CESifo
AREA CONFERENCES 2019

Economics of Digitization

Munich, 22–23 November 2019

There's an App (Update) for That

Understanding Product Updating Under Digitization

Benjamin T. Leyden

20 YEARS CESifo

There's an App (Update) for That

Understanding Product Updating Under Digitization*

Benjamin T. Leyden
Dyson School of Applied Economics and Management
Cornell University
leyden@cornell.edu †

March 3, 2019

The latest version of this paper is available at benleyden.com/app-updating.

Abstract

The digitization of consumer goods gives firms the ability to monetize and update already purchased products, changing firms' product innovation incentives. I develop and estimate a structural model of the smartphone application (app) industry to study how the availability of these tools affects the frequency and content of product updates. Using data from Apple's mobile platform, I employ natural language processing and machine learning techniques to classify product updates and define precise categorical markets. I find that the availability of these tools via digitization generally result in an increase in the frequency of product updates, and, in particular, an increase in the relative frequency of major, feature-adding updates compared to more minor, incremental updates. These results show that the manner in which product digitization changes firms' product innovation incentives has a significant effect on firm behavior, and should be accounted for in future research on digital and digitizing industries.

JEL Classifications: L13, L15, L86, O30
Keywords: Product innovation, endogenous product characteristics, dynamic oligopoly, software, digitization

*Portions of this paper were previously circulated under the title "Consumer Preferences for Product Updates Under Digitization: A Model of Demand for Smartphone Applications."

†I am grateful for guidance from Simon Anderson, Federico Ciliberto, Gaurab Aryal, and many other faculty members and students at the University of Virginia. This work has benefited from comments at IIOC 2017 & 2018, the 2017 ZEW/MaCCI Conference on Innovation and Patenting, the 2017 Searle Center Conference on Internet Commerce and Innovation, EARIE 2017, and NABE TEC 2018, and the HBS Strategy unit. I am indebted to the app developers who have spoken with me about their industry, and, in a some cases, provided data that have been essential to this project. I acknowledge the Bankard Fund for Political Economy and the Radulovacki Summer Research Fund for financial support. All errors are my own.

1 Introduction

> *"It's about post-purchase monetization of the TV... we're continuing to invest in those older TVs to bring them up to feature level comparison with the new TVs... and the reason why we do that is there are ways to monetize that TV."*
>
> — Bill Baxter, *CTO, Vizio*[1]

Consumer goods are becoming increasingly software-dependent and internet-connected. This *digitization* of consumer goods changes the incentives firms face regarding when and how to engage in product innovation. This is because digitization has the potential to extend the interaction between consumers and firms, which has typically ended at the point of sale, by providing firms with two new strategic tools. First, under digitization, firms are able to monetize the use of their products, for example, by including advertisements in the product or by selling additional, instantly available features to consumers. Second, the digital nature of these products makes it possible for firms to update a product after the consumer has purchased it. Together, these two changes—the ability to monetize the *use* of a product, and the ability to update a product past the point of sale—have the potential to change how firms choose to innovate their products.

I study how the availability of use monetization and post-purchase updating via digitization affects product innovation. Specifically, I consider whether product updates are more or less frequent, and how the content of updates changes when firms have access to these tools. I answer these questions through an empirical analysis of consumer and firm behavior in the context of smartphone application (app) markets on Apple's App Store marketplace.

The digitization of consumer durable good industries is rapidly accelerating. Of particular note is a growing collection of home appliances and gadgets often referred to as the "Internet of Things," due to their incorporation and reliance on internet-connected software. Digitization is also evident in the car industry, where electronic car manufacturer Tesla has repeatedly released downloadable updates that increase the quality and functionality of their vehicles. For example, in January of 2015, Tesla released an update that increased acceleration rate of their model P85D car, and in June of 2017, they released an update that improved the semi-autonomous driving capabilities of

[1] Patel (2019)

Tesla cars (Matthews, 2015; Lambert, 2017). It is anticipated that many more product categories will become increasingly digitized in the coming years.

Apps are an extreme case of a digitized durable good as they are fully digital. This provides a unique opportunity to study the effects of the changes brought on by digitization without facing additional modeling and empirical challenges brought forth by the specifics of a particular industry's non-digital production costs and technologies. And, while software has always been a digital product, the modern app industry differs from the more traditional consumer software developed over the last few decades. Modern apps have undergone a process of digitization, as defined in this paper, similar to many other durable good industries. Software, especially on mobile devices, is now nearly always connected to the internet, and with the increase in the speed and ubiquity of the internet, modern app developers are able to take advantage of the changes of digitization as defined above. To be clear, no single characteristic of the modern app industry is completely new, but rather, it is the confluence of these changes that distinguishes modern app stores from traditional software.

Additionally, app markets are an interesting industry in their own right. Modern app markets began in 2008 with the introduction of Apple's App Store, which was quickly followed by Google's Android Market, a precursor to what is now called Google Play.[2] The number of apps in the App Store has ballooned from 500 in 2008 to over two million, and, according to Apple, apps have been downloaded from the App Store over 130 billion times, resulting in over $100 billion in revenue (Apple, 2016, 2017).[3]

I approach the question of how the introduction of use monetization and post-purchase updating to consumer goods via digitization has affected firms' product innovation, or product updating, behavior through the lens of competition in durable goods. Digitization has the potential to change how firms compete by shifting a firm's focus from encouraging consumers to replace existing products toward encouraging consumers to continue to use existing products. In the standard durable goods model, a firm competes first against other firms to make a sale, and then against itself, as firms update their products in order to get the consumer to purchase a replacement product. A risk associated with this second aspect of competition is that if a consumer returns to the store,

[2]Henceforth, any mention of the "App Store" is a reference to Apple's app marketplace.

[3]In a mid-2017 press release, Apple announced that nearly $70 billion had been given to developers (Apple, 2017). As noted in Section 2, developers keep 70% of the revenue their app collects.

that consumer may switch to a competitor's product. The digitization of consumer goods shifts the emphasis in this game away from trying to push the consumer back to the store, and instead toward trying to keep the consumer engaged with their current product through product updates, while earning a continued revenue stream through various forms of use monetization.

The question of how digitization affects the frequency and content of product updates is an inherently empirical one. It is theoretically ambiguous whether the ability to monetize the use of a product and to update a product after the point of sale will lead to increased or decreased updating, as the effect depends on the relative response to updates by potential new purchasers and by previous purchasers choosing whether or not to use the product. For example, Foerderer and Heinzl (2017) find that while updates increase demand, existing users tend to react negatively (as measured by consumer reviews) to updates. This might result in less frequent updating, to the extent that apps rely on advertising or other use-based sources of revenue. Furthermore, it is also unclear how the content of updates will change as a result of digitization. If consumers are relatively responsive to updates, but not to the content of those updates, firms might tend to produce more frequent but lower quality updates. On the other hand, if consumer behavior is particularly attuned to the content of product updates, firms may instead offer less frequent, but higher quality updates.

With this framework in mind, I develop and estimate a structural model of the demand for apps and of developers' dynamic product updating decisions. On the demand side, I make a distinction between the extensive margin—consumers choosing to buy new apps—and the intensive margin—consumers choosing whether to use their already owned apps. In the model, consumers consider the expected future value of owning an app when making extensive-margin, purchase decisions. On the intensive margin, consumers make period-by-period decisions of whether to use their already-purchased apps or not. Using the model of extensive-margin demand, I estimate consumers' preferences over app updates.

This paper differs from other recent empirical work on the demand for durable goods as it allows for quality changes to an existing product. That is, instead of modeling consumers' dynamic product switching decisions, as in Gowrisankaran and Rysman (2007), I model consumers' purchase decisions in light of their expectations over future product updates and the associated future utility they will receive. In my model, the inherent quality or characteristics of a product change following

its developer's decision to update, unlike, for example, in Lee (2013)'s model of demand for video game software, where a product's inherent quality is fixed, and period-to-period changes in the lifetime expected utility of a purchase occur only as a result of changes in the composition of available products and temporal preference changes.

On the supply side, I model developers' dynamic product updating decisions. Developers choose each period whether, and if so, how, to update their app while considering the impact an update will have on future demand and on the behavior of their market-level competitors. Using this model I estimate the fixed costs of producing small, bug-fixing updates, and larger, feature-adding updates. I then conduct a counterfactual analysis in which I "turn off" the digital aspects of the industry and simulate developers' updating decisions. By comparing developers' updating decisions without use monetization and post-purchase updating to observed behavior, I am able to estimate the extent to which digitization has changed developers' updating behaviors, both in terms of the frequency and content of updates.

This work contributes to a growing literature on endogenous product characteristics. While much of this work has focused on static models (see, e.g., Eizenberg (2014); Fan (2013)), there has been some work on dynamic endogenous product characteristic decisions, including Goettler and Gordon (2011), who model innovation in the CPU manufacturing industry, and Sweeting (2013). Sweeting (2013)'s model of radio station format changes is closely related to this model developed in this paper. As in this paper, Sweeting estimates a dynamic model of firms making discrete choices from a menu of options. Additionally, in both cases, firms face somewhat atypical profit functions, where marginal costs are zero and revenue does not result (at least entirely) from consumers' purchase behavior. Unlike Sweeting (2013), though, consumers in my model exhibit forward-looking behavior.

To conduct my analysis, I have created a dataset on the universe of apps on Apple's mobile platform. For both economic and computational reasons, I focus my analysis on a subset of apps in the Productivity category of the App Store, which consists of note-taking apps, task-management apps, and virtual private network apps, among other categories.[4] This data has been collected from

[4]One reason for focusing on Productivity apps is that many other categories are largely made up of *companion* apps, which exist only to complement other products or services. By studying Productivity apps I am able to focus my attention on apps where the app itself is the product, and thus where we would expect to see profit-maximizing behavior from the developers. See Appendix A for a more detailed discussion of this point.

a variety of sources, and includes publicly available information from the App Store, and, in some cases, proprietary sales and revenue data has been collected directly from app developers.

While the app industry is a data-rich environment, much of the information on apps is in the form of unstructured text. This paper contributes to a growing body of work that integrates text data into economic analysis.[5] In particular, I observe text descriptions of every app and every app update. I use machine learning (ML) and natural language processing (NLP) techniques to process these texts for use in estimating my model.

First, I use text descriptions of every app update, called release notes to classify every update as either a Bug Fix update, which makes incremental changes, or a Feature update, which adds new features or functionality. To do so, I classified a set of release notes by hand, which are then used to train a support-vector machine (SVM). The SVM is able to process every update's release notes and determine whether that update is a Bug Fix or Feature update. This distinction serves as a proxy for the quality of an update, and allows my model to account for the heterogeneity in updates.

Second, I develop a method for defining precise categorical markets by again applying NLP and ML techniques to the text app descriptions. This approach is similar to that of Hoberg and Phillips (2016), who use the text provided in annual SEC reports to define annualized product markets. In this case, I map the text descriptions to a vector space, and then identify apps by clustering the descriptions in that space using the X-Means clustering algorithm, a form of unsupervised machine learning. This approach should prove useful for other researchers studying long-tail product industries, such as online retailers, or peer-to-peer transaction markets such as Craigslist.[6]

I find that digital product updates increase extensive-margin, purchase demand, but that consumers do not appear to differentiate based on the content of updates. That is, there is no statistical evidence that consumers are more responsive to Feature updates than Bug Fix updates. While the exact mechanism behind this result is unclear, it suggests that consumers may be poorly informed

[5] E.g., Baker, Bloom, and Davis (2016) develop a new index of economic policy uncertainty using the text of news reports, Wu (2017) analyzes sexism on a popular Economics message board website, and Aryal, Ciliberto, and Leyden (2019) find evidence of tacit collusion in the airline industry using earnings call transcripts. Gentzkow, Kelly, and Taddy (Forthcoming) provide a broad overview of economics research that uses text as data.

[6] This approach generalizes to other forms of descriptive information on products. E.g., product photographs, which are collectable from most online marketplaces, could be used.

about the quality of digital goods prior to purchase. On the supply side, I estimate the fixed costs of production for Feature and Bug Fix updates and find that Feature updates are 28% more expensive to produce than Bug Fix updates. These results, that consumers are not more responsive to Feature updates, and that Feature updates are more costly to produce, coupled with the fact that 44% of the updates in my sample are Feature updates, presents an apparent contradiction, and points to the need for future work understanding consumer behavior on the intensive, or product-use, margin of demand. In Section 6.3, I discuss the role of the intensive margin of demand, where consumers are choosing whether to use previously purchased apps each period, in explaining this apparent contradiction.

Finally, using counterfactual simulations where I simulate developer behavior in the absence of digitization, I find that the availability of use monetization and post-purchase updating generally increases the frequency of updates and the likelihood that a given update is a feature-adding update, as opposed to a bug-fixing, incremental update. These results suggest that we should expect not only to see an increased rate of product updating in many other consumer goods industries, but also an increase in the quality of these updates as quality competition among firms accelerates as a result of digitization.

This paper proceeds as follows: In Section 2, I provide an overview of Apple's App Store marketplace and of the current state of research on the app industry. Given that background, I develop a model of consumer and firm behavior for the app industry in Section 3. Section 4 outlines the data I use to estimate this model, the data processing steps that must be taken in order to estimate the model, and provides preliminary evidence of the effect of digitization on product updating. Section 5 outlines the estimation procedure for both the demand and supply models, and Section 6 discusses the results of that estimation. Finally, in Section 7, I analyze how developers' updating behavior changes under digitization by considering counterfactual simulations that "turn off" the digital aspects of the industry. Section 8 concludes.

2 The App Store

The mobile application (app) industry is characterized by a small number of distinct platforms, or mobile operating systems. In 2015 and 2016, the most prominent platforms in the United States

were Apple's iOS, Google's Android, Microsoft's Windows, and Blackberry. Apple served 43.3% of the U.S. mobile platform market as of March of 2016 (ComScore, 2016).

The industry is a textbook example of a multi-sided market. Consumers purchase hardware, which is exclusively tied to one platform.[7] By purchasing the appropriate hardware, a consumer gains access to the platform's store, which is the primary (and in some cases only) way to purchase apps for use on the device. Consumers in this market generally single-home, though some consumers purchase a phone associated with one platform and a tablet associated with another. For the remainder of this paper, I limit discussion to Apple's mobile platform, called iOS, and its exclusive marketplace, the App Store. While the discussion below is specific to Apple, the general structure of the platform is applicable to the other mobile platforms.

2.1 Apple's iOS Platform

Apple's App Store marketplace is the only approved platform for selling apps for its mobile platform.[8] Developers submit their app to the App Store, and, once approved by Apple, the app is made available to consumers. This approval process ensures that the submitted product conforms to Apple's platform policies. During the sample period this paper considers, this process took approximately a week, though that time has since shortened substantially. This approval process is applied to all new apps, and to all updates to previously approved apps.

Developers set the price of their app, and are able to change that price at any time. Developers are restricted to a finite, discrete set of 94 possible prices. Apps can be free, or priced as low as $0.99 and as high as $999.99. Apple collects 30% of all sales revenue, which is similar to other platforms (Spencer, 2015).[9] Figs. 1a and 1b show a snapshot of the distribution of prices in the App Store.

In addition to setting a price, developers can choose to include advertisements and/or in-app purchases (IAPs). IAPs are options within an app to buy additional content or functionality. As an example, the task management app OmniFocus offers an IAP for the "Pro" version of the product,

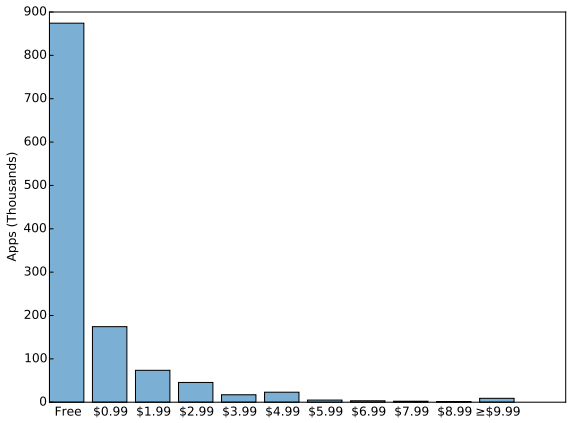
[7] Each platform runs on a corresponding set of hardware devices, primarily mobile phones and tablet computers, though some also run on other devices such as televisions.

[8] Apps can be acquired illegally through online piracy. Piracy is relatively limited in this industry, on the iOS platform, and is not considered here.

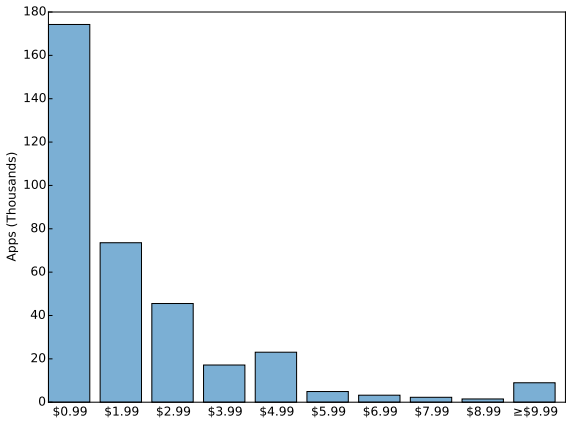
[9] The one exception to this policy, introduced in September 2016, reduces Apple's share of subscription revenue to 15% on all subscriptions older than a year (Goode, 2016). This change occurred after the sample period considered in this paper.

Figure 1: App Store Prices

(a) Distribution App Store Prices (12/31/2014)



(b) Distribution of App Store Prices, Excluding Free Apps (12/31/2014)



which includes additional task organization capabilities. As with app sales revenue, Apple collects 30% of all IAP revenues. Apps are also allowed to offer subscriptions through Apple's payment system or through other means (e.g., a website), but any subscription bought *within* the app must be purchased through Apple's system.

In the early years of the App Store, many developers offered two versions of the same product, a free and a "paid" version. The free version would either be limited in some way, or would contain advertisements. This was, in general, either an attempt to engage in price discrimination or an attempt to create a rudimentary "Free Trial" option for users. This practice has largely fallen out of fashion, particularly among non-game apps.

App production is primarily characterized by the fixed costs associated with designing and programming the app. In almost all cases, app developers face zero marginal costs when selling their app. This is both a function of the digital nature of these products, and the fact that Apple handles all of the "back-end" requirements of selling an app: managing the computer servers that host the files that consumers download, processing all transactions on the platform, and managing the (digital) storefront.

2.2 Existing Research on Smartphone Applications

This paper contributes to a small, but growing literature on smartphone apps. Early work was largely descriptive, but research has since expanded to studying cross-platform entry decisions

(Bresnahan, Orsini, and Yin, 2014; Liu, 2017; Nekipelov, Park, and Liu, 2013), consumer search (Ershov (2018)), firm strategy (Bresnahan, Li, and Yin (2016), Davis, Muzyrya, and Yin (2014), and Yin, Davis, and Muzyrya (2014)), and the relationship between mobile platforms and developers (Gans (2012))

There has been some research on the demand for apps and on app updates, though to my knowledge, no paper has explicitly considered the effect of updates on consumer utility as this paper does. Yin, Davis, and Muzyrya (2014) find that the level of updating by the most successful apps varies by category. Specifically, highly successful game apps update infrequently, while the most successful non-game apps update more frequently. Ghose and Han (2014) estimate a demand model in the style of Berry, Levinsohn, and Pakes (1995) across the App Store and the Google Play Store. They use a nested-logit model, where consumers choose first whether to download a Free or Paid app, then a category of apps, and finally which app to buy. Using this model, Ghose and Han find evidence that consumer demand is higher the more recently an app has been updated.

Of particular relevance to the issues studied in this paper is Comino, Manenti, and Mariuzzo (2018), which considers the effect of app updates on the growth rate of downloads. They find that the growth rate of downloads is positively affected by updates on Apple's platform, while there is no effect in the Google Play Store—a difference they attribute to Apple's strict app review process, which is interpreted as having a positive effect on the overall quality of the apps and app updates on Apple's platform compared to Google's. My paper builds on this idea of app updates affecting demand by directly estimating how updates affect consumer utility, and by modeling the supply-side decision of when and how to update an app.

3 A Structural Model of the App Store

In this section, I develop a model of consumer and developer behavior in the smartphone application (app) industry. Each period developers choose whether, and if so, how, to update their app. Developers earn revenue through both an extensive margin of demand, which captures consumers' app purchase decisions, and an intensive margin of demand, which captures consumers' app use decisions. Extensive-margin revenue comes from new purchases, while intensive-margin revenue can come from three possible sources: in-app purchases (IAPs), advertisements (ads), and subscriptions.

On the demand side, consumers choose each period whether to purchase an app in a given market (extensive margin demand). Consumers make static decisions, but are forward-looking as they consider the expected lifetime value of owning an app. At the same time, consumers choose each period whether to use any of their previously purchased apps (intensive margin demand). This additional margin of demand is important because many apps earn revenue (in whole or in part) from consumers' intensive-margin, *use* decisions. Due to a lack of available data on consumers intensive-margin behavior, I am unable to estimate a structural model of the intensive margin of demand.[10] Instead, I use a reduced form approximation of an app's total revenue when estimating the supply-side app updating model.[11]

3.1 Timing

Time in this model is discrete, and indexed by t. In my empirical estimation of this model a period is a week. The timing for each period is as follows:

1. The developer of app j observes the current state of the market, S_{jt}, and chooses whether, and if so, how, to update their apps.

2. Apple reviews the update submission, and, once approved, propagates the update to the App Store storefront for potential new customers to see, and to all previous purchasers of app j.[12]

3. Market-level demand-shocks are realized, and consumers observe their individual consumer-app match values.

4. Consumers make their extensive margin decisions of what app, if any, to purchase, and their intensive margin decisions of what previously purchased apps, if any, to use.

5. Developers receive their extensive- and intensive-margin revenue.

[10] In Appendix D I outline a basic model of consumer behavior on the intensive margin, which is used to calculate consumers' lifetime expected utility from purchasing an app.

[11] See Section 4.2 for a discussion of how I approximate apps' per-period revenue, and Section 5.2 for a how I estimate the supply-side model.

[12] See Section 2 for a description of this review process. I ignore the possibility that Apple might reject an update.

3.2 A Model of App Demand

As discussed above, consumers face two separate decisions each period: whether to purchase a new app, and whether to use any already owned apps.

Extensive Margin of Demand On the extensive margin of demand, consumer i receives utility u_{ijt} from purchasing app j in period t, and chooses the app that provides the greatest utility. In doing so, the consumer considers the potential future benefits of owning the app. Specifically, the utility from app j, u_{ijt} is

$$\begin{aligned} u_{ijt} &= X_{jt}\beta + \alpha p_{jt} + \xi_{jt} + \Lambda_{ijt} + \epsilon_{ijt} \\ u_{i0t} &= \epsilon_{i0t} \end{aligned} \tag{1}$$

and u_{i0t} is the utility from the outside option of not purchasing an app. X_{jt} is a vector of observable app characteristics, which includes the app's file size and age-appropriateness rating, whether or not the app includes IAPs, ads, and subscriptions, and month, year, and iOS version fixed effects. p_{jt} is the price of the app, which may be \$0.00. ξ_{jt} is the mean unobservable consumer utility from the app, and represents the utility that results from unobserved (to the econometrician) characteristics of the app. Λ_{jt} is the expected, discounted future utility from app j, capturing the fact that consumers anticipate both the future use value of the app, and future updates to the app when making a purchase decision. Finally, ϵ_{ijt} is a period-specific, consumer-app match value. I assume that $\epsilon_{ijt} \sim$ i.i.d. Type I Extreme Value, which admits a closed form solution for the share of consumers purchasing app j at time t, s_{jt}.

$$s_{jt} = \frac{\exp(X_{jt}\beta + \alpha p_{jt} + \Lambda_{jt} + \xi_{jt})}{1 + \sum_k \exp(X_{kt}\beta + \alpha p_{kt} + \Lambda_{kt} + \xi_{kt})} \tag{2}$$

Mean Unobservable Consumer Utility (ξ) Even within the same market, apps are highly differentiated, and, while there are some app characteristics that are observable to the econometrician, many key features are not. These unobserved characteristics are captured in the demand model by ξ_{jt}. ξ_{jt} captures a wide variety of unobserved characteristics including primary functions of an app, subtle features that differentiate it from its competitors, as well as other, broader characteristics that might affect a consumer's demand decision, such as whether it is particularly easy to find through the App Store's user interface.

It is unlikely that many of the unobserved characteristics captured by ξ_{jt}, such as specific functionality of the app, will vary period-to-period in an i.i.d. way. For example, a note-taking app would not offer a search feature one period, remove it the next, and then reinstate it in a third period. To capture this, I assume ξ_{jt} follows an AR-1 process, expressed as

$$\xi_{jt} = \rho\xi_{jt-1} + \eta_{jt}$$

where the unobserved quality of the app is correlated over time according to ρ, but is still subject to i.i.d., mean-zero shocks η_{jt}.

While the process above captures the fact that unobserved product characteristics will be fairly consistent period-to-period, it does not account for the fact that when a developer updates an app, ξ_{jt} is likely to change in a more substantial way than it would absent an update. To account for this, I model app updates as directly affecting ξ_{jt}, by assuming that ξ_{jt} undergoes a one-time, vertical shift as the result of an update. Namely, assume ξ_{jt} follows the law of motion

$$\xi_{jt} = \rho\xi_{jt-1} + g(a_{j,t};\mu) + \eta_{jt} \tag{3}$$

where $a_{j,t}$ is the developer's updating decision in period t, and $g(a_{j,t};\mu)$ captures the effect of the developer's update decision on ξ. In estimation, I use a linear sum of dummy variables, such as

$$g(a_{j,t};\mu) = \mu_{BF}1(a_{j,t} = \texttt{Bug Fix}) - \mu_F 1(a_{j,t} = \texttt{Feature})$$

when considering two types of updates: Feature and Bug Fix updates.

Note that unpacking the variety of possible mechanisms captured by ξ, while potentially interesting, is not necessary in order to answer the questions considered in this paper. All that is necessary in order to see whether consumers respond to updates is that the net effect of updates on consumer behavior is captured.

Expected, discounted future utility (Λ) When making purchase decisions, consumers have expectations over the likelihood of future updates, and over future consumer-app match values which are relevant to the consumer's period-specific, intensive margin, use decisions. In order

to construct a measure of consumers' lifetime expected utilities of owning an app I assume that consumers make independent, binary choices of whether to use each of their previously purchased apps. In each period, after purchasing an app the consumer will receive utility based on either using the app or choosing the outside option of not using that app. Therefore, I define the lifetime expected utility of owning app j, purchased in period t, Λ_{jt}, to be the discounted sum of the per-period expected utility of owning an app, with expectations over the developer of app j's future updating decisions. This takes the form

$$\Lambda_{ijt} = E\left[\sum_{\tau=t+1}^{\infty} \delta^{\tau-t} \ln\left(1 + e^{X_{j\tau}\beta+\xi_{j\tau}}\right)\right] \tag{4}$$

where δ is the discount factor. See Appendix D for the derivation of Λ_{jt}.

3.3 A Model of App Updating

Next, I develop a dynamic model of app updating. Apps in market m are indexed $j \in J_m$. Developers face a choice each period $t = 0, \ldots, \infty$ regarding whether, and if so, how, to update their app. I denote the state for period t from the perspective of firm j as $S_{j,t}$.

Motivated by computational restrictions, I make three simplifying assumptions. First, I restrict developers to a discrete set of potential updates, so that the developer's per-period choice set is $A = \{\emptyset, \texttt{Bug Fix}, \texttt{Feature}\}$, with $\emptyset$ representing the choice to not update the app. Second, there is relatively little within-app variation in the monetization strategies used by developers, and so I assume the choice of how to monetize an app, including the price of the app, is determined when the app enters the App Store, and is otherwise fixed. Finally, multi-app developers are assumed to treat the development of each app separately. The cost of this assumption is that this model will fail to account for any economies of scope in the production of apps. Only 8% of developers in the sample have more than one app in a given market.

The revenue an app earns each period is a function of the app's fixed monetization strategy. Monetization strategies consist of some combination of setting a non-zero price, and including advertisements, subscriptions, or in-app purchases (IAPs) in the app. It is possible that some apps earn revenue in additional ways, such as through accompanying hardware or services that are not sold through the App Store, and/or by selling users' information to third parties (see Kesler,

Kummer, and Schulte (2017)).

Developers choose $a_{jt} \in A$ for each period $t = 0, \ldots, \infty$ in order to maximize

$$E \sum_{t=0}^{\infty} \beta^t \Big(\pi(a_{j,t}, S_{j,t}) + \theta^{\epsilon} \epsilon_{j,t}(a_{j,t}) \Big) \tag{5}$$

where the per-period profit function $\pi(a_{j,t}, S_{j,t})$ is

$$\begin{aligned} \pi_{j,t}(a_{j,t}, S_{j,t}) + \theta^{\epsilon} \epsilon(a_{jt}) = R_j(a_{j,t}, S_{j,t})(1 - \phi) \\ - \theta^{BF} 1(a_{j,t} = \texttt{Bug Fix}) - \theta^{F} 1(a_{j,t} = \texttt{Feature}) + \theta^{\epsilon} \epsilon_{j,t}(a_{j,t}) \end{aligned} \tag{6}$$

R_j is the revenue the app earns, θ^{BugFix} and $\theta^{Feature}$ represent the fixed costs of updating, and ϵ_{jt} is an i.i.d. Type I Extreme Value error, which captures the fact that apps may earn additional revenue in the given period that is not captured by the model and available data. ϕ is the share of revenue that the platform collects. In the case of Apple, $\phi = 0.3$. Note that marginal costs are assumed to be 0, as discussed in Section 2.

R_j is defined as

$$R_j(a_{j,t}, S_{j,t}) = \overbrace{p_j s_{j,t} M_{m,t}}^{\text{Direct Sales Revenue}} + (\texttt{Intensive Margin Revenue}) \tag{7}$$

where intensive margin revenue can come from advertisements, in-app purchases, subscriptions, or any combination of the three. $M_{m,t}$ represents the market size of market m in period t.

Equilibrium When deciding whether to update their app, developers must weigh the current-period fixed cost of updating against present- and future-period revenues that would result from updating. I assume a Markov Perfect Nash Equilibrium (Ericson and Pakes, 1995). In such an equilibrium, strategy σ_j maps from a given state (S_j, ϵ_j) to an action a_j without dependence on t. Then, following Bellman's principle of optimality, the value to a firm in a given state can be expressed as

$$V_j^{\sigma}(S_j, \epsilon_j) = \max_{a_j \in A} [\pi(a_j, S_j) + \theta^{\epsilon} \epsilon_j(a_j) + \beta E \left[V_j^{\sigma}(S_j') | S_j, a_j \right] \tag{8}$$

and, since ϵ is i.i.d. Type I Extreme Value, I can express the optimal strategy for j as a conditional choice probability:

$$P^{\sigma_j}(a, S_j, \sigma_{-j}) = \frac{\exp\left(\nu_j^{\sigma}(a, S_j, \sigma_{-j})\right)}{\sum_{a' \in A} \exp\left(\nu_j^{\sigma}(a', S_j, \sigma_{-j})\right)} \tag{9}$$

where $\nu_j^{\sigma}(\cdot)$ is the choice-specific value function:

$$\nu_j^{\sigma}(a, S_j, \sigma_{-j}) = \frac{1}{\theta^{\epsilon}} \left(\pi(a, S_j) + \beta E\left[V_j^{\sigma}(S_j')|S_j, a_j\right]\right) \tag{10}$$

I assume a Markov Perfect Nash Equilibrium exists. Making this assumption is required due to the fact that this model has continuous state variables (e.g., download size, market size, and ξ_{jt}).[13]

4 Data

4.1 Description of Data

I estimate the model developed in Section 3 using data from the Productivity category on Apple's App Store platform. The category includes calendar apps, task management apps, and other apps meant to improve work or personal productivity in a wide variety of ways. The sample begins on December 31, 2014 and ends on June 29, 2016.

The data used in this project comes from a variety of sources. Product characteristic data, including information regarding product updating, comes directly from the App Store. I observe a number of characteristics for each app, including the app's name, price, file size (in megabytes), age-appropriateness rating, the day the app was first released, the version number of the app, the category classification of the app, and what devices the app is compatible with. I also observe a text description of each app and app update, both written by the developer.

In addition to the characteristic data, I also collect from the App Store three daily sales ranking lists. I observe the Top Free, Top Paid, and Top Grossing lists, which rank apps by either the quantity sold (Top Free and Top Paid) or by the apps' gross revenues (Top Grossing). Finally, I augment this App Store data with data from the analytics company AppFigures on which apps display advertising, and with sales data collected directly from a small number of the developers

[13] As noted in Sweeting (2013), it would be possible to discretize these continuous state variables in order to then apply the proof of Doraszelski and Satterthwaite (2010) that a pure strategy equilibrium exists.

who have apps in my sample.

In order to estimate the structural model, it's important to restrict my sample to those apps that are actively competing on the app store. There are two reasons why such a restriction is necessary. First, apps can be broadly classified as either "companion" or "product" apps, where the former are apps that are companions to, or work in conjunction with, other products or services outside of the App Store (e.g., the Wells Fargo or American Airlines apps), and the latter are apps that are a primary product being sold by the developer. Companion apps should be excluded from this study because, in many cases, their development is not profit-maximizing from the perspective of the App Store. For example, while American Airlines may not make any revenue from its app (particularly during this sample period), it is likely willing to invest in the development of the app because the app supports the company's overall ticket sales business.

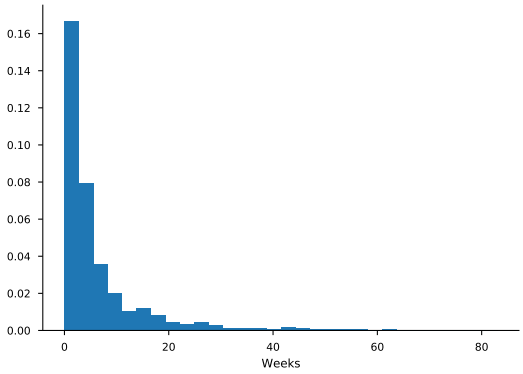
Second, because entry into the app store is relatively inexpensive, and the cost of keeping a poorly selling app on the app store is negligible, there are a large number of "abandoned apps," that are not being actively developed, and that have limited, or even zero, sales during the sample period. In order to restrict my focus to apps that are actively competing on the platform, I limit my sample to apps that reach a ranking of at least 200^{th} on the Top Grossing list for at least 10% of the sample period. This restriction, combined with the decision to focus on the Productivity category allows me to avoid both companion and abandoned apps as much as is possible. See Appendix A for further discussion of both of these issues.

My estimation sample consists of 356 apps. On average, a version of an app lasts 19 weeks before being updated, though there is a wide variance in the frequency of updating. Figure Fig. 2a shows the distribution of the time between updates. To some degree, this heterogeneity can be explained by app-level differences in updating behavior. The average app updates every 19 weeks, though the median app updates every 14 weeks. Fig. 2b shows the distribution of average updating frequency, aggregated at the app-level.

Apps are monetized through some combination of non-zero pricing, subscriptions, advertisements, and in-app purchases (IAPs). 49.5% of apps in the sample have a non-zero price, while 64.6% use at least one form of use-monetization. Fig. 3a shows the percentage of apps using each type of monetization, and Fig. 3b shows the fraction of apps in the sample using each possible monetization strategy.

Figure 2: App Version Ages

(a) Distribution of App Version Ages



(b) Average Version Age by App

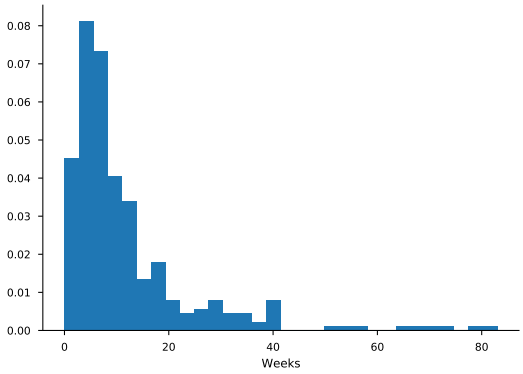
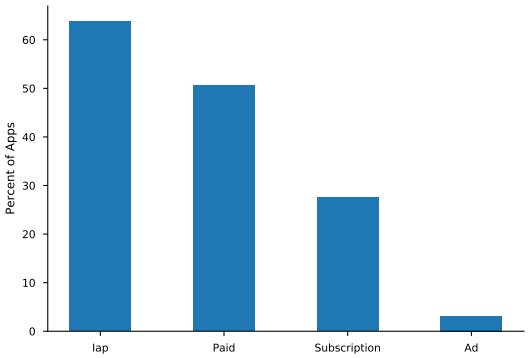


Figure 3: App Monetization

(a) Monetization Types



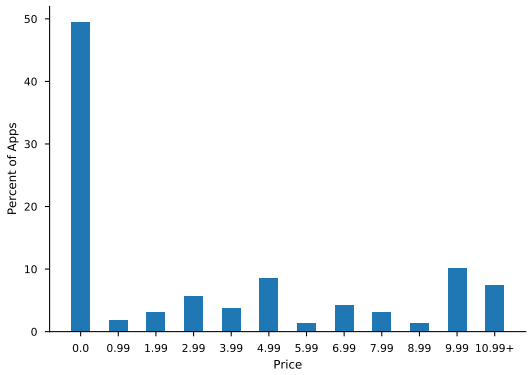
(b) Monetization Strategies

	# Apps	Percent
Paid	126	14.00
IAP	83	9.22
Subscription, IAP	77	8.56
Paid, IAP	39	4.33
Paid, Subscription, IAP	17	1.89
Advertisements, IAP	9	1.00
Paid, Subscription	3	0.33
Paid, Advertisements, IAP	1	0.11
Advertisements, Subscription, IAP	1	0.11

Fig. 4 shows the distribution of prices across the sample period. Prices are relatively constant over the sample period, and only 28 of the 356 apps ever change from free to paid, or vice versa, and these changes only account for 0.32% of the app-weeks in the sample.

In order to estimate the model in this paper I must first address three issues in the data. First, in Section 4.2 I account for the fact that sales data is, in general, unavailable for this industry. In Section 4.3, I use text descriptions of the app updates, called release notes, to classify updates based on their content. Finally, in Section 4.4 I discuss and address the challenge of defining markets in the App Store. In Section 4.5 I present preliminary evidence of the relationship between use monetization and updating.

Figure 4: Distribution of Prices



4.2 Estimating App Sales

Estimating the demand model (Eq. (1)) requires data on period-specific market shares. Here, as in many industries, sales data is proprietary, and direct data on market shares for my entire sample is unavailable. In studies of smartphone apps and other online markets, this issue has been dealt with in two ways. Bajari, Fox, and Ryan (2008) develop a method for estimating demand directly from sales ranking data, but their method requires that observable product characteristics are sufficient for determining a consumer's preference ordering within a market. Given the importance of the unobservable characteristics represented by ξ in my model, and, in particular, the manner in which product updates are incorporated into the ξ process, adopting the estimation approach developed by Bajari, Fox, and Ryan (2008) is not appropriate in this case.

The second approach, which has been widely used in research on apps, is to first estimate a relationship between an app's sales ranking and its level of sales using a secondary source of information about app sales. Then, this estimated relationship is used to predict sales for all products in all periods, and the demand model is estimated using market shares that are computed using these predicted sales values. This approach for mapping sales ranking data to sales quantity data in online markets was pioneered by Chevalier and Goolsbee (2003) and Brynjolfsson, Hu, and Smith (2003) in studying online book sales, and applied to app markets by Garg and Telang (2012), Ghose and Han (2014), and Ershov (2018), among others.[14] In most applications, the relationship between sales ranking and sales quantity is estimated using publicly available ranking data and

[14]This approach has also been applied by app developers, see, e.g., Perry (2015)

a small (relative to the sample of apps being studied) amount of sales data either collected from various developers or collected from small, independent app marketplaces.[15]

Apple provides three types of sales ranking lists, at both the storewide and the category level. The exact algorithms for the ranking lists are unknown, but they roughly rank apps by sales.[16,17] The Top Free list ranks apps with a price of $0.00 by sales quantity, the Top Paid list ranks apps with a non-zero price by sales quantity, and the Top Grossing list ranks all apps by revenue. I use the Top Free and Top Paid ranking lists at the category level to estimate sales because they directly account for apps' period-specific downloads, which is the necessary variable for computing the market shares used in the demand estimation. I assume sales ranking data follows a Pareto distribution, and estimate the following regression.

$$\ln(\text{sales}_{jt}) = \alpha - \beta \ln(\text{rank}_{jt}) + \gamma_{month,year} + \epsilon_{jt} \tag{11}$$

I estimate Eq. (11) using publicly observable ranking data, and the sales quantity data I have collected directly from some of the developers in the sample. Table 1a describes the data used to estimate this relationship. The apps in this dataset are ranked between 5 and 498 during the sample period, with an median ranking of approximately 118^{th}. Column 1 of Table 1b shows the results of estimating the rank-sales relationship for the Top Paid ranking list. Due to data limitations, I am only able to reliably estimate Eq. (11) for the Top Paid ranking list in the Productivity category. In order to also calculate sales for the Free apps in the sample, I follow Ghose and Han (2014) in assuming that the shape, β, of the relationship between an app's ranking and its sales is the same for both the Free and Paid list, but that the scale parameter, α, can differ. I calibrate $\alpha_{Free} = 7.175$ using sales data collected from developers' that produce Free apps.

My estimate of $\beta = 0.424$ is lower than what is typically found in the literature. Garg and Telang (2012) summarize seven studies of this relationship in non-app industries, and find that

[15] Instead of using rankings, Liu (2017) uses app ratings and/or reviews as a proxy for sales. An app rating is an integer between 1 and 5 that the consumer assigns an app, whereas a review is a rating accompanied by a textual comment on the product. This approach can be unreliable, though, as some developers actively prompt users for ratings and/or reviews while others do not.

[16] While Apple's internal policies regarding the ranking algorithm are not known, the view within the industry is that the lists do follow some explicit algorithm, which is not altered to benefit any particular app (e.g., for promotional reasons).

[17] Bresnahan, Li, and Yin (2016) study apps that attempt to game the ranking lists, though not by altering the ranking algorithm itself.

Table 1: Ranking-Sales Estimation

(a) Ranking-Sales Descriptive Statistics

	Downloads	Ranking
Mean	95.01	147.87
Median	36	117.5
Standard Deviation	274.89	107.21
Minimum	10	5
Maximum	4,826	498

Note that the best (i.e., most sales) possible ranking is 1, so the minimum ranking corresponds to the maximum number of sales, and vice versa.

(b) Ranking-Sales Parameter Estimates

	Top Paid	Top Grossing
α	5.231***	7.102 ***
	(0.182)	(0.204)
β	0.424***	0.404 ***
	(0.025)	(0.031)
R^2	0.760	0.738

Regressions include month-year fixed effects. Standard errors are in parenthesis. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$.

estimates range from 0.613 to 1.2. In app markets, Ghose and Han (2014) estimate $\beta = 1.09$, and Ershov (2018) estimates the parameter to be 1.168 for games and 0.926 for non-games.

There are two possible reasons for the difference between my estimates and those found in the literature, particularly those of Ghose and Han (2014) and Ershov (2018). First, my estimates are the only ones calculated using sales data from Apple's App Store, and the shape of the ranking-sales relationship may differ between platforms. Ghose and Han (2014) use data from a third-party (i.e., not Google nor Apple) app marketplace, and Ershov (2018) estimates the relationship for the Google Play Store using the lower bound of sales ranges the store provides publicly.[18]

A second reason is that the shape of the ranking-sales relationship may be changing over time. Using data on Amazon's book sales, Brynjolfsson, Hu, and Smith (2010) find evidence that the shape parameter has decreased in magnitude over time, as weight has shifted to the tail of the distribution. A similar change may be occurring in app markets. The sample period for this paper (12/31/2014 to 6/29/2016) is more recent than other papers that have estimated this relationship, so a smaller estimate of $|\beta|$ might be consistent with their findings. Currently, data limitations restrict my ability to further investigate whether the shape of the ranking-sales relationship has in fact changed over time in the App Store.

Given the results in column 1 of Table 1b, sales quantities are estimated for all apps in the sample based on their ranking each period. For the remainder of this paper, any reference to an app's sales quantity (or market share) should be viewed as a reference to that app's estimated sales

[18] For example, an app with cumulative sales of 45,000 is publicly listed as having "10,000 to 50,000" sales. See (Ershov, 2018, Table A2) for the sales ranges provided.

quantity (or market share). Additionally, I estimate this model for apps' per-period revenues using the Top Grossing list, which are shown in column 2 of Table 1b. These revenue estimates are used in the estimation of the supply model (see Section 5.2).[19]

4.3 Classifying Updates

In order to understand how consumers respond to app updates, it is important to account for the differences between specific updates. Updates serve a variety of purposes—from totally re-inventing an app to fixing a small typo in a rarely-used part of an app—and are released under a variety of circumstances—some updates are part of a large promotional campaign by the developer, while most are released without any fanfare. Developers provide two sources of information about the content and purpose of updates. The first is the update's version number, a sequence of numbers that developers ostensibly use to track the development of an app. The second is the update's release notes, a set of notes outlining the specific changes included in a particular update. I take advantage of both of these sources of information, and develop two distinct classifications of updates.

4.3.1 Classifying Updates by Version Numbers

Every time a developer updates an application, they change the version number, which serves as a developer-defined index of the development of an app. Among software developers, it is customary to use the change in the version number to indicate the magnitude or importance of an update. For example, an update from version 4.3.1 to 5.0.0 typically indicates a major revision, whereas an update from version 4.3.1 to 4.3.2 typically indicates a smaller update. Borrowing vocabulary from the developer community, I classify any update where the first number in the version number changes (e.g., 4.3.1 to 5.0.0) as a Major update, and any update where any of the subsequent numbers change (e.g., 4.3.1 to 4.3.2) as a Point update (so-called because the numbers after the first "point" are changed). However, there are no formal rules regarding version numbering, so updates where the version number changes from, for example, 4.3.1 to 5.0.0 may only include

[19] Note that the extent to which Apple is able to observe an app's revenue, and therefore the extent to which it is able to rank apps by revenue, is limited by the fact that Apple is only able to observe revenue that is collected through its platform. Apple observes all direct sales and IAP revenue, as well as any subscription revenue in cases where the subscription is managed and transacted through Apple's platform. Additionally, Apple ran a now-defunct advertising platform, iAd, during this sample period, however developers could also use competing platforms (e.g., Google AdWords).

minor changes to the app despite the "large" change in the version number.

4.3.2 Classifying Updates by Release Notes

While changes in an app's version number can be informative about the importance of an update, the lack of a clearly defined set of rules for version numbering means they don't provide a clear sense of the *content* of an update. As noted above, many apps in the App Store have Major updates where only small adjustments have been made, while many Point updates include the addition of important new features. Given this, I also make use of each update's release notes in order to classify updates solely by their content.

Specifically, updates are classified as either Bug Fix updates or as Feature updates. Conceptually, Bug Fix updates include changing the app so that it no longer crashes whenever the user performs a certain action, updating the app so that it is able to run on new devices or operating systems, and implementing "under-the-hood" changes, which involve cases where the developer makes changes primarily for the developer's own benefit—for example, by changing the underlying database system to something that the developer finds easier to work with, but that the consumer would be unaware of. Feature updates are those that add additional, user-salient functionality or content to the app. For example, updates that add voice dictation to a note-taking app, or add audio pronunciations to a dictionary app would be Feature updates.

I use a support vector machine (SVM) to classify updates using their release notes. An SVM is a form of supervised machine learning that is trained using a set of pre-classified release notes, and then, once trained, is able to classify the updates in my sample. In practice, I first convert the text release notes into vectors using a combination of natural language processing techniques. I discuss this process in Appendix B. Once the release notes have been "vectorized," I train the SVM using 782 app updates that I have classified by hand. Fig. 5 outlines the set of rules under which the training set was built. Two-thirds of the trained release notes are used to train the SVM, and the remaining third is used to test the performance of the machine. The results of these tests are shown in Table 2a. I use the trained model to classify every app update in my sample.

In Table 2b, I present a cross tabulation of updates according to both the Version Number classification system (Point and Major updates) and the SVM classification system (Bug Fix and Feature updates). 93.1% of updates classify as a Point update, though, notably, 57.7% of those

Figure 5: Release Notes Update Classification Training Rules

Feature Update	**Bug Fix Update**
• Adds additional functionality. • Adds additional content. • Adds support for a new language.	• Any direct mention of fixing bugs. • Performance improvements or anything indicating "under-the-hood improvements." • Adjustments to maintain compatibility with the latest version of iOS. • Changes to existing functionality that imply minor improvements to existing functionality, not the addition of more functionality.

Table 2: Update Classification

(a) Update Classification Precision and Recall

	Precision	Recall	N
Bug Fix	0.92	0.90	185
Feature	0.76	0.79	73
Average/Total	0.87	0.87	258

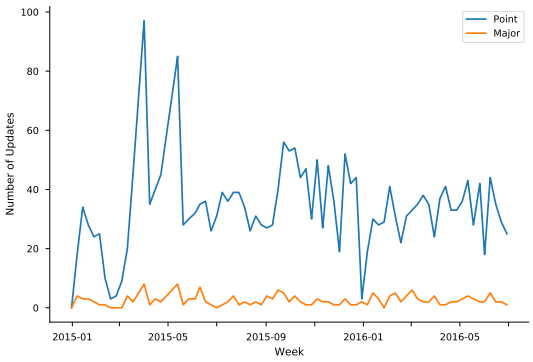
Precision indicates the ratio of the number of correctly predicted cases for a given update type to the total number of cases predicted to be of that type. Recall is the ratio of the number of correctly predicted cases for a given update type to the total number of cases of that type.

(b) Cross Tabulation of Update Classifications

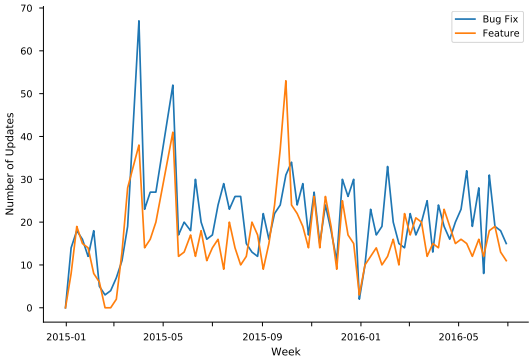
	Point	Major	Total
Bug Fix	1,473	69	1,542
Feature	1,079	119	1,198
Total	2,552	188	2,740

Figure 6: Updates By Type

(a) Version Number Classification



(b) Release Notes Classification



updates add new features or functionality to the app. Part of the concern about using the Version Number classification system is evident in Table 2b, as only two-thirds of Major updates actually add new features or content to the app. Fig. 6 shows the number of updates by type over the sample period. In general, consistent with the cross tabulation, Bug Fix updates outnumber Feature updates, but there is time variation in the ratio of the two types of updates and at times Feature updates outnumber Bug Fix updates. For example, in late September 2015, just following the release of a new model of the iPhone (indicated by the dashed vertical line), there is a spike in the number of Major updates, which outnumbers Minor updates in that period.

4.4 Defining Markets

Finally, estimating the model developed in Section 3 requires defining markets within the Productivity category of the App Store. Properly defining the relevant market for a product has long been a challenge in empirical market analyses, and markets for digital goods provide additional challenges. In this context, the wide variety of monetization methods used make it difficult to develop and employ systematic methods for defining markets.

Traditional, systematic methods for defining markets, such as the commonly employed "small but significant and non-transitory increase in price" (SSNIP) test, are especially difficult to develop and apply due to the challenge of calculating the elasticity of demand when many products are priced at $0.00 (Evans, 2011). Approaches that do not involve price elasticity calculations are also difficult to employ, given the variety of ways products are monetized in this industry. Examples of

this include the recently suggested "small but significant and non-transitory increase in *(exchanged) costs*" (SSNIC) test, where exchanged costs represent exchanges of information or attention in return for a product, and the "small but significant and not-transitory decline of quality" (SSNDQ) test, which was employed in a 2013 case before the Chinese Supreme People's Court (Newman, 2015).

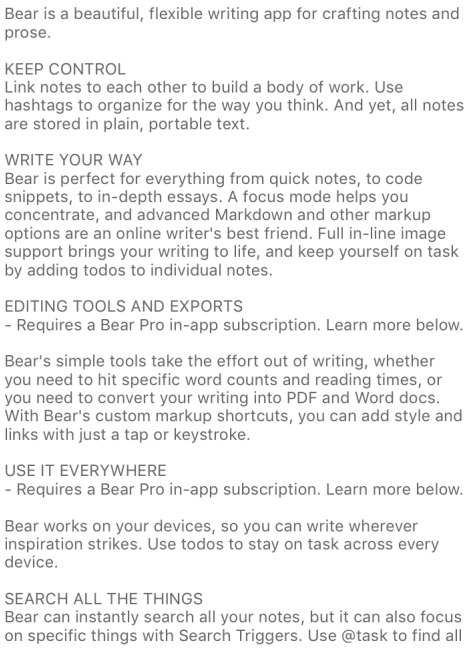
It is not clear, for example, how to define a product's exchanged costs when the firm employs multiple forms of monetization, or even when a firm employs a single form of monetization, but has the *ability* to employ multiple forms of monetization.[20] That is, estimating the SSNIC associated with a paid-upfront price increase of x would have to account for the endogenous response of the product's advertisement level, number and prices of in-app purchases, and/or its subscription price. This would still be abstracting from the endogeneity of *which* forms of monetization a particular product uses. While such an abstraction is reasonable in a typical, non-digital industry, where all firms charge for a product at the point of purchase, or where all firms use a single form of alternative monetization (e.g., advertising), it is less reasonable for app markets given the varied manner in which the products are monetized. Finally, accounting for all of this would be difficult in terms of modeling, and also computationally difficult given the large number of potential products available.

Previous Approaches for Defining Markets in the App Industry In the literature on smartphone apps, markets have generally been defined by adopting the categories used within the marketplaces themselves. That is, in Apple's App Store, each of the 22 consumer-facing categories would each be considered a market.[21] This approach of adopting the platforms' categories is overly broad and can suggest competition where there is none. For example, within Apple's Productivity category, there are a number of apps that allow a user to deploy an encrypted virtual private network in order to secure their internet communications. At the same time, there are many apps that help users manage their photos for the photo-sharing app Instagram. While some consumers may use both, the apps serving one of these two purposes are unlikely to be in direct competition with each other. In order to properly characterize consumer choice, and model firms' strategic

[20] See Newman (2015), particularly pages 66–67, for a discussion of implementing the SSNIC test. Notably, though, this discussion implicitly assumes that the monetization strategy used by firms is fixed, whereas in many digital industries, including apps, it is not.

[21] There are, to some extent, sub-genres defined within the App Store, but during this sample period they were primarily used for Game apps, which are not considered in this paper.

Figure 7: App Description



behavior in these markets, it is important to define markets in such a way as to properly account for demand-side substitution that occurs within markets.

One improvement on this market definition approach is that of Ghose and Han (2014), who offer a more detailed approach by using a nested structure for markets, where consumers first choose whether to buy a paid or free app, and then which category to shop in. However, this too fails to account for the large amount of heterogeneity within product categories.

I define markets by combining apps' categorical classification with additional information provided by the apps' developers. Within the Productivity category I cluster apps using an unsupervised machine learning algorithm based on the text app descriptions provided by the developers. These descriptions are a prominent part of apps' listings on the App Store, and thus developers have a strong motivation to provide a clear description of their product. Fig. 7 shows the description of the note-taking app Bear.

Defining Markets via Clustering I use a combination of natural language processing techniques and an unsupervised clustering algorithm to define markets.[22] First, all app descriptions

[22] See Hoberg and Phillips (2016), who use the text provided in annual SEC reports to define annualized product markets, for a related approach.

undergo the same text pre-processing process outlined in Appendix B. The specific parameters of this process are separately determined for the market definition algorithm.

Once the app descriptions have been processed and vectorized, I define markets using the X-means clustering algorithm (Pelleg and Moore, 2000).[23] X-means clustering is an extension of the k-means clustering algorithm, and uses the Bayesian Information Criterion (BIC) to determine the optimal number of clusters, an issue that the traditional k-means algorithm cannot address. Broadly, the k-means algorithm is a recursive algorithm that places a set of k centroids in the app description vector space, assigns each app to the closest centroid, and then adjusts the location of the centroids to minimize the distance between a centroid and its constituent apps. This process continues until convergence. The X-means algorithm improves on this by beginning at a low number of centroids, running the k-means algorithm, and then again using the k-means algorithm and the BIC to determine whether any of the defined clusters (i.e., markets) should be split in two. It then runs the k-means algorithm with the new number of clusters. The algorithm cycles through these steps until convergence. I discuss both algorithms in more detail in Appendix C.

While the X-means algorithm is able to select the optimal number of clusters (i.e., markets), there is still a question of what parameters should be set for the text pre-processing stage. To address this, I repeat the clustering algorithm across a grid of text processing hyper-parameters, and select the vector of parameters that maximize a scoring metric. In particular, I use the silhouette score, which provides an average measure of how well each app matches with its market, compared to how it matches with the other markets (Rousseeuw, 1987).

[23] I use the X-means clustering implementation in the pyclustering library (Novikov, 2018)

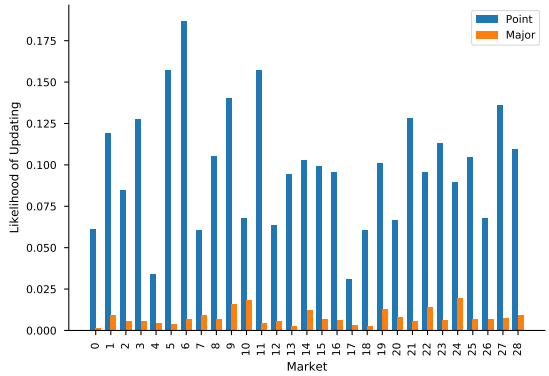
Table 3: Market-Level Summary Statistics

		Monetization												**Updates**				**Characteristics**				
		Price				Price (Paid apps)								Version Number		SVM			Age Appropriateness Rating			
Market	Apps	Min	Mean	Median	Max	Min	Mean	Median	Max	Free	IAP	Ad	Subs	Minor	Major	Bug Fix	Feature	Size	4+	9+	12+	17+
0	10	0.00	4.41	1.99	19.99	0.99	8.68	7.99	19.99	0.491	0.491	0.000	0.108	0.061	0.001	0.037	0.026	2.661	0.892	0.108	0.000	0.000
1	3	0.00	3.10	4.99	4.99	1.99	4.76	4.99	4.99	0.349	0.670	0.000	0.000	0.119	0.009	0.041	0.092	3.076	1.000	0.000	0.000	0.000
2	10	0.00	6.48	2.99	29.99	1.99	11.33	9.99	29.99	0.428	0.687	0.000	0.380	0.085	0.006	0.042	0.048	2.816	0.996	0.000	0.004	0.000
3	15	0.00	3.16	1.99	19.99	0.99	5.93	6.99	19.99	0.467	0.743	0.000	0.415	0.128	0.006	0.080	0.054	3.121	0.929	0.000	0.000	0.071
4	9	0.00	3.69	3.99	7.99	2.99	5.48	3.99	7.99	0.326	0.422	0.000	0.334	0.034	0.005	0.012	0.026	1.310	1.000	0.000	0.000	0.000
5	10	0.00	3.01	0.00	24.99	0.99	7.52	2.99	24.99	0.601	0.702	0.000	0.499	0.157	0.004	0.095	0.069	3.065	0.899	0.101	0.000	0.000
6	6	0.00	3.84	2.99	11.99	0.99	5.73	3.99	11.99	0.330	0.497	0.000	0.169	0.187	0.007	0.069	0.127	3.804	0.835	0.000	0.000	0.165
7	12	0.00	1.09	0.00	4.99	1.99	2.92	2.99	4.99	0.625	0.887	0.000	0.797	0.061	0.010	0.051	0.020	3.035	1.000	0.000	0.000	0.000
8	21	0.00	10.33	3.99	99.99	0.99	14.97	9.99	99.99	0.310	0.433	0.047	0.048	0.105	0.007	0.054	0.061	3.343	0.812	0.049	0.090	0.049
9	5	0.00	5.76	4.99	9.99	4.99	8.31	9.99	9.99	0.307	0.543	0.000	0.230	0.141	0.016	0.073	0.093	4.377	0.454	0.233	0.077	0.236
10	9	0.00	8.55	4.99	49.99	0.99	11.10	4.99	49.99	0.229	0.451	0.000	0.229	0.068	0.018	0.050	0.036	2.769	0.885	0.000	0.115	0.000
11	13	0.00	10.37	5.99	49.99	0.99	16.10	9.99	49.99	0.356	0.519	0.000	0.327	0.157	0.004	0.091	0.072	3.763	0.918	0.000	0.000	0.082
12	9	0.00	5.59	4.99	12.99	1.99	7.31	6.99	12.99	0.235	0.442	0.000	0.112	0.064	0.006	0.047	0.024	3.180	0.761	0.000	0.000	0.239
13	5	0.00	6.57	4.99	14.99	0.99	8.29	4.99	14.99	0.208	0.397	0.000	0.205	0.095	0.003	0.043	0.057	3.439	0.603	0.000	0.000	0.397
14	17	0.00	2.21	0.00	9.99	0.99	7.13	7.99	9.99	0.690	0.756	0.018	0.133	0.103	0.012	0.059	0.057	3.124	0.641	0.000	0.000	0.359
15	6	0.00	1.74	0.99	4.99	0.99	3.31	3.99	4.99	0.473	0.652	0.000	0.000	0.099	0.007	0.099	0.007	2.398	0.513	0.000	0.000	0.487
16	36	0.00	5.31	1.99	24.99	0.99	9.29	7.99	24.99	0.429	0.511	0.000	0.223	0.095	0.006	0.054	0.049	2.427	0.902	0.000	0.000	0.098
17	5	0.00	12.14	4.99	49.99	1.99	16.47	4.99	49.99	0.263	0.263	0.000	0.263	0.031	0.003	0.024	0.010	2.001	1.000	0.000	0.000	0.000
18	5	0.00	4.58	6.99	10.99	0.99	8.61	9.99	10.99	0.468	0.800	0.000	0.200	0.061	0.003	0.037	0.026	5.157	1.000	0.000	0.000	0.000
19	3	0.00	18.38	6.99	119.99	5.99	27.57	14.99	119.99	0.333	0.333	0.000	0.000	0.101	0.013	0.083	0.031	3.680	1.000	0.000	0.000	0.000
20	4	0.00	7.15	8.99	9.99	1.99	7.40	8.99	9.99	0.033	0.300	0.000	0.000	0.067	0.008	0.037	0.037	3.064	0.683	0.000	0.000	0.317
21	24	0.00	6.40	3.99	39.99	0.99	10.48	4.99	39.99	0.390	0.726	0.000	0.385	0.128	0.006	0.056	0.080	2.965	1.000	0.000	0.000	0.000
22	9	0.00	0.71	0.00	4.99	0.99	3.38	2.99	4.99	0.791	0.819	0.124	0.386	0.096	0.014	0.094	0.016	3.106	0.526	0.000	0.159	0.315
23	14	0.00	3.37	0.00	16.99	1.99	7.96	6.99	16.99	0.577	0.688	0.000	0.147	0.113	0.006	0.071	0.050	2.854	0.934	0.000	0.066	0.000
24	12	0.00	0.00	0.00	0.00	—	—	—	—	1.000	1.000	0.479	0.000	0.090	0.019	0.091	0.019	2.380	0.597	0.000	0.403	0.000
25	13	0.00	2.40	0.00	49.99	0.99	5.69	4.99	49.99	0.579	0.579	0.000	0.320	0.105	0.007	0.071	0.040	2.605	0.910	0.000	0.090	0.000
26	18	0.00	3.02	0.99	12.99	0.99	5.61	4.99	12.99	0.461	0.633	0.000	0.180	0.068	0.007	0.040	0.036	2.571	0.988	0.000	0.012	0.000
27	27	0.00	0.76	0.00	19.99	0.99	6.73	4.99	19.99	0.887	0.979	0.090	0.652	0.136	0.007	0.101	0.045	2.513	0.766	0.046	0.152	0.036
28	26	0.00	3.12	0.00	9.99	0.99	7.34	7.99	9.99	0.575	0.700	0.039	0.354	0.110	0.009	0.063	0.057	2.822	0.616	0.005	0.000	0.380
Total	356	0.00	4.62	0.99	119.99	0.99	9.14	6.99	119.99	0.495	0.642	0.029	0.284	0.103	0.008	0.063	0.049	2.897	0.840	0.016	0.041	0.104

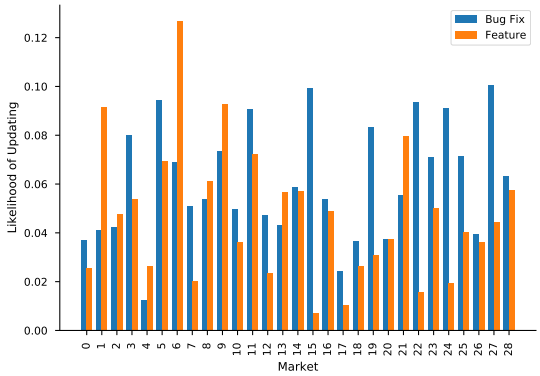
Observations are app-weeks. $N = 24,811$.

Figure 8: Update Types, By Market

(a) Version Number Classification



(b) Release Notes Classification



Market Results This X-means clustering process results in 29 markets. Fig. 9 lists the apps in four of the markets. There are some cases where human judgement might be at odds with the algorithm's market choices, but the market definitions generally appear to be reasonable. Table 3 presents overall summary statistics for each of these markets, as well as for the overall sample. The markets range from a 3-firm market (combination calendar/task management apps) to a 36-firm market (password management/VPN apps). The average market has 12 apps, with the median slightly lower at 10 apps. Within each market, there is a wide variety of monetization strategies in use. In fact, market 24 (social media management apps) is the only market that does not have a combination of Free and Paid apps. Finally, Fig. 8 shows update frequencies by market under both classification methods. Notably, there is a wide level of variation across markets in both the level of updating and the relative level update types.

4.5 Descriptive Evidence

While many apps in the App Store engage in use monetization, many others do not. This provides natural treatment and control groups for developing a preliminary understanding of how the ability to monetize a product's use affects developers' updating behavior. Of course, a natural concern with such an analysis is that there is likely selection on unobservables into use or non-use monetization. Furthermore, analyzing how use- and non-use monetizing apps differ does not capture the effects of firms' abilities to update products past the point of sale on their updating decisions. Thus, it will be necessary to estimate the structural model developed in Section 3 and conduct the proposed

Figure 9: Example Markets

CalenGoo
PocketLife Calendar
Week Calendar
LikeTopix Calendar App
One Place
IAFF Foundation Pro-Calendar
Mom's Daily Planner
CalenMob Pro
Awesome Calendar
iCalendar
CalenMob
Jorte Calendar & Organizer
Calendars by Readdle
Calendars 5
Moleskine Timepage
Shift Work Days
BusyCal

(a) Calendar Apps

Things
Toodledo
Remember The Milk
2Do
TaskTask
HomeRoutines
Pocket Informant
gTasks
Benjamin
Any.do
ATracker PRO
ATracker
Planner Plus
Todo Cloud
Planner Plus
Todoist
FranklinCovey Tasks
Tody
gTasks Pro
OmniFocus 2 for iPhone
Todo
Motivated Moms
OmniFocus 2 for iPad
Focus
OmniPlan 3

(b) Task Management Apps

Print n Share
PrinterShare Mobile
PrinterShare Premium
Printer Pro
PrintCentral Pro for iPhone/iPod Touch
To Print
Quick Print
SMS Export PRO
Quick Print via Google Cloud Print

(c) Printing Apps

Groups
My Contacts Backup Pro
Simpler Contacts Pro
EasilyDo Assistant
Smart Merge Pro
Cleaner Pro
Easy Backup
Cloze
Simpler Contacts
Cleaner
Smart Merge
CCleaner for iOS
Cleaner Master
Interact

(d) Contacts Management Apps

Table 4: Preliminary Evidence of the Effect of Digitization on Product Innovation

	(1) Update	(2) Update	(3) Major	(4) Feature
Free	0.0534***			
	(0.0091)			
Use Monetization		0.0437***	0.0269*	-0.0252
		(0.0087)	(0.0150)	(0.0411)
Price	0.0012	0.0006	0.0005	0.0007
	(0.0008)	(0.0006)	(0.0009)	(0.0029)
Download Size	0.0005***	0.0005***	-0.0002	0.0010***
	(0.0001)	(0.0001)	(0.0001)	(0.0003)
N	24811	24811	2773	2773

Specifications (1) and (2) estimate a linear probability model of the updating decision. Columns (3) and (4) are linear probability models of the type of update, conditional on updating. Observations are app-weeks. Standard errors clustered at the market level, and are in parentheses. Regressions also include controls for the month, year, market, iOS version, the age of the app (in weeks), and the squared age of the app. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

counterfactual analysis (Section 7) to fully answer the questions of how the digitization of consumer goods affects product innovation.

To develop a preliminary understanding of how use monetization relates to product updating, I regress binary indicators of an app's updating choice, y_{jt}, on measures of the app's monetization strategy, m_{jt}, and a set of covariates. In particular, I estimate the linear probability model

$$y_{jt} = \alpha + \beta m_{jt} + X_{jt}\gamma + \epsilon \tag{12}$$

where X_{jt} includes the price and download size of the app, as well as controls for the month, year, market, iOS version, age (in weeks), and squared age of the app. In each case, the coefficient of interest is β, which indicates how a particular monetization strategy m_{jt} relates to the updating behavior.

In specification (1) of Table 4, I regress whether or not app j updated in period t on whether it is a Free app, or one with a price of \$0.00. I find that Free apps are 5.3 percentage points more likely to update than non-free apps. Though—and this finding is consistent across the specifications considered—there is a positive relationship between an app's price and its likelihood of updating.

In specification (2), I estimate the relationship between the decision to update and whether an app uses any form of use monetization (which can include apps with a non-zero upfront price). I find that use-monetizing apps are 4.4 percentage points more likely to update than their paid-only counterparts. Finally, in specifications (3) and (4) I investigate whether there is a relationship between use monetization and the type of updates developers release. I find there is a relationship when distinguishing between updates using the Version Number classification system. Use-monetizing apps are 2.7 percentage points more likely to release a Major update than paid-only apps, conditional on updating. However, there is no evidence of use-monetizing apps behaving differently than paid-only apps when distinguishing between updates using the Release Notes classification approach.

5 Estimating the Structural Model

5.1 Demand Estimation

I estimate the model in Section 3.2 using the sample and markets defined in Section 4. Due to the ξ process, the estimation procedure differs somewhat from the typical approach used to estimate logit demand models. Recall that the demand model (Eq. (1)) implies the market share of app j in period t can be expressed as

$$s_{jt} = \frac{\exp(X_{jt}\beta + \alpha p_{jt} + \Lambda_{jt} + \xi_{jt})}{1 + \sum_k \exp(X_{kt}\beta + \alpha p_{kt} + \Lambda_{kt} + \xi_{kt})}$$

In order to derive an estimating equation for the demand model, I transform market shares by taking the log of both sides (Berry, 1994). This gives

$$\ln(s_{jt}) - \ln(s_{0t}) = X_{jt}\beta + \alpha p_{jt} + \Lambda_{jt} + \xi_{jt} \tag{13}$$

Typically, Eq. (13) can be estimated using ordinary least squares or two-stage least squares. However, because ξ follows the law of motion defined in Eq. (3), and because I need to estimate parameters within that process, an alternative approach is needed.

For clarity, let $y_{jt} = \ln(s_{jt}) - \ln(s_{0t})$ be the left-hand side of Eq. (13). Given this, and plugging

in the ξ law of motion, Eq. (13), can be re-written as

$$y_{jt} = X_{jt}\beta + \alpha p_{jt} + \Lambda_{jt} + \rho\xi_{jt-1} + g(a_{j,t};\mu) + \eta_{jt} \tag{14}$$

In order to estimate the parameters of the model, I quasi-first-difference this model, by subtracting ρy_{jt-1} from y_{jt}, giving

$$y_{jt} - \rho y_{jt-1} = (X_{jt} - \rho X_{jt-1})\beta + \alpha(p_{jt} - \rho p_{jt-1}) + (\Lambda_{jt} - \Lambda_{jt-1}) + g(a_{j,t};\mu) + \eta_{jt} \tag{15}$$

I estimate Eq. (15) using non-linear least squares (NLLS). In estimating the demand model, Λ_{jt} is calculated using forward simulation.[24]. Finally, to estimate the effect of various types of updates (according to the two update classification approaches discussed in Section 4.3), I define $g(a_{j,t};\mu)$ to be a linear sum of dummy variables, such as:

$$g(a_{j,t};\mu) = \mu_{BF} 1(a_{j,t} = \texttt{Bug Fix}) - \mu_F 1(a_{j,t} = \texttt{Feature})$$

Non-linear Price Disutility The model assumes that price has a linear effect on a consumer's utility. This is not necessarily the case. Anecdotal evidence suggests that while consumers may strongly dislike paying for an app, they may not be highly price sensitive conditional on paying for an app. In light of these concerns, I consider three specifications of p_{jt} when I estimate the model in order to account for any possible non-linearities in consumers' price sensitivity. The first specification includes the price p_{jt} of the app, and the second specification uses an indicator variable for whether the price is greater than zero or not. Specification three includes both the app's price, and the "paid" indicator.

Timing and Endogeneity As discussed in Section 2, Apple maintains a strict review process for all app updates. This process creates a delay between the creation of the update (and its submission to Apple), and the release of the update to consumers. Based on conversations with developers, this process took about a week on average during my sample period. Because I aggregate my data to the week level, I assume that the decision of whether, and if so, how, to update an app is made

[24] This process is discussed in more details in Appendix D

prior to the realization of η_{jt}. That is, η_{jt} is assumed to be independent of period-t updating decisions.

Price changes, however, can be made independently from app updates, and are not subject to review, so the standard price endogeneity concern exists. This is dealt with using a two-stage non-linear least squares approach, where period t prices are predicted in a first-stage regression using current and one-period lagged covariates.[25] This is similar to the approach taken in Doraszelski, Lewis, and Pakes (2016).

5.2 Fixed Costs Estimation

I estimate the app updating model using a forward-simulation approach, as in Bajari, Benkard, and Levin (2007), henceforth BBL. BBL propose a two-stage estimator. In the first stage, state transition probabilities and the conditional choice probabilities for firms' updating decisions are estimated. These are then used to approximate the firms' value functions. In the second stage, the approximated value functions are used to estimate the dynamic parameters of the model.

First Stage In the first stage of the supply estimation, I estimate the state transition probabilities for the state variables that vary over time, estimate a reduced form model for approximating apps' revenues, and estimate reduced form policy functions for apps' updating decisions. Table 5 provides a list of state variables in the model, and a description of their evolution process.

Following BBL, I rewrite Eq. (8) to express the value function as a linear function of the

[25] More precisely, let the set of covariates $X_{jt} = (X^1_{jt}, X^2_{jt})$ where X^1 consists of variables that vary over time for a given app, and the elements of X^2 do not. The first stage regression used is $p_{jt} = \gamma_0 X_{jt} + \gamma_2 X^1_{jt} + \epsilon_{jt}$.

Table 5: State Variables

State Variable	Evolution	Level
Paid (i.e., $1(p > 0)$)	Fixed	Firm-level
Retail Price	Fixed	Firm-level
IAP	Fixed	Firm-level
Ad	Fixed	Firm-level
Subscription	Fixed	Firm-level
Download Size	AR-1, when updated	Firm-level
Last-period Update Status	Firm's action	Firm-level
Competitors' Last-period Update Statuses	Competitors' actions	Firm-level
ξ_j	Eq. (3)	Firm-level
ξ_{-j}	Eq. (3)	Firm-level
Number of Firms	Fixed	Market-level
Market Average Download Size	(See note)	Market-level

The average download size for the market is computed as the average of the download sizes for all apps in the market. Those download sizes follow an AR-1 process, changing only when a given app has been updated.

parameters to be estimated.

$$\begin{aligned} V(S_{jt}|\sigma_j,\sigma_{-j}) =& E_{j,\sigma_j,\sigma_{-j}} \sum_{t=0}^{\infty} \delta^t R_j(a_{j,t}, S_{jt}) \\ &-\theta^{BF} E_{j,\sigma_j,\sigma_{-j}} \sum_{t=0}^{\infty} \delta^t 1(a_{jt} = \texttt{Bug Fix}) \\ &-\theta^{F} E_{j,\sigma_j,\sigma_{-j}} \sum_{t=0}^{\infty} \delta^t 1(a_{jt} = \texttt{Feature}) \\ &+\theta^{\epsilon} E_{j,\sigma_j,\sigma_{-j}} \sum_{t=0}^{\infty} \delta^t \epsilon_{jt}(a_{jt}) \\ =& \mathbf{R}_{j,\sigma_j,\sigma_{-j}} - \theta^{BF} \mathbf{F}^{BugFix}_{j,\sigma_j,\sigma_{-j}} - \theta^{F} \mathbf{F}^{Feature}_{j,\sigma_j,\sigma_{-j}} + \theta^{\epsilon} \boldsymbol{\epsilon}_{j,\sigma_j,\sigma_{-j}} \end{aligned} \tag{16}$$

As discussed in Section 3, a structural model of the intensive margin of demand cannot be reliably estimated due to limited data availability. Thus, it is not possible to develop a structural model for app j's revenue, $R_j(S_{k,j}; \theta^D)$, since many apps will depend on intensive- as well as extensive-margin demand for revenue. Instead, I estimate a flexible model of revenue on the current state and a large number of additional covariates that are each a function of the current state.[26] Using the results of this model I am able to predict the revenue an app will earn given the state,

[26] Revenue data for the firms is calculated using the Top Grossing ranking list. See Section 4.2 for more details.

and this prediction is used in the supply estimation. Thus, revenue is treated as observed in the estimation procedure. Letting $\hat{R}(S_{k,j}, \theta^R)$ represent the estimated reduced form revenue model, the value function used in estimating this model is

$$V(S_{jt}|\sigma_j, \sigma_{-j}) = \hat{\mathbf{R}}_{j,\sigma_j,\sigma_{-j}} - \theta^{BF}\mathbf{F}^{BugFix}_{j,\sigma_j,\sigma_{-j}} - \theta^{F}\mathbf{F}^{Feature}_{j,\sigma_j,\sigma_{-j}} + \theta^{\epsilon}\boldsymbol{\epsilon}_{j,\sigma_j,\sigma_{-j}} \tag{17}$$

where $\hat{\mathbf{R}}_{j,\sigma_j,\sigma_{-j}}$ is equivalent to calculating $\mathbf{R}_{j,\sigma_j,\sigma_{-j}}$ using the reduced form revenue model in place of a full structural model of revenue.

As outlined in BBL, the linearity of the firms' value functions in $\theta = (\theta^{BF}, \theta^F, \theta^{\epsilon})$ significantly reduces the computational burden of estimating this model, because it is only necessary to calculate the value functions once, as opposed to re-calculating the values for each candidate parameter during the optimization process. I calculate approximations of $\boldsymbol{R}$, $\boldsymbol{F}^{BF}$, $\boldsymbol{F}^F$, and $\boldsymbol{\epsilon}$ using forward simulation.

Given the estimated policy functions and the reduced form revenue model, I use the following process to calculate a single simulated path for $\boldsymbol{R}$, $\boldsymbol{F}^{BF}$, $\boldsymbol{F}^F$, and $\boldsymbol{\epsilon}$. In these simulations I assume the discount factor $\delta = 0.95$.

1. Initialize $\boldsymbol{R}$, $\boldsymbol{F}^{BF}$, $\boldsymbol{F}^F$, and $\boldsymbol{\epsilon}$ equal to 0.

2. Given the state, calculate the conditional choice probabilities (CCPs) for all firms in the market, and for each firm select an action based on that firm's set of CCPs. $\boldsymbol{F}^{BF}$ or $\boldsymbol{F}^F$ are updated as appropriate (dependent on the chosen action for firm j), as well as $\boldsymbol{\epsilon}$.

3. Update the state variables to reflect changes due to the firms' actions.

4. Given the updated state, calculate the firms' revenues using the revenue model $\tilde{R}$ and update the value of $\boldsymbol{R}$.

5. Repeat steps 2-4 for 50 periods.

I then repeat this process for a total of 250 simulations, and average across those 250 simulated paths. I conduct this process for each observed initial state, and for each alternative updating choice a firm in each observed state could have made.

Stage 2 Given the results of Stage 1, I estimate the dynamic parameters $(\theta^{BF}, \theta^{F}, \theta^{\epsilon})$ using a maximum likelihood estimator. In particular, since ϵ is assumed to be i.i.d. Type I Extreme Value, I can express the optimal strategy for j as Eq. (9), which I replicate below.

$$P^{\sigma_j}(a, S_j, \sigma_{-j}) = \frac{\exp\left(\nu_j^{\sigma}(a, S_j, \sigma_{-j})\right)}{\sum_{a' \in A} \exp\left(\nu_j^{\sigma}(a', S_j, \sigma_{-j})\right)}$$

where $\nu_j(\cdot)$ is the choice-specific value function is

$$\nu_j^{\sigma}(a, S_j, \sigma_{-j}) = \frac{1}{\theta^{\epsilon}}\left(\pi(a, S_j) + \beta E\left[V_j^{\sigma}(S_j')|S_j, a_j\right]\right)$$

Denoting this function when app j is in state S_j as $v_j(a)$, I can construct the loglikelihood function:

$$\ln \mathcal{L}(\theta) = \sum_t \sum_t \left(\sum_{a \in A} 1(a_{jt} = a)\nu_j(a) - \ln\left(\sum_{a \in A} \exp\left(\nu_j(a)\right)\right)\right) \tag{18}$$

I then find the set of dynamic parameters $(\theta^{BF}, \theta^{F}, \theta^{\epsilon})$ that maximize this function.

6 Estimation Results

6.1 Demand Results

In this section, I discuss the results of estimating the demand model presented in Section 3.2. In Table 6 I consider the relationship between an app's monetization strategy and (extensive-margin) consumer demand, and in Table 7 I consider how updates affect consumer demand, and whether consumers respond differentially to different types of updates.

Monetization To understand how the price of an app relates to the demand for that app, I consider three different specifications of the model, the results of which are presented in Table 6. In column (1) I include only the price of the app (which in approximately half of the observations is equal to \$0.00). In column (2), I exclude the price of an app, and instead use a dummy variable, *Paid*, to indicate whether the app is priced above zero or not. Finally, in column (3), I include both the price and the Paid indicator.

As expected, I estimate a negative relationship between price and demand. Further, comparing

Table 6: Effect of Price on Demand

	(1)	(2)	(3)	(4)	(5)
Update	0.1920***	0.1681***	0.1811***	0.1932***	0.1809***
	(0.0360)	(0.0360)	(0.0359)	(0.0360)	(0.0360)
Price	-0.0653***		-0.0537***	-0.0724***	-0.0591***
	(0.0050)		(0.0052)	(0.0063)	(0.0065)
Paid		-1.3227***	-0.9064***		-0.9738***
		(0.1240)	(0.1281)		(0.1277)
In-App Purchase	0.6382***	0.1282	0.1114	0.5985***	0.0426
	(0.1188)	(0.1431)	(0.1394)	(0.1206)	(0.1399)
Advertisements	0.7313**	0.7458**	0.7002**	0.7242**	0.6899**
	(0.3286)	(0.3358)	(0.3263)	(0.3288)	(0.3268)
Subscription	-0.4678***	-0.5404***	-0.6016***	-0.4798***	-0.6235***
	(0.1283)	(0.1323)	(0.1288)	(0.1286)	(0.1291)
ρ	0.7599***	0.7648***	0.7584***	0.7598***	0.7586***
	(0.0041)	(0.0041)	(0.0041)	(0.0041)	(0.0041)

Observations are app-weeks. Standard errors are in parentheses.
Regressions also include controls for the month, year, market, iOS, version, the age of the app (in weeks), and the squared age of the app.
* p $<$ 0.10, ** p $<$ 0.05, *** p ¡ 0.01

specifications (1) and (2), I find that consumers are far more sensitive to an increase in price from free to \$0.99 (the minimum, non-zero price allowed by the platform), than to an equivalent increase in price from any initial, non-zero price. That is, a developer can expect to lose significantly more potential sales by changing their price from \$0.00 to \$0.99, than by increasing their price from, for example, \$4.99 to \$5.99. Specification (3) captures both the consumer response to the price of a product, and the response to whether the product is free or not. Consistent with the findings from columns (1) and (2), the coefficient on the paid indicator in specification (3) is much larger in magnitude than the coefficient on price. This again suggests a non-linear effect of price on utility.

Columns (4) and (5) of Table 6 show the result of estimating the model using non-linear two-stage least squares (NLTSLS), where I instrument for prices using lagged covariates. The price coefficient in specifications (4) and (5) are larger in magnitude compared to their NLLS analogues (specifications (1) and (3)), though the disparity between the NLLS and NLTSLS results are smaller than the difference between the price and price/paid specifications, which suggests that the non-linearity in price sensitivity is the primary cause for bias in specification (1).

Given these estimates, it is clear that consumers are highly reticent to pay for an app, but that

conditional on paying, they are not particularly price sensitive. Using the results from specification (5), we can see that the additional disutility of increasing the price from \$0.00 to \$0.99 is approximately the same as increasing the price from \$0.99 to \$17.99. Specification (5) is the preferred specification from these results, as it is able to capture this non-linearity in price sensitivity, and I use variations of specification (5) in what follows.

Finally, I find that the inclusion of subscriptions in an app is associated with lower utility. IAPs on the other hand, do not have a statistically significant relationship with utility, which may be due to the relatively limited information consumers receive about the inclusion of and content associated with IAPs. Advertisements appear to have a positive relationship with sales, however with only 11 apps across 4 markets displaying ads, it is possible that this is estimated parameter is picking up on app-specific attributes.

Interpreting the monetization estimates as causal effects requires abstracting away from developers' selection into a particular method (or methods) of monetization—which is unlikely to be a reasonable abstraction. Additionally, the potential bias caused by this selection effect is difficult to sign. For example, it could be that apps that use ads are on average lower quality than apps that use IAPs because viewing an ad does not require the consumer to take an explicit action (purchasing the IAP). On the other hand, as noted above, it could be that ad-financed apps are on average higher quality because consumers are more likely to spend extended periods of time using higher-quality products and therefore monetizing the time spent viewing is optimal relative to trying to sell product add-ons to the consumer. Since apps rarely change their monetization strategy—i.e., which *types* of monetization they use—understanding this selection effect would require modeling apps' entry decisions into the available forms of monetization, which is beyond the scope of this project, and is left to future work.

Updates Table 7 shows the results of estimating the demand model under five different classifications of updates. Across all specifications, I find that updates have a positive, and statistically significant effect on demand, and that there is a non-zero correlation in the quality of apps over time, with the estimates of ρ all falling around 0.758. Thus, while updates don't have an enormous direct effect on an app's quality, the effect of an update persists, via the ξ_{jt} AR-1 process, across multiple periods. These results are consistent with the finding in Foerderer and Heinzl (2017),

Table 7: Effect of Updates on Demand

	(1)	(2)	(3)	(4)	(5)
Update	0.1809***				
	(0.0360)				
Update, Paid		0.1256**			
		(0.0551)			
Update, Free		0.2207***			
		(0.0468)			
Major			0.5962***		
			(0.1277)		
Point			0.1487***		
			(0.0372)		
Feature				0.1580***	
				(0.0521)	
Bug Fix				0.1988***	
				(0.0464)	
Major, Feature					0.5168***
					(0.1600)
Major, Bug Fix					0.7338***
					(0.2103)
Point, Feature					0.1139**
					(0.0551)
Point, Bug Fix					0.1741***
					(0.0475)
Price	-0.0591***	-0.0585***	-0.0593***	-0.0590***	-0.0593***
	(0.0065)	(0.0065)	(0.0065)	(0.0065)	(0.0065)
Paid	-0.9738***	-0.9412***	-0.9732***	-0.9742***	-0.9739***
	(0.1277)	(0.1299)	(0.1277)	(0.1276)	(0.1277)
In-App Purchase	0.0426	0.0434	0.0367	0.0429	0.0366
	(0.1399)	(0.1398)	(0.1399)	(0.1399)	(0.1399)
Advertisements	0.6899**	0.6841**	0.6892**	0.6827**	0.6730**
	(0.3268)	(0.3265)	(0.3268)	(0.3269)	(0.3271)
Subscription	-0.6235***	-0.6328***	-0.6223***	-0.6244***	-0.6237***
	(0.1291)	(0.1292)	(0.1292)	(0.1291)	(0.1292)
ρ	0.7586***	0.7584***	0.7587***	0.7586***	0.7587***
	(0.0041)	(0.0041)	(0.0041)	(0.0041)	(0.0041)

Observations are app-weeks. Standard errors are in parentheses.
Regressions also include the controls for download size, month, year, market, iOS, version, the age of the app (in weeks), and the squared age of the app.
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Ghose and Han (2014), and Comino, Manenti, and Mariuzzo (2018) who all find evidence that updates increase demand.

Column (1) shows this primary finding, that updates increase the demand for apps. In under to better understand how updates affect demand, I consider a number of different classification methods for updates. The first is presented in column (2) of Table 7. In this case, I distinguish between updates made by Paid and Free apps. While the estimated coefficients indicate that updates made to Free apps have a larger effect that updates to Paid apps, I am unable to conclude that Paid and Free apps have a statistically significant difference in the effect of updates on demand.[27]

In column (3), I employ the Version Number classification approach described in Section 4.3. This method classifies an update as a Major update when the the first digit of an app's version number changes (e.g., 2.5 to 3.0), and as a Point update when any subsequent number in the version number changes (e.g., 2.5 to 2.6). I find that Major updates have an approximately four times bigger effect on demand than Point updates do, though both types of updates increase demand.[28] However, it's unclear how to interpret this finding. This is because Major updates conflate two effects. On one hand, Major updates ostensibly represent larger, more important updates to software, as convention has traditionally dictated that a change in the first digit signifies a major revision of the product. However, developers use version numbers to both track the development of an app, and as a marketing tool. For example, announcing a "Brand new version 5.0!" can increase media coverage and word-of-mouth. Furthermore, through the Release Notes classification approach, I find that 36.7% of Major updates in my sample only make minor changes to the product. Thus, I cannot distinguish whether consumers respond more to Major than Minor updates because of the content of the update or if they are merely responding to the fact that there has been a (possibly high profile) update.

In order to better understand whether consumers are responding to the *content* of an update, I next the model with updates categorized based solely on their content, using the Release Notes classification method discussed in Section 4.3. I present the results of this specification in column (4) of Table 7. Under this classification approach, updates that introduce new features or significant changes are labeled as a Feature update, and updates that make only minor adjustments are labeled

[27] Using a Wald test, I cannot reject the hypothesis that updates from Paid and Free app have the same effect at the 10% level ($p = 0.185$).

[28] This difference is statistically significant at the 1% level ($p = 0.001$).

as a Bug Fix update. While both Feature and Bug Fix updates increase demand, I fail to find evidence that Feature and Bug Fix updates have different effects on demand.[29]

Finally, I interact the Version Number and Release Notes classification systems. I present the results of estimating the model with this interaction in column (5) of Table 7. The results in this case reflect that of the last two. Namely, Major updates, regardless of whether they are Feature or Bug Fix updates, have a larger effect on demand than Point updates, again regardless of whether they add features or fix bugs. Within the Major and Point classifications, there is no statistically significant difference between Feature and Bug Fix updates. However, there is a statistically significant difference between each type of Major update and each type of Point update.[30]

One possible explanation for the finding that consumers do not differentiate based on the content of an update is that consumers may not have a strong understanding of the quality of an app. If consumers are poorly informed about the quality of a product, then the specifics of how it has been updated may be uninformative about the degree to which overall quality has changed. In this case, it may be appropriate to think of apps as experience goods, à la Nelson (1970), where consumers make purchase decisions based on an initial belief about the quality of the product, but do not learn the true quality of the product until they have had a chance to use it.[31] Of course, other explanations—e.g., rational inattention to the details of updates—are possible. Whether and how consumers respond to Feature versus Bug Fix updates on the intensive margin remains an open research question—though, one that is currently stymied by significant data collection challenges.

6.2 Fixed Cost Results

Table 8 shows the results of estimating the dynamic app updating model. The cost of a Bug Fix update, θ^{BF}, is \$3,355.97, and the cost of a Feature update is 28% higher at \$4,279.17. While

[29] The difference is not statistically significant at the 10% or lower level (p=0.543).

[30] As in previous cases, I use a Wald test to test for the equality of the coefficients. The resulting p-values are: p=0.410 for Major-Feature vs. Major-Bug Fix, p=0.017 for Major-Feature vs. Point-Feature, p=0.039 for Major-Feature vs. Point-Bug Fix, p=0.004 for Major-Bug Fix vs. Point-Feature, p=0.009 for Major-Bug Fix vs Point-Bug Fix, and p=0.391 for Point-Feature vs Point-Bug Fix.

[31] This hypothesis speaks to a primary concern of developers, that consumers are not informed enough about the products they are purchasing, exemplified by the developer Kevin Hoctor, who notes, "The limited information a prospective customer has prior to a purchase is one of the problems with Apple's App Store. People are supposed to fork over money for apps, but only get to see five screenshots and a few paragraphs of text before making a decision—that just doesn't cut it." (Hoctor, 2013)

Table 8: Dynamic Parameter Estimates

θ^{BF}	3,355.97***
	(276.43)
θ^{F}	4,279.17***
	(412.89)
θ^{ϵ}	383.45**
	(173.18)

Bootstrapped standard errors are reported in parenthesis.

these costs may at first seem small, it's important to view them in context. Across the sample, the average weekly gross revenue is $1,436.55, with the median slightly lower at $1,217.00. Thus, a Minor update costs just over two weeks of revenue, and a Major update costs approximately the same as 3 weeks of revenue. Of course, as shown in Section 6.1, updates increase demand, so developers likely earn their revenue back in less time than the average revenue figures would suggest.

6.3 Discussion

Taken together, the demand and supply results appear to present a contradiction. I find that Feature updates are more expensive to produce than Bug Fix updates (Table 8), while at the same time failing to find evidence that Feature and Bug Fix updates have different effects on demand (Table 7). Furthermore, developers produce a relatively large number of Feature updates, as shown in Table 2b. The fact that developers do in fact regularly produce Feature updates suggests that the different types of updates may indeed serve different strategic purposes, but that those purposes are not focused on spurring new *extensive margin* demand. Instead, Feature updates may have an outsized effect on consumer behavior through the intensive margin of demand. That is, Feature updates may increase the likelihood that a consumer uses an already-owned app in a given period by more than a Bug Fix update, which might in turn result in an increased stream of revenue to the developer via use monetization. This possibility is consistent with the idea that consumers' failure to differentiate between different types of updates may result from an ex-ante lack of information about the full functionality and user experience of an app, but that once they have had time to use the product they learn the true quality of the product, and as a result become more responsive to more substantial improvements to the product.

7 Counterfactual Analysis

Having constructed and estimated a model of consumers' demand for apps and developers' app updating decisions, I now return to my primary research question: How the digitization of consumer goods—characterized by the addition of two new strategic tools—affects firms' updating behavior. In particular, I investigate how both the frequency and content of updates changes in the presence of these tools. Since these tools have been available to developers since the introduction of the App Store, I am unable to take advantage of some form of a natural experiment in order to answer this question. Instead, I conduct counterfactual simulations wherein I "turn off" these two aspects of digitization and simulate how developers would choose to update in their absence. The results of this counterfactual exercise serve as a proxy for how developers would behave in a "traditional" durable goods industry. I compare the results of this counterfactual exercise to observed behavior in order to estimate an effect of digitization on firms' updating behavior.

Model Restrictions In practice, turning off the two aspects of digitization I study means making two specific restrictions in the model. First, I artificially restrict the revenue developers can earn to revenue earned via the extensive, or purchase, margin of demand. Revenue from advertisements, subscriptions, and in-app purchases is all fixed at zero in the counterfactual regime. Put another way, firms no longer consider the effect of updates on consumers' intensive margin behavior. Second, I remove the ability for consumers to receive updates to apps they have already purchased.

In terms of the model developed in Section 3, these two restrictions are represented by first changing the firm's revenue function, R_j from

$$R_j(a_{j,t}, S_{j,t}) = \overbrace{p_j^{Retail} s_{j,t}^{ext} M_{m,t}}^{\text{Direct Sales Revenue}} + (\texttt{Intensive Margin Revenue})$$

to

$$R_j^{counter}(a_{j,t}, S_{j,t}) = \overbrace{p_j^{Retail} s_{j,t}^{ext} M_{mt}}^{\text{Direct Sales Revenue}} \tag{19}$$

and by changing consumers lifetime expected utility from purchasing an app from

$$\Lambda_{ijt} = E\left[\sum_{\tau=t+1}^{\infty} \delta^{\tau-t} \ln\left(1 + e^{X_{j\tau}\beta + \xi_{j\tau}}\right)\right]$$

where consumers have expectations over future updates and future consumer-app match values, to

$$\Lambda_{ijt}^{counter} = \sum_{\tau=t+1}^{\infty} \delta^{\tau-t} \ln\left(1 + e^{X_{j\tau}\beta + \xi_{j\tau}}\right) \tag{20}$$

where the consumers' expectations are only over future consumer-app match values, and not over future updates, and $\xi_{j\tau}$ is not affected by the developer's updating decisions in periods $\tau' > \tau$.

Solving the Model and Forward Simulation For computational reasons, I assume firms make static product updating decisions. That is, in the counterfactual simulations, firms do not consider how a decision in period t will affect the state of the market in period $t+1$. Because of this, I expect that this counterfactual will *understate* the level of updating in the counterfactual regime, and thus will result in my potentially *overstating* the effect of digitization on the frequency of updating. However, the results should still give a clear sense of the direction of the effect.

Accounting for Counterfactual Prices It is necessary to account for the fact that prices are likely to differ between the observed and counterfactual regimes. As discussed in Section 3.3, my model of app updating focuses on developers' product updating decisions, and assumes that prices are fixed. In order to account for the fact that prices will differ in the counterfactual regime, I assume firms play a two-stage game, where in the first stage they set prices, and in the second stage they engage in the infinite-horizon updating game defined in Section 3.3. In practice, I solve this two-stage game backwards, first by calculating each firms' value under each possible pricing equilibrium, and then by determining the Nash equilibrium in the first-stage pricing game. In order to make this exercise computationally feasible I restrict firms to selecting a price from the set $\{0.99, 9.99, 19.99, 29.99, 39.99\}$.

Counterfactual Results I simulate updating in six markets, each with between three and six apps. Table 9 presents the unconditional likelihood of an app in the given market updating in a given period for both the counterfactual regime and the observed, digital regime. The "Counterfactual" columns present the frequency of updates when products are treated as traditional, non-digital consumer goods, and the "Digital" column presents the frequency of app updates under digitization. For each type of updating, Overall, Bug Fix updates, and Feature updates, I present the difference

Table 9: Static Counterfactual Results

	Overall Updates			Bug Fix Updates			Feature Updates			Feature Conditional on Updating		
Market	Counterfactual	Digital	Δ	Counterfactual	Digital	Δ	Counterfactual	Digital	Δ	Counterfactual	Digital	Δ
1	10.6%	13.3%	2.7pp	4.1%	4.1%	0.0pp	6.5%	9.2%	2.7pp	0.61	0.69	0.08
6	12.4%	19.6%	7.2pp	5.1%	6.9%	1.8pp	7.3%	12.7%	5.4pp	0.59	0.65	0.06
9	9.8%	16.6%	6.8pp	4.4%	7.3%	2.9pp	5.4%	9.3%	3.9pp	0.55	0.56	0.01
18	8.1%	6.3%	-1.8pp	5.7%	3.6%	-2.1pp	2.4%	2.7%	0.3pp	0.30	0.43	0.13
19	7.5%	11.4%	3.9pp	6.1%	8.3%	2.2pp	1.4%	3.1%	1.7pp	0.19	0.27	0.09
20	5.7%	7.5%	1.8pp	2.3%	3.8%	1.5pp	3.4%	3.7%	0.3pp	0.60	0.49	-0.10

between the level of updating for digital products and traditional products, Δ. Positive values of Δ indicate an increased level of updating in the Digital regime compared to the Counterfactual regime.

I find that overall updating is higher under digitization in all but one market. These increases cover a range of 1.8 to 7.2 percentage points, representing an increase in the rate of updating of between 25% and 70% relative to a baseline of the "non-digital" counterfactual level. These increases are driven by an increase in both Feature and Bug Fix updates, though the rate of Feature updates increases by slightly more than Bug Fix updates. Indeed, in five of the six markets, the chance of a given update being a Feature update increases by between 10% and 45%. That said, as with the increase in the frequency of updating, I find that in one of the simulated markets the relative frequency of Feature updates falls.

These results speak to the importance of accounting for intensive-margin behavior when studying both digital and digitizing industries. The fact that digitization is found to result in a 25% to 70% increase in the rate of product innovation shows that any analysis of firm behavior in this industry must account for the intensive-margin incentives firms face. This presents both new opportunities—a more nuanced understanding of non-price competition—and new difficulties—data on relevant intensive-margin prices and quantities can be difficult to collect—for economic analysis.

8 Conclusion

As consumer durable goods undergo a process of digitization, firms gain access to new strategic tools, including the abilities to monetize the use of their products and to continue to update their product after a consumer has purchased it. I find that the availability of these two tools leads to

more frequent product updating in Apple's App Store marketplace, and, in general, a higher relative frequency of feature-adding updates compared to more incremental, bug-fixing updates. There is, however, wide variation in these results across markets, which suggests that the specifics of an individual market play an important role in determining how digitization affects firms' incentives to provide more substantial updates, and is an area for future research. Additionally, this paper also identifies a possible concern as products become increasingly digital. My finding that consumers do not differentiate between Feature and Bug Fix updates when making their extensive-margin (purchase) decisions suggests that the saliency of product functionality and, more generally, quality will be an important issue for firms selling digital or partially digital products.

In answering the question of how digitization affects firms' updating behavior, I have contributed a new method for defining markets, which allows economists to use high-dimensional product descriptions (e.g., text, images) in order to group products into markets. In addition, this work has contributed to a growing body of research that uses text as a primary form of data.

As an increasing number of consumer durable good industries undergo digitization it is important to understand how the changes brought on by digitization affect firm behavior. While the effects estimated in this paper are specific to the smartphone application industry, the qualitative results are relevant beyond the App Store. We should expect to see more frequent product updating via digital updates in other industries, an expectation that is already coming to fruition in some markets, such as in the market for electronic cars, or, as noted by the quote that began this paper, in the television set industry.

As for future economic research on this and related industries, there remains substantial work to be done on understanding consumer behavior on the intensive margin of demand (i.e., product use behavior). Understanding how consumers behave *after* they have purchased a product will be increasingly important as more and more products earn a flow revenue from product use via advertisements, subscriptions, and within-product micro-transactions.

References

Anderson, Simon P., André de Palma, and Jacques-François Thisse. 1992. *Discrete Choice Theory of Product Differentiation*. The MIT Press. 60

Apple. 2016. "Apple - WWDC 2016 Keynote." URL https://www.youtube.com/watch?v=n5jXg{_}NNiCA. 3

———. 2017. "Developer earnings from the App Store top $70 billion." URL https://www.apple.com/newsroom/2017/06/developer-earnings-from-the-app-store-top-70-billion/. 3

Aryal, Gaurab, Federico Ciliberto, and Benjamin T. Leyden. 2019. "Public Communication and Tacit Collusion in the Airline Industry." 6

Bajari, Patrick, Lanier Benkard, and Jonathan Levin. 2007. "Estimating dynamic models of imperfect competition b." *Econometrica* 75 (5):1331–1370. 35

Bajari, Patrick, Jeremy T. Fox, and Stephen P. Ryan. 2008. "Evaluating wireless carrier consolidation using semiparametric demand estimation." *Quantitative Marketing and Economics* 6 (4):299–338. 19

Baker, Scott R., Nicholas Bloom, and Steven J. Davis. 2016. "Measuring Economic Policy Uncertainty." *The Quarterly Journal of Economics* 131 (4):1593–1636. 6

Berry, Steven T. 1994. "Estimating Discrete-Choice Models of Product Differentiation." *The RAND Journal of Economics* 25 (2):242. 33

Berry, Steven T., James Levinsohn, and Ariel Pakes. 1995. "Automobile Prices in Market Equilibrium." *Econometrica* 63 (4):841–890. 10

Bresnahan, T, J Orsini, and Pl Yin. 2014. "Platform Choice By Mobile Apps Developers." *NBER working paper* . 10

Bresnahan, Timothy, Xing Li, and Pai-ling Yin. 2016. "Paying Incumbents and Customers to Enter an Industry: Buying Downloads." *Working Paper* 94305. 10, 20

Brynjolfsson, Erik, Yu (Jeffrey) Hu, and Michael D. Smith. 2003. "Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers." *Management Science* 49 (11):1580–1596. 19

———. 2010. "The Longer Tail: The Changing Shape of Amazon's Sales Distribution Curve." *Social Science Research Network* (September):1–13. 21

Chevalier, Judith and Austan Goolsbee. 2003. "Measuring Prices and Price Competition Online: Amazon.com and BarnesandNoble.com." *Quantitative Marketing and Economics* 1 (2):203–222. 19

Comino, Stefano, Fabio Maria Manenti, and Franco Mariuzzo. 2018. "Updates Management in Mobile Applications: iTunes vs Google Play." *Journal of Economics & Management Strategy* (May):1–33. 10, 42

ComScore. 2016. "U.S. Smartphone Subscriber Market Share." URL https://www.comscore.com/Insights/Rankings/comScore-Reports-January-2016-US-Smartphone-Subscriber-Market-Share. 8

Davis, Jason P, Yulia Muzyrya, and Pai-Ling Yin. 2014. "Experimentation Strategies and Entrepreneurial Innovation: Inherited Market Differences in the iPhone Ecosystem." *INSEAD Working Papers Collection* (24):1–40. 10

Doraszelski, Ulrich, Gregory Lewis, and Ariel Pakes. 2016. "Just Starting Out: Learning and Equilibrium in a New Market." *NBER Working Paper* . 35

Doraszelski, Ulrich and Mark Satterthwaite. 2010. "Computable Markov-perfect industry dynamics." *RAND Journal of Economics* 41 (2):215–243. 16

Eizenberg, Alon. 2014. "Upstream innovation and product variety in the U.S. home PC market." *Review of Economic Studies* 81 (3):1003–1045. 5

Ericson, Richard and Ariel Pakes. 1995. "Markov-perfect industry dynamics: A framework for empirical work." *The Review of Economic Studies* 62 (1):53–82. 15

Ershov, Daniel. 2018. "The Effect of Consumer Search Costs on Entry and Quality in the Mobile App Market." 10, 19, 21

Evans, David S. 2011. "The antitrust economics of free." *Competition Policy International* 7 (1):70–89. 25

Fan, Ying. 2013. "Ownership Consolidation and Product Characteristics: A Study of the US Daily Newspaper Market." *American Economic r* 103 (5):1598–1628. 5

Foerderer, Jens and Armin Heinzl. 2017. "Product Updates: Attracting New Consumers versus Alienating Existing Ones." *SSRN Electronic Journal* URL https://www.ssrn.com/abstract=2872205. 4, 40

Gandhi, Amit, Zhentong Lu, and Xiaoxia Shi. 2013. "Estimating Demand for Differentiated Products with Error in Market Shares." 56

Gans, Joshua S. 2012. "Mobile Application Pricing." :1–24. 10

Garg, R and Rahul Telang. 2012. "Estimating App Demand from Publicly Available Data." *Available at SSRN 1924044* 37 (4):1–22. 19, 20

Gentzkow, Matthew, Bryan T. Kelly, and Matt Taddy. Forthcoming. "Text as Data." *Journal of Economic Literature* . 6

Ghose, A.a and S.P.b Han. 2014. "Estimating demand for mobile applications in the new economy." *Management Science* 60 (6):1470–1488. 10, 19, 20, 21, 27, 42

Goettler, Ronald L and Brett R Gordon. 2011. "Does AMD Spur Intel to Innovate More?" *Journal of Political Economy* 119 (6):1141–1200. 5

Goode, Lauren. 2016. "Apple's new subscription offerings are now available to App Store developers." URL https://www.theverge.com/2016/9/2/12774758/apple-developers-app-store-new-subscription-rules. 8

Gowrisankaran, Gautam and Marc Rysman. 2007. "Dynamics of Consumer Demand for New Durable Goods." *Journal of Political Economy* 120 (6):1173–1219. 4

Hoberg, Gerard and Gordon Phillips. 2016. "Text-Based Network Industries and Endogenous Product Differentiation." *Journal of Political Economy* 124 (5):1423–1465. URL http://www.journals.uchicago.edu/doi/10.1086/688176. 6, 27

Hoctor, Kevin. 2013. "In-App Purchase - The Future is Here." URL http://blog.hoctor.com/in-app-purchase-the-future-is-here/. 43, 60

Honnibal, Matthew and Ines Montani. 2017. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." *To appear* . 56

Kesler, Reinhold, Michael Kummer, and Patrick Schulte. 2017. "Access to User Data , Market Power and Innovation in Online Markets: Evidence from the Mobile App Industry." 14

Lambert, Fred. 2017. "Tesla pushes new Autopilot 2.0 update with truly "silky smooth" control algorithm." 3

Lee, Robin S. 2013. "Vertical Integration and Exclusivity in Two-Sided Markets." *The American Economic Review* 103 (7):2960–3000. 5

Liu, Yongdong. 2017. "Mobile App Platform Choice." :1–66. 10, 20

Matthews, Lee. 2015. "Tesla P85D software update reduces the car's already insane 0-60 time." URL https://www.geek.com/apps/tesla-p85d-software-update-reduces-the-cars-already-insane-0-60-time-1614741/. 3

Nekipelov, Denis, Minjung Park, and Yongdong Liu. 2013. "Timely versus quality innovation: The case of Mobile Applications on iTunes and Google Play." 10

Nelson, Phillip. 1970. "Information and Consumer Behavior." *Journal of Political Economy* 78 (2):311–329. 43, 60

Newman, John M. 2015. "Antitrust in Zero-Price Markets: Foundations." *University of Pennsylvania Law Review* 164 (1). URL http://heinonline.org/HOL/Page?handle=hein.journals/pnlr164{&}id=153{&}div={&}collection=. 26

Novikov, Andrei. 2018. "annoviko/pyclustering: pyclustering 0.8.2 release." URL https://doi.org/10.5281/zenodo.1491324. 28

Patel, Nilay. 2019. "Taking the smarts out of smart TVs would make them more expensive." URL https://www.theverge.com/2019/1/7/18172397/airplay-2-homekit-vizio-tv-bill-baxter-interview-vergecast-ces-2019. 2

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830. 57

Pelleg, Dau and Andrew Moore. 2000. "X-means: Extending K-means with Efficient Estimation of the Number of Clusters." In *In Proceedings of the 17th International Conf. on Machine Learning*. Morgan Kaufmann, 727–734. 28, 58

Perry, Charles. 2015. "The Shape of the App Store." URL dazeend.org/2015/01/the-shape-of-the-app-store/28/. 19

Quan, Thomas W. and Kevin R. Williams. 2015. "Product variety, across market demand heterogeneity and the value of online retail." *Working Paper* November. 56

Rousseeuw, Peter J. 1987. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." *Journal of Computational and Applied Mathematics* 20 (C):53–65. 28

Spencer, Graham. 2015. "A Beginner's Guide to App Store Pricing Tiers." URL https://www.macstories.net/stories/a-beginners-guide-to-app-store-pricing-tiers/. 8

Sweeting, Andrew. 2013. "Dynamic Product Positioning in Differentiated Product Markets: The Effect of Fees for Musical Performance Rights on the Commercial Radio Industry." *Econometrica* 81 (5):1763–1803. 5, 16

Wu, Alice H. 2017. "Gender Stereotyping in Academia : Evidence from Economics Job Market Rumors Forum." (August). 6

Yin, Pai Ling, Jason P. Davis, and Yulia Muzyrya. 2014. "Entrepreneurial innovation: Killer apps in the iPhone Ecosystem." *American Economic Review* 104 (5):255–259. 10

Appendices

A Sample Restrictions

A.1 Companion versus Product Apps

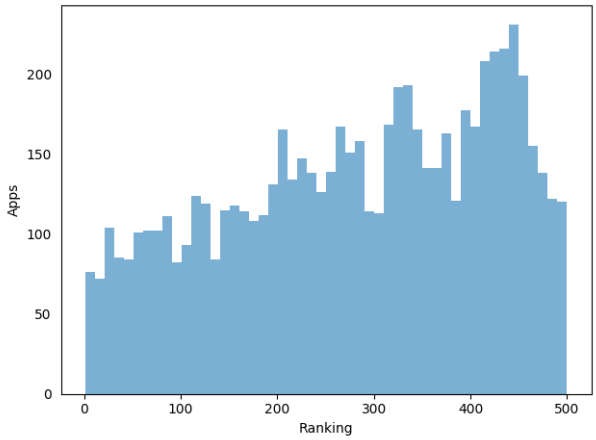
Apps can be classified as either companion or product apps. Companion apps are those that work in conjunction with an existing product or service. For example, the American Airways app, which offers detailed flight information, flight check-in, and a consumer's boarding pass on the day of the flight, serves as a companion to American Airways flights. On its own, the app provides no value. Similarly, the brick-and-mortar retailer Target's app serves as a companion to shopping at Target either in the store, where it provides information about what aisle certain products are available in, or online.

Product apps are those that are themselves the primary product being offered. The weather app, Dark Sky, which provides forecasts and weather radar, is produced as a standalone product. This distinction is important, because while product apps can reasonably be expected to be developed in a standard profit-maximizing way, there is no such expectation for a companion app as it is just a (possibly small) piece of a much larger product offering. Modeling the pricing and updating of a companion app would require modeling the larger product offering. For example, decisions made regarding Target's apps could be viewed as profit maximizing from the perspective of Target's overall retail operation, but directly observing the revenue collected through the app would likely not rationalize the costs of development. Further, from a demand perspective, consumer preferences for a companion app will be highly dependent on products, pricing, and competition outside of the app industry—demand for companion apps is largely, if not entirely, a derived demand.

A.2 Accounting for Abandoned Apps

A second issue in defining a sample in the App Store stems from the fact that, as discussed in Section 2, entry into the App Store is relatively inexpensive. Thus, many apps have such a low level of sales (never, or rarely achieving a sales ranking associated with non-zero sales) that it is hard to consider them as behaving strategically in the marketplace. It is important to re-emphasize the fact that, since app development primarily consists of fixed costs, developers face minimal costs

Figure 10: Distribution of Productivity Apps' Best Ranking



in keeping an abandoned product in the store. Unlike the production of physical goods, continuing to sell an abandoned, or no longer actively developed app does not, in general, result in continued variable expenses.

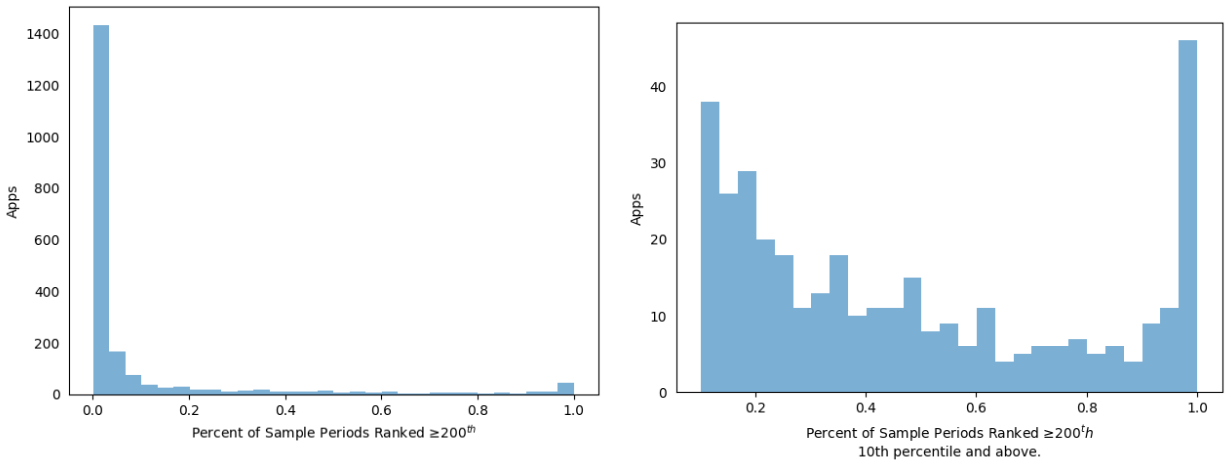
Considering, for a moment, only the apps in the Productivity category, only 10.6% ever reach a sales ranking better than 500^{th} in the sample period, and only 3.2% ever reach a ranking better than 200. Fig. 10 shows the distribution of apps' best rankings for those that reach a ranking of at least 500 on at least one occasion. However, even among apps that do reach a particular ranking threshold, few apps are able to repeatedly maintain sales above that threshold. Fig. 11 shows that among apps that achieve at least a ranking of 200, nearly all spend less than 10% of the sample period at or exceeding that threshold.

There are several possible reasons for why so many apps enter the store, yet never achieve a meaningfully high ranking. One possibility is that there is a high degree of randomness in app store success, and so the nearly 90% of Productivity apps that entered the store but never reached a ranking of at least 500 had an ex-ante positive expected profit, but upon entry, "drew" a bad shock. Note that this disparity between expected and realized outcomes could be due to a high level of volatility in the marketplace, or to a general high degree of uncertainty regarding consumer demand on the part of the developer. Another possibility, which explains *some* of the consistently low-ranking apps in the store, though certainly not all, is that many apps are hobbies or passion projects for developers. That is, some apps may only exist to scratch a particular developer's own itch.

Figure 11: Persistence Above Ranking Threshold of 200

(a) Periods ranked above 200

(b) Periods ranked above 200 (10^{th} percentile and above.)



This issue would be especially problematic when estimating the structural model in Section 3, because keeping abandoned apps in the sample would introduce an enormous number of zeroes in the market-share data, which is inconsistent with the logit demand model developed in Section 3.2. While methods do exist for allowing this, they generally require an assumption of sampling error in shares in order to explain zero-shares (see Gandhi, Lu, and Shi (2013)). This is inappropriate in the current context, as many products on the App Store actually do have zero sales throughout much of, and in many cases, all of, the sample period.[32] I employ the restrictions outlined in Section 4 to limit my sample to apps that are actively competing on the platform, and that do not face this massive zero-sales concern.

B Natural Language Processing

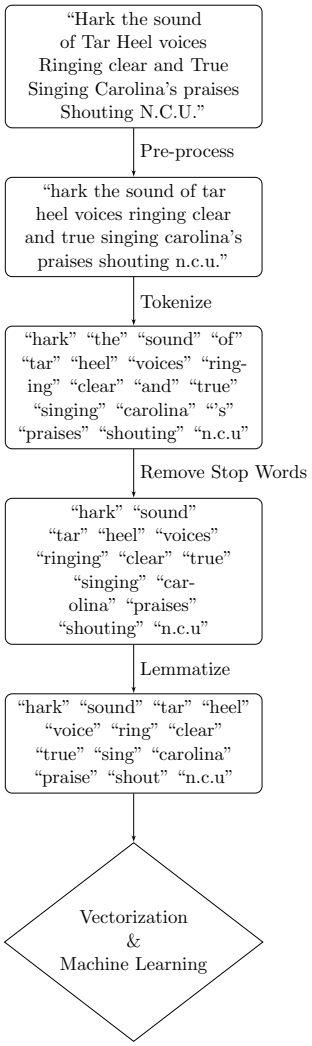
To classify updates and define markets using text data, I must first clean and transform the data to use in the relevant machine learning algorithms. Essentially, these unstructured text documents must be converted to vectors, to which standard mathematical operations can be performed. This section outlines my approach for doing this. In practice, I use a combination of the spaCy and Scikit-learn libraries to implement the text processing work outlined below (Honnibal and Montani,

[32] Quan and Williams (2015) develop an alternative method for handling zero-shares, but it requires cross-market products. As there is no distinction between local app markets and a national app market, applying their method is not possible in this case.

2017; Pedregosa et al., 2011).

Each text document, whether an update's release notes or an app's description, is first tokenized. Tokenization is the practice of reducing a document to a set of individual word "tokens." I next lemmatize these tokens, which reduces the words to their dictionary form. Finally, I remove all punctuation and non-letter characters (e.g., bullet points and emojis), a standard list of common English "stop words," such as "a" and "the," as well as a list of words related to Apple and its products, such as "iPhone" and "iOS." Removing these words allows me to focus my analysis on the words that indicate the purpose of app updates, rather than, for example, the frequency with which a developer mentions Apple's platform. Fig. 12 provides an example of applying these steps to a short text.

Figure 12: Natural Language Processing Flow Chart



Additionally, I optimally select a number of other options, including allowing for multi-word phrases ("n-grams"), removing words that only appear a few times across all descriptions (such as proper nouns), and removing any words that appear in a large percentage of the descriptions. The parameters for these options are chosen by iterating over a parameter grid, and choosing the best vector of parameters by maximizing a scoring metric. Ultimately, this process produces a "bag of words" for each description. Given the bag of words for each description, I map each document to a vector space where each element of a vector represents a particular word or n-gram. Rather than simply using an indicator for which words are present in a document, I use term frequency, inverse-document frequency weighting. This weighting procedure places greater emphasis on words that appear frequently in a document relative to the word's overall frequency in the set of release notes, under the assumption that such words are a more valuable signal for differentiating a particular set of release notes than words that appear frequently across all documents. The weight for

word i in release notes j is

$$weight_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \times \ln\left(\frac{N}{n_i}\right) \tag{21}$$

where f_{ij} is the frequency of word i in document j, N is the total number of release notes (i.e., the total number of updates), and token i appears in n_i of the release notes. This process results in a set of sparse, high-dimensional vectors representing each release note.

C X-Means Clustering

To define markets, I use the X-means clustering algorithm, which is itself an extension of the k-means clustering algorithm. I describe both below. The X-means clustering algorithm was developed by Pelleg and Moore (2000).

k-Means Clustering Given a set of description vectors $X = \{x_1, \ldots, x_N\}$, and an exogenously determined number of clusters k, k-means clustering allocates each app description $x \in X$ to one of the k clusters, characterized by a centroid in the vector space defined by X. Broadly, x_i is in cluster j if and only if it is more similar to the x's in j than to those in any other cluster.

More precisely, define

$$\gamma_{ij} = \begin{cases} 1, & x_i \text{ is in cluster } j \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

and let δ_j be the centroid of cluster j. Letting $\gamma = \{\gamma_{ij}\}$ and $\delta = \{\delta_j\}$, the k-means clustering algoritm then solves

$$\min_{\gamma,\delta} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{k} \gamma_{ij} ||x_i - \delta_j||^2 \tag{23}$$

Eq. (23) is solved using a recursive, two-step process. In the first step it optimally selects γ given δ, and in the second step it optimally selects δ given the γ determined in the first step. Specifically, the objective in each step is

$$\text{Step 1:} \qquad \min_{j'} ||x_i - \delta_{j'}||^2 \ \forall i \tag{24}$$

$$\text{Step 2:} \qquad \sum_{i=1}^{N} \gamma_{ij}(x_i - \delta_j) = 0, \ \forall j \tag{25}$$

This two-step process is repeated until the convergence of γ and δ.

X-Means Clustering Given a minimum number of clusters, k_{min}, the algorithm conducts k-means clustering on the sample, as described above. Then, for each cluster, run the k-means algorithm with $k = 2$. The Bayesian Information Criterion (BIC) for the original cluster, and the split cluster. The BIC is used to determine whether each cluster should be split, and is therefore used to determine the new number of clusters. The process then repeats, using the new number of clusters. This continues until convergence.

D Intensive Margin of Demand

In this appendix section I outline a simple model of intensive-margin, or use demand for apps. This model is useful for both understanding the incentives firms face regarding the intensive margin, and the model is used in order to simulated consumers' lifetime expected utility from owning an app, which is used in the estimation of the demand model. See Sections 3.2 and 5.1 for more details on the extensive margin model and estimation procedure.

Model Each period consumers choose whether to use any of the apps they already own. That is, consumer i, owning apps $j \in J_i$ makes $|J_i|$ independent decisions of whether to use each app or not. Specifically, consumers earn utility w_{ijt} from using app j at time t where

$$\begin{aligned} w_{ijt} &= X_{jt}\tilde{\beta} + \tilde{\xi}_{jt} + \tilde{\epsilon}_{ijt} \\ w_{i0t} &= \epsilon_{i0t} \end{aligned} \tag{26}$$

and the fraction of previous purchasers who choose to use app j in period t is s_{jt}^{int}, which is derived in the same way as s_{jt}.

$$s_{jt}^{int} = \frac{\exp(X_{jt}\tilde{\beta} + \tilde{\xi}_{jt})}{1 + \sum_k \exp(X_{kt}\tilde{\beta} + \tilde{\xi}_{kt})}$$

Eq. (26) differs from Eq. (1) in three ways. First, consumers do not consider the price of the app when making a use decision, as they have already paid that one-time cost. Second, $\Lambda_{jt} = 0$, reflecting the fact that period-specific use decisions are independent across periods.[33]

Finally, a primary concern developers have about selling entirely digital goods is that consumers may not be well-informed about the quality of apps prior to purchase.[34] In light of this, I do not restrict the intensive-margin preference parameter $\tilde{\beta}$ and unobserved product quality $\tilde{\xi}$ to be equal to their extensive-margin counterparts. Furthermore, I assume that any difference in $(\tilde{\beta}, \tilde{\xi})$ is unknown to the consumer prior to making a purchase, which fits with the notion that apps may in some ways be experience goods à la Nelson (1970).

Derivation of Lifetime Expected Utility (Λ_{jt}) Consumers anticipate that apps will be updated in future periods, and thus consider the future utility of using the product when making their purchase decision. Specifically, consumers have expectations over future updates, and over future consumer-app match values ϵ_{ijt}, which are relevant to the consumer's period-specific, intensive margin, use decision. Thus, conditional on the developer's updating choices, the expected value of owning previously purchased app j in period τ, when the consumer can either choose to use app j or not is the expected utility of Eq. (26). Since the error in the intensive margin demand model is assumed to be i.i.d. Extreme Value Type 1, this expectation, is defined by the standard "logsum" formula for the double exponential (Anderson, de Palma, and Thisse, 1992), which is

$$Ew_{ij\tau} = \ln\left(1 + e^{X_{j\tau}\beta + \xi_{j\tau}}\right) \tag{27}$$

Therefore, the expected future, discounted value of owning app j, Λ_{jt} is the discounted sum of

[33] It may be the case that in some app markets period t use affects the utility received in periods $\tau > t$. This is not considered here due in to data limitations. Games, the primary category where one might imagine this matters is not considered in this paper.

[34] E.g., "The limited information a prospective customer has prior to a purchase is one of the problems with Apple's App Store. People are supposed to fork over money for apps, but only get to see five screenshots and a few paragraphs of text before making a decision — that just doesn't cut it." (Hoctor, 2013)

Eq. (27), with expectations over the developer of app j's future updating decisions. Namely,

$$\begin{aligned}\Lambda_{ijt} &= E\left[\sum_{\tau=t+1}^{\infty} \delta^{\tau-t} E w_{ijt}\right] \\ &= E\left[\sum_{\tau=t+1}^{\infty} \delta^{\tau-t} \ln\left(1+e^{X_{j\tau}\beta+\xi_{j\tau}}\right)\right]\end{aligned} \tag{28}$$

where δ is the discount factor. The symmetry of the model implies, $\Lambda_{ijt} = \Lambda_{jt}\ \forall i$.

Note that the formation of the expected future value Λ is entirely a function of consumers' extensive margin preferences β and the extensive-margin unobserved product quality ξ. While the values of β and ξ may differ between the two margins, consumers are only able to form expectations over what they know *prior* to purchasing a given app. Since the intensive-margin values, if they differ at all, are only learned *after* the point-of-sale, Λ cannot account for these differences.

Calculating $\Lambda_{j,t}$ I calculate $\Lambda_{j,t}$ using forward simulation. Broadly, the process of calculating $\Lambda_{j,t}$ is similar to what is used to estimate the supply model. For each candidate parameter in the demand estimation I do the following: For each observation, simulate forward 50 periods, calculating $Ew_{ij\tau}$ for each simulated period τ. Then, I take the discounted sum of the 50 $Ew_{ij\tau}$ values. I repeat this process 50 times, and average across all of those simulations to get Λ_{jt}. In light of the potentially high computational cost of these simulations, I calculate and save each simulated path for each observation outside of the estimation procedure. Then, for each candidate parameter in the estimation procedure I calculate $Ew_{ij\tau}$ for each period of each path, take the discounted sum for each path, and then average across all of the simulations.

Calculating Firm Revenue Given w_{ijt} it is possible to define the number of "active users" of app j, AU_{jt}. Specifically,

$$AU_{jt} = s_{jt}^{ext} M_{m\tau} + s_{jt}^{int} \sum_{\tau=t_0}^{t-1} s_{j\tau}^{ext} M_{m\tau} \tag{29}$$

That is, $AU_{j,t}$ is the number of consumers who purchased the product in period t, plus the number of customers who have previously purchased the app and have chosen to use it in period t.

Given this measurement of an app's *intensive-margin market size*, one can imagine models similar to the intensive use model that characterize the share of an app's active users that purchase

an IAP ($s_{j,t}^{IAP}$) and that purchase, or make a continued payment for a subscription ($s_{j,t}^{sub}$), as appropriate depending on the app's monetization strategy. In most cases it would be reasonable to assume all active users who use the app in period t view any ads presented in the app, so $s_{j,t}^{ads} = s_{j,t}^{int}$. In the interest of brevity, and since such models do not enter into the estimation approach used in this paper, I omit a more precise discussion of these models.

Given this, R_j is defined as

$$R_j(S_{kjt}) = \overbrace{p_j^{Retail} s_{j,t}^{ext} M_{mt}}^{\text{Sales Revenue}} + \overbrace{\left(\sum_l (p_{j,l}^{IAP} s_{j,l,t}^{IAP}) + p_j^{Ads} s_{j,t}^{ads} + p_j^{sub} s_{j,t}^{sub} \right) AU_{j,t}}^{\text{Use Revenue}} \tag{30}$$

where, as noted above, s_{jt}^{x} is the share making a purchase of $x \in \{IAP, Sub\}$, as appropriate, and $AU_{j,t}$ is the set of active users for app j in period t. l indexes the number of IAPs offered by the app, which in many cases is greater than one. Note that marginal costs are assumed to be 0, as discussed in Section 2. Because I cannot observe intensive-margin behavior, I use a reduced form approximation of revenue when estimating the supply model (see Section 5.2.