UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FACULTY OF BUSINESS ADMINISTRATION AND ECONOMICS

Working Paper Series

Working Paper No. 2021-08

The Fate of the App: Economic Implications of Updating under Reputation Resetting

Dominik Gutt, Jürgen Neumann, Wael Jabr and Dennis Kundisch

May 2020

The Fate of the App:

Economic Implications of Updating under Reputation Resetting¹

Dominik Gutt

Erasmus University Rotterdam
gutt@rsm.nl

Jürgen Neumann

Paderborn University
juergen.neumann@uni-paderborn.de

Wael Jabr

Pennsylvania State University
wjabr@psu.edu

Dennis Kundisch

Paderborn University
dennis.kundisch@uni-paderborn.de

May 8, 2020

Abstract:

Online reviews on digital platforms are essential in building reputation. Existing reviews, however, become less informative for products whose attributes change over time. Digital platforms have explored reputation update mechanisms that adequately reflect this change. One such distinctive mechanism in the context of apps is that reputation, and thus the underlying review history, is reset with the release of a new app update. While this reputation resetting mechanism might have been intended to ensure that an app's reputation always reflects its current characteristics, it results in complete loss of app reputation irrespective of its prior quality. The implications of this loss of reputation on app market performance remains unknown. This is further complicated by settings where app updating and thus reputation resetting may be required by the hosting platform in compliance with its release of a new operating system. Exploiting an instrumental variables approach on a panel data set from the Apple App Store, we find that when reputation resetting is platform-driven, the effects are asymmetrical across reputation levels. Apps with low prior reputation enjoy a sharp increase in demand while those with high prior reputation experience a sharp decline in demand. Further, our results indicate that the effects on high prior reputation apps are longer-lived than those on low prior reputation apps. Interestingly, for developer-driven updates we find symmetrical effects. Hence, our results reveal that the reputation mechanism arguably designed to ensure accurate reputation has benefits but also substantial drawbacks, specifically for the platform's best performing apps.

Keywords: Online Reviews, App Store, Update, Reputation Mechanism, Digital Platform

JEL Classification: M15, M31, O32, D47

¹ The authors thank participants at the International Conference on Information Systems 2019 for valuable feedback. Our paper also benefited from helpful comments from Rodrigo Belo, Zike Cao, Jelle de Vries, Likoebe Maruping, Arun Rai, Hendrik Schmitz, and Steffen Zimmermann. The work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901) under the project number 160364472-SFB901.

1 Introduction

“This is becoming quite critical. We have been holding a release back, because we did not want the rating reset, and we need to provide an update for iOS 11.”
(Apple Developer Forums 2017)

Since the launch of mobile app platforms, the market for apps has been experiencing rapid growth, thus creating enormous business opportunities. The success of these platforms relies not only on third-party developers creating apps and on buyers purchasing them, but also on those buyers subsequently reviewing the apps they purchased (Lee and Raghu 2014). These crowd-sourced reviews have evolved over time into a well-established reputation mechanism that attempts to overcome quality uncertainties of products sold online (Luca 2017; Tadelis 2016). They are then aggregated and displayed on the platform resulting in an electronic reputation system that incorporates the number of reviews (volume) and the average rating (valence). The reputation that forms as a result became an essential factor in the functioning of online platforms making it the focus of a long stream of research (Babić Rosario et al. 2016; Cheung and Thadani 2012; Gutt et al. 2019a; Hong et al. 2017).

At the same time, platforms have grappled with the problem of ensuring the relevance of this reputation, especially when the product or service being reviewed undergoes fundamental changes over time, making older reviews not quite reflective of the current product or service. For instance, a restaurant might get new management, get a new chef or menu, yet Yelp’s reputation system will still show, amongst others, an overall review valence and the review volume that equally aggregates all the reviews over the lifetime of the restaurant. To address this effect, some platforms rely on assigning smaller weight to older reviews or censoring them all together. For example, Amazon switched in 2015 its average rating derivation to a weighted approach that gives greater weight to newer, more helpful, and verified consumer reviews (Bishop 2015) and AirBnB just shows the reviews posted during the last twelve months.

In the context of mobile apps that we study, ensuring the relevance of reputation becomes even more pressing as apps are prone to alterations with the release of new versions sometimes changing the software’s entire functionality (Dillet 2017). Developers frequently push updates for enhancing the app

functionality (*feature update*, e.g., Förderer and Heinzl 2017; Fleischmann et al. 2016) and for fixing bugs or ensuring compatibility with the platform operating system (*maintenance update*, e.g., Kemerer and Slaughter 1999; Banker et al. 1998). For instance, the developers of the social networking app WhatsApp pushed 16 feature and maintenance updates over a period of 6 months between May and November 2019 (based on the app version history). In response, the two main players in mobile app platforms have adopted a diametrically opposed approach. While the Android Store did not intervene in the shaping of the reputation mechanism, Apple's App Store had in place a mechanism directly tied to releases of app updates: with each app update the platform resets the entire review history displayed to consumers,² thus creating a reputation discontinuity and making the app almost start anew. As such, after an app update is released, the number of reviews (volume), the average rating (valence), and the review details are all automatically reset thus making only readily visible those reviews (and corresponding aggregations) posted subsequent to the latest release. This mechanism is more reflective of the current release. However, such mechanism also implies that, irrespective of the magnitude of the change in characteristics resulting from an app update, the app's reputation is reset.

Cognizant that consumer reviews play a critical role in app reputation and purchasing decisions (Babić Rosario et al. 2016; Ghose and Han 2014), this unique mechanism clearly has critical ramifications for developers. Some might feel tempted to use updates strategically to reset their reputation in order to be less at the mercy of negative reviewers and more in control of their app's reputation. For other developers, perhaps those selling slightly more reputable apps, losing their well-established reputation as a result of an update is unwelcome. Either way, both groups of developers face a risky choice that the platform imposes on them. As such, an app update is no longer just a simple functionality modification tool but can be used rather like a strategic reputational instrument. Under this scenario, the conventional wisdom is that developers are in control of the decision to update, releasing enhancements of features or bug fixes, which we label *deliberate*. However, developers are frequently compelled to push updates to ensure compatibility with the platform (i.e., with a new version of the platform operating system) (Song et al. 2018). We label these updates as *driven by the platform*. As

² While at the time of our data collection this policy was in place, Apple recently adjusted this policy to make the resetting optional to the developer.

prior literature has recognized (Barry et al. 1999), most often platform-driven updates are released to ensure compatibility with the newly released operating system, yet in our research context they still result in reputation resetting.

While the literature on online reviews has primarily focused on environments where product characteristics are implicitly assumed to be time invariant (Chevalier et al. 2018), research focusing on reputation mechanisms that account for variation in those inherent characteristics over time has mostly been analytical (Aperjis and Johari 2010; Dellarocas 2006a; Kovbasyuk and Spagnolo 2019). Our work fills this gap by studying a unique resetting mechanism and by adopting an empirical lens. We seek to answer the following research questions:

1. *What is the effect of a reputation resetting mechanism on app demand?*
2. *How does the role of the platform and prior app reputation influence this relationship between reputation resetting and app demand?*

Apple's App store provides a unique setup to answer these research questions by analyzing the implications of the reputation resetting, and thus reputation discontinuity on subsequent app market performance in the form of app demand. It enables us to disentangle the effect of reputation resetting from the effect of changes in app characteristics triggered by the app update. Based on monthly panel data collected between March 2014 and December 2015, we investigate the implications of app updating in the form of a maintenance update driven either by the platform or deliberately by the developer. With developers having to ensure compatibility of their apps with a new version of Apple's operating system (iOS) and thus releasing maintenance updates but effectively keeping the app characteristics unchanged, we are able to study the isolated effect of reputation resetting on app demand. Of course, the worry is potential endogenous self-selection into updating. However, Apple's iOS update creates an exogenous variation in the updating behavior of app developers circumventing such concerns. With developers also releasing *feature* updates triggering reputation resetting, we study the implications of those updates on app performance as well.

Our results show that when maintenance app updates are made to comply with the release of a new operating system version (say Apple releases iOS 8 to replace iOS 7), those updates are detrimental to high-reputation apps, while low-reputation apps substantially benefit from this platform-driven update. In a way, even a prior high reputation that an app builds over time does not shield it from the harm of loss of reputation. In contrast, the deliberate release of updates (either maintenance or feature) is associated with an increase in app demand both for high- and low-reputation apps. Consequently, one would expect developers of high-reputation apps to align their deliberate feature updates with the release of a new iOS version, to limit the number of reputation resets and mitigate its negative effects. Yet, surprisingly developers do not seem to take advantage of this alignment, even though iOS release dates are known in advance.

The novelty of our work lies in examining a setting where developers' decision to release an update is tied to both substantial changes to their app's reputation and the compatibility with the platform's newly released operating system. This setting contrasts with the extant literature that studies software updates as a production-based (Ji et al. 2011) or competitive-based (Tiwana 2015) decision thus disregarding the requirements for compatibility and reputational impact. In fact, given Apple's reputation resetting policy, app updates become more than part of software development operations but also part of reputation management. While previous research reveals that frequently releasing app updates is a reliable developer strategy to ensure business survival (Pervin et al. 2019), our results suggest that updating under a reputation resetting mechanism is not always beneficial for economic performance. We find that the detrimental effect of platform-driven app updates is a novel aspect developers should consider in their software update strategy (see for example Ji et al. 2011 and Doğan et al. 2011, whose focus is on update strategy irrespective of the platform). In particular, our findings uncover the tension between ensuring an app's operational stability and its loss of online reputation.

Our work contributes to the literature on platform reputation mechanisms and advances our understanding of reputation discontinuity. In this way, our work is a direct response to the call for more research on platforms that has been identified as an emergent theme at the operations management and information systems interface (Ba and Nault 2017). First, although there is anecdotal (Apple Developer Forums 2017) as well as academic evidence (Comino et al. 2019) on the link between app updating

behavior, app reputation, and app performance, we are not aware of any empirical study that investigates this interplay with a focus on reputation discontinuity. On a more general level, prior studies have, by and large, examined how changes in reputation (i.e., volume and valence of reviews) affect product performance (e.g., demand) (e.g., Chevalier and Mayzlin 2006; Duan et al. 2008; Anderson and Magruder 2012). These studies have almost exclusively investigated the demand effect when reputation is built incrementally – e.g., increasing the volume of reviews by an additional review or changing the average rating. For instance, prior empirical literature has gleaned these insights exclusively from data sets where review volume increases gradually but where decreases in volume are technically infeasible. Only a few theoretical papers (e.g., Dellarocas 2005, 2006b; Aperjis and Johari 2010) studied truncating reputation by limiting the availability of past reviews either in terms of a time window or in terms of a fixed number of recent reviews. Our study extends the literature on reputation mechanisms in digital platforms by being the first to explicitly focus on the effects of removing reputation (which we label as reputation discontinuity) instead of incrementally improving or deteriorating reputation.

Second, we find indications for divergent effects on app performance depending on prior reputation (high or low), update types (maintenance or feature) and sources (driven by the platform or deliberately by the developer). Specifically, we uncover scenarios where low-quality apps make use of the reputation discontinuity to enjoy improved performance. Similarly, we uncover scenarios where high-quality apps suffer from updating. Interestingly, while the platform reputation resetting mechanism is supposedly put in place to ensure that app reputation adequately reflects app quality after an app update, we illustrate how this reputation discontinuity can have asymmetrical effects on apps when updates have to be released to comply with the platform operating system. We show how those asymmetrical effects on high- versus low-reputation apps have an unexpected detrimental effect on high-reputation apps. Further, our results indicate that the effects on demand of reputation resetting on apps with high prior reputation is longer-lived than those effects on low prior reputation apps. We thus provide new insights relevant to information systems research and platform economics research that will benefit practitioners and researchers alike. In doing so, we make use of a novel set of instrumental variables that leverage an exogenous event driving app updates, an approach we believe can also be extended to other phenomena in the study of apps.

2 Theoretical Background

In online platforms, reputation of a product, service or seller is derived essentially from crowd-sourced reviews and the information they contain. Such reputation has been shown to be quite influential on market performance resulting in improved sales, larger profits and many other benefits (e.g. Rice 2012; Babić Rosario et al. 2016; Gu and Ye 2014; Sun and Xu 2018; Wang et al. 2019). A common finding in this literature is that reputation systems help reduce information asymmetries between buyers and sellers and reduce uncertainties. This stream of research however implicitly assumes that product characteristics are time-invariant (as noted by Chevalier et al. 2018). Once those inherent characteristics start to vary over time, as is the case with app updates for example, reputation systems that are not able to map reviews to their app's respective release become less informative. Consequently, researchers have recently argued that platforms should adopt a more active role to control the information displayed in order to ensure its relevance and accuracy (Tadelis 2016).

Two possible truncation approaches online platforms may employ consist of limiting the availability of past reviews (memory length) either in terms of a time window (e.g., reviews just from the last twelve months) or in terms of a fixed number of recent reviews. Consequently, aggregated metrics are calculated based on this truncated set of reviews. To study this phenomenon, the limited theoretical literature in information systems research on the economics of reputation mechanisms analyzes truncating information by partially or totally discounting older reviews. For instance, Dellarocas (2005) and (2006b) study a reputation mechanism that updates a monopoly seller's reputation publishing only the k most recent ratings and find that such mechanism induces higher levels of cooperation and market efficiency compared to when all ratings are published. To further explore memory length, Aperjis and Johari (2010) analyze the implications of the size of the optimal number of k ratings to keep; however, their focus is on seller truthfulness in describing the items for sale.

The broader theoretical literature on the economics of reputation has focused on the effect of product or seller reputation and the implications it has on seller's performance when the informational policy of the platform includes truncating information, i.e., limiting memory length (Elul and Gottardi 2015; Kovbasyuk and Spagnolo 2019; Kremer et al. 2014). Though in the context of lending and bankruptcy,

Elul and Gottardi (2015) build on a long stream of literature to show that when forgetting agents' past reputation occurs at fixed and shorter intervals (i.e., every 5 years instead of 10 for instance), agents act in a myopic way thus increasing their rates of default since the risk of reputational loss is minimal. In a context similar to ours where the intermediary platform hosts buyers' feedback about their transactions with sellers, Kovbasyuk and Spagnolo (2019) focus on a market with heterogeneous sellers whose characteristics change over time. They study the effects of erasing past buyers' feedback on trade as well as welfare and find that it can be welfare maximizing for a platform hosting a reputation system to apply resets to high-reputation sellers more frequently than to low-reputation sellers. The underlying mechanism at play encourages buyers to not only buy from high-reputation sellers but also from sellers with no existing reviews. This in turn produces large positive information externalities to the market via newly posted reviews on non-rated sellers.

What is common in this stream of literature is the focus on a sliding window for truncating reviews to derive reputation and the implications of increasing or decreasing the size of that window. In reality, while these mechanisms exist, some platforms have also adopted alternatives that are triggered by changes in underlying product characteristics. Once those characteristics change (e.g., a release of an app update) the reputation is reset. These mechanisms have found no coverage in the literature so far. Our paper contributes to fill this void in the literature by empirically studying such a reputation resetting mechanism.

3 Research Framework & Hypotheses

Kovbasyuk and Spagnolo (2019) offer a stylized model that is closely related to our reputation resetting mechanism. Their paper offers an analytical lens and focuses on the overall market performance, ours offers an empirical lens with a focus on the sellers' perspective. We also extend the general notion behind this model to incorporate a novel perspective which accounts for *who* is driving the reputation resetting. In their setting, only the seller decides on new product versions but has no control over reputation resetting, while in ours both the seller and the platform may drive the release of a new product version and thus trigger a reputation reset.

Our reputational setup is therefore as follows: A market consists of a number of sellers (specifically developers) each selling only one product (specifically app). Each developer has a true quality type that is either high in case of a high-quality app or low in case of a low-quality app. Hence, developer type and app type are equivalent (we use those two terms interchangeably). The app type is private information to its developer and can change over time due to efforts by the developer to improve app quality. Once a developer has sold an app to a buyer, the buyer can leave a review that contributes to the app's reputation, which perfectly identifies its quality type. In particular, the buyer either leaves a positive review for a high type app or a negative review for a low type app. Alternatively, buyers can decide to not leave a review at all. This market setup gives rise to three pools of apps:

- 1) a pool of *non-reviewed apps* for which buyers cannot yet observe the true product type,
- 2) a pool of *high-reputation apps* representing all high type apps that have been positively reviewed by at least one buyer, and
- 3) a pool of *low-reputation apps* containing low type apps that have been negatively reviewed by at least one buyer.

Subsequent buyers observe the three pools of apps prior to making a purchase and only consider buying a non-reviewed app (i.e., from pool 1) or a high-reputation app (i.e., from pool 2). Buyers will not buy a low-reputation app (i.e., from pool 3), because of its reputational signal. While it is expected that buyers would typically choose from the high-reputation pool, they might also want to experiment and choose from the non-reviewed pool. One such experimentation instance is when their valuation for a high-reputation app is lower than that app's price. Eventually, low-reputation apps drop out of the market – unless they can improve their reputation or, in the extreme, erase it. If reputation resetting is available, developers with apps in the low-reputation pool might want to choose such resetting in order to re-enter the pool of non-reviewed apps and, thus, get the chance again to be purchased by buyers willing to experiment. On the other hand, developers in the high-reputation pool aim to maintain high ratings for their apps in order to continue selling them.

However, if reputation resetting is not optional but rather required by the hosting platform (as is the case with the Apple App Store) whenever an app update is released, this app is moved to the pool of non-reviewed apps irrespective of whether it belonged to the pool of high- or low-reputation apps before

the update. Because developers have to ensure their software compatibility with the hosting platform's operating system, anytime the platform pushes its own update of the operating system, the developers are also pushed to update their apps. We label these updates as *driven by the platform*. Alternatively, developers might release feature enhancements or bug fixes independent of platform updates anytime. We label these updates as *deliberate*. Both types of updates result in reputation resetting. Note that while a deliberate update is initiated by single developers at any time, a platform update typically will induce multiple apps to be updated at the same time. For illustration purposes in Figure 1 we show how different dynamics between these two forms of reputation resetting play out.

Note that a developer of a low-reputation app pushes a deliberate update on her own schedule thus moving it to the pool of non-reviewed apps (top part of the figure). Several developers of both high- and low-reputation apps push a platform-driven update of their apps around the same time after the platform has updated its operation system. All these apps are moved to the pool of the non-reviewed apps (lower part of the figure).

Figure 1: Dynamics of Reputation Resetting

Next, we delineate our hypotheses that focus on the demand faced by developers and the dynamics of apps switching between the three pools of non-reviewed, high-reputation, and low-reputation apps. First, we focus on reputation resetting for low-reputation apps. The empirical literature has shown that a low reputation causally leads to lower demand (Babić Rosario et al. 2016). Thus, a reputation resetting removes the unfavorable signal buyers rely on to learn about the low app quality and helps the app to start anew by placing it back into the pool of non-reviewed apps that buyers are willing to experiment with. In this pool, buyers cannot distinguish between apps that have been moved from the pool of high-

reputation apps to this pool as a result of resetting and apps that have been moved from the pool of low-reputation apps also as a result of resetting. Being able to remove the negative reviews they have received by deliberately updating, one expects that low-reputation apps would benefit from reputation resetting, thus resulting in an increased demand. Thus, we formulate our first hypothesis as follows:

Hypothesis 1: Demand for a low-reputation app increases after a deliberate update.

Furthermore, even if an app update is driven by the platform, it results again in the removal of negative reviews and thus the signal of poor quality. Hence, one expects again that low-reputation apps would experience an increased demand after reputation resetting. Therefore, we formulate an analogous hypothesis for the case of platform-driven updates.

Hypothesis 2: Demand for a low-reputation app increases after a platform-driven update.

Note that for both deliberate and platform-driven updates of low-reputation apps, we expect to observe an increase in demand after reputation resetting (as a result of buyer's experimentation).

Next, we consider the case of high-reputation apps. In contrast to low-reputation apps, high-reputation apps enjoy a positive demand due to the signal conveyed through their reputation (Babić Rosario et al. 2016). Thus, a resetting of the reputation after an update should be associated with a decrease in demand once the good quality signal is gone. Whenever an update of a high-reputation app is pushed, the app is moved from the pool of high-reputation apps to the pool of non-reviewed apps making it indistinguishable from low-quality apps in the same pool. Thus, the app loses the high reputational signal. Consequently, we expect that the demand for high-quality apps will decrease after an update driven by the platform. Hence, we have our third hypothesis.

Hypothesis 3: The demand for a high-reputation product decreases after its reputation is reset due to a platform-driven update.

Note that the mechanism laid out in our research framework implies that a developer of a high-quality app, which is already in the pool of high-reputation apps, has little incentive to push a deliberate update given the exposure the app already has. As a result, we do not deduce a hypothesis for the case of deliberate updates of high-reputation apps unlike in the case of low-reputation apps. Still, we recognize that there may be reasons outside of our research framework (e.g., market pressure, competitors, customer needs) that may contribute to a developer pushing a deliberate update for an app in the pool

of high-reputation apps. Therefore, the implications of deliberate updates for previously high-reputation apps will also be analyzed.

4 Data

Our data consists of app sales ranks, app reviews, and app release notes with each update, all of which we scraped from AppAnnie.com³ between March 2014 and December 2015, during which two major iOS updates were made. We collect the app sales rank data for apps listed in the top 500 charts on the Apple App Store under three chart types (free, paid, and grossing) across 24 different categories (including games, business, education, lifestyle, etc.). We also collect all online reviews from the App Store to derive an app's average rating and number of ratings at any given point in time. In particular, whenever an app entered the top 500 during our observation period, we crawled all review information on this app until December 2015 even if that app has left the top 500 before that date.⁴ We provide in the Appendix (Figure A1) additional details and snapshots of what consumers see when navigating reviews. While a consumer can still choose to display the volume, the average star rating and reviews for previous versions, by default, they see the volume, the average star rating and reviews corresponding to the current version of the app. Additionally, app developers provide release notes for every update of their app, which we also collected. After we drop observations of apps with no reviews in a single month, our final data set comprises 7,245 apps that we use in our analyses. Throughout our study, we make an important distinction between “app” and “unique app”. For example, in our data set, there is an app for Pokémon fans called “Pokemini – Dress and Decorate!”. We refer to this as “app”, it is one of the 7,245 apps in our data set.⁵ This app is listed in the chart types “free” and “grossing”. Moreover, in the grossing chart type, the app is listed in the category “Social Networking”. In the “free” chart type, it is listed in the category “Social Networking” and in the category “Games”. We refer to these different

³ Opposed to Apple App store rankings, which are displayed when consumers search for an app, AppAnnie provides sales rank data based on actual downloads (Scott 2012).

⁴ In the case of an app dropping out of the top 500 chart, we set its sales rank to 501. When an app drops out of the top 500, it is more likely that its actual sales rank is higher than 501. Therefore, the estimates in our subsequent analyses for apps that experience a sales rank increase and drop out of the top 500 chart are essentially a lower bound. Likewise, for apps that see a decrease in sales rank and (re-)enter the top 500, the estimates are also a lower bound because we assume that an app's sales rank prior to entering the top 500 was 501, when its actual sales rank is most likely higher. Hence, imputing a sales rank of 501 to apps outside of the top 500 chart is a very conservative approach that does not bias our results in the direction set out by our hypotheses.

⁵ In Table A2 in the Appendix, we provide an overview over how many apps are comprised in each category.

instances of “Pokemini – Dress and Decorate!” in the various chart types and categories as “unique apps”.

Following common practice, we classify updates into maintenance and feature updates. To that end, and in line with the literature (Kemerer and Slaughter 1999; Telang and Wattal 2007), we conducted a catchword-based content analysis of the release notes of the apps in our data set. To that end, two co-authors conducted an exploratory assessment of a random sample of release notes to better design our catchword-based implementation. They then identified the catchwords that are systematically used to characterize maintenance updates and the ones used to characterize feature updates. Next, after a review of the terms and an identification of the most frequent ones referencing update type, all co-authors agreed on using the terms “bug”, “fix”, “minor”, “iOS”, “iOS 8”, or “iOS 9” to identify a maintenance update and using the terms “new” or “added” to identify feature updates, while allowing for both upper- and lower-case spelling. To mitigate concerns over potential misclassification of updates, two co-authors independently checked the validity of this textual classification on a sub-sample of 40 randomly selected release notes. To do this, each of the authors hand-coded whether a release note indicated a maintenance or a feature update. Afterwards, we calculated the interrater agreement between the manual coding and our catchword implementation using Krippendorff’s Alpha, a conservative measure of interrater agreement. The joint interrater agreement between the manual coding and our catchword-based approach is 0.76 for maintenance updates and 0.79 for feature updates. These values exceed 0.7, a threshold identified in the literature for sufficient interrater agreement (Lombard et al. 2002). Thus, we conclude that our catchword-based approach has reliably identified maintenance and feature updates. As a last step, each of the app updates are classified as either *maintenance* or *feature* if the release note contains at least one of the corresponding catchwords. We note that in the cases where a release note contained words from both lists of catchwords, we classified the update as *feature* as well. Our approach has several advantages over an alternative approach used in the literature that relies on version numbers to classify update types (e.g., as in Comino et al. 2019). Simply relying on changes in version numbers (say from 3.5 to 3.5.1, 3.6, or 4.0) masks the purpose of the update (feature or maintenance). Such version numbering also does not describe compliance with the platform operating

system update (i.e., a new iOS version). Additionally, different developers might have different conventions for naming app versions. Nevertheless, in a robustness check included in the Appendix, we adopt the alternative classification of using version numbers and demonstrate that our main results remain, by and large, qualitatively unchanged.

The main variable of interest to proxy demand in our analysis is the natural logarithm of the monthly sales rank $LN(RANK)$ (as in prior literature, e.g., Brynjolfsson et al. 2003; Garg and Telang 2013). Moreover, applying a logarithmic transformation to ranked data facilitates the interpretation of results by allowing for percentage interpretation as opposed to absolute rank interpretation. For instance, a one-unit rank increase might indicate a significant change for an app that is already highly ranked, say ranked second, but might make little difference for a low ranked app, say ranked 400th. The other two main variables of interest are dummy variables capturing the update type, maintenance (*MAINT*) or feature (*FEAT*). Moreover, we also calculate an app's rating average in a given month (*AVG*) and the number of ratings (*NUM*) taking the most recent reputation resetting as starting point (which is also reflected in the App Store).

To control for additional time-variant factors that might affect an app's sales rank, we also incorporate information on developer and competitor behavior. To control for the effect of the feature updates released by a focal app on its app demand, we determine the update frequency of feature updates for each app (*UP_FRQ_FEAT*). Prior literature has consistently highlighted that a focal app's closest competitors come from the same app category, as apps in the same category, for instance, represent possible substitutes of each other (Arora et al. 2017; Liu et al. 2014). A focal app's sales rank might suffer if its competitors are particularly active in releasing feature updates for their apps. To control for this competitive effect, we calculate the number of feature updates for an app's category (*NUM_FEAT_CAT*) to measure the feature updating behavior of competitors. Additionally, following an approach introduced in prior literature (Gutt et al. 2019b), we calculate an app category's Herfindahl Hirschmann Index (*HHI*) based on the total volume of reviews that apps in a particular category received. This variable allows for measuring and controlling for market concentration and the level of competition an app faces. Because an app's closest competitors are usually from other apps within the

same category (e.g., WhatsApp is a close competitor of Facebook Messenger, Telegram Messenger, and Viber Messenger, which are all in the category “Social Networking”), we compute the *HHI* at the category level. We calculate the *HHI* at the category level by summing the total squared percentage market shares of reviews for each app, normalized between 0 and 1. An *HHI* of 0 indicates high competition, while larger values imply less competition. If, for example, a category consists of two apps with 200 reviews each, the total review count of that category is 400 and the *HHI* for this market is $\frac{(50\%)^2 + 50\%^2}{10,000} = 0.5$.

Table 1 displays the descriptive statistics of our data set. Note that $LN(RANK)$, *MAINT*, *AVG*, and *UP_FRQ_FEAT* are computed for app i , in category c , chart type r , in year y , and in month t . *NUM_FEAT_CAT* and *HHI* are computed at the category level.

Table 1: Descriptive Statistics

Variable	N	Mean	Std. Dev.	Min	Max
$LN(RANK)$	197,813	4.8122	1.2410	0	6.2166
<i>MAINT</i>	197,813	0.1301	0.3364	0	1
<i>AVG</i>	197,813	0.1145	0.3184	0	1
<i>NUM</i>	197,813	3.5382	1.1657	1	5
<i>UP_FRQ_FEAT</i>	197,813	0.1122	0.1667	0	1
<i>NUM_FEAT_CAT</i>	197,813	61.048	38.0984	4	247
<i>HHI</i>	197,813	0.0929	0.1309	0	0.9952

4.1 Distinguishing High- and Low-Reputation Apps

As in prior literature (Babić Rosario et al. 2016; Chevalier and Mayzlin 2006; Zervas et al. 2017), we adopt two dimensions to characterize reputation of an app, namely review volume and valence. Naturally, high-reputation apps are expected at the higher end of volume and valence in their market, while low-reputation apps are expected at the lower end of volume and valence. While we are specifically interested in analyzing the impact of reputation resetting upon updating on high- and low-reputation apps, rather than simply splitting our sample in high- and low-reputation apps, we adopt a finer grained operationalization and follow prior literature in splitting the apps in four reputation

quartiles (Clemons et al. 2006; Dewan and Hsu 2004; Gao et al. 2015; Manes and Tchetchik).⁶ In particular, we split the apps into four quartiles depending on the volume of ratings (as captured by the variable *NUM*). This approach is in line with recent work by Schoenmüller et al. (2019) who conclude that “the relation between volume and sales is more robust than the relation between valence and sales” based on a review of three meta-analyses (Babić Rosario et al. 2016; Floyd et al. 2014; Ya et al. 2015). Table 2 depicts summary statistics for each quartile, where the first quartile is the one with the low-reputation apps, while the fourth quartile is the one with the high-reputation apps. Reassuringly, we find that valence for each of the four quartiles is consistent with our operationalization based on volume⁷ (apps of the fourth reputation quartile have the strongest reputation with the most reviews and the highest average rating, followed by apps in the third quartile, the second quartile, and quartile 1).

Table 2: App Reputation Quartiles

Variable	Reputation	Mean <i>NUM</i>	Mean <i>AVG</i>	Mean <i>RANK</i>
1 st Quartile of <i>NUM</i>	Low	4.038	3.172	193.268
2 nd Quartile of <i>NUM</i>		14.688	3.406	155.949
3 rd Quartile of <i>NUM</i>		42.633	3.617	131.776
4 th Quartile of <i>NUM</i>	High	302.265	4.003	116.135

We find that apps with the highest reputation also have the best mean sales rank, as one would expect. Likewise, the sales rank deteriorates gradually towards the quartile with the low-reputation apps.

5 Methods

Testing for Hypothesis 1

In Hypothesis 1, we study the demand for a low-reputation app after a deliberate update. We estimate a simple fixed effects model that tests the positive correlation between the deliberate decision of developers to push an app update, thus resetting its reputation, and demand. Because the decision to

⁶ In a robustness check, we also median-split apps into two groups and find that the results of our main models testing all three hypotheses remain qualitatively the same.

⁷ An alternative option would have been to split app quartiles based on valence (*AVG*). The drawback, however, is that it is not uncommon for apps to have a very high valence but only very few reviews (e.g., at time of app launch). This makes valence by itself a noisy indicator of quality. In fact, our data echoes this phenomenon.

update is endogenously made by the app developer, this analysis is correlational. Next, to tease out the causality effect of reputation resetting on demand, we introduce our identification strategy.

Testing for Hypotheses 2 and 3

Naturally, identifying a causal relationship between reputation resetting and sales ranks poses a challenge. Even though we control for a wide range of time-varying factors that could influence the demand for an app, there might still be unobservable endogenous factors that threaten the validity of our results. The most relevant concern is that an app developer can endogenously decide when to reset an app's reputation, i.e., when to release an update, which is the case for deliberate updates. On the one hand, developers of an app in reputation quartile 1 could exploit the reputation resetting by frequently releasing maintenance updates simply to reset their low reputation profile. On the other hand, it is possible that developers of apps with high reputation might be able to intuitively determine when the demand gain from, for instance, fixing a bug and pushing an app update outweighs the demand loss from losing high reputation. Therefore, developers of high-reputation apps might self-select into updating only when they deem updates beneficial to the market performance of their apps. Moreover, prior literature has documented that app developers tend to release updates whenever their reputation decreases (Comino et al. 2019). Thus, app update decisions could be endogenously determined by app reputation. To mitigate such potential endogeneity, we need a source of exogenous variation that compels app developers to release an update. The release of an iOS update is such an exogenous shock that allows us to study the causal effects of reputation resetting on app demand. Specifically, we study the effects of such updates on the demand for low-reputation apps (H2) and high-reputation apps (H3) after a platform-driven update.

We implement the exogenous shock, i.e., the iOS update, as instrumental variables (IVs) in a two-stage least squares (2SLS) estimation. We exclude feature updates in our testing of the hypotheses to mitigate potential concerns that the relationship between reputation resetting and demand is confounded by changes in app inherent characteristics. We therefore only include maintenance updates, which are rather free from these potential confounding effects.

We identify two exogenous shocks, namely, the two major iOS updates iOS8 in September 2014 and iOS9 in September 2015 and collect data around both events. In Figure 2, we depict the timing of the two major iOS updates as well as the surrounding lag and lead months in our data.

For each iOS release, iOS 8 and iOS9, we show two diamond-lines. The longer one is for the actual month of the release (September), the shorter one for the next month (October). Both are used as IVs, as explained in the text. The checkered areas mark the lag months and the striped areas mark the lead months.

Figure 2: Timeline of Events

The rationale behind our IV identification strategy is as follows: Apple releases new versions of the iOS each year in September. Following such iOS update, developers often have to update their apps to resolve compatibility issues. Developers might even preview some of the changes due to beta versions and presentations at the annual Apple Worldwide Developers Conference (Förderer 2020). Still, while developers can endogenously decide whether to release a maintenance update in compliance with the new iOS, they have little influence over the extent of issues or potentials introduced by the iOS update and the resulting necessity to update their app. For developers this is hard to predict to a large extent and hence almost random. As anecdotal evidence, an Apple developer blog (Sharma 2020) writes that app compatibility with different iOS versions is the number one challenge to iOS app developers and lists future-proofing of an app as a great challenge even calling it “somewhat an unpredictable task”. Even though developers may prescreen upcoming iOS changes, developers often complain that many changes ensuing an iOS can remain undocumented and without announcement by Apple, hence potentially catching app developers off guard (Hainzinger 2020).

We introduce two IVs reflecting the timing of the iOS update. One instrument indicates the month of the iOS update (i.e., September) whereas the other indicates the month right after the update (i.e., October) (see Figure 2). We chose these two months as IVs as we expect the majority of platform-driven (maintenance) updates to be released right after an iOS update. The descriptive statistics on

update frequency presented in Figure 3 supports our approach given the spikes in maintenance updates in the month of an iOS update and the first month after the iOS update. With these variables, we instrument the *MAINT* update dummy variable in a multi-instrument 2SLS estimation. We note that, technically, our IV approach identifies the *local average treatment effect* (LATE) on the treated apps, as discussed for instance by Imbens (2010). The treated apps in our case are the apps of developers who “comply” with the treatment (the iOS update) and release a maintenance update in response. Hence, the coefficients of the IV estimations apply to the causal effect of the subpopulations of apps whose developers release a maintenance update during or shortly after the iOS update.

The identification of the causal effect using our IV approach rests upon the assumption that our IVs fulfill the *relevance criterion*, i.e., that they are significantly correlated with the endogenous variable *MAINT*. Figure 3 provides a model-free visual support for the relevance criterion, thus justifying our choice of IVs (the iOS release months and the two following months). We depict the number of maintenance updates for each month of our observation period. The diamond-lines in this figure represent our IVs. Reassuringly, the percentage of maintenance updates released by developers increases sharply in the months of the iOS updates (September 2014 and September 2015) and remains high in the following month (October 2014 and October 2015), with peaks more pronounced in 2014. Figure 3 also shows that this trend is consistent across apps in all four quartiles.

Figure 3: Monthly Percentage of Apps within a Group that Release a Maintenance Update

In addition to the model-free evidence, the assumption about the relevance criterion can also be formally tested using the F-Test of the first stage regressions. We conduct this test and find that our IVs are significantly correlated with the maintenance updates (see the F test results in Table 3).

We next address the *exclusion criterion* that our IVs are not directly related to the sales rank outcome variable. While this criterion cannot be tested (Angrist and Pischke 2009), we argue that the release of an iOS update has no direct effect on the sales rank of an app. It is rather the case that it affects the sales rank through the reputation resetting that happens to apps whose developers release a maintenance update to comply with the new iOS version, which is exactly the relationship we would like to leverage in our IV approach. One exception of concern is when the iOS update introduces new functionalities that selectively promotes certain apps and, thus, boosts their sales rank. Therefore, we revisit this exception when conducting robustness checks.

5.1 Empirical Model

To test our hypotheses, we estimate the following equation:

$$\begin{aligned} \ln(RANK)_{i,t} = & \beta_0 + \sum_q \beta_q (MAINT_{i,t} * QUART_{i,t-1,q}) + \sigma MAINT_{i,t} \\ & + \sum_q \gamma_q QUART_{i,t-1,q} + \alpha X_{i,t} + \theta_i + \delta_t + \varepsilon_{i,t} \end{aligned} \quad (1)$$

$RANK_{i,t}$ represents the sales rank of a unique app i in month t . Note that each listing of an app in one of multiple categories and/or multiple chart types at the same time is treated as a unique app.

$MAINT_{i,t}$ is a dummy variable indicating whether app i has received a maintenance update in month t . As the reputation of apps is reset each time the developer releases an app update, $MAINT_{i,t}$ indicates reputation resetting of apps. Note that there is no category and chart type subscript, as once the app is updated this is reflected across categories and charts. To measure the differential impact of reputation resetting via maintenance updates across different app reputation quartiles, we introduce three app reputation quartile dummies ($QUART_{i,t-1,q}$, $q \in \{1,2,3\}$, with quartile 4 being the base case), which we interact with $MAINT_{i,t}$. In particular, we use $QUART_{i,t-1,q}$ in $t - 1$, i.e., the rating quartile prior to the maintenance update, so that this variable reflects the app reputation quartile that is being reset upon an update. Naturally, we also introduce stand-alone terms of $MAINT_{i,t}$, and $QUART_{i,t-1,q}$. Our main coefficients of interest are β_q indicating the impact of rating resetting across the different reputation quartiles. Note that the coefficient of the stand-alone term $MAINT_{i,t}$ identifies the impact of rating resetting for the fourth reputation quartile (as this quartile is the base case). Moreover, the stand-alone term of $QUART_{i,t-1,q}$ serves as a control variable for the direct relationship between app reputation and demand.

Vector X comprises NUM_FEAT_CAT , UP_FRQ_FEAT , and HHI as control variables. Note that at any given month, an app is assigned to a specific iOS release (iOS8 and iOS9) if this month lies within six months lag or six months lead⁸ from the iOS update. Additionally, for each unique app, we introduce

⁸ Note that for the iOS9 update we could just collect three lead months.

two dummy variables, one for the year surrounding the first iOS update (iOS8) and one for the second update (iOS9). $\theta_{i,t}$ is a vector comprising these unique app fixed effects (FEs). To control for potentially additional unobserved seasonal effects, we introduce half-year fixed effects δ_t . $\varepsilon_{i,t}$ is a random error term. For deliberate reputation resetting, we run a fixed effects model as defined in equation (1). For platform-driven reputation resetting, we run the same model but instrument for *MAINT* using our IVs as explained earlier.

6 Model Results

Table 3 depicts our results of estimating the model in equation (1). Column (1) displays the simple fixed effects model to test H1 and column (2) depicts the 2SLS model with fixed effects to test H2 and H3. Note that a negative sign of a coefficient indicates a decrease in the numerical sales rank, i.e., an improvement in sales. Also note that to determine the overall effect for low-reputation apps (quartile 1), one has to combine σ , the direct effect of *MAINT* as the base case with respect to high-reputation apps (quartile 4) and β_1 , the interaction effect of *MAINT* and *QUART* for low-reputation apps (quartile 1). Lastly, note that in the simple fixed effects model testing H1, developers are free to choose the timing of a maintenance update, and thus the reputation resetting.

The coefficient σ of *MAINT* in column (1) suggests that when developers reset their reputation deliberately, apps in the fourth reputation quartile experience a statistically significant decrease in sales rank by around 2% ($100 * (e^{-0.01976} - 1)$). The interaction term for low-reputation apps (quartile 1) is statistically insignificant and, thus, indistinguishable from zero. The combination of these two effects points to a decrease in sales rank. As a result, for low-reputation apps (quartile 1), we find support for H1 due to the significant decrease in sales rank upon a deliberate reputation resetting. In summary, we conclude that when app developers can decide for themselves when to release a maintenance update, and reset the reputation of their app, this is beneficial though small in magnitude for the app's sales rank for both high- and low-reputation apps. This may be taken as evidence that app developers can avoid negative consequences of reputation resetting when deliberately deciding the timing of a maintenance update. This also confirms the endogeneity concern and the need to account for it.

Table 3: Model Results

	(1) FE	(2) 2SLS + FE
VARIABLES	$LN(RANK)$	$LN(RANK)$
MAINT	-0.0197** (0.0099)	0.3067** (0.1427)
MAINT * QUART₁	0.0194 (0.0154)	-0.7138*** (0.2255)
MAINT * QUART₂	0.0206 (0.0142)	-0.0029 (0.2588)
MAINT * QUART₃	0.0053 (0.0136)	-0.1532 (0.2088)
QUART₁	0.2233*** (0.04215)	0.3448*** (0.0421)
QUART₂	0.1659*** (0.0476)	0.1759*** (0.0400)
QUART₃	0.0995*** (0.0401)	0.1279*** (0.0401)
AVG	-0.0159*** (0.0042)	-0.0134*** (0.0042)
NUM_FEAT_CAT	0.0006*** (0.0001)	0.0006*** (0.0001)
HHI	-0.1597*** (0.0237)	-0.1496*** (0.0237)
UP_FRQ_FEAT	-0.1081*** (0.04258)	-0.0638 (0.04258)
Unique App FE	✓	✓
Half-Year FE	✓	✓
Observations	197,813	197,029
Groups	22,463	21,679
2SLS 1. Stage F Test	-	47

Note: *** p<0.01, ** p<0.05, * p<0.1, Robust standard errors clustered on the app level in parentheses. A group is represented by a unique app in a given period (March 2014 to February 2015 or March 2015 to December 2015).

Next, we discuss the results of the 2SLS when developers' maintenance updates are driven by the platform as a result of an iOS update. We start with the base case, i.e., the analysis of the high-reputation apps. The coefficient of *MAINT* in column (2) suggests that upon a platform-driven update, high-reputation apps experience a statistically significant and sizeable increase in sales rank by 35% ($100 * (e^{0.30673} - 1)$). Therefore, we find support for H3 that demand for a high-reputation app decreases after its reputation is reset due to a platform-driven update. We find a substantial and statistically significant decrease in sales rank for low-reputation apps (quartile 1) by 51% ($100 * (e^{-0.71384} - 1)$), relative to the base case. The total effect on apps of the first app reputation quartile can be computed by subtracting the effect of *MAINT * QUART₁* from the effect of *MAINT* (35% – 51%) which yields a decrease in sales rank of 16%. Therefore, we also find support for H2 that demand for a low-reputation

app increases after its reputation is reset due to a platform-driven update. In sum, our results provide empirical evidence that high-reputation apps are substantially hurt by platform-driven updates whereas low-reputation apps benefit from them.

7 Further Analyses

7.1 Effects of Feature Updates on Demand

So far, we have focused on updates of type maintenance as the main mechanism for resetting app reputation. Yet, feature updates also represent a way through which developers can reset the reputation of their apps. Intuitively, the rationale for app developers to release feature updates should be to enhance the app by providing new features and functionalities to appeal to existing and attract new customers. Thus, on the one hand, feature updates in general should have a positive relationship with demand. On the other hand, feature updates also reset an app's reputation. For app developers it is only advisable to release a feature update when the expected increase in demand via the new feature update outweighs the potential loss in demand due to the reputational resetting. We empirically test the relationship between reputation resetting via feature updates and demand by re-estimating our 2SLS model replacing *MAINT* with *FEAT*, which is a dummy variable indicating whether an app has received a feature update in a month t .

In expectation of a positive sales effect from feature updates, it seems plausible that developers might strategically postpone some of their feature updates until the next iOS update. For example, anticipating an iOS update in September, instead of releasing a feature update in July or August, app developers might post-pone feature updates to align them with maintenance updates that might be necessary anyway in September. In this way, app developers would reap two benefits. First, they would only reset their reputation once instead of twice. Second, the expected positive association of feature updates with demand could partially offset the potentially negative effect of platform-driven rating resetting on demand. To delve into the question whether app developers are strategically postponing their feature updates to align them with iOS updates, we use two specifications to code the dummy variable *FEAT*. In the first specification, we recode the dummy variable *FEAT* to now require that the release notes contain keywords for both types of updates. That is, *FEAT* now equals 1 when both feature and

maintenance updates are packaged together in one update. If app developers aligned their feature updates with iOS updates, we would expect the first stage of this regression to be highly significant. That is, feature updating activity would be explained well by our IVs. The results of estimating this model are shown in column (1) of Table A3 in the Appendix. The F-test of the first stage ($F = 3$) is well below the conventional threshold of 10 and well below the first stage of the 2SLS regression of the baseline results ($F = 47$, see Table 3) (Stock and Yogo 2005). Moreover, the coefficient of *FEAT* grows quite large, which is a typical sign of weak instrument bias, i.e., a low F-statistic, corroborating our interpretation. This suggests that developers do not strategically align feature updates with iOS updates. In the second specification, in column (2) of Table A3, we recode the dummy variable *FEAT* in a way that it equals 1 for pure feature updates (i.e., no reference to maintenance in the release notes). Again, the first stage of the regression exhibits an insignificant F-statistic underlining that feature updates are not explained well by our IVs.

In summary, developers on average do not strategically align feature updates with iOS platform updates. We conclude this because neither updates that are both, feature and maintenance, nor pure feature updates are significantly correlated with our IVs.

7.2 App Demand Dynamics

So far, we have restricted our analysis to the immediate impact of reputation resetting on app demand in the month of resetting. In an additional analysis, we are interested in examining the dynamics of app demand multiple months before and after the reputation resetting. This analysis holds value in two ways: Firstly, by analyzing demand before resetting, this empirical exercise helps establishing that our analysis is not confounded by pre-existing spurious dynamics in app demand prior to reputation resetting. In fact, it could be possible that apps exhibit large natural fluctuations in demand due to the competitive nature of the app market. If this was the case, this could confound our analysis. Secondly, by analyzing demand after resetting, we can investigate potential dynamic evolutions of the effect of reputation resetting on demand over time.

To this end, we estimate our 2SLS model replacing the dependent variable at $t = 0$ with six different specifications where $t \in \{-3, -2, -1, 1, 2, 3\}$. Therefore, we account for sales rank in one, two, and

three lead and lag months. If the changes in app demand are in fact driven by the resetting, we would expect insignificant relationships between maintenance updates in response to iOS updates and app demand prior to the update (i.e., $\in \{-3, -2, -1\}$). Yet, the theory is silent with regards to the dynamic evolution of this effect over time.

To derive insights into these dynamics, we introduce Figure 4 where we plot the coefficients of maintenance updates for high-reputation (quartile 4) and low-reputation apps (quartile 1) from our 2SLS specification. For high-reputation apps the left panel of Figure 4 depicts sharply increasing and statistically significant coefficients in the month of the update as well as the following two lead months, indicating a steep decline in demand. Specifically, these apps witness the sharpest decrease in demand one month after the update (i.e., coefficient of approx. 0.69 which corresponds to an increase of 100%), in the following month the decrease in demand diminishes in magnitude (i.e., coefficient of approx. 0.52 which corresponds to an increase of 68%) but remains statistically significant, suggesting a relatively long-lived effect. The effect turns insignificant in lead month three after reputation resetting. For low-reputation apps, the right panel of Figure 4 indicates that directly in the month of the update, these apps witness a sharp and statistically significant increase in demand, reflected by a negative coefficient of approx. -0.71 and then the effect slightly decreases in the first month after the update to -0.52 (statistically significant at the 5% level). Afterwards, in the second period after the update the effect almost vanishes and turns statistically insignificant. These coefficients translate into an overall effect of a decrease in sales rank of -16% in the month of the update and turns positive for all three months after the update, suggesting, that the demand enhancing effect of reputation resetting on low-reputation apps is short-lived. This is in contrast to the long-lived effects of reputation resetting for apps with high reputation.

Reassuringly, across apps with high and low reputation, we find, by and large, statistically insignificant changes in app ranks before the reputation reset, suggesting that our effect is not driven by unobserved spurious dynamics in app rankings. As before, the F-tests for all the models estimated in Figure 4 are well above 10.

Figure 4: Dynamics of Ranks for Apps in Quartiles 1 and 4 before and after an Update

8 Robustness Checks

We conduct several checks to ensure the robustness of our results. The estimation results are included in the Appendix.

First, one might be concerned that our results are driven by the coding of our focal variables *MAINT* and *FEAT*. In particular, when catchwords for maintenance and feature updates both coincided in a release note, we categorized this update as a feature update. In a robustness check, we conduct an alternative coding where we code coinciding maintenance and feature catchwords as maintenance updates. We re-run our 2SLS regression and the results remain qualitatively unchanged, as depicted in Table A4.

Second, one might be concerned that the changes introduced by an iOS update enables new functionalities to selected apps that can leverage these functionalities to increase their sales rank.

Analyzing the content of the iOS updates⁹, we found that demand for the app “Shazam” might have been directly affected by the iOS 8 update. With iOS 8, Shazam was integrated into Siri which might have boosted the app’s sales, thus impacting its rank and the ranks of the other apps in the same category. To this end, we exclude all apps of the category “music” from our sample. Our results remain qualitatively unchanged, as depicted in column (2) of Table A5, which provides evidence that this does not threaten the robustness of our results.

Third, iOS 8 and iOS 9 introduced features such as Continuity, Spotlight, 3D Touch, and Picture in Picture. If apps leverage these newly-enabled functionalities when releasing a maintenance app in response to the iOS update, this could confound the effect of reputation resetting on sales rank. In fact, it might soften the detrimental effect for apps with high prior reputation and it might increase the beneficial effect of rating resetting for apps with low prior reputation. Again, we followed a catchword-based content analysis of the release notes to identify app updates that leverage these newly introduced features of iOS8 and 9. We used the catchwords “continuity”, “spotlight”, “3D”, “picture-in-picture” (we allowed for both, upper and lower case spelling) for the content analysis. We exclude all apps mentioning these catchwords in their release notes. Again, we find qualitatively unchanged results (column (1) of Table A5), and conclude that this potential concern does not bias our results. Along the same logic, we identify apps that rely on Apple’s functionality “Carplay”. As the release of Apple’s Carplay functionality lies in our observation period, we identify app updates mentioning this feature in their release notes. We exclude these observations from our sample and re-run our analysis and present the results in the Appendix (column (3) of Table A5) which reconfirm that our results are not biased by apps that strategically leverage this new functionality in their updates.

Fourth, even though we believe that our catchword-based update classification strategy is more suitable in our context than relying on version number changes to identify feature and maintenance updates (e.g., used by Comino et al. 2019 and Liang et al. 2019), we demonstrate that our results are not simply an artifact of our classification strategy. To that end, we derive our variables *MAINT* and *FEAT* based on

⁹ More information about the iOS 8 and iOS 9 updates can be found at <https://support.apple.com/en-us/HT208000#8> and <https://support.apple.com/en-us/HT208010#9>, respectively (accessed on 28th February 2020).

changes in version numbers. In particular, in line with Liang et al. (2019), we code a version update to be a feature update when there is a change in the second digit of the app's version number. For example, under our definition, an update from 1.8.0 to 1.9.0 is a feature update, and an update from version 2.0.0 to 2.0.1 is a maintenance update. Moreover, an update is a feature update when there is a change from 1.8.0 directly to 2.0. We run our baseline model using this classification and find that our focal 2SLS results remain qualitatively unchanged, as depicted in Table A6 of our Appendix.

Fifth, we demonstrate that our focal effect is consistent across the different chart types of free, grossing, and paid apps. Therefore, we re-estimate our 2SLS specifications separately for these three chart types, the results are depicted in Table A7 of the Appendix. Our results remain qualitatively unchanged to our baseline results shown in column (2) of Table 3. However, the significant decrease in reputation resetting for apps in quartile 4 seems to be most pronounced for grossing apps.

Finally, to alleviate concerns that our results may be driven by partitioning apps into four reputation quartiles, we split apps in only two groups (above and below the median rank). Our 2SLS baseline results remain qualitatively unchanged, as depicted in Table A8.

9 Implications and Conclusion

When a product or service being reviewed undergoes fundamental changes over time (as is the case for apps), older reviews are not quite reflective of the current product or service anymore. As a result, platforms have grappled with the problem of ensuring that the reputation system displays relevant reviews and aggregated metrics (such as review volume and valence). To that end, platforms have pursued different approaches for addressing this problem. Two widely adopted and studied approaches limit the availability of past reviews either in terms of a time window or in terms of a fixed number of recent reviews. In contrast, our paper is the first to empirically study reputation discontinuity in the form of reputation resetting after an update. The goal of this research is therefore to provide insights on the economic consequences of a unique reputation mechanism that ties app updating and reputation resetting. Building on the analytical model by Kovbasyuk and Spagnolo (2019), our study provides novel insights on the developers' and platforms' perspective regarding reputation resetting. To overcome potential endogeneity concerns and establish causality in our results we make use of an

instrumental variables approach that relies on platform updating in the form of new iOS releases. It enables us to disentangle the effect of reputation resetting from the effect of changes in app characteristics triggered by the app update. Our data set covers two major iOS releases (iOS 7 and iOS 8) and consists of 7,245 apps from the Apple App Store. In line with our expectations, we uncover an asymmetrical impact with winners and losers in this unique setting. Apps with good prior reputation suffer from platform-driven updates whereas apps with bad prior reputation benefit from them. This illustrates how reputation discontinuity can have asymmetrical effects on apps when updates have to be released to comply with the platform operating system. In contrast, the deliberate release of updates (either maintenance or feature) is associated with an increase in app demand both for high- and low-reputation apps. Surprisingly, developers of high-reputation apps seem not to align feature updates with an iOS release to mitigate the negative effects of reputation resetting.

9.1 Implications for Theory and Research

To the best of our knowledge, our study is the first to highlight that reputation resetting mechanisms give rise to a new set of pressures in management, namely the tension between an app's operational stability and its loss of reputation. While prior studies have advocated that frequent updating of apps is rather beneficial for their survival on app platforms (Pervin et al. 2019), we showcase that, with reputation resetting, there might be asymmetric consequences on app demand that need to be factored into developers' app updating decision. Recent analytical work on optimizing software updating strategies has, however, not yet incorporated this aspect (Ji et al. 2011) and not accounted for the interplay between software updating and reputation systems. Hence, we contribute empirical evidence supporting the claim to account for platform-driven reputation resetting when analyzing updating strategies. With the prior literature's call (Ba and Nault 2017; Kumar et al. 2018), our paper exemplifies again the role of reputation systems as part of software production processes.

Our study contributes empirical evidence to a research area that has been mainly supported by analytical modelling so far (e.g., Elul and Gottardi 2015; Kovbasyuk and Spagnolo 2019; Kremer et al. 2014). In particular, our results suggest that heterogeneous market dynamics postulated in these models can indeed be observed in the field. That is, we confirm the negative impact of reputation resetting on high-

reputation sellers and the positive impact of reputation resetting on low-reputation sellers. We also highlight asymmetry in the longevity of the demand impact across high- and low-reputation apps. We find that demand decreases due to reputation discontinuity for high-reputation apps are longer-lived than demand increases due to reputation discontinuity for low-reputation apps. This offers new room to refine existing analytical models to account for the asymmetry in the longevity of the demand impact after reputation resetting.

Furthermore, we focus on the seller side of the market in contrast to the previous literature in this domain that has mainly focused on studying market efficiency. In line with this, we extend existing theoretical knowledge by incorporating the sellers' ability to deliberately reset their reputation into our research framework. In fact, we do find that sellers are able to use deliberate reputation resetting to their advantage.

This study also contributes to the literature on the relationship between platforms and app developers. While there is research on how developers can use updates to boost sales (Comino et al. 2019) as well as research on the impact of platform updates on app developers (Song et al. 2018), this work combines these two streams in a novel setting. It investigates the interplay between platform-driven developer behavior and a platform-controlled reputation mechanism. Our results show that the interplay between platform updates and rules set by the platform can yield unexpected and unintended outcomes for app developers. While the platform has arguably envisioned a reputation resetting mechanism that is tied to updates so it is more relevant to customers, we find that this mechanism disproportionately harms high-reputation apps. Our study thus raises awareness for a more general class of problems, namely, when platform complementors (in our case, app developers) are simultaneously affected by the platform's rules (in our case, reputation resetting mechanism) and actions (in our case, new version of the operating system).

9.2 Practical Implications

We uncover the implications of the reputation resetting mechanism for apps that are harmful to high-reputation apps but beneficial to low-reputation apps with high-reputation apps suffering a decrease in demand after a maintenance update driven by the platform. An immediate practical implication of our

results is that platform managers need to be aware that an app-resetting policy can have disproportionate detrimental demand effects for a large share of the apps on their platform. Hence, platforms have to balance the need to keep an app's reputation reflective of the current app version and the risk of disproportionately hurting an app by deleting its entire rating history through platform-driven updates. Our findings add to the current push for building effective online review systems that account for both technological but also managerial design decisions (e.g., in our case to decide about the adequate reputation resetting approach).

Our findings also provide guidance to developers. Unlike common wisdom that updates always help improve app sales, we find that updates do not necessarily increase app demand when a reputation resetting mechanism is in place. Thus, even though such a mechanism makes intuitively sense because of the dynamic nature of app characteristics and functionalities, one must be aware of the associated asymmetric consequences on economic performance. For platforms, that implies considering the extent of changes to app characteristics in the reputation resetting mechanisms. If platforms can reliably identify product updates that only slightly affect the app inherent characteristics, it might be reasonable to refrain from resetting the respective seller's reputation. It is worthwhile noting that Apple has recently moved away from mandatory reputation resetting with each update to a setup where developers get to choose whether or not to reset their history. We take Apple's move from mandatory towards optional resetting as anecdotal confirmation of our findings that mandatory reputation resetting is hurting high-reputation apps and not low-reputation apps. Therefore, switching to an optional resetting mitigates the drawbacks for the platform's best performing apps.

9.3 Limitations and Future Research

This research exhibits several strengths including an empirical strategy that exploits platform updates as an exogenous event for reputation resetting. Furthermore, due to the unique composition of our data set, which consists of online reviews as well as release notes, we can provide empirical arguments supporting the assumptions underlying this strategy. Naturally, our work also comes with limitations that are avenues for future research. First, despite the efforts we undertake in our empirical analysis, the developer's decision to update, and thus the decision to reset one's reputation, remains an endogenous

one. Future research could analyze developer behavior qualitatively to examine the determinants of the updating decision. Second, even though we employ a textual classification of updates that has been validated by previous research, future research could use a more sophisticated textual analysis to further classify different types of updates. Third, there may be another factor, namely marketing efforts to boost app demand that we do not observe in our study. In this case, the magnitude of the found negative effect for high-reputation apps can therefore be taken as a lower bound. However, future research could study the implications of marketing efforts on the demand of low-reputation apps. Fourth, given the drawbacks of reputation resetting that we have found, future research could develop new approaches to measure the extent to which digital products change and devise novel reputation mechanisms to alleviate the disadvantages of reputation resetting.

References

- Anderson, M., and Magruder, J. 2012. "Learning from the Crowd: Regression Discontinuity Estimates of the Effects of an Online Review Database," *Economic Journal* (122:563), pp. 957-989.
- Angrist, J. D., and Pischke, J. S. 2009. *Mostly Harmless Econometrics : An Empiricist's Companion*. Princeton: Princeton University Press.
- Aperjis, C., and Johari, R. 2010. "Optimal Windows for Aggregating Ratings in Electronic Marketplaces," *Management Science* (56:5), pp. 864-880.
- Apple Developer Forums. 2017. "App Store Ratings with iOS 11 Version." Retrieved July 20, 2019, from <https://forums.developer.apple.com/thread/86921>
- Arora, S., ter Hofstede, F., and Mahajan, V. 2017. "The Implications of Offering Free Versions for the Performance of Paid Mobile Apps," *Journal of Marketing* (81:6), pp. 62-78.
- Ba, S., and Nault, B. R. 2017. "Emergent Themes in the Interface between Economics of Information Systems and Management of Technology," *Production & Operations Management* (26:4), pp. 652-666.
- Babić Rosario, A., Sotgiu, F., De Valck, K., and Bijmolt, T. H. A. 2016. "The Effect of Electronic Word of Mouth on Sales: A Meta-Analytic Review of Platform, Product, and Metric Factors," *Journal of Marketing Research* (53:3), pp. 297-229.
- Banker, R. D., Davis, G. B., and Slaughter, S. A. 1998. "Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study," *Management Science* (44:4), pp. 433-450.
- Barry, E., Kemerer, C., and Slaughter, S. 1999. "Toward a Detailed Classification Scheme for Software Maintenance Activities," *Americas Conference on Information Systems*, Milwaukee, WI, pp. 726-728.
- Bishop, T. 2015. "Amazon Changes Its Key Formula for Calculating Product Ratings and Displaying Reviews." Retrieved July 20, 2019, from <https://www.geekwire.com/2015/amazon-changes-its-influential-formula-for-calculating-product-ratings/>
- Brynjolfsson, E., Hu, Y. J., and Smith, M. D. 2003. "Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers," *Management Science* (49:11), pp. 1580-1596.
- Cheung, C. M. K., and Thadani, D. R. 2012. "The Impact of Electronic Word-of-Mouth Communication: A Literature Analysis and Integrative Model," *Decision Support Systems* (54:1), pp. 461-470.
- Chevalier, J. A., Dover, Y., and Mayzlin, D. 2018. "Channels of Impact: User Reviews When Quality Is Dynamic and Managers Respond," *Marketing Science* (37:5), pp. 688-709.
- Chevalier, J. A., and Mayzlin, D. 2006. "The Effect of Word of Mouth on Sales: Online Book Reviews," *Journal of Marketing Research* (43:3), pp. 345-354-354.

- Clemons, E. K., Gao, G. G., and Hitt, L. M. 2006. "When Online Reviews Meet Hyper-Differentiation: A Study of the Craft Beer Industry.," *Journal of Management Information Systems* (23:2), pp. 149-171.
- Comino, S., Manenti, F. M., and Mariuzzo, F. 2019. "Updates Management in Mobile Applications: iTunes Vs Google Play," *Journal of Economics & Management Strategy* (28:3), pp. 392-419.
- Dellarocas, C. 2005. "Reputation Mechanism Design in Online Trading Environments with Pure Moral Hazard," *Information Systems Research* (16:2), pp. 209-230.
- Dellarocas, C. 2006a. "How Often Should Reputation Mechanisms Update a Trader's Reputation Profile?," *Information Systems Research* (17:3), pp. 271-285.
- Dellarocas, C. 2006b. "Strategic Manipulation of Internet Opinion Forums: Implications for Consumers and Firms," *Management Science* (52:10), pp. 1577-1593.
- Dewan, S., and Hsu, V. 2004. "Adverse Selection in Electronic Markets: Evidence from Online Stamp Auctions," *Journal of Industrial Economics* (52:4), pp. 497-516.
- Dillet, R. 2017. *Techcrunch*, Retrieved July 20, 2019, from <https://techcrunch.com/2017/06/07/ios-app-developers-will-be-able-to-keep-app-store-ratings-when-releasing-an-update/>
- Doğan, K., Ji, Y., Mookerjee, V. S., and Radhakrishnan, S. 2011. "Managing the Versions of a Software Product under Variable and Endogenous Demand," *Information Systems Research* (22:1), pp. 5-21.
- Duan, W., Gu, B., and Whinston, A. B. 2008. "Do Online Reviews Matter? -- an Empirical Investigation of Panel Data," *Decision Support Systems* (45:4), pp. 1007-1016-1016.
- Elul, R., and Gottardi, P. 2015. "Bankruptcy: Is It Enough to Forgive or Must We Also Forget?," *American Economic Journal* (7:4), pp. 294-338.
- Fleischmann, M., Amirpur, M., Grupp, T., Benlian, A., and Hess, T. 2016. "The Role of Software Updates in Information Systems Continuance -- an Experimental Study from a User Perspective," *Decision Support Systems* (83), pp. 83-96.
- Floyd, K., Freling, R., Alhoqail, S., Cho, H. Y., and Freling, T. 2014. "How Online Product Reviews Affect Retail Sales: A Meta-Analysis," *Journal of Retailing* (90:2), pp. 217-232.
- Förderer, J. 2020. "Interfirm Exchange and Innovation in Platform Ecosystems: Evidence from Apple's World Wide Developers Conference," *Management Science* (Articles in Advance).
- Förderer, J., and Heinzl, A. 2017. "Product Updates: Attracting New Consumers Versus Alienating Existing Consumers," *International Conference on Information Systems*, Seoul, Korea: AIS.
- Gao, G., Greenwood, B. N., Agarwal, R., and McCullough, J. S. 2015. "Vocal Minority and Silent Majority: How Do Online Ratings Reflect Population Perceptions of Quality," *MIS Quarterly* (39:3), pp. 565-590.
- Garg, R., and Telang, R. 2013. "Inferring App Demand from Publicly Available Data," *MIS Quarterly* (37:4), pp. 1253-1264.

- Ghose, A., and Han, S. P. 2014. "Estimating Demand for Mobile Applications in the New Economy," *Management Science* (60:6), pp. 1470-1488.
- Gu, B., and Ye, Q. 2014. "First Step in Social Media: Measuring the Influence of Online Management Responses on Customer Satisfaction," *Production & Operations Management* (23:4), pp. 570-582.
- Gutt, D., Neumann, J., Zimmermann, S., Kundisch, D., and Chen, J. 2019a. "Design of Review Systems – a Strategic Instrument to Shape Online Reviewing Behavior and Economic Outcomes," *The Journal of Strategic Information Systems* (28:2), pp. 104-117.
- Gutt, D., Herrmann, P., and Rahman, M. S. 2019b. "Crowd-driven Competitive Intelligence: Understanding the Relationship between Local Market Competition and Online Rating Distributions," *Information Systems Research* (30:3), pp. 980-994.
- Hainzinger, B. 2020. "iOS Developer Changes in iOS 13 You Missed." *App Developer Magazine* Retrieved Feb 20, 2020, from <https://appdeveloper magazine.com/ios-developer-changes-in-ios-13-you-missed/>
- Hong, H., Xu, D., Wang, G. A., and Fan, W. 2017. "Understanding the Determinants of Online Review Helpfulness: A Meta-Analytic Investigation," *Decision Support Systems* (102), pp. 1–11.
- Imbens, G. W. 2010. "Better Late Than Nothing: Some Comments on Deaton (2009) and Heckman and Urzua (2009)," *Journal of Economic Literature* (48:2), pp. 399-423.
- Ji, Y., Kumar, S., Mookerjee, V. S., Sethi, S. P., and Yeh, D. 2011. "Optimal Enhancement and Lifetime of Software Systems: A Control Theoretic Analysis," *Production & Operations Management* (20:6), pp. 889-904.
- Kemerer, C. F., and Slaughter, S. 1999. "An Empirical Approach to Studying Software Evolution," *IEEE Transactions on Software Engineering* (25:4), pp. 493-509.
- Kovbasyuk, S., and Spagnolo, G. 2019. "Memory and Markets." under review at Review of Economic Studies.
- Kremer, I., Mansour, Y., and Perry, M. 2014. "Implementing the "Wisdom of the Crowd"," *Journal of Political Economy* (122:5), pp. 988-1012.
- Kumar, S., Mookerjee, V., and Shubham, A. 2018. "Research in Operations Management and Information Systems Interface," *Production & Operations Management* (27:11), pp. 1893-1905.
- Lee, G., and Raghu, T. S. 2014. "Determinants of Mobile Apps' Success: Evidence from the App Store Market," *Journal of Management Information Systems* (31:2), pp. 133-169.
- Liang, C., Shi, Z., and Raghu, T. S. 2019. "The Spillover of Spotlight: Platform Recommendation in the Mobile App Market," *Information Systems Research* (30:4), pp. 1296-1318.
- Liu, C. Z., Au, Y. A., and Choi, H. S. 2014. "Effects of Freemium Strategy in the Mobile App Market: An Empirical Study of Google Play," *Journal of Management Information Systems* (31:3), pp. 326-354.

- Lombard, M., Snyder-Duch, J., and Campanella Bracken, C. 2002. "Content Analysis in Mass Communication," *Human Communication Research* (28:4), pp. 587-604.
- Luca, M. 2017. "Designing Online Marketplaces: Trust and Reputation Mechanisms," *Innovation Policy and the Economy* (17:1), pp. 77-93.
- Manes, E., and Tchetchik, A. "The Role of Electronic Word of Mouth in Reducing Information Asymmetry: An Empirical Investigation of Online Hotel Booking," *Journal of Business Research* (85), pp. 185-196.
- Pervin, N., Ramasubbu, N., and Dutta, K. 2019. "Habitat Traps in Mobile Platform Ecosystems," *Production & Operations Management* (28:10), pp. 2594-2608.
- Rice, S. C. 2012. "Reputation and Uncertainty in Online Markets: An Experimental Study," *Information Systems Research* (23:2), pp. 436-452.
- Schoenmüller, V., Netzer, O., and Stahl, F. 2019. "The Extreme Distribution of Online Reviews: Prevalence, Drivers and Implications." Columbia Business School.
- Scott, J. 2012. "Stateside: Getting to grips with App Annie's App Store stats." Retrieved July 20, 2019, from <https://www.pocketgamer.biz/feature/47064/stateside-getting-to-grips-with-app-annies-app-store-stats/>
- Sharma, S. 2020. "Top 10 Challenges Faced by iOS Developers While Developing iPhone Apps in 2020." *Insights* Retrieved Feb 20, 2020, from <https://www.credencys.com/blog/challenges-faced-by-ios-developers/>
- Song, P., Xue, L., Rai, A., and Zhang, C. 2018. "The Ecosystem of Software Platform: A Study of Asymmetric Cross-Side Network Effects and Platform Governance," *MIS Quarterly* (42:1), pp. 121-142.
- Stock, J. H., and Yogo, M. I. 2005. "Testing for Weak Instruments in Linear IV Regression," in *Identification and Inference in Econometric Models: Essays in Honor of Thomas J. Rothenberg*, D.W.K. Andrews and J.H. Stock (eds.). Cambridge, UK: Cambridge University Press.
- Sun, H., and Xu, L. 2018. "Online Reviews and Collaborative Service Provision: A Signal-Jamming Model," *Production & Operations Management* (27:11), pp. 1960-1977.
- Tadelis, S. 2016. "Reputation and Feedback Systems in Online Platform Markets," *Annual Review of Economics* (8), pp. 321-340.
- Telang, R., and Wattal, S. 2007. "An Empirical Analysis of the Impact of Software Vulnerability Announcements on Firm Stock Price," *IEEE Transactions on Software Engineering* (33:8), pp. 544-557.
- Tiwana, A. 2015. "Evolutionary Competition in Platform Ecosystems," *Information Systems Research* (26:2), pp. 266-281.
- Wang, Y., Goes, P., Wei, Z., and Zeng, D. 2019. "Production of Online Word-of-Mouth: Peer Effects and the Moderation of User Characteristics," *Production & Operations Management* (28:7), pp. 1621-1640.

- Ya, Y., Vadakkepatt, G. G., and Joshi, A. M. 2015. "A Meta-Analysis of Electronic Word-of-Mouth Elasticity," *Journal of Marketing* (79:2), pp. 19-39.
- Zervas, G., Proserpio, D., and Byers, J. W. 2017. "The Rise of the Sharing Economy: Estimating the Impact of Airbnb on the Hotel Industry," *Journal of Marketing Research (JMR)* (54:5), pp. 687-705.

Appendix

Table A2: List of App Categories with their respective Number of Apps user in our Sample

Category	Number of Apps
Books	239
Business	488
Catalogs	443
Education	447
Entertainment	513
Finance	525
Food and Drink	480
Games	567
Health and Fitness	346
Lifestyle	553
Medical	329
Music	362
Navigation	244
News	328
Newsstand	189
Photo and Video	338
Productivity	238
Reference	206
Shopping	163
Social Networking	225
Sports	164
Travel	227
Utilities	296
Weather	93
TOTAL	8,003

Note: The sum of the number of apps in the different app categories (8,003) exceeds the number of apps (7,245) because one unique app can be in multiple categories.

Table A3: Results for Feature Updates

	(1) 2SLS + FE – Feature & Maintenance together	(2) 2SLS + FE – Only Feature
VARIABLES	<i>LN(RANK)</i>	<i>LN(RANK)</i>
<i>FEAT</i>	-0.0080	-0.92196**
	(0.40253)	(0.40437)
<i>FEAT * QUART₁</i>	-1.90744*	2.71335***
	(1.04691)	(1.02755)
<i>FEAT * QUART₂</i>	-0.06189	0.65765
	(0.5116)	(0.77649)
<i>FEAT * QUART₃</i>	-0.28285	0.61082
	(0.43927)	(0.57283)
<i>QUART₁</i>	0.41147***	-0.05458
	(0.10729)	(0.09976)
<i>QUART₂</i>	0.17140**	0.0604
	(0.07302)	(0.08834)
<i>QUART₃</i>	0.13719*	0.00581
	(0.07688)	(0.08073)
Control Variabels	✓	✓
Unique App FE	✓	✓
Half-Year FE	✓	✓
Observations	197,029	197,029
Groups	21,679	21,679
First Stage F Test	3	8

Note: *** p<0.01, ** p<0.05, * p<0.1, Robust standard errors clustered on the app level in parentheses. A group is represented by a unique app in a given period (March 2014 to February 2015 or March 2015 to December 2015).

Table A4: Results for Recoding of Updates to Maintenance when Maintenance and Feature Update Catchwords are Found in Release Notes

	(1) FE	(2) 2SLS + FE
VARIABLES	$LN(RANK)$	$LN(RANK)$
MAINT	0.00362	0.15907**
	(0.00819)	(0.08018)
MAINT * QUART₁	-0.00371	-0.62250***
	(0.01426)	(0.20631)
MAINT * QUART₂	-0.00242	0.0328
	(0.01287)	(0.25298)
MAINT * QUART₃	-0.01709	-0.13338
	(0.0123)	(0.20924)
QUART₁	0.22766***	0.33417***
	(0.01426)	(0.04027)
QUART₂	0.17018***	0.17238***
	(0.01258)	(0.0474)
QUART₃	0.10376***	0.12633***
	(0.00982)	(0.04091)
Control Variabels	✓	✓
Unique App FE	✓	✓
Half-Year FE	✓	✓
Observations	197,813	197,029
Groups	22,463	21,679
First Stage F Test	-	44

Note: *** p<0.01, ** p<0.05, * p<0.1, Robust standard errors clustered on the app level in parentheses. A group is represented by a unique app in a given period (March 2014 to February 2015 or March 2015 to December 2015).

Table A5: 2SLS + FE Results Excluding Updates Mentioning Particular Functionalities

	(1) No iOS8 & iOS9 Functionalities	(2) No Music Category	(3) No Carplay
VARIABLES	$LN(RANK)$	$LN(RANK)$	$LN(RANK)$
MAINT	0.31346**	0.33375**	0.30807**
	(0.14122)	(0.14798)	(0.14258)
MAINT * QUART₁	-0.72948***	-0.71724***	-0.71506***
	(0.22752)	(0.23304)	(0.22541)
MAINT * QUART₂	0.00821	0.01411	-0.00436
	(0.26418)	(0.27613)	(0.25874)
MAINT * QUART₃	-0.13958	-0.10688	-0.15706
	(0.20748)	(0.22265)	(0.20896)
QUART₁	0.34677***	0.34709***	0.34496***
	(0.04241)	(0.04346)	(0.04212)
QUART₂	0.17427***	0.17674***	0.17605***
	(0.04848)	(0.05084)	(0.0476)
QUART₃	0.12481***	0.12220***	0.12851***
	(0.03971)	(0.04266)	(0.04006)
Control Variabels	✓	✓	✓
Unique App FE	✓	✓	✓
Half-Year FE	✓	✓	✓
Observations	195,873	187,940	197,021
Groups	21,679	20,711	21,679
First Stage F Test	44	43	47

Note: *** p<0.01, ** p<0.05, * p<0.1, Robust standard errors clustered on the app level in parentheses. A group is represented by a unique app in a given period (March 2014 to February 2015 or March 2015 to December 2015).

Table A6: Results for Alternative Definition of Maintenance and Feature Updates Based on the Version Number of the Update

	(1) FE	(2) 2SLS + FE
VARIABLES	$LN(RANK)$	$LN(RANK)$
MAINT	-0.03579***	0.52865**
	(0.01088)	(0.22818)
MAINT * QUART₁	0.06441***	-1.31041***
	(0.01626)	(0.43138)
MAINT * QUART₂	0.04144***	0.05289
	(0.01554)	(0.49491)
MAINT * QUART₃	0.03326**	-0.33247
	(0.01441)	(0.30637)
QUART₁	0.21581***	0.42654***
	(0.01407)	(0.07223)
QUART₂	0.16203***	0.17227**
	(0.01237)	(0.08258)
QUART₃	0.09438***	0.16113***
	(0.00981)	(0.05865)
Control Variabels	✓	✓
Unique App FE	✓	✓
Half-Year FE	✓	✓
Observations	197,813	197,029
Groups	22,463	21,679
First Stage F Test	-	12

Note: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$, Robust standard errors clustered on the app level in parentheses. A group is represented by a unique app in a given period (March 2014 to February 2015 or March 2015 to December 2015).

The only difference with our baseline results in Table 3 in the main body of the paper is that in Column (1), the coefficient of $MAINT * QUART_1$ is statistically significant, positive, and larger than the coefficient of $MAINT$, suggesting that deliberate maintenance updates are detrimental to the sales ranks of apps in quartile 1. In fact, this might point to the fact that apps in quartile 1 have to release maintenance updates when customers are experiencing problems with the app, which might decrease their sales rank. This endogeneity concern underscores even more the need to use the exogenous variation of our instrument to estimate the differential relationship of maintenance updates for apps in quartile 1 and quartile 4.

Table A7: 2SLS + FE Results for Different Chart Types (Free, Grossing, Paid) Separately

	(1) Free	(2) Grossing	(3) Paid
VARIABLES	<i>LN(RANK)</i>	<i>LN(RANK)</i>	<i>LN(RANK)</i>
MAINT	0.10939	0.42207**	0.68431
	(0.14974)	(0.21107)	(0.44537)
MAINT * QUART₁	-0.58611***	-0.56545*	-1.32665*
	(0.20994)	(0.32472)	(0.72144)
MAINT * QUART₂	-0.1352	0.15647	0.01052
	(0.24921)	(0.37261)	(0.79278)
MAINT * QUART₃	0.22082	-0.58456*	-0.50788
	(0.21281)	(0.32102)	(0.64597)
QUART₁	0.38281***	0.28157***	0.31774***
	(0.04573)	(0.05935)	(0.09484)
QUART₂	0.23300***	0.12192*	0.13538
	(0.05325)	(0.06518)	(0.10525)
QUART₃	0.07558	0.17997***	0.13468
	(0.04768)	(0.05727)	(0.08413)
Control Variabels	✓	✓	✓
Unique App FE	✓	✓	✓
Half-Year FE	✓	✓	✓
Observations	83,210	73,539	40,280
Groups	9,211	8,079	4,389
First Stage F Test	18	20	8

Note: *** p<0.01, ** p<0.05, * p<0.1, Robust standard errors clustered on the app level in parentheses. A group is represented by a unique app in a given period (March 2014 to February 2015 or March 2015 to December 2015).

Table A8: Results for Maintenance Updates and Median Split

	(1) FE	(2) 2SLS + FE
VARIABLES	<i>LN(RANK)</i>	<i>LN(RANK)</i>
<i>MAINT</i>	-0.01747**	0.23444**
	(0.00727)	(0.11048)
<i>MAINT * LOWREP</i>	0.01829*	-0.37860**
	(0.01072)	(0.17874)
<i>LOWREP</i>	0.09280***	0.16114***
	(0.00818)	(0.03215)
Control Variabels	✓	✓
Unique App FE	✓	✓
Half-Year FE	✓	✓
Observations	197,813	197,029
Groups	22,463	21,679
First Stage F Test	-	47

Note: *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$, Robust standard errors clustered on the app level in parentheses. A group is represented by a unique app in a given period (March 2014 to February 2015 or March 2015 to December 2015). In this median split, the base case is again the high-reputation apps (i.e., all apps above the median rank). LOWREP is a dummy variable that identifies apps with reputation below the median reputation in the sample.