a clear signal of this. Hence, a worthwhile area of future investigation refers to the analysis of the optimal policy in terms

of value generation towards the distribution of applications.

Finally, we should mention that in our data the only indicator of app performance is related to first-time downloads. It could be interesting to extend our analysis in order to check whether updates impact also other dimensions of app performance such as, for instance, the degree of app usage.

Appendices

A Mathematical appendix

Proof of Proposition 1. On top of the scenario (U,N) discussed in the text, in order to characterise the equilibrium we need to determine the pay-offs of the two types of developers in the other scenarios, namely when they both update (U,U), when none of them update (N,N) and when only the developer of type H updates, (N,U). Following a procedure similar to the one used for the definition of $L_{U,N}$ and $H_{U,N}$, the discounted values of returns of developer of type L and H are, respectively:

$$L = \frac{\gamma\,\eta - \Delta + \tau}{2\gamma\,\eta}\,(\delta\,L) + \left(1 - \frac{\gamma\,\eta - \Delta + \tau}{2\gamma\,\eta}\right)\left(\overline{R} + \delta\,H\right) - \phi,$$

$$H = \frac{\gamma\,\eta - \Delta - q_H + \tau}{2\gamma\,\eta}\,(\delta\,L) + \left(1 - \frac{\gamma\,\eta - \Delta - q_H + \tau}{2\gamma\,\eta}\right)\left(\overline{R} + \delta\,H\right) - \phi.$$

Solving the system of these two expressions it is easy to derive the pay-offs for the two types of developers when they always update, $L_{U,U}$ and $H_{U,U}$:

$$L_{U,U} = \frac{(\Delta + \gamma\,\eta - \tau)\,\overline{R} + (\delta\,q_H - 2\,\gamma\,\eta)\,\phi}{(1-\delta)\,(2\,\gamma\,\eta - \delta\,q_H)},$$

$$H_{U,U} = \frac{(\Delta - \delta\,q_H + \gamma\,\eta + q_H - \tau)\,\overline{R} + (\delta\,q_H - 2\,\gamma\,\eta)\,\phi}{(1-\delta)\,(2\,\gamma\,\eta - \delta\,q_H)}.$$

Similarly, the discounted values of returns of developer of type L and H when they never update are, respectively:

$$L = \frac{\eta + \tau}{2\eta}\,(\delta\,L) + \left(1 - \frac{\eta + \tau}{2\eta}\right)\left(\overline{R} + \delta\,H\right),$$

$$H = \frac{\eta + \tau - q_H}{2\eta}\,(\delta\,L) + \left(1 - \frac{\eta + \tau - q_H}{2\eta}\right)\left(\overline{R} + \delta\,H\right),$$

and the pay-offs $L_{N,N}$ and $H_{N,N}$ are, therefore:

$$L_{N,N} = \frac{(\tau - \eta)\,\overline{R}}{(\delta - 1)\,(\delta\,q_H - 2\,\eta)}, \qquad \text{and} \qquad H_{N,N} = \frac{(\delta\,q_H - \eta - q_H + \tau)\,\overline{R}}{(1-\delta)\,(\delta\,q_H - 2\,\eta)}.$$

Finally, the discounted values of returns of the two developers when only the type H updates are:

$$L = \frac{\eta + \tau}{2\eta}\,(\delta\,L) + \left(1 - \frac{\eta + \tau}{2\eta}\right)\left(\overline{R} + \delta\,H\right),$$

$$H = \frac{\gamma\,\eta - \Delta - q_H + \tau}{2\gamma\,\eta}\,(\delta\,L) + \left(1 - \frac{\gamma\,\eta - \Delta - q_H + \tau}{2\gamma\,\eta}\right)\left(\overline{R} + \delta\,H\right) - \phi,$$

and the solution to the system is:

$$L_{N,U} = \frac{(\tau - \eta)\,\gamma\,\left(\overline{R} - \delta\,\phi\right)}{(1-\delta)\,(\delta\,\tau\,(\gamma - 1) + \delta\,(\Delta + qH) - 2\,\gamma\,\eta)}$$

$$H_{N,U} = \frac{(\gamma\,(\delta\,\tau - \eta) + (\delta - 1)\,(\Delta + qH - \tau))\,\overline{R} - \gamma\,\phi\,(\delta\,(\eta + \tau) - 2\,\eta)}{(1-\delta)\,(\delta\,\tau\,(\gamma - 1) + \delta\,(\Delta + qH) - 2\,\gamma\,\eta)}.$$

Note that in order for the probabilities characterising the various scenarios to take values in the $(0,1)$ interval, the following conditions on the parameters need to hold: $|\tau| < \eta$, $|\tau - q_H| < \eta$, $|\tau - \Delta| < \gamma\eta$ and $|\tau - \Delta - q_H| < \gamma\eta$.

Let us start with the equilibrium where type L always updates while type H does never; in order for (U,N) to be the an equilibrium it must be that $L_{U,N} - L_{N,N} > 0$ and $H_{U,N} - H_{U,U} > 0$. Using the above expressions:

$$L_{U,N} - L_{N,N} = \frac{(\delta\,(\tau - q_H) - \eta - (1-\delta)\eta)\left(\gamma\,(\delta\,qH - 2\,\eta)\,\phi + (\Delta + (\gamma - 1)\tau)\,\overline{R}\right)}{(1-\delta)\,(\gamma\,(\eta + \delta\,(\tau - q_H)) + \delta\,(\Delta - \tau) + \gamma\,\eta)\,(\delta\,qH - 2\,\eta)}.$$

This difference is positive for $\phi < \overline{\phi}$, where $\overline{\phi} = \frac{(\gamma-1)\tau + \Delta}{(2\,\eta - \delta\,q_H)\gamma}\,\overline{R}$; note, in fact, that $\delta\,(\tau - q_H) - \eta - (1-\delta)\eta < 0$ as $|\tau - q_H| < \eta$, that $\gamma\,(\eta + \delta\,(\tau - q_H)) + \delta\,(\Delta - \tau) + \eta\,\gamma > 0$ as $|\tau - q_H| < \eta$ and $|\tau - \Delta| < \eta\gamma$ and that $\delta\,qH - 2\,\eta < 0$ as $q_H < 2\eta$.[26]

As the difference $H_{U,N} - H_{U,U}$ is concerned, using the expressions of $H_{U,N}$ and $H_{U,U}$ found above, it follows that:

$$H_{U,N} - H_{U,U} = \frac{(\gamma\,\eta\,(1-\delta) + \Delta\,\delta + \gamma\,\eta - \delta\,\tau)\left((\delta\,q_H - 2\,\gamma\,\eta)\,\phi + (\Delta + (\gamma - 1)(\tau - q_H))\,\overline{R}\right)}{(1-\delta)\,(\gamma\,(\eta + \delta\,(\tau - q_H)) + \delta\,(\Delta - \tau) + \gamma\,\eta)\,(\delta\,q_H - 2\,\gamma\,\eta)}.$$

This difference is positive for $\phi > \underline{\phi}$, where $\underline{\phi} = \frac{(\gamma-1)(\tau - q_H) + \Delta}{2\,\eta\,\gamma - \delta\,q_H}\,\overline{R}$; note, in fact, that $\gamma\,\eta\,(1-\delta) + \Delta\,\delta + \gamma\,\eta - \delta\,\tau > 0$ as $|\tau| < \eta$, that $\gamma\,(\eta + \delta\,(\tau - q_H)) + \delta\,(\Delta - \tau) + \gamma\,\eta > 0$ as $|\tau - q_H| < \eta$ and $|\tau - \Delta| < \eta\gamma$ and that $\delta\,qH - 2\gamma\eta < 0$ as $q_H < 2\eta$.

Therefore for $\phi \in (\underline{\phi}, \overline{\phi}]$, the developer of type L updates while that of type H does not. Simple algebra reveals that the equilibrium (U,N) exists, formally that $\overline{\phi} - \underline{\phi} > 0$; note that $\overline{\phi} > 0$ and $\underline{\phi}$ that can be either positive or negative.

Let us now consider the equilibrium (U,U) where both types of developers update. This equilibrium exists if $L_{U,U} - L_{N,U} > 0$ and $H_{U,U} - H_{U,N} > 0$. As seen above, the latter condition holds for $\phi < \underline{\phi}$; using the expressions of $L_{U,U}$ and $L_{N,U}$, the former condition holds for $\phi < \frac{(\gamma-1)\,\tau + \Delta}{2\,\gamma\,\eta - \delta\,q_H}\overline{R}$. Note that $\underline{\phi} < \frac{(\gamma-1)\,\tau + \Delta}{2\,\gamma\,\eta - \delta\,q_H}\overline{R}$, therefore (U,U) is an equilibrium for $\phi < \underline{\phi}$.

The last equilibrium we are left with is (N,N), where no one updates. This equilibrium exists if $L_{N,N} - L_{U,N} > 0$ and $H_{N,N} - H_{N,U} >$. According to what already seen when

[26] The conditions $\tau > q_H - \eta$ and $\tau < \eta$ imply that $q_H < 2\eta$.

discussing the equilibrium (U,N), the latter condition olds for $\phi > \overline{\phi}$; using the expressions of $H_{N,N}$ and $H_{N,U}$ presented above, the former condition holds for $\phi > \frac{(\gamma-1)(\tau-q_H)+\Delta}{(2\eta-\delta q_H)\gamma}\overline{R}$. Note that $\overline{\phi} > \frac{(\gamma-1)(\tau-q_H)+\Delta}{(2\eta-\delta q_H)\gamma}\overline{R}$, therefore (U,U) is an equilibrium for $\phi > \overline{\phi}$.

In order to conclude the proof, we need to show that (N,U) - the L type does not update, while the H type does - cannot be an equilibrium. Looking at the previous analyses, the developer of type L prefers not to update if $\phi > \frac{(\gamma-1)\tau+\Delta}{2\gamma\eta-\delta q_H}\overline{R}$, while that of type H prefers to update if $\phi < \frac{(\gamma-1)(\tau-q_H)+\Delta}{(2\eta-\delta q_H)\gamma}\overline{R}$. Therefore the equilibrium exists if and only if $\frac{(\gamma-1)(\tau-q_H)+\Delta}{(2\eta-\delta q_H)\gamma} - \frac{(\gamma-1)\tau+\Delta}{2\gamma\eta-\delta q_H} > 0$ or, after some algebraic manipulation, if $\delta\tau(\gamma-1)+\Delta\delta+\delta q_H-\delta\tau-2\gamma\eta > 0$. It is possible to prove by contradiction that this latter condition never holds; the lhs of the inequality is increasing in δ, and for $\delta = 1$, it reduces to

$$q_H > 2\gamma\eta - \Delta - \tau(\gamma-1). \tag{3}$$

We know that the model exists for $|\tau - \Delta - q_H| < \gamma\eta$ and for $|\tau| < \eta$; the first condition implies that $q_H < \tau - \Delta - \gamma\eta$. One can check that when this latter condition holds and provided that $\tau < \eta$, condition (3) is never satisfied. If it does not hold for $\delta = 1$, it never holds for any $\delta \in (0,1)$, and the scenario (N,U) can never be an equilibrium. □

Proof of Corollary 1. Part *i*) can be easily proved by verifying that $d\underline{\phi}/d\Delta > 0$ and $d\overline{\phi}/d\Delta > 0$; therefore for both types of the developer the incentives to update increase with Δ. To prove part *ii*) of the corollary we need to show that $\overline{\phi}$ and $\underline{\phi}$ are greater than zero for $\Delta = 0$. Given the parameter restrictions discussed in Proposition 1, it is possible to check that this is always the case for $\overline{\phi}$, while the sign of $\underline{\phi}$ is ambiguous. This implies that when $\Delta = 0$ the set of parameters such that it is optimal to update is always non empty for the developer of type L, while it may be empty or not for type H. □

B Instrumental variable estimations

B.1 The contribution of each set of instruments

In order to highlight the contribution of each set of instruments in the identification of the growth equation, it is useful to compare the IV regressions shown in the paper with those obtained when omitting one set of instruments at a time. For the sake of brevity, in this appendix we focus on the regressions obtained employing the original dataset, but things do not change when estimations are conducted using the sample of multihomed apps.

Comparing the results shown in Table 2, with the results obtained when omitting one set of instruments at a time reported in Table 9, it follows that:

- Type I instruments are useful in order to improve the identification of lagged growth of downloads. When omitting these instruments, the F-stat for lagged growth and the Under-identification χ^2-stat drop substantially.

- Type II instruments appear to be (mildly) useful in order to ensure the exogeneity of the instruments. Omitting them we fail to reject the Over-identification test for the case of Google Play (see column (4)). These instruments do not appear to play a major role in the other regressions.

- Type III instruments: comparing the estimates of the paper with those shown in columns (3) and (6), it follows that the these instruments are useful in order to improve the identification of the variable *Update*. When omitting them, the F-stat for Update and the Under-identification χ^2-stat drop substantially.

It deserves also to be noticed that the main coefficient of interest in these estimates (the coefficient of Update) is fairly stable all through the regressions obtained when omitting Type I and Type II instruments. The only noticeable change occurs when we omit the last set of instruments; nonetheless, provided that these instruments are useful for the identification of the variable Update itself, this should not be too worrying.

We are aware of the fact that the test for over-identifying restrictions rests on the crucial (and untestable) assumption of having enough exogenous instruments. However, as observed by Murray (2006), in case the instrumental variables are grounded in different rationales, a failure to reject the over-identifying restrictions provides more comfort about the exogeneity of the instruments. As we argue in the paper, our set of instrumental variables is based on different rationales and in Murray's spirit this should alleviate concerns about their exogeneity. On similar lines, also the fact that the coefficient of the variable of interest (*Update*) remains stable in all the regressions in the table provides further evidence about the appropriateness of our IV strategy.

[Table 9 about here.]

B.2 First stage regressions

[Table 10 about here.]

References

Anderson, T. and C. Hsiao (1981). Estimation of Dynamic Models with Error Components. *Journal of the American Statistical Association*, 76(374):598–606.

Arellano, M. and Bond, S. (1991). Some tests of specification for panel data: Monte Carlo evidence and an application to employment equations. *The review of economic studies*, 58(2):277–297.

Arellano, M. and Bover, O. (1995). Another look at the instrumental variable estimation of error-components models. *Journal of econometrics*, 68(1):29–51.

Berry, S., Levinsohn, J., and Pakes, A. (1995). Automobile Prices in Market Equilibrium. *Econometrica*, 63:841–890.

Bond, S. (2002). Dynamic Panel Data Models: a Guide to Micro Data Methods and Practice. The Institute for Fiscal Studies, UCL, WP09/02.

Bresnahan, T., Davis, J., and Yin, P.-L. (2014). Economic Value Creation in Mobile Applications. In Jaffe, A. and Jones, B., editors, *The Changing Frontier: Rethinking Science and Innovation Policy*. Chicago Scholarship Online, Chicago.

Carare, O. (2012). The Impact of Bestseller Rank on Demand: Evidence From the App Market. *International Economic Review*, 53(3):717–742.

Chevalier, J. A. and Mayzlin, D. (2006). The Effect of Word of Mouth on Sales: Online Book Reviews. *Journal of Marketing Research*, 43(3):345–354.

Datta, D. and Sangaralingam, K. (2013). Do App Launch Times Impact their Subsequent Commercial Success? *IEEE 2013 International Conference on Cloud Computing and Big Data*, Dec 2013:205–210.

Diamond, P. A. (1971). A Model of Price Adjustment. *Journal of Economic Theory*, 3(2):156–168.

Duan, W., Gu, B., and Whinston, A. B. (2008). Do Online Reviews Matter? An Empirical Investigation of Panel Data. *Decision Support Systems*, 45(4):1007–1016.

Duan, W., Gu, B., and Whinston, A. B. (2009). Informational Cascades and Software Adoption on the Internet: an Empirical Investigation. *Management Information Systems Quarterly*, pages 23–48.

Garg, R. and Telang, R. (2013). Inferring App Demand From Publicly Available Data. *Management Information Systems Quarterly*, 37(4):1253–1264.

Ghose, A. and Han, S. P. (2014). Estimanting Demand for Mobile Applications in the New Economy. *Management Science*, 60(6):1470–1488.

Gollier, C., Koehl, P., and Rochet, J. (1997). Risk-Taking Behavior with Limited Liability and Risk Aversion. *The Journal of Risk and Insurance*, 64(2):347–370.

Google (2015). Mobile App Marketing Insights. How Consumers Really Find and Use Your Apps. May 2015.

Greenbaum, J. (2005). This is Just In: Consumers Hate Their Software Vendors. *InformationWeek*. Available at: http://www.informationweek.com/.

Hausman, J. A. (1996). Valuation of New Goods Under Perfect and Imperfect Competition. In Bresnahan, T. and Gordon, R., editors, *The Economics of New Goods*, chapter 5, pages 209–248. Chicago University Press, Chicago.

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.

Lee, G. and Raghu, T. S. (2014). Determinants of Mobile Apps Success: Evidence from App Store Market. *Journal of Management Information Systems*, 31(2):133–170.

Lee, Y.-J., Hosanagar, K., and Tan, Y. (2015). Do I follow My Friends or the Crowd? Information Cascades in Online Movie Ratings. *Management Science*, 61(9):2241–2258.

Leyden, B. (2018). There's An App (Update) For That. Understanding Product Updating Under Digitization. Paper presented at the 2017 Conference on Internet Commerce and Innovation, The Searle Center on Law, Regulation, and Economic Growth.

Li, X. and Hitt, L. M. (2008). Self-selection and Information Role of Online Product Reviews. *Information Systems Research*, 19(4):456–474.

Liu, Y. (2006). Word of mouth for movies: Its dynamics and impact on box office revenue. *Journal of marketing*, 70(3):74–89.

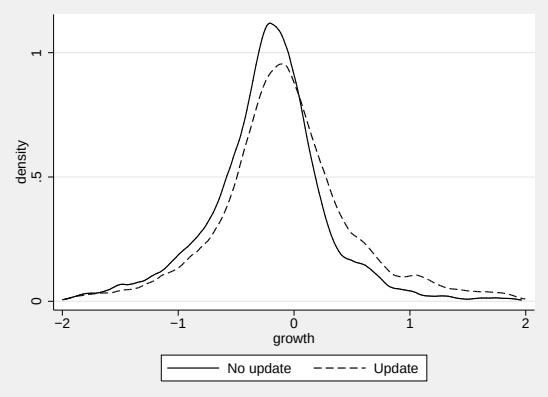
Murray, M. P. (2006). Avoiding Invalid Instruments and Coping with Weak Instruments. *Journal of Economic Perspectives*, 20(4):111–132.

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

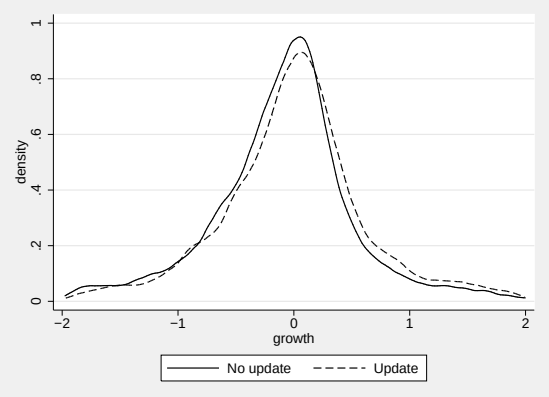
Sankaranarayanan, R. (2007). Innovation and the Durable Goods Monopolist: The Optimality of Frequent New-version Releases. *Marketing Science*, 26(6):774–791.

Yin, P.-L., Davis, J. P., and Muzyrya, Y. (2014). Entrepreneurial Innovation: Killer Apps in the iPhone Ecosystem. *American Economic Review*, 104(5):255–259.

Figure 1: Downloads growth density function



(a) iTunes



(b) Google Play

Table 1: Summary statistics

	iTunes			Google Play		
	N	mean	std. dev.	*N*	mean	std. dev.
Free	14,759	0.917	0.276	15,981	0.999	0.033
Price (if free=0)	1,229	2.759	2.907	17	2.447	1.148
In-app	14,759	0.561	0.496	15,981	0.297	0.457
Local	14,759	0.349	0.477	15,981	0.374	0.484
Age (in months)	14,759	19.350	15.997	15,981	15.038	13.993
Number of updates (all time)	14,759	10.082	9.891	15,981	33.434	78.495
Update (sampling period)*	8,534	0.453	0.498	7,990	0.542	0.498
# apps same developer	14,759	3.585	5.163	15,981	2.983	4.583
Number countries	14,759	2.030	1.491	15,981	1.877	1.395
Monthly downloads	14,759	58,142	166,451	15,981	249,840	1,244,288
Download growth**	8,657	-0.160	0.646	8,224	-0.026	0.763
			Subsample (A)			
Free	3,700	0.958	0.202	2,956	0.999	0.032
Price (if free=0)	157	3.160	2.603	3	3.873	0.203
In-app	3,700	0.609	0.488	2,956	0.349	0.477
Local	3,700	0.335	0.472	2,956	0.365	0.481
Age (in months)	3,700	23.848	15.529	2,956	20.664	13.849
Number of updates (all time)	3,700	13.403	10.837	2,956	54.908	91.956
Update (sampling period)*	3,700	0.411	0.492	2,956	0.507	0.500
# apps same developer	3,700	4.140	5.896	2,956	3.982	6.072
Number countries	3,700	2.379	1.632	2,956	2.398	1.667
Monthly downloads	3,700	69,533	157,134	2,956	476,337	2,189,596
Download growth**	3,700	-0.170	0.527	2,956	-0.070	0.688
			Subsample (B)			
Free	1,466	0.977	0.151	1,120	1.000	0
Price (if free=0)	34	2.325	1.885	0	-	-
In-app	1,466	0.622	0.485	1,120	0.472	0.499
Local	1,466	0.335	0.472	1,120	0.405	0.491
Age (in months)	1,466	25.960	16.883	1,120	22.758	14.559
Number of updates (all time)	1,466	15.700	11.643	1,120	65.546	100.517
Update (sampling period)*	1,466	0.503	0.500	1,120	0.609	0.488
# apps same developer	1,466	4.611	6.895	1,120	4.693	6.675
Number countries	1,466	2.760	1.757	1,120	2.556	1.753
Monthly downloads	1,466	103,761	213,553	1,120	556,097	1,345,918
Download growth**	1,466	-0.168	0.517	1,120	-0.026	0.689

*Updates and download growth are in first differences and hence have less observations. **Growth rates are expressed as log changes.

Table 2: Download growth estimates by store (growth equation)

	iTunes			GP		
	OLS		GMM	OLS		GMM
	(1)	(2)	(3)	(4)	(5)	(6)
Update	0.188^a	0.245^a	0.296^a	0.061^c	0.103^c	0.130
	(0.017)	(0.028)	(0.045)	(0.025)	(0.046)	(0.080)
Lag growth	-0.278^a	-0.379^a	-0.131^a	-0.194^a	-0.323^a	-0.165^a
of downloads	(0.023)	(0.028)	(0.022)	(0.021)	(0.031)	(0.026)
Age	-0.000	0.000	†	0.008^a	0.009	†
	(0.001)	(0.001)	†	(0.001)	(0.005)	†
Lag in-app	0.009	0.045	0.235	-0.006	-0.092	0.243
	(0.020)	(0.074)	(0.318)	(0.029)	(0.115)	(0.218)
Lag free	-0.088^c	-0.122	-0.095	0.104^b	-0.070	‡
	(0.045)	(0.083)	(0.468)	(0.033)	(0.045)	‡
Lag # apps	0.000	-0.039	-0.052^a	0.001	-0.037^c	-0.026^c
same developer	(0.002)	(0.021)	(0.015)	(0.002)	(0.018)	(0.012)
Tests of hypotheses						
F-stat lagged growth			603.11^a			428.89^a
F-stat update			106.40^a			49.59^a
Underidentif. χ^2-stat			298.00^a			182.69^a
Overidentif. J-test			4.88			9.76
Dev FE	NO	YES	YES*	NO	YES	YES*
N	3,700	3,700	3,700	2,956	2,956	2,956
R^2	0.129	0.158	0.181	0.195	0.448	0.281

Significance level: $^a p < 0.001$, $^b p < 0.01$, $^c p < 0.05$. Clustered (by developer) standard error in parenthesis. Time and category dummies are included. *Variables in first difference. †Cancels out in first differences. ‡No observations, as almost all top 1,000 apps are free. Instruments: Type I: L2.(growth, update), Type II: D2.(own-update, L1.own-growth, dummy for single-app developer), Type III: D1.L1.(age update, number of updates); the term "own" stands for average value of the variable for the other apps distributed by the same developer.

Table 3: Download growth estimates by store for multihomed apps (growth equation)

	iTunes				GP			
	OLS		GMM		OLS		GMM	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Update	0.214^a		0.287^a		0.150^b		0.266^b	
	(0.033)		(0.059)		(0.043)		(0.097)	
Update both		0.313^a		0.317^c		0.240^a		0.329^b
		(0.052)		(0.130)		(0.062)		(0.104)
Update own		0.172^c		0.881^c		0.123		0.044
		(0.079)		(0.382)		(0.072)		(0.145)
Lag growth	-0.291^a	-0.379^a	-0.124^b	-0.195^a	-0.309^a	-0.286^a	-0.177^a	-0.180^a
of downloads	(0.038)	(0.049)	(0.037)	(0.052)	(0.037)	(0.051)	(0.038)	(0.043)
Tests of hypotheses								
F-stat lagged growth			214.20^a	63.82^a			155.09^a	67.25^a
F-stat update			62.86^a				14.87^a	
F-stat update both				15.90^a				18.12^a
F-stat update own				2.57^b				9.02^a
Underidentif. χ^2-stat			161.08^a	33.37^b			88.15^a	69.70^a
Overidentif. J-test			2.52	5.30			0.68	6.34
Dev FE	NO	YES	YES*	YES*	NO	YES	YES*	YES*
N	2,143	1,074†	1,466	676†	1,714	1,074†	1,120	676†
R^2	0.304	0.354	0.173	0.165	0.351	0.388	0.284	0.304

Significance level: $^a p < 0.001$, $^b p < 0.01$, $^c p < 0.05$. Clustered (by developer) standard error in parenthesis. Time and category dummies are included. *Variables in first difference. †Sample of apps observed in both stores in the same period. Instruments (columns 3 and 7): Type I: L2.(growth, update), Type II: D2.(own-update, L1.own-growth, dummy for single-app developer), Type III: D1.L1.(age update, number of updates). Instruments (columns 4 and 8): Type I: L2.(growth, update own, update both), Type II: D2.(own-update own, own-update both, L1.own-growth, dummy for single-app developer), Type III: D1.L1.(age update, age update both, number of updates, number of updates both). In the instruments, the term "own" stands for average value of the variable for the other apps distributed by the same developer. Controls: L1.(download growth, in-app, dummy if free, number other apps by the developer), and dummies for time and category.

Table 4: Download growth treatment-effect estimates by store

	iTunes				GP			
	# of nearest neighbours used for matching							
	1	25	50	100	1	25	50	100
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
	Whole sample							
Update	0.166^a	0.146^a	0.152^a	0.158^a	0.033	0.015	0.009	0.012
	(0.025)	(0.031)	(0.031)	(0.030)	(0.027)	(0.027)	(0.027)	(0.027)
N	5,695	5,695	5,695	5,695	4,856	4,856	4,856	4,856
	Sample of multihomed apps							
Update	0.215^a	0.296^a	0.363^a	0.316^a	0.102^c	0.169^b	0.174^b	0.155^b
	(0.028)	(0.034)	(0.041)	(0.040)	(0.060)	(0.063)	(0.053)	(0.045)
N	2,143	2,143	2,143	2,143	1,714	1,714	1,714	1,714

	Sample of multihomed apps: multinomial logit inverse-probab. weighting model	
Update both	0.238^a	0.175^a
	(0.042)	(0.046)
Update own	0.203^c	0.086
	0.081	0.057
N	693	846

Dependent variable: download growth. Treatment: update dummy. Controls: L1.(download growth, in-app, dummy if free, number other apps by the developer), age of the app, dummies for 5 categories, dummies for developer (developers with less that 6 apps –10 apps for the multinomial estimation– are treated as a single fringe developer). Significance level: $^a p < 0.001$, $^b p < 0.01$, $^c p < 0.05$. Abadie-Imbens robust standard error in parenthesis.

Table 5: Update estimates by store (update equation)

	iTunes				GP			
	OLS		GMM		OLS		GMM	
	(1)	(2)	(3)	(4) (mh)	(5)	(6)	(7)	(8) (mh)
Lag update	0.183^a	-0.151^a	0.026	0.072	0.285^a	-0.092^c	0.248^a	0.350^a
	(0.022)	(0.027)	(0.030)	(0.053)	(0.023)	(0.036)	(0.038)	(0.070)
Lag growth of downloads	-0.038^c	-0.009	-0.084^a	-0.092^c	-0.013	-0.011	-0.017	-0.012
	(0.015)	(0.017)	(0.024)	(0.044)	(0.014)	(0.018)	(0.023)	(0.041)
Age	0.001	-0.001	†	†	0.002^c	-0.001	†	†
	(0.001)	(0.002)	†	†	(0.001)	(0.002)	†	†
Lag agever	-0.021^a	-0.010	0.148^a	0.204^a	-0.022^a	-0.021^a	0.247^a	0.296^a
	(0.002)	(0.005)	(0.014)	(0.025)	(0.003)	(0.005)	(0.017)	(0.040)
Lag in-app	0.020	0.133	-0.550^b	-0.405	0.136^a	0.167^c	-0.174	-0.117
	(0.021)	(0.089)	(0.130)	(0.239)	(0.024)	(0.076)	(0.099)	(0.124)
Lag free	0.044	0.039	0.334^c	‡	-0.329^a	0.004	‡	‡
	(0.040)	(0.077)	(0.131)	‡	(0.027)	(0.017)	‡	‡
Lag # apps same developer	-0.004^a	-0.019	-0.009	-0.021	-0.002	0.010	-0.006	-0.012
	(0.001)	(0.014)	(0.010)	(0.017)	(0.002)	(0.014)	(0.011)	(0.016)
Tests of hypotheses								
F-stat lagged growth			152.82^a	40.16^a			105.79^a	53.10^a
F-stat update			425.15^a	187.65^a			475.13^a	202.19^a
Underidentif. χ^2-stat			111.41^a	38.28^a			82.70^a	37.88^a
Overidentif. J-test			3.53	2.50			3.12	0.85
Dev FE	NO	YES	YES*	YES*	NO	YES	YES*	YES*
N	3700	3700	3700	1466	2956	2956	2956	1120
R^2	0.092	0.450	0.135	0.135	0.197	0.565	0.053	-0.010

Significance level: $^a p < 0.001$, $^b p < 0.01$, $^c p < 0.05$. Clustered (by developer) standard error in parenthesis.Time and category dummies are included. *Variables in first differences. †Cancels out in first differences. ‡No observations, as almost all top 1,000 apps are free. Instruments: Type I: L2.(update), Type II: D2.(L1.own-growth, dummy for single-app developer), Type III: D1.L1.(number of countries app is available in top 1,000 position); the term "own" stands for average value of the variable for the other apps distributed by the same developer.

Table 6: Robustness check for update estimates by store (update equation)

	In-app purchases				Free apps (iTunes only)		Local apps	
	Yes		No					
	iTunes	GP	iTunes	GP	Yes	No	iTunes	GP
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Lag growth of downloads	-0.062^{c}	-0.015	-0.129^{b}	-0.021	-0.076^{b}	-0.112^{b}	-0.358^{b}	-0.054
	(0.028)	(0.027)	(0.045)	(0.036)	(0.026)	(0.042)	(0.106)	(0.143)
Lag update	-0.022	0.298^{a}	0.100^{c}	0.227^{a}	0.037	-0.137	0.154^{c}	0.268^{b}
	(0.037)	(0.081)	(0.050)	(0.044)	(0.032)	(0.113)	(0.062)	(0.077)
Tests of hypotheses								
F-stat lagged growth	127.21^{a}	67.06^{a}	88.01^{a}	77.07^{a}	179.35^{a}	31.35^{a}	26.70^{a}	10.84^{a}
F-stat update	343.88^{a}	212.43^{a}	270.01^{a}	414.44^{a}	541.80^{a}	28.29^{a}	265.81^{a}	265.78^{a}
Under identif. χ^2-stat	78.60^{a}	43.87^{a}	40.24^{a}	52.24^{a}	106.46^{a}	16.31^{a}	7.10^{c}	12.22^{b}
Over identif. J-test	0.16	0.15	1.57	0.04	0.38	0.37	0.37	0.23
N	2,255	1,032	1,445	1,924	3,543	157	1,239	1,078
R^2	0.195	0.011	0.054	0.056	0.131	0.165	-0.062	0.029

Significance level: $^{a}p < 0.001$, $^{b}p < 0.01$, $^{c}p < 0.05$. Clustered (by developer) standard error in parenthesis. Control variables: L1.(in-app, dummy if free, # apps same developer, age version); time and category dummies are included. All regressions are in first difference. Instruments: Type I: L2.(update), Type II: D2.(L1.own-growth, dummy for single-app developer), Type III: D1.L1.(number of countries app is available in top 1,000 position); the term "own" stands for average value of the variable for the other apps distributed by the same developer.

Table 7: Download growth and major and minor update estimates (iTunes)

	Growth (1)	Major update (2)	Minor update (3)
Major update†	0.338	0.049^{c}	-0.373^{a}
	(0.185)	(0.022)	(0.075)
Minor update†	0.275^{a}	-0.004	0.106^{b}
	(0.047)	(0.010)	(0.032)
Lag growth	-0.129^{a}	-0.010	-0.081^{b}
of downloads	(0.022)	(0.007)	(0.025)
Tests of hypotheses			
F-stat lagged growth	466.04^{a}	103.81^{a}	103.81^{a}
F-stat major update	16.58^{a}	452.92^{a}	452.92^{a}
F-stat minor update	81.68^{a}	293.30^{a}	293.30^{a}
Under identif. χ^2-stat	263.63^{a}	109.86^{a}	109.86^{a}
Over identif. J-test	7.43	2.45	4.31
N	3,568	3,568	3,568
R^2	0.185	0.261	0.103

Significance level: $^{a}p < 0.001$, $^{b}p < 0.01$, $^{c}p < 0.05$. Clustered (by developer) standard error in parenthesis. †Lagged in columns (2) and (3). All regressions are in first difference. Control variables: L1.(in-app, dummy if free, # apps same developer, age version –age minor and major version in columns 2 and 3–), and time dummies. Instruments (column 1): Type I: L2.(download growth, update minor, update major), Type II: D2.(L1.own-growth, own-update minor, own-update major), Type III: D1.L1.(age minor version, age major version, number of updates). Instruments (columns 2 and 3): Type I: L2.(update minor, update major), Type II: D2.L1.(own-growth, own-update minor, own-update major), Type III: D1.L1.(number of countries app is available in top 1,000 position); the term "own" stands for average value of the variable for the other apps distributed by the same developer.

Table 8: Country-level download growth estimates

	iTunes			GP		
	OLS		GMM	OLS		GMM
	(1)	(2)	(3)	(4)	(5)	(6)
Update	0.135^a	0.215^a	0.218^a	0.024	0.042	0.071
	(0.016)	(0.028)	(0.040)	(0.021)	(0.040)	(0.063)
Lag growth of	-0.266^a	-0.413^a	-0.130^a	-0.286^a	-0.379^a	-0.217^a
downloads	(0.031)	(0.028)	(0.020)	(0.015)	(0.017)	(0.020)
Tests of hypotheses						
F-stat lagged growth			784.99^a			865.54^a
F-stat update			84.89^a			53.62^a
Under identif. χ^2-stat			182.32^a			129.42^a
Over identif. J-test			5.05			9.09
N	7,602	7,602	7,602	5,842	5,842	5,842
R^2	0.143	0.432	0.192	0.290	0.476	0.361
Dev FE	NO	YES	YES*	NO	YES	YES*

Significance level: $^a p < 0.001$, $^b p < 0.01$, $^c p < 0.05$. Clustered (by developer) standard error in parenthesis. *Variables in first difference. Control variables: age, L1.(in-app, dummy if free, # apps same developer), time and category dummies. Instruments: Type I: L2.(growth, update), Type II: D2.(own-update, L1.own-growth, dummy for single-app developer), Type III: D1.L1.(age update, number of updates); the term "own" stands for average value of the variable for the other apps distributed by the same developer.

Table 9: Download growth estimates excluding a type of instruments

	iTunes			GP		
	no Type I (1)	no Type II (2)	no Type III (3)	no Type I (4)	no Type II (5)	no Type III (6)
Lag growth of downloads	-0.421^b (0.143)	-0.133^a (0.022)	-0.130^a (0.023)	-0.155 (0.204)	-0.163^a (0.026)	-0.165^a (0.027)
Update	0.242^a (0.047)	0.304^a (0.046)	0.240 (0.136)	0.206 (0.118)	0.152 (0.082)	-0.176 (0.198)
Lag in-app	0.064 (0.289)	0.241 (0.318)	0.211 (0.327)	0.254 (0.242)	0.236 (0.219)	0.161 (0.228)
Lag free	0.007 (0.555)	-0.091 (0.469)	-0.065 (0.472)	† †	† †	† †
Lag # apps same developer	-0.032 (0.017)	-0.051^b (0.015)	-0.055^b (0.016)	-0.028 (0.016)	-0.025 (0.013)	-0.022 (0.013)
F-Stat lagged growth	4.5^a	1030.4^a	807.1^a	5.8^a	726.7^a	581.6^a
F-stat Update	137.9^a	168.5^a	12.2^a	53.0^a	73.5^a	10.1^a
Under identif. χ^2-stat	8.2	296.6^a	32.1^a	15.8^b	178.5^a	37.5^a
Over identif. J-test	1.6	3.1	4.2	5.4	7.6^c	4.8
N	3716	3700	3700	2965	2956	2968
R^2	0.333	0.181	0.181	0.271	0.280	0.258

Significance level: $^a p < 0.001$, $^b p < 0.01$, $^c p < 0.05$. Clustered (by developer) standard error in parenthesis. Regressions are in first differences. †No observations, as almost all top 1,000 apps are free. Control variables: L1.(download growth, in-app, dummy if free, number apps by the developer), time and category dummies. Instruments: Type I: L2.(growth, update), Type II: D2.(own-update, L1.own-growth, dummy for single-app developer), Type III: D1.L1.(age update, number of updates); the term "own" stands for average value of the variable for the other apps distributed by the same developer.

Table 10: First-stage estimates

	Growth equation				Update equation			
	iTunes		GP		iTunes		GP	
	L1.growth (1)	update (2)	L1.growth (3)	update (4)	L1.growth (5)	L1.update (6)	L1.growth (7)	L1.update (8)
Type I instruments:								
L2.growth	-1.207^a	-0.008	-1.259^a	-0.003				
	(0.019)	(0.015)	(0.024)	(0.015)				
L2.update	-0.034	0.124^a	-0.032	-0.133^a	-0.201^a	-0.731^a	-0.184^a	-0.524^a
	(0.021)	(0.021)	(0.023)	(0.020)	(0.027)	(0.018)	(0.030)	(0.012)
Type II instruments:								
D2.(own-update)	0.006	0.124^a	0.038	0.127^a				
	(0.017)	(0.022)	(0.025)	(0.027)				
D2.L1.(own-growth)	0.024	0.002	0.016	0.003	0.102^b	0.012	0.099^b	0.001
	(0.015)	(0.010)	(0.015)	(0.011)	(0.034)	(0.010)	(0.029)	(0.008)
D2.(dummy single app developer)	-0.005	-0.039	-0.015	-0.043	0.051	0.033^c	-0.070	0.034^c
	(0.020)	(0.021)	(0.028)	(0.024)	(0.035)	(0.016)	(0.042)	(0.016)
Type III instruments:								
D1.L1(age update)	-0.022^c	0.078^a	-0.044^a	0.179^a	-0.000	-0.157^a	-0.036^b	-0.252^a
	(0.009)	(0.008)	(0.011)	(0.012)	(0.014)	(0.016)	(0.014)	(0.015)
D1.L1.(number updates)	0.089^a	-0.291^a	-0.001	0.000				
	(0.016)	(0.020)	(0.001)	(0.001)				
D1.L1.(number countries)					0.580^a	0.034^b	0.557^a	0.001
					(0.024)	(0.010)	(0.031)	(0.008)
Controls:								
D1.L1.(in-app)	-0.214	-0.250	-0.082	-0.127	-0.061	0.432^a	-0.218	0.170^a
	(0.159)	(0.156)	(0.141)	(0.097)	(0.226)	(0.050)	(0.207)	(0.040)
D1.L1.(free)	0.093	0.176			1.371	0.015		
	(0.292)	(0.137)			(0.826)	(0.113)		
D1.L1.(# apps same developer)	0.042^a	-0.002	0.012	-0.002	0.032	0.004	0.026^c	-0.005
	(0.011)	(0.007)	(0.016)	(0.010)	(0.017)	(0.009)	(0.013)	(0.008)
N	3700	3700	2956	2956	3700	3700	2956	2956
R^2	0.698	0.297	0.664	0.188	0.260	0.558	0.321	0.632

Significance level: $^a p < 0.001$, $^b p < 0.01$, $^c p < 0.05$. Clustered (by developer) standard error in parenthesis. Regressions include time and category dummies. The term "own" stands for average value of the variable for the other apps distributed by the same developer.