Updates Management in Mobile Applications. iTunes vs Google Play

Stefano Comino
Dipartimento di Scienze Economiche e Statistiche
Universita di Udine, Udine (Italy)

Fabio M. Manenti
Dipartimento di Scienze Economiche ed Aziendali “M. Fanno”
Universita di Padova, Padova (Italy)

and Franco Mariuzzo
Centre for Competition Policy
School of Economics
University of East Anglia

CCP Working Paper 15-4 v3*

Abstract

This paper focuses on a specific strategy that developers of mobile applications may use to stimulate demand: the release of updates. We start with a stylised theoretical analysis to describe the developer's decision to release an update. Its predictions are then tested by using an unbalanced panel with the top 1,000 apps in iTunes and Google Play for five European countries. We show that while in iTunes updates increase the rate of growth of downloads, in Google Play their effect is not significant. We argue that the lack of quality control by Google Play can lead to an excess of updating. We also find that the past performance of the app influences the decision to release an update, but only in iTunes. This finding is in line with our theoretical analysis and can again be interpreted on the basis of the different way of governing the release of updates in the two stores.

***** Earlier versions of this paper have been published as CCP Working Papers 15-4 and 15-4v2, with the same title. This version contains a new section on apps multihoming.

Contact Details:
Franco Mariuzzo F.Mariuzzo@uea.ac.uk

The authors wish to acknowledge the contribution of the ESRC who supported their research through Centre for Competition Policy funding (ref: RES-578-28-0002).

ISSN 1745-9648

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

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

May 17, 2016

Abstract

This paper focuses on a specific strategy that developers of mobile applications may use to stimulate demand: the release of updates. We start with a stylised theoretical analysis to describe the developer's decision to release an update. Its predictions are then tested by using an unbalanced panel with the top 1,000 apps in iTunes and Google Play for five European countries. We show that while in iTunes updates increase the rate of growth of downloads, in Google Play their effect is not significant. We argue that the lack of quality control by Google Play can lead to an excess of updating. We also find that the past performance of the app influences the decision to release an update, but only in iTunes. This finding is in line with our theoretical analysis and can again be interpreted on the basis of the different way of governing the release of updates in the two stores.

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

JEL Codes: L10, L63, M31.

*We thank the seminar audiences at the 2015 Telecom ParisTech annual workshop on the Economics of Information and Communications Technologies, and at the 2016 Annual Scientific Seminar of the Florence School of Regulation. Paper presented also at the 2015 Meeting of the French Economic Association and Conference of the Centre for Competition Policy, at the 2016 conference of the Industrial Organization Society and of Royal Economic Society. The authors wish to acknowledge the financial support from Progetto di Ateneo - Università di Padova, 2012-14 and the contribution of the ESRC for funding our research assistant Martin Graffenberger from Centre for Competition Policy and further assistance from Elena Mengo (funding ref: RES-578-28-0002). Authors also thank Joeffrey Drouard, Thierry Penard, Lisa George, Toker Doganoglu 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).

1 Introduction

> 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 simple, but the execution takes patience, practice, and a plan. 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, "blog.appannie.com/updating-your-app-chad-mureta/"

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, in turn, was released only two weeks after version 6.1.0. Since November 2011, when it was launched on iTunes, Diamond Dash has been updated nearly 60 times, on average more than once per month. Diamond Dash is not an exception. The frequent release of updates is a common feature among the apps in our sample: in Google Play and iTunes, the two most popular stores, apps are updated on average every 13 days and every 58 days, respectively.

With millions of apps available in various app stores, developers face a tremendous challenge. Not only they do struggle to catch the attention of prospective users (Bresnahan et al., 2015), but they fiercely compete for usage too. Competition among developers is so harsh that app markets are said to be "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 compete 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 the "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 in the hyper-competitive app markets. Only those apps that are able to get noticed and to attract user's attention can survive and possibly thrive.[1]

Besides the high degree of competition, another relevant feature characterising app markets is the skewness of the distribution of downloads and usage. According to www.appbrain .com, on Google Play about 1.4 million apps out of 2 million have less than one thousand

[1] In the marketing jargon, the techniques aimed at increasing the transmission of products information by stimulating consumers' "word-of-mouth" is referred to as buzz marketing (see Rosen, 2009).

downloads each while, by comparison, just eighteen 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 completely disappear, it is even more compelling to adopt an effective management of updates to attract customer attention.

In this article we analyse the strategy of releasing frequent updates by exploiting the differences across the two main app stores. Interestingly, iTunes and Google Play follow different policies regarding the publication 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 not providing any lasting entertainment value to users are not published. On top of this, apps that include undocumented or hidden features inconsistent with the description of the app are rejected as well. By contrast, publication in Google Play does not go through a similar quality check; apps and updates can be published instantaneously by developers with a "simple click of the mouse". 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.[2]

These differences in quality control may have important consequences on developers' decisions to update their apps and on the effects of such decisions on performance. In our analysis, we approximate app performance with downloads; Figure 1 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 negative values of the growth rate of downloads. This is a preliminary evidence showing that updates have a different impact on app performance in the two stores.

[FIGURE 1 ABOUT HERE]

In this paper, we start presenting a very stylised theoretical framework to explain the basic intuition for the developer's decision about whether or not to update their mobile application. We show that a developer releases an update when they observe a worsening of the performance, in the attempt to revive the app. Interestingly, our analysis shows that

[2]The website www.appbrain.com estimates that nearly 13% of Google Play apps are of low quality.

a developer might also be tempted to release an update of low quality; by stimulating the buzz surrounding their app, the developer might succeed in attracting users' attention even with an update making little improvements to the software code.

We then use an unbalanced panel of the top 1,000 apps on iTunes and Google Play in five European countries to empirically test the predictions of the theoretical analysis. Consistent with the preliminary evidence of Figure 1, we find that the release of an update positively affects downloads in iTunes while it has no significant impact in Google Play. We attribute this finding to the institutional differences characterising the two stores, and in particular to a "quality-check" effect. As argued above, in Google Play there is not a strict quality control; this implies that, consistent with the theoretical framework, developers publish both high and low-quality updates so that the overall effect on downloads is not significant.[3]

Estimates for iTunes apps 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, the 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.

Our empirical analysis proceeds by presenting further evidence for the different institutional characteristics of the two stores being responsible for the discrepancy of updates on app performance. We focus on multihomed apps in an attempt to identify the "quality-check" effect. Then, for iTunes only[4], we pursue a more in-depth analysis distinguishing between major (significant changes in app functionalities) and minor updates (bug fixing and minor changes). Our empirical investigation suggests that the latter are more likely to be employed by developers as a strategic tool to counter a drop in performance. Finally, three additional sets of regressions are presented in order to check for the robustness of our results.

The rest of the paper is organised as follows. In Section 1.1 we discuss the relevant literature on mobile apps. In Section 2, we present a stylized theoretical framework that we use to derive the testable predictions. Sections 3 and 4 are devoted to presenting the data and the empirical strategy we employ in the estimations. The results of the empirical investigation are discussed in Sections 5 and 6. Section 7 concludes.

[3]A similar exercise is performed also in McIlroy et al. (2015). The authors focus on the top free apps available in Google Play and group them into two categories: "frequently updated" and "infrequently updated". Frequently updated apps are shown to have a significantly lower percentage of negative ratings when compared to apps for which developers release a lower number of upgrades.

[4]As we clarify below, the distinction between minor and major updates is available for iTunes only.

1.1 Literature review

The astonishing growth of mobile ecosystems has attracted the attention of several scholars and the literature that studies the characteristics and the functioning of app markets is already quite developed. Bresnahan et al. (2015) stress the role of mobile development platforms in lowering the cost of R&D and stimulating the creation of an enormous number of potentially very valuable products. However, the authors observe that the sheer volume of apps available for download makes the matching between consumers and products the most challenging issue developers face. With hundreds of thousands of applications, developers struggle to make their apps visible, competition for consumer attention is pervasive and takes place not only among apps performing similar functions. In these hypercompetitive markets, developers need to find ways to let their products emerge from the mass of available apps.

Ghose and Han (2014) present one of the first estimates of the demand for mobile applications. The authors quantify the consumer preferences for different app characteristics based on a structural model that combines a random coefficients nested logit demand model with the pricing equations of software developers. The analysis is based on daily information on the top-400 free apps and the top-400 paid apps in iTunes and Google Play. The authors estimate downloads by means of a calibration exercise, relating app ranking to the number of downloads. They find that demand is larger when app description is more accurate (measured in terms of description length and number of screenshots), when the app has the in-app purchase option, and when it is older (they measure both the age of the app and of the version). User reviews are found to play a significant role as their volume and positive valence stimulate downloads. A finding which is relevant to our investigation is that, in both platforms, demand is boosted by the number of previous versions of the same app.

The two authors also find that demand increases with the number of apps published by the same developer, and when the app is available both on iTunes and Google Play; these results suggest the existence of so-called "portfolio" effects due to complementarities or other forms of scope economies. The presence of portfolio effects is confirmed by Lee and Raghu (2014) who find that broadening app offerings across multiple categories is a key determinant for developers to survive in the top charts. Their analysis is based on weekly observations of the top-300 apps of the Free, Paid, and Top Grossing charts in iTunes. More closely related to our paper, they also find that continuous updates have a positive impact on apps success.

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 investigate 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 any update.

Carare (2012) investigates in detail the role of top-ranked charts in favouring the matching between users and developers, thus stimulating app demand. The work is based on the top-100 paid apps available in the US iTunes store. The author shows that the bestseller status of the top-ranked apps is a very important determinant of consumer willingness to pay, and that the effect of rank declines very steeply for the top 10 apps and becomes negligible for

apps ranked higher than 50. 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 list of top-ranked apps may have a valuable effect in terms of stimulating additional downloads.[5]

In this paper we focus on the role of updates in stimulating demand. As a matter of fact, as argued in the introduction, developers update their apps very frequently. This strategy is not only specific to mobile applications but, although with much lower frequency, it is also commonly observed in the "traditional" desktop computer software (see Greenbaum, 2005). Following Sankaranarayanan (2007), the release of a new version of "traditional" 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. A common rule in app stores is that updates must be made available free of charge to anyone who has previously downloaded the app.[6] As a consequence, developers cannot exploit their installed base of users by trying to re-sell upgrades of their software.[7]

2 The decision to update: a stylised framework

App performance is highly skewed, and a handful of mobile applications obtains the lion's share of the market. With the huge mass of apps competing for download, producing a high quality software is not enough to guarantee a brilliant performance in terms of downloads and usage. Success depends heavily on the ability of the developer to stimulate consumer's attention. The frequent release of new versions of the app (updates) is a strategy developers can use for this purpose.

A new version of the software is likely to attract the interest of bloggers, social networks and journalists of specialised magazines, thus stimulating what we call the "buzz" surrounding the app and, possibly, downloads. Similarly, updates might be useful also 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 steer up users' curiosity, thus inducing them to start using the app again.

In the following pages, we present a very stylised model useful to interpret our empirical analysis. The model studies the choice of a developer about whether or not to release a new version of her app. Following our previous discussion, key features of the model are: *i*) users' decision to download/use the app depends on the perceived quality of the software, which, in turn, is determined by both its intrinsic/technical quality and the buzz surrounding it; *ii*)

[5]The role of top-ranked charts in boosting downloads is studied also in Ifrach and Johari (2014) and Garg and Telang (2013).

[6]On this point see http://digitalmediadiet.com/?p=1292.

[7]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.

app performance is discontinuous and it can be very high or very low; and *iii*) the release of an updated version stimulates the buzz around the app.

More formally, consider a developer who has already published two (or more) apps in the store; the developer has to decide whether or not to update one of her apps (that we indicate as the "focal" app). In taking this decision the developer aims at maximising returns (downloads and usage), net of further possible costs to update the software. Let q_P denote the quality of the focal app perceived by potential users. We model perceived quality as the sum of two components: the intrinsic quality of the software, q_I, and the impact of the buzz around the app, b. Both software intrinsic quality and the buzz depend on the released version. Formally, we express perceived quality as $q_P(u) = q_I(u) + b(u)$, where $u = 0$ for the current version of the app, and $u = 1$ in the case where the developer releases an update. Users can perceive the app as of high quality if its intrinsic quality is high and/or it is surrounded by a positive buzz (i.e. good users' reviews, positive discussions on dedicated blogs, etc.).

The crucial assumption of the model is that the decision to release an update stimulates the buzz surrounding the app. This assumption is made on practical grounds considering that updates tend to stimulate discussions or comments in dedicated blogs/magazines/websites or on social networks. Clearly, the increased buzz can be either positive or negative: bloggers, journalists and regular users might positively or negatively welcome the new version of the software. In other words, the augmented buzz may improve or worsen app perceived quality. Formally, we assume that an update makes app perceived quality more uncertain.

Consistently, we assume b as the realization of a random variable, while q_I, for the sake of simplicity, is assumed to be deterministic.

According to the empirical evidence outlined in the introduction, we assume that returns are discontinuous. Formally, we assume that there exists a threshold level τ for app perceived quality such that:

$$\begin{array}{ll} \text{if } q_P \geq \tau & \text{returns are } \overline{R} = \overline{r} + \rho\overline{r} \\ \text{if } q_P < \tau & \text{returns are } \underline{R} = \underline{r} + \rho\underline{r}, \end{array}$$

where $\overline{r} > \underline{r} \geq 0$. $\overline{r}$ and $\underline{r}$ represent the returns of the focal app while $\rho\,\overline{r}$ and $\rho\,\underline{r}$ measure the impact of downloads/usage of this app on the returns of the other app distributed by the developer. In other words, $\rho\, r$ indicates the increase/decrease in the other application returns due to the performance of the focal app. The two apps can be either complements ($\rho > 0$) or substitutes ($\rho < 0$). Therefore, an increase in downloads/usage of the focal app can either contribute to stimulate returns from the other app or it can cannibalise the latter. Complementarity may emerge from a "branding effect" or from cross advertising between the two apps.[8] Substitutability, instead, may occur when the two apps address the same or similar user needs: a crowding out effect.

The current version of the focal app has perceived quality $q_P(0) = q_I(0) + b(0)$, where $b(0)$ is the realization of a random variable uniformly distributed over the segment $(B-\eta, B+\eta)$,

[8]The model only takes into account complementarity/substitutability looking at the demand side, but complementarity/substitutability can also emerge on the production side.

with density function $f(b(0)) = 1/(2\eta)$. The total expected returns generated by the current version of the focal app amount to:[9]

$$\left(\int_{B-\eta}^{\tau-q_I(0)} \frac{1}{2\eta} db(0)\right) \underline{R} + \left(\int_{\tau-q_I(0)}^{B+\eta} \frac{1}{2\eta} db(0)\right) \overline{R} = \frac{\left(\overline{R} - \underline{R}\right)\left[B - \tau + q_I(0)\right]}{2\eta} + \frac{\overline{R} + \underline{R}}{2}. \quad (1)$$

When the focal app is updated, the developer incurs the (development) cost ϕ, and the new version of the software has perceived quality $q_P(1) = q_I(1) + b(1)$, where:

- $q_I(1) = q_I(0) + \Delta$, with $\Delta \lesseqgtr 0$. In other words, the new version of the software may have a higher or smaller intrinsic quality;[10]

- $b(1)$ is the realization of a random variable uniformly distributed over the segment $(B - \eta\gamma, B + \eta\gamma)$, with $\gamma \geq 1$, according to the density function $f(b(1)) = 1/(2\eta\gamma)$. Following the above discussion, we assume that the release of the update stimulates the buzz around the app, thus increasing the uncertainty about its perceived quality ($\gamma \geq 1$).[11]

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

$$\left(\int_{B-\eta\gamma}^{\tau-q_I(0)-\Delta} \frac{1}{2\eta\gamma} db(1)\right) \underline{R} + \left(\int_{\tau-q_I(0)-\Delta}^{B+\gamma\eta} \frac{1}{2\gamma\eta} db(1)\right) \overline{R} - \phi = \frac{\left(\overline{R} - \underline{R}\right)\left(B - \tau + q_I(0) + \Delta\right)}{2\eta\gamma} + \frac{\overline{R} + \underline{R}}{2} - \phi. \quad (2)$$

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

By comparing expressions (1) and (2) one can easily see that the developer updates the focal application when the increase in total expected returns is higher than the cost of developing the update; formally, the developer chooses $u = 1$ when:

$$(\overline{R} - \underline{R})[(\tau - q_I(0) - B)(\gamma - 1) + \Delta] \geq 2\eta\gamma\phi. \quad (3)$$

[9] We focus on the most interesting case with interior solution, i.e. with events $q_P(i) \geq \tau$ and $q_P(i) < \tau$ occurring with positive probability, for $i \in \{0, 1\}$. This, in turn, imposes restrictions on the parameters as discussed in the proof of Remark 1.

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

[11] We are implicitly assuming that the buzz is not influenced by the intrinsic quality of the new version of the software ($b(1)$ does not depend on Δ). Admittedly, this is a strong assumption which, however, allows us to derive quite neatly the interesting result shown in Remark 1. More generally, a positive correlation between intrinsic quality and buzz (positive buzz is more likely when Δ is large) would make high-quality updates more profitable while reducing the benefits of releasing low-quality ones. However, also in this case, the main message of our argument would still be valid: developers might be tempted to release updates even of low quality (Δ small).

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

In other words, when the developer expects a poor performance of the current version of the app, she releases an update in the hope of stimulating positive buzz and, via this channel, returns. Clearly, this decision is profitable provided that the development cost ϕ is small compared to the increase in expected returns.

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

It is interesting to observe that the developer might find it profitable to release an update even when its intrinsic quality is not greater than that of the current version of the software; in other words, condition (3) may hold also for $\Delta \leq 0$. This occurs when the expected perceived quality of the app is very low so that the developer anticipates that the current version is very likely to obtain low returns. In this case, she has everything to gain from releasing an update thus stimulating the buzz and increasing the uncertainty about the app perceived quality. The update may turn out to have either a very low or a very high perceived quality, depending on the realisation of the buzz. 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. In other words, when the expected perceived quality of the current version is very low an update (even of lower intrinsic quality) represents a sort of "bet for resurrection" strategy that the developer might find it profitable to pursue.[12]

The following Remark formalises these arguments:

Remark 1 *When the expected perceived quality of the current version of the app is very low, the developer may decide to release an update even if it does not improve the intrinsic quality of the app,* $\Delta \leq 0$.

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

[12]This strategy is reminiscent of the effect of limited liability in firm's 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.

2.1 Testable predictions

The above theoretical analysis can be used to derive some testable predictions. As discussed above, the developer releases an update when the expected perceived quality of the current version of the app is low. It is reasonable to assume that the developer forms expectations on perceived quality by looking at the app past performance. Hence, condition (3) suggests that:

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

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

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

According to Remark 1, developers may be tempted to release an update also when the new version of the software is of the same or lower intrinsic quality. They may do so in order to try reviving a poor performing app via the effect on buzz. The remark suggests an interesting prediction related to the institutional differences between the iTunes and Google Play stores. As we argued in the introduction, while iTunes has a formal quality check for publishing apps and related updates, such a check is not implemented in Google Play. Therefore, in principle, low quality updates can be released in Google Play but they cannot in iTunes. This amounts to saying that Remark 1 is more likely to apply to Google Play apps. Because of this "quality-check" effect, we expect that updates are more likely to improve the app performance in iTunes, where only high quality new versions can be published, than in Google Play. In other words, we expect that:

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

The rest of the paper is devoted to empirically test our predictions. Unfortunately, no information about app usage or revenues is present in our data and the only available information for app performance is the number of downloads. Therefore, in the estimations, we approximate the performance of each app with its growth rate of downloads.

3 The data

The data used in this study is a combination of information obtained from the consulting analytics Priori and data downloaded from the web site AppAnnie.com. Priori provided us with monthly data on the top 1,000 most downloaded apps in iTunes and Google Play in five

European countries (Germany, France, Italy, Spain, and the UK) for the period September 2013-February 2014. According to Priori (2014), the top 1,000 apps covers about 60% of the market in each country (e.g. in October 2013 our dataset covers 55.3% of total downloads in iTunes in UK and 62.09% in Italy).

For each app, the 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 (in a scale from 1 to 5), the number of user's ratings,[13] 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 deserves to be noted that except for downloads, all the other information is not country-specific, i.e. it is the same everywhere.

With the exception of the 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's 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.[14] Only first-time installations are counted as downloads in our data. Users' upgrades of already installed applications are not counted as downloads.

For iTunes apps, we complemented Priori data with additional information taken from the App Annie web site. Among the collected information, the most relevant one is the type of update.[15] Conventionally, software developers keep track of the different versions of their products by means of a three-digit sequence, where the first digit identifies major updates and the second and third digits minor updates of decreasing significance. So, for example, the version of a given app can be 2.12.4 meaning that there have been two major updates and sixteen minor ones (12+4). It is customary to consider minor updates as new

[13]Neither user rating nor user rating count are employed in our econometric application. User rating is the average rating obtained by the app since its publication and given its cumulative nature, it varies only marginally from month to month. For this reason it is ill-suited for our estimations. User rating count, instead, is problematic since in several cases the information is updated with delay and it is not possible to ascertain the time period to which it refers to.

[14]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.

[15]For each app, we also collected information about the degree of compatibility (app for iPhone/iPod touch, for iPad only, for iPhone only or Universal), the size of the code (data collected in May 2014), and the age restriction (4+, 9+, 12+, 17+). On top of this, we re-classified apps into "corporate" apps (apps used by companies as an additional distribution channel - e.g. airlines or banking apps, newspaper or TV network apps etc. - or to provide information about their service) vs "pure" apps (applications that perform a specific task and that can be considered as a product *per se*).

versions of the software 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. For each update in iTunes during the sampling period, we distinguish major updates from minor ones; for major updates, we also recorded the date of release in the store. Finally, it deserves to be noted that, once published, the new version of the software is available worldwide.

3.1 Descriptive statistics

One of the purposes of our research is to estimate the determinants of the developers' decision to update an app; in particular, we look at how the past performance affects the decision to update (see Conjecture 1). Given that updates apply to all countries worldwide, it seems sensible to assume that the decision to release a new version of the software is based on the overall performance of the app in terms of downloads. For this reason, we aggregate the data from the five European countries; following this aggregation, monthly downloads are the sum of the downloads in the five countries in a given month. All the other variables are not country-specific and are not affected by aggregation.

[TABLE 1 ABOUT HERE]

Summary statistics for the two stores are provided in Table 1. In the top panel we display the full sample. From the original 30,000 observations per store (1,000 apps, during 6 months, in 5 countries) we are left with 15,983 observations for Google Play, and 14,755 for iTunes. This drop 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. In the bottom panel of Table 1 we display the descriptive statistics for the selected sample on which we base our empirical analysis; as shown only a small proportion (about 20 per cent) of the full sample is of use for our regressions. The reason for this further drop in sample size is that our econometric specification requires the apps to be observed in at least four subsequent periods. As we highlight below, a significant number of apps enter the top 1,000 ranking in only few months and, therefore, it cannot be employed in the econometric analysis. The sample considered in the regressions is composed of 2,956 observations for Google Play and 3,699 for iTunes.

The main characteristics of the full 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 the 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.

- During the six-month period of observation, the probability that an app is updated at least once or more in a given month is 54.2% in Google Play and 45.3% in iTunes (see the variable Update).[16]

- 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 developer distributes about 3 top-ranked apps (see the 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 the variable Number of countries).

- The number of major updates for iTunes apps is a small fraction of the overall number of updates (about 2 updates out of 10 are major).

When looking at the selected sample (bottom part of Table 1), we observe that it consists of apps older than in the full sample. Apps in the selected sample have a larger number of updates, but this is mainly due to their older age. On both platforms, the proportion of updates during the period of observation is slightly smaller than in the full sample. The number of downloads is higher for the apps in the selected sample but, in terms of growth rate of downloads, the growth rate is slightly lower. In iTunes, the number of apps distributed by the same developer is sensibly higher in the selected sample.

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., 2015).

[16]Update is a binary variable which takes value 1 if the app is updated at least once in a given month.

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. In our data, the overall number of different iTunes apps that we observe during the six-month period in the five countries is 11,017;[17] 18.49% of these apps are observed every month while a substantial percentage (about 44%) of apps appear only in one period. The level of turnover is even larger in Google Play. For the Android store, the overall number of different applications is 13,029 and the share of apps that appears only in one month in the top 1,000 list is about 50%. Finally, we pinpointed multihomed apps. We defined as multihomed apps those that we observe in at least one month and country both in iTunes and Google Play.[18] We have that 20.88% of the full sample of apps are multihoming in iTunes and 16.84% in Google Play.

4 The econometric model

For each app store separately, the unit of observation in our empirical analysis is the mobile application, indexed by j, distributed in the top 1,000 rank position in a country labelled c and in a month denoted by t. We deal with a longitudinal dataset which has a large cross-section of apps of size J, and limited number of periods and countries of size T and C, respectively. Asymptotics relies on the largest dimension, which is the apps dimension.

We study the dynamics of the rate of growth of downloads and of the decision to release an update; therefore, our econometric specification is made of two equations: the growth and the update equations. One key variable in the growth equation is *update*, denoted by $u_{jt} = \{0, 1\}$, a binary variable taking value 1 in the case where the publisher releases an update in period t. In regards to the update equation, following our theoretical analysis, the key variable is the past performance of the app, measured in terms of the rate of growth of downloads.

As argued previously, when a developer updates her app, the new version of the software is available in all countries. Therefore, it is sensible to assume the decision about whether or not to release an update to be based on the overall performance of the app. For this reason, in our analysis, we obtain for each app an aggregate measure of downloads by summing the amount of downloads the app received in each country; we then use this aggregate measure to compute the app growth rate of downloads. Formally, letting Q_{jct} be the amount of

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

[18] For the matching between the two stores we used the name of the app. This required an elaborated manual check of inconsistencies in names. For example, the Yahoo app was called "Yahoo" in one store and "Yahoo!" in the other. Thus, to identify the multihoming apps we had to homogenise the names of the apps.

downloads for app $j \in \mathcal{J}$ distributed in country $c \in \mathcal{C}$ in the period $t \in \mathcal{T}$, the rate of growth of downloads is $g_{jt} = \frac{\ln \sum_{c \in \mathcal{C}} Q_{jct}}{\ln \sum_{c \in \mathcal{C}} Q_{jc,t-1}}$; we treat the number of downloads as zero if the app does not appear in the top 1,000 ranking in country c, in the relevant period.[19]

We model the growth and the update equations as linear autoregressive distributed lag models of order 1, made of observable and unobservable app characteristics. A set of controls is also included, yielding the following system of linear equations:[20]

$$\begin{aligned} g_{jt}^k =& \phi_{11}^k g_{j,t-1}^k + \phi_{12}^k u_{jt}^k + h_{1t}^k + \mathbf{x}_{1jt}^k \boldsymbol{\beta}_1^k + \alpha_{1j}^k + \varepsilon_{1jt}^k, \quad t = 3, \cdots, 6 \\ u_{jt}^k =& \phi_{21}^k g_{j,t-1}^k + \phi_{22}^k u_{j,t-1}^k + h_{2t}^k + \mathbf{x}_{2jt}^k \boldsymbol{\beta}_2^k + \alpha_{2j}^k + \varepsilon_{2jt}^k, \quad k \in \{iTunes, GP\}. \end{aligned} \tag{4}$$

We have chosen to model the binary variable u_{jt}^k 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. For short panels, as is the case here, it is common to let the time effect be fixed and hence exclude it from the composite error term which, for each equation $l = \{1, 2\}$, is the sum of the app unobserved heterogeneity, α_{lj}^k, and the pure idiosyncratic error term, ε_{ljt}^k. We assume the idiosyncratic error to be uncorrelated both over time and apps. We account for the time effect on growth or updates with the function h_{lt}^k. The vector $\mathbf{x}_{ljt}^k$ includes a set of controls: in-app option, number of apps of the same developer, free, age. The variable update is endogenous and further endogeneity is brought in by the presence of unobserved heterogeneity.

As both the lagged dependent variable and the current value of the update variable are expected to be correlated with the app unobserved heterogeneity, we follow the dynamic linear panel model literature and take the first difference of both sides of the system of equations (4); this approach is aimed at removing inconsistencies of the parameters driven by such correlation and leads to the following system of equations in first differences:

$$\begin{aligned} \Delta g_{jt}^k =& \phi_{11}^k \Delta g_{j,t-1}^k + \phi_{12}^k \Delta u_{jt}^k + \tau_{1t}^k + \Delta \mathbf{x}_{1jt}^k \boldsymbol{\beta}_1^k + \Delta \varepsilon_{1jt}^k, \quad t = 4, 5, 6 \\ \Delta u_{jt}^k =& \phi_{21}^k \Delta g_{j,t-1}^k + \phi_{22}^k \Delta u_{j,t-1}^k + \tau_{2t}^k + \Delta \mathbf{x}_{2jt}^k \boldsymbol{\beta}_2^k + \Delta \varepsilon_{2jt}^k, \end{aligned} \tag{5}$$

where τ_{lt}^k denotes the first difference of the original function of the time effect, h_{lt}^k. The first difference of each equation is nothing but an econometric transformation of the original equations, aimed at estimating unbiased original parameters. Hence, the interpretation of the coefficients is still based on the original equations (those not in first difference).

Though in the first difference structural system of equations (5) we have eliminated the unobserved heterogeneity and the aforementioned issue of correlation, still the system cannot

[19] As robustness check, instead of assuming zero downloads when the app does not appear in the top 1,000 ranking, we also 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 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 shown in Section 5.

[20] Below, we clarify how we deal with the possible simultaneity between the rate of growth of downloads and the change in versions.

be estimated consistently by OLS. We also deal with other sources of correlation. In the first equation we cope with the correlation between $g^k_{j,t-1}$ and $\varepsilon^k_{1j,t-1}$, in addition to that between the update variable and download growth, which causes $E(\Delta u^k_{jt}, \Delta\varepsilon^k_{1jt}) \neq 0$. Similarly, in the second equation we tackle the endogeneity between $\Delta\varepsilon^k_{2jt}$ and $\Delta u^k_{j,t-1}$, in addition to the endogeneity between $\Delta\varepsilon^k_{2jt}$ and $\Delta g^k_{j,t-1}$.

In the next subsection we discuss the instruments that we use to correct for such multifaceted endogeneity. For notational convenience in what follows we omit reporting superscript k.

4.1 Instruments

We need to identify the coefficients associated to the following endogenous variables: the first differences of the lagged growth rate of downloads, $\Delta g_{j,t-1}$, which enters both equations, and contemporaneous and lagged updates, Δu_{jt} and $\Delta u_{j,t-1}$ that appear both in the growth and the update equation. We select a set of instruments that are believed to be *strong* and *valid*. We exploit the time series dimension of the panel and the fact that a sizeable number of developers have several apps listed in the top 1,000 ranking (see variable # of apps same developer in Table 1).

Below we describe the most relevant instruments. The complete list of instruments is relegated, along with applicable tests on the validity and strength, to a footnote of the results tables.

- We instrument lagged growth with a two-period lagged growth, as suggested in Anderson and Hsiao (1981), and with average lagged growth of other apps produced by the same developer. Multi-app developers share common underlying resources that, either via substitutability or complementarity, determine correlated growth trajectories for their apps portfolio. We define the average growth of non j^{th} apps distributed by developer f as $\overline{g}_{-jt} = \frac{\sum_{l\in(\mathcal{J}_{f,-j,t})} g_{lt}}{J_{ft}-1}$, where $\mathcal{J}_{f,-j,t}$ is the set of apps, excluding app j, distributed by developer f and that appear in the top 1,000 ranking in at least one of the five countries in our sample in period t; J_{ft} is the number of apps distributed by developer f in the top 1,000 in period t, including app j. In the case where the developer has only app j in the top 1,000 in period t, $J_{ft} = 1$, the variable takes value zero. Similarly, the variable is set to zero in the first period. This works as a source of normalisation, but has the advantage of not dropping additional observations in the instrumentation, while not compromising the results, as confirmed by the statistical tests on our instruments. To balance the replacement of a missing value with a 0 for single-app developers, we add as additional instrument a dummy variable that takes value one if the developer has only one app in the top ranking in the period, and zero otherwise. To be precise, we use as instrument for $\Delta g_{j,t-1}$ the second difference of the first lag of the constructed variable and work with the assumptions

$$\begin{aligned} &Cov\left[\left(\overline{g}_{-j,t-1} - 2\overline{g}_{-j,t-2} + \overline{g}_{-j,t-3}\right), \left(g_{-j,t-1} - g_{-j,t-2}\right)\right] \neq 0, \\ &Cov\left[\left(\overline{g}_{-j,t-1} - 2\overline{g}_{-j,t-2} + \overline{g}_{-j,t-3}\right), \left(\varepsilon_{kjt} - \varepsilon_{kj,t-1}\right)\right] \approx 0, \quad t = 4,\ 5,\ 6 \end{aligned} \tag{6}$$

where t starts from period 4 rather than 5 due to the aforementioned normalisation. Because of possible correlation between $\overline{g}_{-j,t-1}$ and $\varepsilon_{kj,t-1}$ one may question the validity of the second equation in (6). In the results section we will provide statistical evidence of its validity. If the suggested instrument were to be problematic, i.e. not valid, we would have replaced it with the first difference of the second lag of $\overline{g}_{-jt}$. The choice between these two alternative instruments is a trade-off between validity and strength, and one can play around with them, depending on the contingent situation.

- The instruments chosen to deal with the endogeneity of updates, both in their contemporaneous and lagged value, follow the logic that we have just detailed for the endogeneity of lagged growth. A first instrument suitable to correct for the biaseness caused by the endogeneity of both time variants of the update variable is the two-period lagged value of the update, $u_{j,t-2}$. Furthermore, we instrument Δu_{jt} with the second difference of the average update of the apps belonging to $\mathcal{J}_{f,-j,t}$, $\overline{u}_{-j,t}$, and, similarly, we instrument $\Delta u_{j,t-1}$ with the lag of the second difference of the average update. The formula for this generated variable is the logical variant of the one provided for the growth variable, so we omit discussing it further. Also in this particular case, controlling for complementarity or substitutability (which we do by using the number of successful apps listed by the same developer) buffers the correlation with updates for other apps produced by the same developer.

As stated earlier, we document the full list of instruments in the footnote of the results table. In all our estimates of each of the two equations we use the same set of instruments, unless the test for validity fails, in which case we omit the problematic instrument(s). In the next section we discuss the main results.

5 Results

Table 2 shows the estimates of system (5) for iTunes and Google Play, respectively.[21] Estimations are in first differences, hence time invariant variables are not identified.[22] At the bottom of the table we display the results of various tests of hypotheses related to strength and validity of the selected instruments. The list of the chosen instruments is given in a note to the table. The F-tests, as well as the under identification and the weak identification (also based on an F-statistic) tests get the unambiguous message across that instruments

[21] Table 2 presents the second stage of the instrumental variables estimation. First stage estimations for columns (1), (2), (4) and (5) are in the Appendix.

[22] In a very limited number of cases, apps change status from free to paid (and viceversa) or from having to not having the in-app purchase option (and viceversa). Therefore, we have included *Inapp* and *Free* among the regressors as control variables. Nonetheless, we are aware that for both of them changes in status might be endogenous; still, we do not instrument them. as otherwise we would have to deal with too many endogenous variables and we would loose focus. For this reason one needs to be cautious in the interpretation of the estimated coefficients of these two regressors.

are strong. Also, instruments are valid, as confirmed by the J over identification test, where only for the growth equation for Google Play they are mildly non valid.

[TABLE 2 ABOUT HERE]

Columns (1) and (2) report the estimates of the growth equation for the two stores and show that:

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

2. Consistent with Conjecture 3, we find that the release of an update has different effects on the two platforms. In iTunes the growth rate of downloads is positively affected by the variable *Update*; quantitatively we find that having released an update increases the rate of growth of downloads by about 30%. This evidence is in line with previous results (see Ghose and Han, 2014; Lee and Raghu, 2014). On the contrary, updating an app on Google Play does not have a significant impact on downloads. This finding is in contrast with what shown in Ghose and Han (2014). Nonetheless we can interpret it on the basis of our theoretical analysis. Remark 1 suggests that in order to stimulate the buzz surrounding the app, developers might also be tempted to release low quality updates; this strategy is more likely to be implemented on Google Play where developers are not subject to any screening concerning the quality of their applications. From these considerations the non significance of the *Update* coefficient for Google Play might be due to the fact that developers release both high and low quality updates which, on average, do not significantly impact on downloads. On the contrary, the strict control on app quality implemented by Apple ensures that updates are of high quality, thus explaining the positive impact of the variable *Update*. Therefore, we attribute the different consequences of updating in the two stores to what we define as the "quality-check" effect.

3. Both in iTunes as well as in Google Play, the growth rate of downloads increases with the number of other applications by the same developer listed in the top 1,000 during the same month. This result is reminiscent of the "portfolio effect" highlighted by Lee and Raghu (2014) and can be due to the presence of complementarities among apps that developers might exploit through, for instance, a "branding effect" or cross-advertising.

4. In both stores, downloads are negatively affected by the presence of the in-app purchase option; this result, which contrasts what shown in Ghose and Han (2014), can be interpreted as the classical negative effect of price on demand. This interpretation is

also suggested by the fact that the vast majority of the apps in our sample are free and the in-app purchase option represents the only form of pricing.

5. In iTunes, free apps are characterised by a larger growth rate of downloads.[23]

Columns (4) and (5) show the estimates of the update equation. Our theoretical analysis suggests that an important determinant of the developers' decision to release a new version of an app is the past performance, here measured in terms of lagged growth rate of downloads, g_{t-1} (Conjecture 1).

This prediction is confirmed in the case of iTunes. For this store, we find that the app past performance affects the decision to release an update: the coefficient of the variable g_{t-1} is negative and statistically significant, meaning that, other things equal, a poor performance of the app in the previous period makes it more likely for the developer to release an update. This amounts to saying that developers use updates in reaction to downturns in downloads. The same does not occur when considering Google Play, where past performance does not have an impact on the decision to release an update. This difference between the two stores can, again, be interpreted on the basis of the institutional characteristics of iTunes and Google Play. As mentioned, iTunes apps must pass a quite severe quality control and this limits the freedom of developers to publish updates. As a consequence, developers who have written an update, and are ready to publish it in the store, may decide to delay publication until when it is really necessary, i.e. when the performance of the app worsens and triggers a response to counter the drop in downloads. In contrast, on 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 where they later need to counter a drop in downloads, they can further distribute another update of any quality. In this environment, updates are continuously published, thus diluting the impact of past performance on the decision to release a new version of the app.

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

1. On Google Play, developers who updated their apps in the previous period are more likely to update them again. This suggests a form of persistency in the decision to release new versions of the apps and confirms our argument according to which Google Play developers keep on updating their apps continuously with a very high frequency.

2. On both platforms, apps with in-app purchases are more likely to be updated. Developers of these apps have more to gain from increasing user engagement and, via this channel, revenues from in-app purchases.[24]

[23] As regards Google Play, the dummy variable *Free* is omitted, provided that nearly all apps in the sample are free of charge.

[24] As mentioned above, the interpretation of the coefficient of the variable *Inapp* must be taken with caution, due to potential problems of endogeneity. On top of that, when a developer introduces the in-app purchases option, she often adds new features to the app which, in turn, require the release of an update; this fact might contribute explaining the large and significant coefficient of the variable *In-app*.

3. The number of other apps by the same developer listed in the top 1,000 during the same month moderately increases the likelihood to update apps on iTunes while it is not statistically significant on Google Play. This confirms Conjecture 2 but only for iTunes apps. As discussed above, estimates of the growth equation of system (5) suggest the presence of complementarities among apps, both on iTunes and on Google Play; according to Conjecture 2, this implies that the probability of updating an app should increase with the number of other apps from the same developer.

5.1 Focusing on iTunes: Major *vs* minor updates

As discussed in Section 3, for iTunes apps, we collected additional information about the type of updates by distinguishing major updates (i.e. significant changes in functionalities) from minor ones (i.e. bug fixing and minor changes to the software code). This allows us to investigate more closely the developers' strategy about updates. We have re-estimated the two equations of system (5) by considering major updates and minor updates separately. The results are displayed in columns (3) and (6-7) of Table 2.

Estimates of the growth equation confirm the results obtained in the general estimation: app updates (both major and minor) positively impact the growth rate of downloads. Even though with a lower statistical significance, the coefficient of the variable *Major update* is slightly larger than that of *Minor update*; major updates imply a more marked improvement in software quality and this translates into a stronger impact on downloads.

The main result obtained in the general estimation of the update equation is confirmed for minor but not for major updates; only for bug fixing and minor changes in the software code, the past performance impacts the decision to update that app (column 6). This finding reveals that the strategic use of updates described by the theoretical analysis is particularly suited to minor updates. This result can be interpreted if one considers the different nature of major and minor updates; while minor updates can be developed with greater ease, major ones require much more development effort and time. It seems therefore that only minor updates are used strategically in reaction to poor performances. In contrast, major updates are inherently less suitable for this purpose.

5.2 Identifying the quality-check effect

In Table 2 we showed that the release of an update has a very different impact on the growth rate of downloads in the two stores. We interpreted the non significance of the variable *Update* in Google Play as evidence of the fact that, absent any quality check, developers release both high and low quality updates which, on average, do not impact on downloads. In principle, there is an alternative explanation for the diverging results in the estimations for the two stores which is related to a potential selection effect: developers operating on iTunes might be systematically different from those on Google Play thus determining a differentiated effect of updates on downloads. For example, highly skilled developers may prefer to operate in iTunes rather than in Google Play. If this is the case, the heterogeneity

in the results we obtain for the two stores might be partially or entirely due to developers' selection rather than to the effect of the iTunes quality check.

In order to show the relevance of the "quality check" effect, in this section, we estimate the growth equation of Google Play restricting the attention to the sample of the multihomed apps, that is apps that are present in both stores. For these apps, the developer is obviously the same, the selection problem is not an issue and this allows us to isolate the quality check effect.

For multihomed apps we can try to assess the quality of updates in Google Play by exploiting the information about whether or not the app was updated also in iTunes. It reasonable to assume that if a developer simultaneously releases a new version of her software in the two stores, the two upgrades are alike, i.e. of similar quality. As the release of updates in iTunes goes through a strict screening, this amounts 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, it is "not quality-checked". Column (2) of Table 3 shows the estimates of the growth equation of Google Play multihomed apps where we distinguish between quality-checked ($UpdateBoth$) and non quality-checked updates ($UpdateOwn$). Quality-checked updates have a positive and significant effect on downloads while non quality-checked ones do not impact on the app performance. This result shows that when we neutralise the developers's selection, the quality-check effect still plays a crucial role.

[TABLE 3 ABOUT HERE]

As a robustness exercise, in column (1) of Table 3 we provide the same estimation for iTunes multihomed apps. Clearly, in iTunes all updates are quality checked and therefore the distinction between $UpdateBoth$ and $UpdateOwn$ simply reveals whether an update takes place on both platforms or in iTunes only. The estimates confirm that updates have a positive and significant impact on downloads; interestingly, the coefficients of $UpdateBoth$ and $UpdateOwn$ are almost identical in magnitude, thus reinforcing our argument about the role of the quality check in iTunes.

6 Robustness

We devote this section to present three additional sets of regressions that we estimate to check the robustness of our results. In the former two, we re-estimate the update equation. In the first set of regressions we restrict our analysis to local apps, namely apps whose downloads are highly concentrated in one of the five countries of the sample; the second set of regression is intended to check the robustness of the rate growth of downloads as a measure of app performance. In the final estimates, we focus on the growth equation and we conduct a country by country analysis.

6.1 Update equation: local apps

In testing Conjecture 1, we have considered the past performance of the app in the five countries included in our sample (measured in terms of the growth rate during the previous month); hence, we have implicitly assumed that those five are the reference countries for the developers' decisions. However, it might be the case that developers base their strategies by looking at the performance in a wider set of countries or in a set of countries different from that of our sample. As a matter of fact, when published in the store an update becomes available worldwide; hence, the decision to release a new version of the software might be based on the worldwide performance of the app. Alternatively, a developer might decide to update the app on the basis of the growth rate experienced in a set of countries different from our ones (e.g. US-based developers might make their decisions by looking at the performance in the US). In both cases, by considering France, Germany, Italy, Spain and the UK we would not control for the right set of countries used in determining whether or not to make an update.

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

The results of this estimation are given in Table 4 and largely confirm our main findings. The past performance of the app does not affect the decision to release an update on Google Play, while it does on iTunes, even though with a smaller magnitude and statistical significance than in the general estimation. Again, for iTunes apps, g_{t-1} does not have any statistically significant effect on the release of major updates, while it negatively affects the decision to publish the minor ones.

[TABLE 4 ABOUT HERE]

6.2 Update equation: measuring app performance

Our theoretical analysis suggests that a developer releases an update to counter a poor app performance. In the estimations we approximate the performance of an app with its growth rate of downloads. Nonetheless, another important dimension affecting performance is app usage. If a developer cares more about usage than downloads, then she might update her app following a decline in the former more than in the latter. If this were the case, our estimations would not capture the correct determinants of updates.

Unfortunately, our dataset does not provide any information about app use. In an attempt to test the validity of the results shown in Table 2, we re-run separate estimations

of the update equations focussing on sub-categories of apps for which usage is likely to be crucial for performance. In particular, we run separate regressions distinguishing between apps with/without the in-app purchase option and between free/paid apps (this latter distinction in iTunes 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.

In Table 5 we show the estimated coefficient of the lagged growth rate of downloads for these sub-categories of apps. In iTunes, g_{t-1} negatively and significantly impact on the developer's decision to update both in the sample of apps 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. In any case, the fact that for these apps g_{t-1} 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 growth rate of downloads is smaller but still negative and significant.

As far as Google Play apps are concerned, independently on the presence of the in-app purchase option, past downloads do not impact on the decision to release an update. Also this result suggests that the growth rate of downloads is an appropriate proxy for measuring app performance.

[TABLE 5 ABOUT HERE]

6.3 The growth equation: a country by country estimation

Estimations have been obtained by aggregating downloads in the five countries of our sample. This approach is fully appropriate for the estimation of the update equation, given that the decision to release a new version of the software is based on the overall performance of the app. Conversely, nothing prevents us from estimating the growth equation, where the impact of updates on downloads is investigated, on a country by country basis. The aim of the country by country analysis we present in this section is twofold. On the one side, this exercise allows us to check the robustness of our main findings at a country level. On the other side, and more relevantly, we are able to address a sample selection issue affecting our estimations. As discussed above, the econometric specification we employ requires the apps to be observed in at least four subsequent months; this implies that estimations are based on the sample of the very successful apps, i.e. those that persistently appear in the top 1,000.

We conduct two series of country by country estimations (see Table 6). In the first series, we simply re-estimate the growth equation country by country using our dataset (we refer to it as the "original" dataset). In the attempt to mitigate the sample selection due to our econometric specification, in the second set of estimations we are able to enlarge the sample

including several apps that are observed less than four consecutive periods in a given country (we refer to it as the "augmented" dataset). A simple example is useful to illustrate how we created the augmented dataset. Consider iTunes apps, and suppose that a given application appears in the top 1,000 only in Germany and in Italy; on top of this, in Germany the app is in the top 1,000 only in September, while in Italy it is among the top 1,000 all through the six-month period. This implies that when running the country by country regressions of the growth equation, the app will be included in the estimations for Italy but not in those for Germany, as in this latter country the app is not present in at least four consecutive periods. Nonetheless, for this app we have all the non-country specific information (that is, we know everything about it except downloads); the idea of the augmented sample is to exploit the non-country specific information that we have for Italy in order to include this app also in the regression for Germany. More specifically, in the augmented sample for Germany, the app is included for all the periods by using the non-country specific information obtained from Italy and by simulating downloads when needed (namely, for all periods except September, when actual downloads in Germany are observed). The simulated values are random numbers drawn from a uniform distribution over the interval 0 to the number of downloads obtained by the app ranked 1,000 in Germany during the relevant month. For both platforms, we apply this procedure to all countries; an app is included in the augmented sample of the country whenever it is listed in the top 1,000 in at least one month in that country.

[TABLE 6 ABOUT HERE]

With this procedure we are able to increase the sample size substantially; for instance, in the case of Germany, observations increase by nearly 50%, from 1,519 to 2,217. In order to reduce the simulation bias induced by the creation of fictitious downloads, we re-run the procedure and the estimations a hundred times. The values shown in Table 6 are the sample means of these estimations.

By comparing the estimates of the original dataset with those of the augmented sample, we are able to say something about the sample selection affecting our estimations. In the augmented sample the bias due to the selection of the apps is less severe as it includes not only the apps which are listed for at least four consecutive periods but also those which were originally listed for a fewer number of periods. As shown in Table 6, the results based on the original dataset and those on the augmented sample are comparable both in terms of sign and of statistical significance of the coefficient of the variable *Update*;[25] these results are also in line with what is shown in Table 2: on Google Play, updates do not have an impact on the performance of the apps, while they increase downloads on iTunes. This evidence suggests that, though numerically important, the sample selection characterising our regressions does not appear to alter the estimates dramatically.

[25]The table reports only the coefficients of *Update*, the most important variable for the growth equation.

7 Conclusions

App developers release new versions of their mobile applications with an extremely high frequency. In this paper, we focus on app updates and present a very 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 has shown that updates play a quite different role in stimulating downloads on iTunes and Google Play: while in the former updates boost downloads, their effect is not significant in the latter. These results are consistent with the prediction of our theoretical analysis which suggests that the lack of a quality check on Google Play can cause too frequent updating: developers release both high and low quality updates, which, on average, do not impact downloads. In the attempt to identify better the role of the update quality control in the two stores, we run additional estimations by restricting the analysis to the sample of multihomed apps; the outcomes of these estimations confirm the relevance of the "quality-check" effect.

In addition, and still in line with the predictions of our theoretical analysis, we have found that on iTunes developers are more likely to release an update when their apps experience a decline in downloads. On the contrary, on Google Play, the past performance of the app has no impact on the decision to release a new version of the software. Again, we interpret this difference between the two stores on the basis of the different way of regulating the release of updates. Moreover, we have also found that there is a certain degree of persistence in updating and that apps with the in-app purchase option are more frequently upgraded.

In the paper we also have investigated more in details the decision to upgrade the app in iTunes by distinguishing between major and minor updates. While we have found that both types of updates positively impact on the growth rate of downloads, the worsening of the app performance increases the likelihood of releasing minor updates but it has no impact on the decision to publish major ones. We interpret this last result as evidence of the fact that minor updates are better suited to be used as a strategic tool to counter poor past performances.

In our estimates we measure app performance with the rate of growth of downloads. Unfortunately, we are not able to control for another important determinant of performance, namely app usage. We leave this interesting extension to future research.

Appendix A

In this section we examine the first stage of the 2SLS regression. We limit the analysis to the estimates of growth and updates reported in columns (1-2) and (4-5) of Table 2. As discussed at great length in the article, lagged growth and update are endogenous in the first equation of the system (4). Similarly, lagged update and lagged growth are endogenous in the second equation of the system. We present the estimates in Table 7. Most of the variables appear in first differences to remove the unobserved heterogeneity in the original equations. In a couple of cases the variables are presented in second differences, indicated with Δ^2. Along with the variables treated as exogenous in the second stage of the regression, we have included the second difference of within-developer average update ($\Delta^2 Ownupdate$) and the first difference of within-developer average lagged growth ($\Delta Own g_{t-1}$), and the two-period lag of the endogenous variables (g_{t-2} and $Update_{t-2}$). Furthermore, we control for the second difference of single app developers binary variable ($\Delta^2 Singleapp$).

As expected, two-period lagged growth is highly correlated with the first difference of growth, and so is the first difference of past age of the version ($\Delta Agever_{t-1}$). These two variables are statistically significant both for iTunes and Google Play, and largely contribute to the strength of the instruments, as documented in the bottom panel of Table 2. Turning the attention to the update equation, in addition to the first difference of the lagged age of the version and the two-period lagged update variable, also highly correlated with the update variable in both stores are: $\Delta^2 Ownupdate$ and $\Delta In\text{-}app$, the first difference of in-app purchase. Finally, there are correlations that are store-specific, which we refer to the table.

[TABLE 7 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.

Bresnahan, T., Davis, J., and Yin, P.-L. (2015). *Economic Value Creation in Mobile Applications*. in A. Jaffe and B. Jones eds: The Changing Frontier: Rethinking Science and Innovation Policy, Chicago Scholarship Online.

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

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

Garg, R. and Telang, R. (2013). Inferring App Demand From Publicly Available Data. *MIS 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/.

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.

McIlroy, S., Ali, N., and Hassan, A. E. (2015). Fresh Apps: an Empirical Study of Frequently-Updated Mobile Apps in the Google Play Store. *Empirical Software Engineering*, pages 1–25.

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

Rosen, E. (2009). *The Anatomy of Buzz Revisited: Real-life Lessons in Word-of-Mouth Marketing*. Crown Business.

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.

Table 1: Summary statistics from the five countries

	Google Play			iTunes		
	N.	mean	std. dev.	N.	mean	std. dev.
			Full sample			
			Priori data			
Free	15,983	0.999	0.033	14,755	0.917	0.276
Price (if free=0)	17	2.447	1.148	1,229	2.759	2.907
In-app	15,983	0.297	0.457	14,755	0.561	0.496
Local	15,983	0.374	0.484	14,755	0.349	0.477
Age (in months)	15,983	15.036	13.993	14,755	19.341	15.991
Number of updates (all time)	15,983	33.430	78.487	14,755	10.079	9.990
Update (sampling period)	7,991*	0.542	0.498	8,531	0.453	0.498
# apps same developer	15,983	2.983	4.583	14,755	3.586	5.163
Number countries	15,983	1.876	1.395	14,755	2.030	1.491
Monthly downloads	15,983	249,803	1,244,139	14,755	58,156	166,472
Growth downloads**	8,225*	-0.026	0.763	8,591	-0.160	0.646
			App Annie data			
Age major version (in months)				13,723	10.536	9.924
Number major versions				13,852	2.055	2.004
Update major version				8,175	0.037	0.189
Size				13,851	63.934	149.595
			Selected sample			
			Priori data			
Free	2,956	0.999	0.032	3,699	0.958	0.202
Price (if free=0)	3	3.873	0.203	157	3.160	2.603
In-app	2,956	0.349	0.477	3,699	0.610	0.488
Local	2,956	0.365	0.481	3,699	0.335	0.472
Age (in months)	2,956	20.664	13.849	3,699	23.840	15.524
Number of updates (all time)	2,956	54.908	91.956	3,699	13.400	10.837
Update (sampling period)	2,956	0.507	0.500	3,699	0.411	0.492
# apps same developer	2,956	3.982	6.072	3,699	4.141	5.897
Number countries	2,956	2.398	1.667	3,699	2.380	1.632
Monthly downloads	2,956	476,337	2,189,596	3,699	69,551	157,152
Growth downloads**	2,956	-0.070	0.688	3,660	-0.170	0.527
			App Annie data			
Age major version (in months)				3,577	12.595	9.777
Number major versions				3,596	2.281	2.192
Update major version				3,596	0.035	0.184
Size				3,596	64.183	149.023

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

Table 2: First difference growth and update equations[†]

	Growth equation g_t			Update equation u_t			
	iTunes	GP	iTunes maj-min	iTunes	GP	iTunes major	iTunes minor
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Lag growth of downloads (g_{t-1})	-0.130^a	-0.163^a	-0.129^a	-0.084^a	0.001	-0.010	-0.078^a
	(0.021)	(0.026)	(0.021)	(0.025)	(0.014)	(0.007)	(0.025)
Update (u_t)[††]	0.300^a	0.125		0.029	0.245^a		
	(0.045)	(0.081)		(0.030)	(0.038)		
Major update (u_{1t})[††]			0.308^c			0.048^b	-0.365^a
			(0.179)			(0.022)	(0.075)
Minor update (u_{2t})[††]			0.284^a			-0.004	0.097^a
			(0.046)			(0.010)	(0.032)
In-app	-0.353^b	-0.367^b	-0.333^c	0.710^a	0.310^a	0.241^b	0.494^a
	(0.175)	(0.176)	(0.183)	(0.096)	(0.129)	(0.111)	(0.159)
Free	1.645^a		1.803^a	-0.133^c		-0.017	-0.145^b
	(0.393)		(0.443)	(0.072)		(0.018)	(0.073)
Number of other apps of the developer	0.050^a	0.062^a	0.055^a	0.018^c	-0.005	0.001	0.025^b
	(0.020)	(0.012)	(0.020)	(0.011)	(0.009)	(0.005)	(0.012)
Lag age version				0.148^a	0.251^a		
				(0.014)	(0.017)		
Lag age minor version						-0.004	0.184^a
						(0.003)	(0.016)
Lag age major version						0.045^a	-0.027^a
						(0.005)	(0.005)
Tests of hypotheses							
F-stat lagged growth (g_{t-1})	715.67^a	498.70^a	474.54^a	209.87^a	728.70^a	107.67^a	128.26^a
F-stat Update (u_t)[††]	122.89^a	58.68^a		563.33^a	435.11^a		
F-stat Major update (u_{1t})[††]			16.61^a			448.23^a	537.21^a
F-stat Minor update (u_{2t})[††]			80.73^a			293.68^a	346.54^a
Under identif. χ^2-stat	296.04^a	180.84^a	275.66^a	111.74^a	293.03^a	110.03^a	110.17^a
Weak identif. F-stat	123.13^a	114.18^a	77.80^a	194.26^a	431.35^a	95.30^a	114.34^a
Over identif. J-test	4.360	9.806^b	6.018	0.339	4.840	2.151	0.566
Observations	3,699	2,956	3,567	3,699	2,956	3,567	3,567
Uncentered R-squared	0.187	0.287	0.191	0.139	0.058	0.264	0.114

[†]Notes: all regressions are in first difference and include period dummy variables. Standard errors are in parentheses and clustered by app. Superscripts a, b, c indicate parameters or tests which are significant at 1%, 5% and 10%, respectively.

[††]The variable in the update equation is lagged.

Instruments: For other apps by the same developer, second difference of average: lagged growth (columns 1-4, 6-7), growth (column 5), update (columns 1-2), lagged update (column 5), update minor (column 3), update major (column 3), lagged update minor (column 6), lagged update major (columns 6-7). Two-period lags of: growth (columns 1-3, 5), update (columns 1-2, 4-5), update minor (columns 3, 6-7), update major (columns 3, 6-7). First difference of: lagged age version (columns 1-2), lagged age minor version (column 3), lagged age major version (column 3), lagged number of versions (columns 1-3), number of countries the app is distributed (column 5), lagged number of countries the app is distributed (columns 4, 6-7).

Table 3: Growth equation for multihomed apps[†]

	iTunes (1)	GP (2)
Lag growth of downloads (g_{t-1})	-0.132^a (0.039)	-0.167^a (0.037)
UpdateBoth	0.255^a (0.114)	0.455^a (0.126)
UpdateOwn	0.279^a (0.104)	-0.048 (0.124)
In-app	-0.473 (0.513)	-0.691^a (0.333)
Number of other apps of the developer	0.037 (0.022)	0.071^a (0.024)
Tests of hypotheses		
F-stat lagged growth (g_{t-1})	122.11^a	92.48^a
F-stat UpdateBoth	13.91^a	12.71^a
F-stat UpdateOwn	22.64^a	4.36^a
Under identif. χ^2-stat	85.765^a	60.058^a
Weak identif. F-stat	7.944^a	6.895^a
Over identif. J-test	4.937	11.852
Observations	1,443	1,103
Uncentered R-squared	0.171	0.268

[†]Notes: all regressions are in first difference and include period dummy variables. Standard errors are in parentheses and clustered by app. Superscripts a, b, c indicate parameters or tests which are significant at 1%, 5% and 10%, respectively.
Instruments: See list of instruments in footnote Table 2.

Table 4: First difference update equation for local apps[†]

	Update equation u_t			
	iTunes	GP	iTunes major	iTunes minor
	(1)	(2)	(3)	(4)
Lag growth (g_{t-1})	-0.065^b	-0.024	0.008	-0.073^b
	(0.032)	(0.035)	(0.011)	(0.031)
Update (u_{t-1})	0.079	0.247^a		
	(0.051)	(0.056)		
Major update ($u_{1,t-1}$)			0.027	-0.613^a
			(0.044)	(0.130)
Minor update ($u_{2,t-1}$)			0.001	0.205^a
			(0.015)	(0.051)
In-app	0.836^a	0.451	0.191	0.547^b
	(0.096)	(0.278)	(0.136)	(0.269)
Number of other apps of the developer	0.014	-0.023	-0.001	0.045
	(0.030)	(0.020)	(0.010)	(0.029)
Lag age version	0.127^a	0.226^a		
	(0.016)	(0.026)		
Lag age minor version			-0.005	0.181^a
			(0.005)	(0.021)
Lag age major version			0.038^a	-0.032^a
			(0.005)	(0.007)
Tests of hypotheses				
F-stat lagged growth (g_{t-1})	305.73^a	98.60^a	239.82^a	176.54^a
F-stat Update (u_t)[††]	198.18^a	164.06^a		
F-stat Major update (u_{1t})[††]			242.24^a	192.41^a
F-stat Minor update (u_{2t})[††]			159.60^a	116.72^a
Under identif. χ^2-stat	177.11^a	143.92^a	190.65^a	196.59^a
Weak identif. F-stat	182.58^a	163.02^a	148.95^a	108.82^a
Over identif. J-test	3.389	0.379	3.680	2.564
Observations	1,239	1,078	1,198	1,198
Uncentered R-squared	0.100	0.043	0.255	0.047

[†]Notes: all regressions are in first difference and include period dummy variables. Standard errors are in parentheses and clustered by app. Superscripts a, b, c indicate parameters or tests which are significant at 1%, 5% and 10%, respectively.
[††]The variable in the update equation is lagged.
Instruments: Instruments: See list of instruments in footnote Table 2.

Table 5: Robustness check on app usage

Update equation: g_{t-1} coefficient

	iTunes	GP
In-App=1	-0.056^b	-0.021
	(0.028)	(0.019)
	[2,193]	[943]
In-App=0	- 0.132^a	0.009
	(0.046)	(0.018)
	[1,426]	[1,898]
Free=1	-0.078^a	
	(0.026)	
	[3,527]	
Free=0	-0.109^b	
	(0.047)	
	[149]	

†Regressions are in first difference and include all regressors, but *free* and *in-app*, and instruments as in Table 2. Standard errors are in parentheses. Superscripts a, b, c indicate parameters or tests which are significant at 1%, 5% and 10%, respectively.

Table 6: Country by country estimates

iTunes	Ger	Spa	Country Fra	UK	Ita
			original dataset		
# of obs.	1,519	1,389	1,607	1,572	1,515
Update (u_t)	0.302^a	0.287^a	0.255^a	0.105^b	0.298^a
	(0.067)	(0.052)	(0.053)	(0.050)	(0.061)
			augmented dataset		
# of obs.	2,217	2,121	2,195	2,131	2,160
Update (u_t)	0.305^a	0.282^a	0.354^a	0.293^a	0.327^a
	(0.088)	(0.085)	(0.084)	(0.090)	(0.084)
Google Play	Ger	Spa	Country Fra	UK	Ita
			original dataset		
# of obs.	1,303	968	1,208	1,184	1,179
Update (u_t)	-0.094	-0.080	0.173	-0.061	0.172
	(0.092)	(0.107)	(0.110)	(0.093)	(0.139)
			augmented dataset		
# of obs.	1,803	1,688	1,912	1,674	1,790
Update (u_t)	-0.071	-0.212	0.139	0.065	0.139
	(0.125)	(0.184)	(0.169)	(0.153)	(0.154)

[†]Regressions are in first difference and include the same regressors and instruments as in 2. Standard errors are in parentheses. Superscripts a, b, c indicate parameters or tests which are significant at 1%, 5% and 10%, respectively.

Table 7: First stage regressions

	iTunes		GP	
	Δg_{t-1}	Δu	Δg_{t-1}	Δu
ΔIn-app	0.013	0.595^{a}	0.264^{c}	0.330^{b}
	(0.100)	(0.091)	(0.114)	(0.113)
ΔFree	-0.864^{b}	0.044		
	(0.268)	(0.106)		
Δ# other apps dev	-0.012	0.012	-0.007	-0.010
	(0.009)	(0.010)	(0.010)	(0.008)
Δ^2Ownupdate	0.005	0.120^{a}	0.038	0.128^{a}
	(0.017)	(0.022)	(0.025)	(0.027)
Δ^2Singleapp	0.010	-0.032	-0.012	-0.049^{c}
	(0.019)	(0.022)	(0.027)	(0.024)
ΔOwng$_{t-1}$	0.025	0.001	0.017	0.002
	(0.015)	(0.010)	(0.014)	(0.011)
g_{t-2}	-1.204^{a}	-0.019	-1.260^{a}	-0.002
	(0.019)	(0.014)	(0.024)	(0.015)
u_{t-2}	-0.018	0.091^{a}	-0.034	-0.134^{a}
	(0.021)	(0.022)	(0.023)	(0.020)
ΔAgever$_{t-1}$	-0.039^{a}	0.093^{a}	-0.042^{a}	0.181^{a}
	(0.008)	(0.010)	(0.011)	(0.012)
Δ# versions$_{t-1}$	0.070^{a}	-0.267^{a}	-0.001	0.000
	(0.015)	(0.021)	(0.001)	(0.001)
t_4	0.027	0.025	0.089^{b}	0.066^{b}
	(0.023)	(0.021)	(0.031)	(0.024)
t_5	0.094^{a}	-0.040	0.420^{a}	0.009
	(0.022)	(0.022)	(0.032)	(0.026)
Cons	-0.218^{a}	0.074^{a}	-0.153^{a}	-0.050^{c}
	(0.017)	(0.018)	(0.029)	(0.021)
$\overline{R}^2$	0.699	0.306	0.663	0.189
N.	3699	3699	2956	2956