Consumer Preferences for Product Updates Under Digitization: The Demand for Smartphone Applications

Benjamin T. Leyden*

Preliminary, Please Do Not Cite

August 16, 2017

Abstract

The digitization of durable goods gives firms the ability to monetize and update already purchased products, shifting firms' focus from pushing consumers toward buying replacement products to locking in consumers through incremental updates. As a first step in understanding the effects of digitization, I investigate consumers' preferences for product updates in Apple's app ecosystem. Using a demand model that differentiates between large, feature-adding updates, and smaller, bug-fixing updates, I find that updates have a positive, non-transitory effect on demand, but that consumers do not differentiate based on the content of updates. Additionally, I find that consumers dislike paying for apps, but that conditional on paying, price sensitivity is low.

Keywords: demand estimation, digitization, durable goods, innovation, mobile apps
JEL Classification: D12, L10, L63, O30

*The University of Virginia, Department of Economics, btl3yv@virginia.edu. I am grateful to my advisors Simon Anderson and Federico Ciliberto for their guidance. This work has benefited from additional conversations with many faculty members and graduate students at the University of Virginia. I am indebted to the app developers who have spoken with me about their industry, and, in a number of 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

Consumer durable goods are becoming increasingly software-dependent and internet-connected. This *digitization* of durable goods is changing the manner in which consumers and firms interact. Whereas this interaction has typically ended at the point of sale, durable goods manufacturers today are able to continue to monetize consumers, for example, by including advertisements in the product or by selling additional, instantly available features to consumers. Furthermore, the digital nature of these products makes it possible for firms to update a product long after the consumer has purchased it. Together, these two changes — the increased 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 the nature of competition in durable good industries by shifting emphasis away from trying to get consumers to purchase replacement products, in favor of trying to keep consumers *locked in* so that a firm can continue to monetize an existing user's use of their current product.

This paper takes a first step in understanding how digitization is affecting durable goods markets by investigating whether and how consumers respond to digital product updates, and consumers' preferences over the various forms of post-sale monetization in smartphone application (app) markets on Apple's mobile platform. While many durable good industries are in the early stages of digitizing, apps are fully digitized durable goods, and thus provide an early opportunity to study these questions as other industries continue to make this transition. In my model, I allow product updates to affect apps' qualities and for the effect of an update to persist over multiple periods. The model also allows for multiple types of updates, capturing the fact that different app updates serve different purposes. In order to conduct this analysis, I develop a method for defining markets in so-called "long-tail" product industries, those with thousands, if not millions, of niche products, where traditional methods may fail.

The digitization of durable good industries is rapidly accelerating. Of particular note is a new paradigm in durable goods markets that is often referred to as the Internet of Things. The Internet of Things consists of products in which internet-connected software is an integral component of the product. This connectedness provides firms with the technological ability to easily send updates to consumers. For example, manufacturers of so-called "smart bulbs" are able to send updates to internet-connected lightbulbs, adding features that allow a bulb to change a room's lighting as the

weather changes, for particular activities such as watching a movie, and even when the consumer's favorite sports team wins a game. Digitization is also evident in the car industry. Electronic car manufacturer Tesla released a free, downloadable update in January of 2015 that increased the 0 to 60 miles per hour acceleration time of their model P85D car by approximately a tenth of a second (Matthews, 2015). 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 — they are fully digital. Apps provide a unique opportunity to study the effects of the changes brought on by digitization — the increased number of options for monetization, and the ability to update frequently and long past the point of purchase — without facing additional modeling and empirical challenges brought forth by the specifics of a particular industry's non-digital production costs and technologies.

Additionally, app markets are themselves compelling. 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.[1] 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 $70 billion in revenue (Apple, 2016).[2]

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. In fact, modern apps have undergone digitization 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.

As the technologies in smartphones have advanced, the different roles phones and tablets play in users' lives have expanded, and with that, there has also been a dramatic expansion in the role of apps in users' lives. While the initial 500 App Store apps consisted of games, some productivity tools, and various utilities, today apps can be used to control various household appliances and

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

[2] In a mid-2016 address to app developers, Apple CEO Tim Cook announced that nearly $50 million had been given to developers. As is discussed in Section 2, developers receive 70% of the revenue their app collects.

electronics, to broadcast information worldwide at the click of a button, and to track health and interact with doctors.

This paper extends the literature on consumer demand for durable goods to digitized durable goods, and in particular, to goods that are updated without requiring an additional purchase from the consumer. Gowrisankaran and Rysman (2012) develop a model of dynamic demand for durable goods, which serves as the basis for much of the recent research on the subject.[3] Lee (2013) studies consumer demand for video game software (in conjunction with the corresponding hardware), but in his model software is considered to be a fixed product, with period-to-period changes in the lifetime expected utility of a software purchase occurring only due to temporal preference changes. Consumers in his model have expectations over future values of the lifetime expected utility of purchasing a video game, thus reducing the consumer's dynamic purchase problem to an optimal stopping problem.[4]

This paper differs from Lee (2013) and other recent papers on demand for durable goods as it allows for quality changes to an existing product. That is, unlike the software in Lee (2013), the inherent quality or characteristics of a product are not fixed, and unlike Gowrisankaran and Rysman (2012), firms are able to update products without requiring the consumer to purchase the newer version.

Developing and estimating a dynamic demand model where products can be updated period-to-period and along multiple dimensions presents significant challenges, and current methodologies are ill-equipped to deal with such a model. Notably, not all recent work on durable good industries has used a dynamic model. Eizenberg (2014) studies the effect of CPU innovation on product variety in the PC industry. In doing so, Eizenberg employs a static model of both supply and demand as current methodologies are unable to account for multi-product firms' endogenous pricing and product choices. Additionally, given the supply-side changes brought on by digitization (the increased ability to monetize a product and the ability to update a product past the point of purchase), there is value in developing a demand model that is able to capture demand in a tractable way that can serve as an input to a supply model.[5] With this is mind, this paper treats consumer

[3] See Schiraldi (2011) and Zhao (2006).

[4] See Zhou (2016) for another study of the video game industry, which takes a similar approach to studying software.

[5] This work is ongoing.

demand as static, but allows for the effect of product updates on consumers' utilities to accumulate over time. Developing and estimating a dynamic demand model for apps with endogenous updating is left for future work.[6]

Additionally, 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), Liu, Nekipelov, and Park (2013)), consumer search (Ershov (2016)), 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 et al. (2014) find that the level of updating by the most successful apps, deemed "killer apps," varies by category. Specifically, highly successful game apps update infrequently, while the most successful non-game apps frequently update. 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 first choose whether to download a Free or Paid app, then a category of apps, and finally which app to buy.

The closest paper to this one is Comino, Manenti, and Mariuzzo (2016), 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.

Methods developed in this paper to define markets in the app industry are relevant to research on so-called long-tail product industries, where there are many products, most of which lack mainstream appeal. While there has been research on entry (see Aguiar and Waldfogel (2016)) and price competition (see Quan and Williams (2015)) in long-tail industries, this paper is, to

[6]Goettler and Gordon (2011) estimate a model of the supply and demand for CPUs that endogenizes product updating with forward-looking consumers. However, the updates are only in one dimension.

my knowledge, the first to systematically define markets beyond adopting the (often enormous) pre-defined categories or genres from the industry marketplace.

In this paper, I first develop a method for defining app markets, where apps are partitioned into categorical markets based on the textual descriptions developers provide to prospective users. Given this industry structure, I then estimate a discrete choice logit model of demand for smartphone apps on Apple's iOS platform. The model incorporates app updates into consumers' utilities in a manner that abstracts from the highly differentiated (and often non-overlapping) specifics of app updates. Updates are classified along two dimensions. First, based on changes in the app's version number, and second, using textual descriptions of each update written by the developer. These descriptions provide information on the specific content of each update, which allows me to classify each update either a feature-adding or bug-fixing update.

Estimating the model on a sample of productivity apps from Apple's App Store, I find strong period-to-period persistence in the quality of apps. Updates increase demand, though new consumers do not appear to differentiate based on the content of an update as the effect on demand of both Bug Fix and Feature updates is approximately the same. Despite this, Bug Fix updates are more common than Feature updates, which suggest that updates may serve additional purposes beyond simply spurring new purchases. For example, updates, of one or both types, may also be an important tool for maintaining consumer lock-in, focused on keeping consumers engaged with the product – and thus contributing to advertising or other forms of revenue – after they have purchased it. Finally, I find that there is a strong disutility associated with paying for apps; but that, conditional on paying, consumers do not appear to be particularly price sensitive.

Section 2 provides an overview of the smartphone app industry, within the context of Apple's iOS platform. Following that, Section 3 describes the data used in this paper. Sales data is hard to come by in the app industry, and Section 3.1 discusses how I overcome that issue. Section 3.2 covers how I classify app updates as either Bug Fix or Feature updates. A primary contribution of this paper is the development of a methodology for defining markets within app marketplaces, and this is discussed in Section 3.3. Section 4 presents a model of app demand that allows for multiple types of updates, and allows the effect of updates to persist over multiple periods. Finally, Section 5 presents the results of this estimation and Section 6 concludes.

Table 1: U.S. Market Shares of Mobile Platforms

Platform	Share (%)
Android	52.9
Apple	43.3
Microsoft	2.7
Blackberry	1.0
Other	0.1

Source: comScore (2016). Data current as of March, 2016.

2 Institutional Background

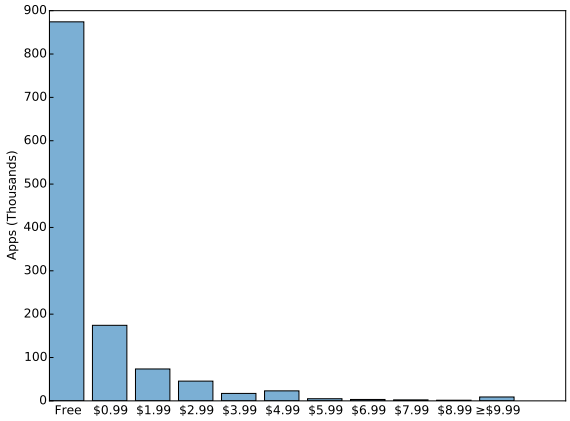
The mobile app industry is characterized by a small number of distinct platforms, or mobile operating systems. The most prominent platforms in the United States are Apple's iOS, Google's Android, Microsoft's Windows, and Blackberry. The U.S. market shares of these platforms are shown in Table 1.

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 typically the primary (and in some cases only) way to download or purchase apps for use on the device. Consumers in this market generally single-home, though some number of consumers purchase a phone associated with one platform and a tablet associated with another platform.

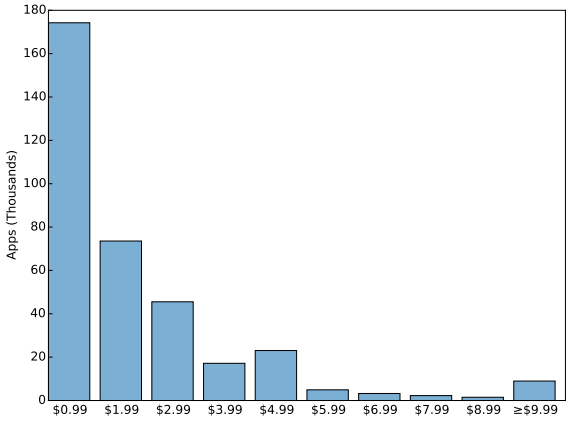
For the remainder of this paper, I limit discussion to Apple's iOS platform and its corresponding App Store marketplace. While the discussion below is specific to Apple, the general structure of the platform is applicable to the other mobile platforms.

[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. Microsoft's Windows platform is unique in that it runs on both hand-held devices (phones and tablets) and traditional laptop and desktop computers.

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)

2.1 Apple's iOS Platform

App developers submit their iOS apps to Apple for inclusion in the App Store. The App Store is the only approved marketplace for selling an iOS app.[8] Following submission, an app must go through an approval process to ensure it does not violate a set of regulations Apple places on apps. Once approval is granted, the app is made available on the App Store.

Developers set the price of their app, and are able to change that price at any time. That said, price changes tend to coincide with the release of app updates. Developers are restricted to a finite, discrete set of prices. Apple allows developers to choose from 94 possible prices. An app 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] Figures 1a and 1b show a snapshot of the distribution of prices in the App Store.

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 has largely fallen out of fashion,

[8] Apps can be acquired illegally through online piracy. Piracy is relatively limited in this industry, particularly for the iOS platform and for the genre of apps considered in this paper, 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.

particularly among non-game apps.

Other forms of monetization In addition to setting a price, developers can choose to include advertisements and/or in-app purchases (IAPs). IAPs are opportunities 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, which includes additional task organizational 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.[10]

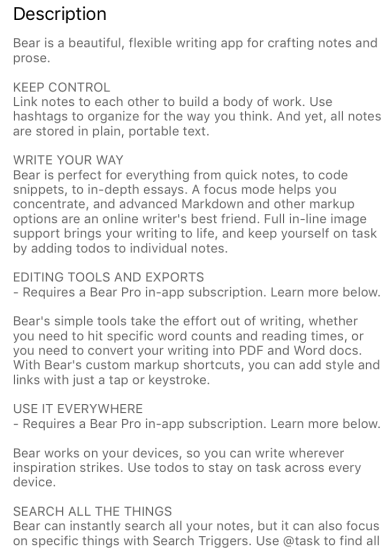
Companion versus Product Apps Apps can be classified as either companion or product apps. Companion apps are those that serve as a companion to an existing product or service. For example, the American Airways app, which offers detailed flight information, flight check-in, and a consumer's boarding pass 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.[11] 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 a derived demand.

[10] This policy results in a number of interesting responses from firms that monetize their product through a subscription. Most firms either offer subscriptions within the app or through a website or other means. This can penalize firms who offer subscriptions through a website, as they're not allowed to advertise those subscriptions within the app. A small number of firms offer subscriptions both within the app and on a website, and either charge different prices, often raising the price in the app to recover Apple's 30% cut, or they charge the same price in both locations but receive 30% less revenue from in-app subscription sales.

[11] Accounting for 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 retail operation, but directly observing the revenue collected through the app would likely not rationalize the costs of development.

Figure 2: Example of an App Description



For example, demand for the American Airways app will almost entirely be driven by demand for American Airways flights.

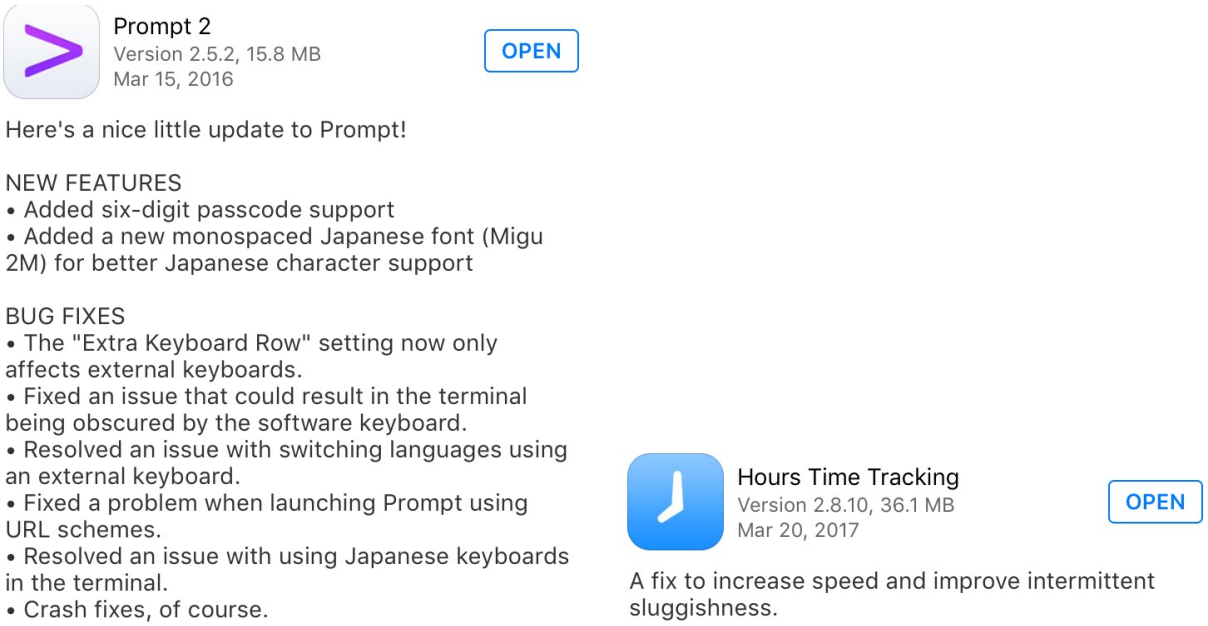
3 Data

I collect app characteristic and ranking data from Apple's App Store, and a limited amount of sales data directly from iOS app developers. The full sample consists of all apps in the App Store from December 31, 2014 to June 29, 2016. Currently, I limit this sample to just apps within the Productivity category for the remainder of this paper.[12] I aggregate daily data to the app-week level. In total, there were 1,241,910 iOS apps in the App Store on December 31, 2014 and 1,823,735 iOS apps in the App Store on June 29, 2016. Over the sample period I observe 64,506 apps in the Productivity category.

I observe a number of characteristics for each app, including the app's name, price, file size (in megabytes), age-appropriateness rating, the name of the app's developer, 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. In addition, I observe a textual description of each app and release notes for every app update, both written by the developer. An example of an app description and the release notes for an update are shown in Figures 2 and 3, respectively. Finally, I use data on the

[12] See Appendix A for more information on the App Store and app categories.

Figure 3: Update Type Examples



(a) Feature Update

(b) Bug Fix Update

presence of advertisements in apps that has been collected by the app analytics firm AppFigures.[13]

In addition to the data described above, I use data on the presence of advertisements in apps that was collected by the app analytics company AppFigures.

Section 3.1 covers how I approximate app sales using the ranking data I have collected. Section 3.2 discusses the two methods I use to classify app updates, and Section 3.3 develops a method for defining markets. Section 3.4 provides summary statistics and other details about the sample that is used in estimation.

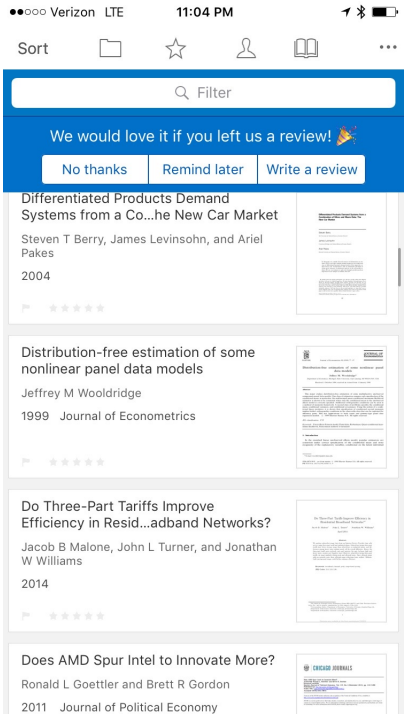
3.1 Estimating App Sales

A fundamental challenge of research on apps, as in research on many other industries, is the proprietary nature of sales data. This has been dealt with in a couple of ways. Liu (2017) uses app ratings and/or reviews as a proxy for sales.[14] This can be problematic, as some developers actively prompt users for ratings and/or reviews while others do not. Figure 4 shows an example of an app

[13] Section 3.2 describes the Feature and Bug Fix updates distinction that is referenced in Figure 3

[14] 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.

Figure 4: Example of an app prompting the user for a review



prompting the user for a review.[15]

A second approach, which has been used more widely in research on apps is to assume a particular probability distribution for sales data. 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 (2016), among others. In most applications, the relationship between sales ranking and sales quantity is estimated using ranking data and a small (relative to the sample of apps being studied) amount of sales data either collected from various developers or from small, independent app marketplaces.

Apple provides three types of 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. While the rules governing the lists are not known, the view within the industry is that the lists do follow some explicit algorithm, which is not altered to benefit some apps (e.g., for promotional reasons).[16] The

[15]The app pictured in Figure 4 uses a two-step screening process when prompting the user for a review. First (not pictured), it asks the user if they are unhappy, confused, or happy with the app. If the user picks one of the first two choices, he or she is presented with information about how to contact the developer for support. Only after selecting "I'm happy" is the user presented with the prompt to leave a review.

[16]Bresnahan et al. (2016) study apps – primarily games – that attempt to game the ranking lists.

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 apps with zero and non-zero prices by revenue.

I use the Top Free and Top Paid ranking lists at the category level to estimate sales because they most directly accounts for apps' downloads, which is the necessary variable for computing the market shares used in the demand estimation. Following Chevalier and Goolsbee (2003), I assume sales ranking data follows a Pareto distribution. Specifically, given a set of apps $j \in J$ and some threshold level of sales $\bar{S}$,

$$\Pr[\text{sales}_{jt} > \bar{S}] = \left(\frac{k}{\bar{S}}\right)^{\beta} \tag{1}$$

Given a sufficient number of apps, this can be rewritten as

$$\frac{\text{rank}_{jt}}{|J|} = \left(\frac{k}{\bar{S}}\right)^{\beta} \tag{2}$$

Taking logs, the assumption of a Pareto sales ranking distribution provides the estimable relationship

$$\ln(\text{rank}_{jt}) = \overbrace{\big(\beta \ln(k) + \ln(|J|)\big)}^{\alpha} - \beta \ln(\text{sales}_{jt}) \tag{3}$$

I estimate Equation (3) using publicly observable ranking data, and sales quantity data I have collected directly from some of the developers in the sample. Table 2 shows the results of estimating the rank-sales relationship for the Top Paid ranking list. Due to data limitations, I am only able to estimate Equation (3) for the Top Paid raking 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 $\alpha_{Free} = 7.326$ using sales data collected from developers' that produce Free apps. The results of this paper are robust to using different values for this parameter within a reasonable range of 7.326.

The estimate for β is lower than is typical in the literature. Garg and Telang (2012) summarize seven studies of this relationship in non-app industries, and find that estimates range from 0.613

Table 2: Ranking-Sales Relationship Estimates (Top Paid List)

α	5.231*** (0.182)
β	0.424*** (0.025)

Includes month-year controls. Standard errors are in parenthesis. R-squared=0.760.
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

to 1.2. In app markets, Ghose and Han (2014) estimate $\beta = 1.09$, and Ershov (2016) 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 (2016). 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 or Apple) app marketplace, and Ershov (2016) estimates the relationship for the Google Play Store using the lowerbound of sales ranges the store publically provides.[17]

A second reason is that the shape of the ranking-sales relationship may be changing. 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 $|\beta|$ would be consistent with the findings of Brynjolfsson et al. (2010). 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 Table 2, 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 quantity (or market share calculated using estimated sales quantities).

[17]For example, an app with cumulative sales of 45,000 is publicly listed as having "10,000 to 50,000" sales. See (Ershov, 2016, Table 15) for the sales ranges provided.

3.2 Classifying App Updates

In order to understand how consumers respond to app updates, it is necessary to account for the differences between specific updates, of which there are many. 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 of updates. The first is the update's version number, a sequence of numbers that developers 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.

3.2.1 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. That said, 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 minor changes to the app despite the "large" change in the version number.

Borrowing vocabulary from the developer community, I code 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).

3.2.2 Classifying Updates by Support Vector Machine

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. In fact, 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 to the app. For example, adding voice dictation to a note-taking app, or adding audio pronunciations to a dictionary app would be Feature updates.

In order to categorize updates using their release notes, I train and use a support vector machine (SVM). An SVM is a form of supervised machine learning that trains a model using a "training set" of pre-classified app updates. Once trained, the model can take other update release notes as an input and return an estimated update type. In order to train the SVM, I classified 1,280 app updates by hand. Figure 5 outlines the set of rules under which the training set was built.

Prior to classifying the updates, the updates' release notes undergo pre-processing to prepare them for the SVM algorithm. Each update's release notes is tokenized and lemmatized, and all punctuation and non-letter characters (e.g., bullet points and emojis) are removed.[18] I remove a standard list of common English "stop words," such as "a" and "the." I also remove a list of words related to Apple and its products, such as "iPhone" and "iOS," from the descriptions.

Additionally, I optimally select a number of other options, including allowing for n-grams, removing words that only appear a couple of 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 combination 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

[18]Tokenization is the practice of reducing a document to a set of individual word "tokens," and lemmatization is the practice of reducing tokens, i.e., words, to their dictionary form. E.g., "drinks", "drank", and "drinking" all reduce to "drink."

Figure 5: Update Classification Training Rules

Feature Update

- Adds additional functionality.
- Adds support for a new language

Bug Fix Update

- 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 3: Update Classification Precision and Recall

	Precision	Recall	N
Bug Fix	0.90	0.90	309
Feature	0.74	0.74	114
Average/Total	0.86	0.86	423

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.

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 frequency across all documents. The weight for word i in release notes j is

$$weight_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \times \ln\left(\frac{N}{n_i}\right) \tag{4}$$

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.

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 3.[19] Figure 6a shows the number of Major and Point updates over the sample period, and Figure 6b shows the ratio of Major to Point updates. Similarly, Figure 6c and Figure 6d show the level of and ratio of Bug Fix Updates and Feature updates over the sample period. Point updates vastly outnumber Major updates, but there is time variation in the ratio of the two types of updates. Bug fix updates tend to be more common than feature updates, though there are periods when the number of Feature updates outnumber the number of Bug Fix updates. For example, just following

[19] Alternative machine learning methods are being explored in an attempt to increase the precision and recall of the predictor.

Table 4: Cross Tabulation of Update Classifications

	Major	Point	Total
Bug Fix	72	1,498	1,570
Feature	116	1,032	1,148
Total	188	2,530	2,718

Major/Point update classification is determined based on how an update changes the app's version number. Bug Fix/Feature update classification is determined using a support vector machine that has been trained using a hand-coded set of Bug Fix and Feature updates.

the release of a new model of the iPhone (indicated by the dashed vertical line), there is a spike in the number of Feature updates, which outnumbers Bug Fix updates in that period.

Table 4 provides a cross-tabulation of the classification of updates. Notably, while a majority of Major updates are Feature updates, there is still a considerable number of Bug Fix updates that are released as Major updates.

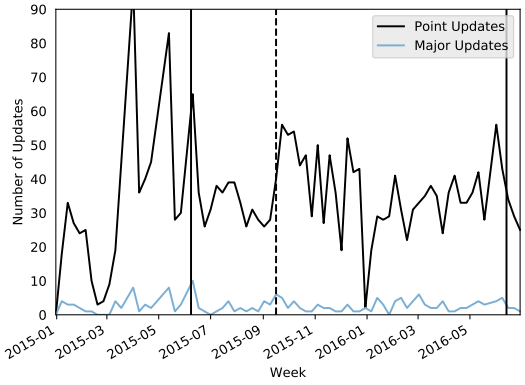
3.3 Defining Markets

Properly defining the relevant market for a product has long been a challenge in empirical market analyses. Digital goods markets provide additional challenges. First, the existence of a "long tail" of products in, for example, app markets, online book markets such as Amazon.com, and online music markets such as iTunes, dramatically expands the potential choice set for consumers. Second, the wide variety of monetization methods used in a particular app market make it difficult to develop and employ systematic methods of market definitions.

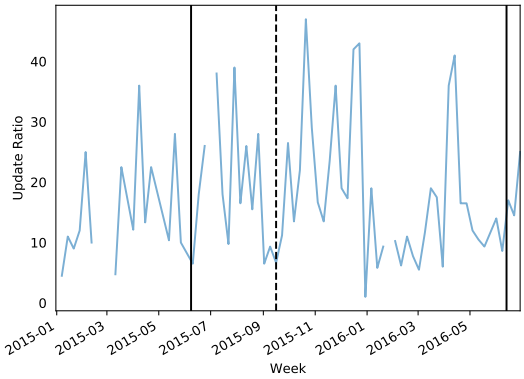
Traditional, systematic methods for defining markets — such as the SSNIP test — are especially difficult to develop and apply due to the difficulty in calculating the elasticity of demand when many products are priced at $0.00. Other approaches that do not involve demand elasticity estimations, such as the Critical Loss approach, are also difficult given the variety of ways products are monetized in this industry. It is not clear, for example, how to define a product's critical loss when the firm has additional forms of monetization. That is, estimating the critical loss associated by 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

Figure 6: Updates By Type

Version Number Classification

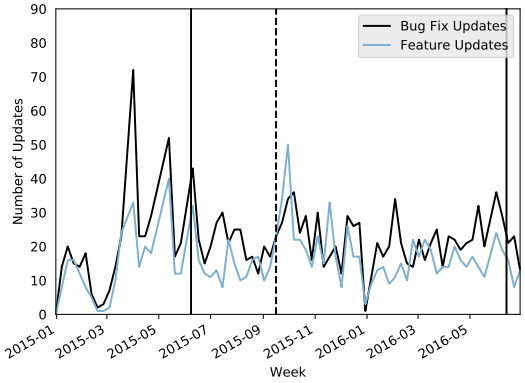


(a) App Updates by Type

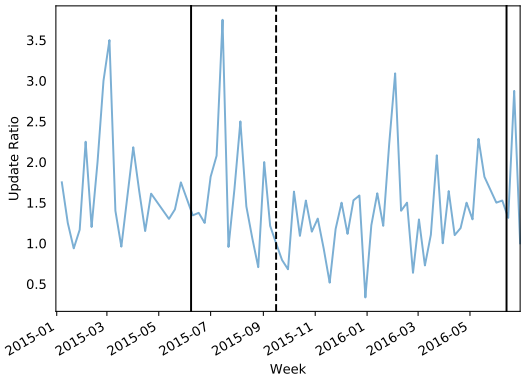


(b) Ratio of Point to Major Updates

SVM Classification



(c) App Updates by Type



(d) Ratio of Bug Fix to Feature Updates

The vertical lines mark the beginning (solid) and end (dashed) of the period in which Apple offers developers access to a pre-release version of their next major platform upgrade. Many developers hold off on updating their apps in anticipation of the release of the new version of the platform.

would still be abstracting from the endogeneity of *which* forms of monetization a particular product uses. While these abstractions are reasonable in typical product industry, where products are paid for at purchase, they are less reasonable for app markets given the varied manner in which the products are monetized. Accounting for all of this would be difficult in terms of modeling, and also computationally difficult given the large number of potential products available.

3.3.1 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.[20] Papers that explicitly consider the relationship between multiple app platforms, such as Liu (2017), generally define some mapping between the sets of categories on each platform.

This approach of adopting the platforms' categories is overly broad and suggests competition where there is none. For example, within Apple's Productivity category, there are a number of apps focused on helping a user manage their photos on the photo-sharing app Instagram, and there are a number of apps focused on writing computer code on mobile devices. 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 — in forthcoming work — in order to properly model strategic firm 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.

Ghose and Han (2014) take a more detailed approach by using a nested structure for markets. They consider a model 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 a given category — in the case of the demand analysis in this paper, Productivity — I cluster apps using unsupervised machine learning techniques, based on the textual descriptions provided by the developers. These descriptions are a prominent part of

[20] There are, to some extent, sub-genres defined within the App Store, but these are primarily used for Game apps, which are not considered in this paper.

the products' listings on the App Store, and thus developers have a strong motivation to provide a clear description of their product.

In order to deal with the "long tail" of apps I restrict my sample to only those apps that appear to be actively competing for sales in the App Store. Many apps, while listed on the store, effectively have no day-to-day sales. Much of the research on apps has implicitly made a similar restriction, usually due to data constraints, by limiting their sample to apps that achieve a particular sales ranking for the given period. I take a more detailed approach and limit my sample to apps that have achieved a particular sales ranking for at least a given fraction of its time in the store.

3.3.2 Accounting for Abandoned Apps

As discussed in Section 2, entry into the app store is relatively inexpensive. This presents an additional challenge in defining markets in a meaningful way, as 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 little to no costs in keeping an abandoned product in the store. Unlike the production of physical goods, continuing to sell an abandoned app does not, in general, result in continued variable expenses.

Among 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. Figure 7 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. Figure 8 shows that among apps that achieve at least a ranking of 200, nearly all spend less than 20% 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

Figure 7: Distribution of Productivity Apps' Best Ranking

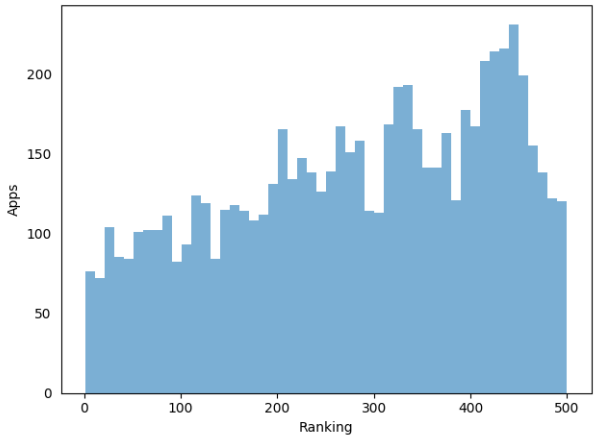
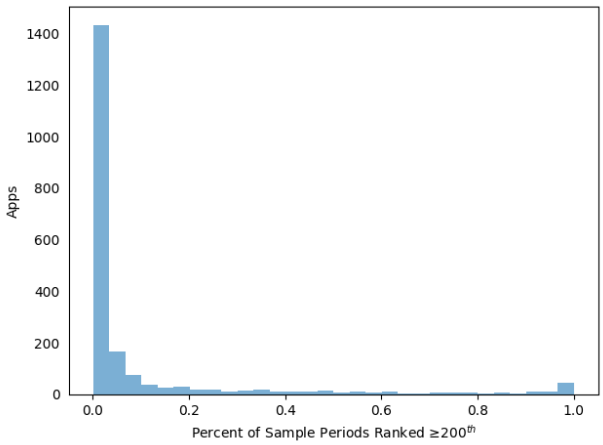
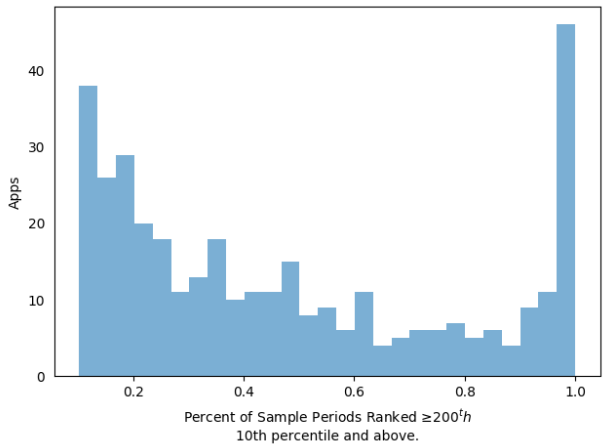


Figure 8: Persistence Above Ranking Threshold of 200



(a) Periods ranked above 200



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

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.

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 4.1. 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.[21]

In order to model and estimate consumer demand for new app downloads, it is important to identify which apps in the store form consumers' choice sets versus those that are abandoned. Note that this distinction is different than the matter of apps as companion versus products apps discussed in Section 2. I restrict the sample to apps that achieve a ranking of 200 for at least 10% of the sample periods. The threshold of 200 corresponds to selling 20 copies in an average period. This reduces the number of apps from 64,506 total Productivity apps during the sample period to 420 apps.[22]

3.3.3 Market Definitions via Clustering

Having restricted my sample in the way described in Section 3.3.2, I then use a combination of natural language processing techniques and unsupervised clustering algorithms to define markets. First, all app descriptions undergo the same text pre-processing process outlined in Section 3.2. The specific parameters of this process are set separately for the market definition algorithm.

Once the app descriptions have been processed and vectorized, I define markets using k-means clustering.[23] 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 element $x \in X$ 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

[21] 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.

[22] A future version of this paper will present estimation results under alternative sample parameters.

[23] Ongoing work will consider other clustering algorithms.

if it is more similar to the x's in j than to those in any other cluster.

More precisely, define

$$\gamma_{ij} = \begin{cases} 1, & x_i \text{ is in cluster } j \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

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

$$\min_{\gamma,\delta} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{k} \gamma_{ij} ||x_i - \delta_j||^2 \tag{6}$$

Equation (6) 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

$$\text{Step 1:} \qquad \min_{j'} ||x_i - \delta_{j'}||^2 \ \forall i \tag{7}$$

$$\text{Step 2:} \qquad \sum_{i=1}^{N} \gamma_{ij}(x_i - \delta_j) = 0, \ \forall j \tag{8}$$

This two-step process is repeated until the convergence of γ and δ. The optimal number of markets (i.e., clusters) and the optimal parameters for the text pre-processing stage are determined by repeating the clustering algorithm across a parameter grid and then selecting the set 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).

3.4 Market Results

After restricting the sample to apps that ranked at or better than 200^{th} for at least 10% of the sample periods, the k-means clustering algorithm finds 39 distinct markets. Table 5 provides summary statistics for each market, and for the overall sample. Figure 10 shows update levels by market, and, as an example of the market results, Figure 9 lists the apps in four of the markets.

Figure 9: Example Markets

Drawing Desk
Paint
Doodle
Sketch
Tayasui Sketches
Graphic

(a) Market 36 - Drawing

VPN Express
TunnelBear VPN
Astrill VPN Client
VyprVPN
HideMyAss! Pro VPN
VPN Unlimited
VPN Seed4.Me
Unlimited Free VPN
VPN Master

(b) Market 5 - Virtual Private Network

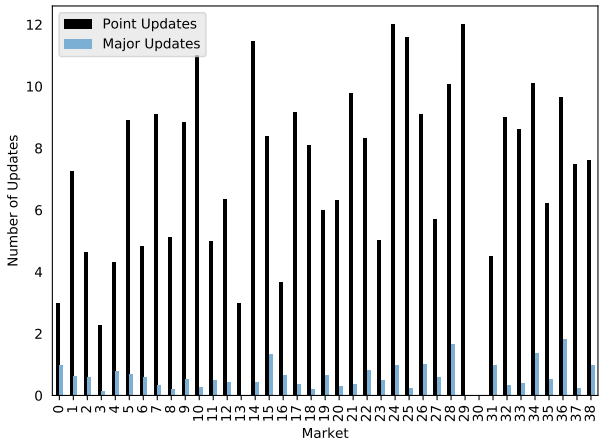
5000 Followers Pro
FBLikes
5,000 Followers for Instagram
TwitterBoost
Followers Jackpot
Gain Followers
Ins Followers
Mega Followers
Follower Boost for Instagram

(c) Market 34 - Instagram Management

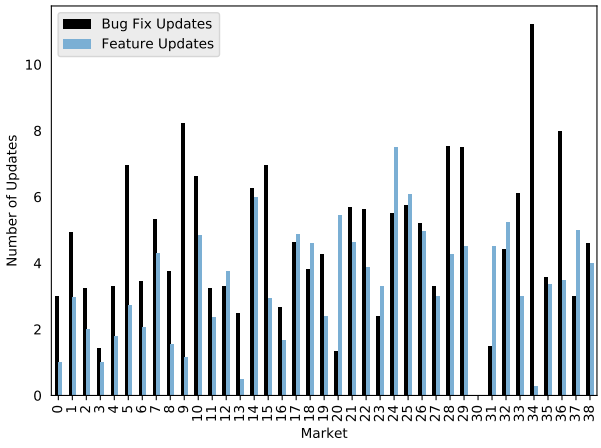
Groups
Simpler Contacts Pro
Smart Merge Pro
Cleaner Pro
Text 2 Group Pro
Easy Backup
Simpler Contacts
Cleaner
Smart Merge
Text 2 Group
Cleaner Master for iOS

(d) Market 4 - Text Messaging

Figure 10: Update Types, By Market



(a) Version Number Classification



(b) SVM Classification

Table 5: Summary Statistics

		Monetization												**Updates**				**Characteristics**				
		Price				Price (Paid apps)								Version Numbers		SVM			Age Appropriateness Rating			
Market	Apps	Min	Mean	Median	Max	Min	Mean	Median	Max	Free	IAP	Ad	Subs	Bug Fix	Feature	Bug Fix	Feature	ln(size)	4+	9+	12+	17+
0	1	0.00	3.68	3.99	4.99	2.99	3.84	3.99	4.99	0.041	1.000	0.000	0.000	0.041	0.014	0.041	0.014	2.539	1.000	0.000	0.000	0.000
1	32	0.00	6.61	3.99	49.99	0.99	9.37	4.99	49.99	0.295	0.552	0.000	0.075	0.100	0.008	0.068	0.041	2.375	0.885	0.066	0.017	0.032
2	5	0.00	12.50	11.99	29.99	3.99	14.32	11.99	29.99	0.127	0.177	0.000	0.000	0.065	0.008	0.045	0.028	2.753	1.000	0.000	0.000	0.000
3	7	0.00	3.16	2.99	9.99	1.99	4.40	3.99	9.99	0.282	0.424	0.000	0.000	0.031	0.002	0.019	0.014	2.633	0.720	0.144	0.136	0.000
4	11	0.00	1.26	0.00	4.99	0.99	3.07	2.99	4.99	0.591	0.876	0.000	0.003	0.059	0.011	0.045	0.024	3.027	1.000	0.000	0.000	0.000
5	9	0.00	0.00	0.00	0.00	—	—	—	—	1.000	1.000	0.000	0.389	0.123	0.010	0.097	0.037	2.271	0.867	0.013	0.000	0.120
6	23	0.00	8.60	6.99	119.99	0.99	13.73	9.99	119.99	0.374	0.542	0.000	0.103	0.066	0.008	0.047	0.028	3.376	0.914	0.000	0.000	0.086
7	30	0.00	11.31	9.99	49.99	0.99	12.14	9.99	49.99	0.069	0.272	0.000	0.000	0.125	0.004	0.074	0.058	3.535	0.890	0.034	0.000	0.076
8	8	0.00	0.93	0.00	3.99	1.99	2.93	2.99	3.99	0.683	0.764	0.165	0.000	0.064	0.002	0.048	0.019	2.621	0.406	0.000	0.358	0.236
9	8	0.00	0.00	0.00	0.00	—	—	—	—	1.000	1.000	0.131	0.822	0.122	0.007	0.113	0.016	2.041	0.760	0.000	0.000	0.240
10	13	0.00	0.38	0.00	4.99	2.99	3.38	2.99	4.99	0.887	0.978	0.087	0.309	0.151	0.004	0.090	0.068	2.856	0.839	0.000	0.161	0.000
11	8	0.00	5.78	4.99	12.99	2.99	7.72	6.99	12.99	0.251	0.376	0.000	0.007	0.068	0.007	0.044	0.032	3.256	0.731	0.000	0.000	0.269
12	5	0.00	2.76	0.00	9.99	3.99	6.27	4.99	9.99	0.560	0.465	0.000	0.233	0.088	0.006	0.047	0.050	2.493	0.975	0.000	0.025	0.000
13	2	0.00	0.00	0.00	0.00	—	—	—	—	1.000	1.000	0.000	0.000	0.051	0.000	0.042	0.008	2.673	1.000	0.000	0.000	0.000
14	19	0.00	3.50	0.00	16.99	0.99	7.95	8.99	16.99	0.560	0.669	0.000	0.396	0.157	0.006	0.086	0.081	3.820	0.730	0.000	0.000	0.270
15	7	0.00	0.00	0.00	0.00	—	—	—	—	1.000	1.000	0.589	0.000	0.109	0.020	0.092	0.039	2.505	0.271	0.000	0.729	0.000
16	3	0.00	1.00	0.00	2.99	2.99	2.99	2.99	2.99	0.667	0.667	0.000	0.000	0.050	0.009	0.036	0.023	2.104	1.000	0.000	0.000	0.000
17	11	0.00	0.31	0.00	3.99	2.99	3.31	2.99	3.99	0.907	0.907	0.000	0.610	0.125	0.005	0.063	0.067	3.325	1.000	0.000	0.000	0.000
18	9	0.00	7.93	4.99	39.99	1.99	12.09	6.99	39.99	0.344	0.783	0.000	0.261	0.110	0.003	0.052	0.063	2.715	1.000	0.000	0.000	0.000
19	6	0.00	6.59	4.99	15.99	0.99	7.99	4.99	15.99	0.176	0.176	0.000	0.000	0.083	0.010	0.059	0.033	2.739	1.000	0.000	0.000	0.000
20	13	0.00	4.93	4.99	9.99	0.99	5.96	4.99	9.99	0.172	0.172	0.000	0.077	0.085	0.004	0.018	0.073	3.016	0.733	0.000	0.089	0.178
21	19	0.00	3.05	2.99	8.99	0.99	4.92	4.99	8.99	0.380	0.646	0.000	0.254	0.134	0.005	0.076	0.064	2.959	1.000	0.000	0.000	0.000
22	6	0.00	18.84	4.99	99.99	0.99	22.84	6.99	99.99	0.175	0.510	0.000	0.009	0.112	0.012	0.075	0.054	3.312	1.000	0.000	0.000	0.000
23	8	0.00	0.43	0.00	19.99	0.99	7.58	4.99	19.99	0.943	0.975	0.000	0.983	0.068	0.006	0.032	0.044	2.738	0.687	0.156	0.000	0.156
24	4	0.00	1.27	0.00	9.99	0.99	5.07	4.99	9.99	0.750	1.000	0.000	0.750	0.162	0.014	0.074	0.101	2.995	1.000	0.000	0.000	0.000
25	4	0.00	6.00	0.00	24.99	2.99	12.16	2.99	24.99	0.507	0.507	0.000	0.075	0.158	0.003	0.079	0.082	3.111	1.000	0.000	0.000	0.000
26	15	0.00	3.25	0.00	9.99	1.99	7.15	4.99	9.99	0.545	0.618	0.000	0.043	0.123	0.014	0.070	0.068	2.827	0.254	0.000	0.000	0.746
27	5	0.00	4.43	0.00	19.99	3.99	9.01	7.99	19.99	0.508	0.593	0.000	0.319	0.077	0.008	0.044	0.041	3.091	0.992	0.000	0.008	0.000
28	11	0.00	0.15	0.00	1.99	0.99	1.56	1.99	1.99	0.901	1.000	0.027	0.199	0.136	0.023	0.102	0.059	3.162	0.799	0.000	0.000	0.201
29	2	7.99	8.99	8.99	9.99	7.99	8.99	8.99	9.99	0.000	1.000	0.000	0.628	0.162	0.000	0.101	0.061	3.631	1.000	0.000	0.000	0.000
30	3	7.99	12.66	9.99	19.99	7.99	12.66	9.99	19.99	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.029	1.000	0.000	0.000	0.000
31	2	0.00	2.33	0.00	4.99	1.99	4.87	4.99	4.99	0.521	1.000	0.000	0.000	0.063	0.014	0.021	0.063	3.054	1.000	0.000	0.000	0.000
32	6	0.00	2.03	2.99	4.99	0.99	3.96	3.99	4.99	0.487	0.487	0.000	0.005	0.112	0.005	0.044	0.078	1.721	1.000	0.000	0.000	0.000
33	10	0.00	3.42	0.99	19.99	0.99	6.79	7.99	19.99	0.497	0.698	0.000	0.378	0.117	0.005	0.083	0.041	2.935	0.899	0.000	0.000	0.101
34	9	0.00	0.00	0.00	0.00	—	—	—	—	1.000	1.000	0.495	0.146	0.136	0.020	0.152	0.004	2.283	0.507	0.000	0.493	0.000
35	6	0.00	3.22	1.99	9.99	0.99	4.27	2.99	9.99	0.246	0.507	0.000	0.101	0.085	0.007	0.048	0.046	2.034	1.000	0.000	0.000	0.000
36	3	0.00	0.59	0.00	2.99	0.99	2.94	2.99	2.99	0.800	0.800	0.000	0.000	0.130	0.022	0.103	0.049	4.071	1.000	0.000	0.000	0.000
37	4	0.00	2.28	0.00	8.99	8.99	8.99	8.99	8.99	0.747	1.000	0.000	1.000	0.103	0.003	0.041	0.068	2.842	1.000	0.000	0.000	0.000
38	5	0.00	2.21	0.00	9.99	9.99	9.99	9.99	9.99	0.779	0.798	0.000	0.596	0.104	0.014	0.063	0.055	3.028	0.617	0.000	0.000	0.383
Total	352	0.00	4.66	0.99	119.99	0.99	9.14	6.99	119.99	0.490	0.638	0.029	0.207	0.106	0.008	0.066	0.049	2.902	0.838	0.016	0.041	0.105

Observations are app-weeks. $N = 28,704$

4 Demand Model

In this section I develop a model of consumers looking to purchase an app in a given market.

4.1 Model

Consumers are randomly assigned to a market m each period. Once assigned to a market, consumers choose which, if any, app to buy. Consumers in this model are myopic, and thus do not consider the value of future use of the app nor the potential for any future updates to the app when making their purchase decision. Consumer i receives utility u_{ijt} from purchasing app j in period t, and chooses the app that provides the greatest utility. Specifically, utility from app j is

$$u_{ijt} = X_{jt}\beta + \alpha p_{jt} + \xi_{jt} + \epsilon_{ijt} \tag{9}$$

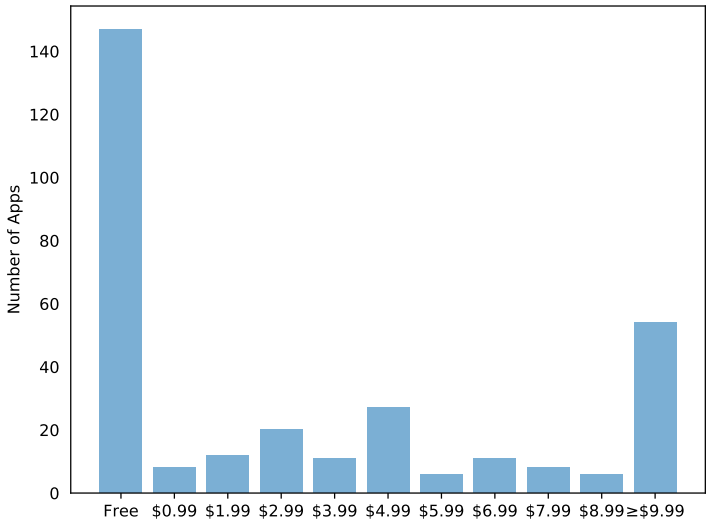
$$u_{i0t} = \epsilon_{i0t} \tag{10}$$

where u_{i0t} represents 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 and subscriptions, and month and iOS version fixed effects. p_{jt} is the price of the app, and ϵ_{ijt} is a per-period consumer-app match shock. 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. I assume that $\epsilon_{ijt} \sim$ i.i.d. Type I EV, which admits a closed form solution for the share of consumers purchasing app j at time t, s_{jt}.

$$s_{jt} = \frac{\exp(X_{jt}\beta + \alpha p_{jt} + \xi_{jt})}{1 + \sum_k \exp(X_{kt}\beta + \alpha p_{kt} + \xi_{kt})} \tag{11}$$

Mean Unobservable Consumer Utility (ξ) Even within the same market, apps are highly differentiated. 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.

Figure 11: Distribution of Prices in Estimation Sample (On 12/31/14)



I allow app updates to directly affect ξ_{jt} by assuming that ξ_{jt} follows the law of motion

$$\xi_{jt} = \rho\xi_{jt-1} + \mu_{BF}\mathbb{1}(\text{Bug Fix})_{jt} + \mu_F\mathbb{1}(\text{Feature})_{jt} + \eta_{jt} \tag{12}$$

where $\mathbb{1}(\tau)_t$ is an indicator function for whether or not an update of type τ has occurred in period t, and η_{jt} is assumed to be an i.i.d., mean 0 shock.

Note that unpacking the variety of possible mechanisms captured by ξ 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.

4.2 Estimation

I estimate the model in Section 4.1 using the sample defined in Section 3.4. The distribution of prices for this sample is shown in Figure 11.

In order to derive an estimating equation for the demand model, I transform market shares (Equation (11)) following the standard Berry (1994) inversion. This gives

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

In general, Equation (13) can be estimated using OLS or 2SLS. However, in this case such

estimation methods aren't possible, because ξ follows the law of motion defined in Equation (12).

For clarity, let $y_{jt} = \ln(s_{jt}) - \ln(s_{0t})$. Given this, and taking note of Equation (12), Equation (13) can be re-written as

$$y_{jt} = X_{jt}\beta + \alpha p_{jt} + \rho\xi_{jt-1} + \mu_{BF}\mathbb{1}(\text{Bug Fix})_t + \mu_F\mathbb{1}(\text{Feature})_t + \eta_{jt} \tag{14}$$

In order to estimate the parameters of this model, I quasi-first-difference this model, giving

$$\begin{aligned} y_{jt} - \rho y_{jt-1} = (X_{jt} - \rho X_{jt-1})\beta + \alpha(p_{jt} - \rho p_{jt-1}) \\ + \mu_{BF}\mathbb{1}(\text{Bug Fix})_t + \mu_F\mathbb{1}(\text{Feature})_t + \eta_{jt} \end{aligned} \tag{15}$$

I estimate Equation (15) using non-linear least squares (NLLS).

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 consumers may strongly dislike paying for an app, but conditional on paying for an app, they may not be highly price sensitive. Of course, the opposite could be true, that the disutility from paying sharply increases in price. In light of these concerns, I consider three specifications of p_{jt} in Section 4.2 in order to approximate any possible non-linearities. Specification (1) includes the price p_{jt} of the app, and specification (2) uses an indicator variable for whether the price is greater than zero or not. Specification (3) includes both the "paid" indicator and the app's price.

Timing and Endogeneity As discussed in Section 2, Apple maintains a strict review process for all app updates. This review process ensures that product updates do not introduce violations of the Apple's platform regulations. This process creates a delay between the creation of the update (and its submission to Apple), and the release of the update to consumers. Given this delay, I assume that the decision of whether, and if so, how, to update an app is made prior to the realization of η_{jt}. That is, η_{jt} is assumed to be independent of period-t updating decisions.

Price changes 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 least squares approach, where period t prices are predicted in a first-stage regression using current and one-period

lagged covariates.[24] This is similar to the approach taken in Doraszelski, Lewis, and Pakes (2016).

5 Results

5.1 Primary Results

Table 6 shows the results of estimating Equation (15) using non-linear least squares (NLLS) under three specifications: specification (1) includes the price of the app, specification (2) uses an indicator variable for whether the app charges a non-zero price or not, and specification (3) includes both the price and the "paid" indicator. For the reasons discussed below, (3) is the preferred specification. The first set of results groups all updates together, and the second set of results classifies the updates as either Major or Point updates according to the Version Number classification method discussed in Section 3.2.1.

Updates I find that updates increase the demand for apps, and that there is a strong correlation in the quality of apps over time, with estimates of ρ generally just above 0.86. Thus, while updates don't have a large effect on quality, that effect is persistent (through ξ_{jt}).

As the Version Number classification results show, consumers differentiate between Major and Point updates, with a Major update having approximately twice the effect of a Point update. That said, it is unclear *why* consumers respond differently to Major updates than to Point updates. The reason for this is that Major updates conflate two effects. First, in many cases, Major updates reflect the addition of important new functionality to an app, which may be the reason consumers respond more to Major updates. At the same time, however, many developers view an app's version number as a promotional tool. For example, announcing a "Brand new version 5.0!" can increase media coverage and word-of-mouth. Thus, the results shown in Table 6 cannot distinguish whether or not consumers respond to the content of an update or 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 additionally categorize updates based solely on their content, using the SVM classification method discussed in Section 3.2.2. Under the SVM classification method, updates are categorized as either

[24] 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 6: NLLS Demand Estimates

	Update			Version Number Classification		
	(1)	(2)	(3)	(1)	(2)	(3)
Update	0.0938***	0.0854***	0.0892***			
	(0.0262)	(0.0262)	(0.0262)			
Major				0.1829**	0.1780*	0.1815*
				(0.0925)	(0.0927)	(0.0926)
Point				0.0864***	0.0776***	0.0818***
				(0.0271)	(0.0271)	(0.0271)
Price	-0.0267***		-0.0331***	-0.0267***		-0.0331***
	(0.0059)		(0.0049)	(0.0059)		(0.0049)
Paid		-0.1795***	-0.1374**		-0.1804***	-0.1383***
		(0.0532)	(0.0535)		(0.0532)	(0.0535)
IAP	0.3684***	0.3189*	0.2837	0.3657**	0.3152*	0.2801
	(0.1426)	(0.1833)	(0.1795)	(0.1426)	(0.1834)	(0.1796)
Ad	-0.1963***	-0.1896***	-0.1858***	-0.1960***	-0.1904***	-0.1865***
	(0.0565)	(0.0574)	(0.0573)	(0.0565)	(0.0574)	(0.0573)
Subscription	-0.0989***	-0.2109***	-0.1836***	-0.0987***	-0.2113***	-0.1841***
	(0.0221)	(0.0486)	(0.0487)	(0.0221)	(0.0487)	(0.0487)
$\ln(size)$	-0.1658***	-0.1953***	-0.1904***	-0.1662***	-0.1966***	-0.1917***
	(0.0530)	(0.0567)	(0.0566)	(0.0530)	(0.0568)	(0.0566)
ρ	0.8610***	0.8643***	0.8614***	0.8610***	0.8643***	0.8615***
	(0.0033)	(0.0033)	(0.0033)	(0.0033)	(0.0033)	(0.0033)

Observations are app-weeks. Standard errors are in parentheses.
All specifications include controls for apps' age appropriateness ratings, iOS platform version, and month.
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Table 7: Interacting Update Classifications (NLLS Estimates)

	(1)	(2)	(3)
(Major, Feature)	0.0492	0.0540	0.0516
	(0.1175)	(0.1177)	(0.1176)
(Major, Bug Fix)	0.3982***	0.3776**	0.3906***
	(0.1488)	(0.1490)	(0.1489)
(Point, Feature)	0.0826**	0.0717*	0.0764*
	(0.0407)	(0.0407)	(0.0407)
(Point, Bug Fix)	0.0890***	0.0816**	0.0853**
	(0.0341)	(0.0341)	(0.0340)
Price	-0.0268***		-0.0333***
	(0.0059)		(0.0049)
Paid		-0.1805***	-0.1383***
		(0.0533)	(0.0535)
IAP	0.3648**	0.3141*	0.2789
	(0.1426)	(0.1834)	(0.1795)
Ad	-0.1967***	-0.1907***	-0.1869***
	(0.0565)	(0.0574)	(0.0573)
Subscription	-0.0987***	-0.2113***	-0.1841***
	(0.0221)	(0.0487)	(0.0487)
$\ln(size)$	-0.1644***	-0.1943***	-0.1893***
	(0.0530)	(0.0568)	(0.0566)
ρ	0.8610***	0.8643***	0.8614***
	(0.0033)	(0.0033)	(0.0033)

Observations are app-weeks. Standard errors are in parentheses. All specifications include controls for apps' age appropriateness ratings, iOS platform version, and month.
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Feature-adding or Bug-fixing updates. I interact the SVM classification with the VN classification to create four categories of updates: (Major, Feature), (Major, Bug Fix), (Point, Feature), and (Point, Bug Fix). Table 7 shows the results of estimating the model with these update categories. I find that within the Major and Point update categories, there is no statistically significant difference between the coefficients of the Feature and Bug Fix updates at the 5% level.[25]

Finally, I consider whether consumers respond differently to updates based on whether an app has a non-zero price. It is possible that consumers may be more discerning when purchasing a Paid app, as the cost of trying and learning about the app is higher. Table 8 shows the results of allowing the update coefficients to differ by whether the app is Paid or Free. I find no evidence that Feature updates are viewed differently than Bug Fix updates in either the Paid or Free app

[25] A Wald test fails to reject the hypothesis that (Major, Feature) updates have the same effect on demand as (Major, Bug Fix) updates (p=0.073), and similarly for the Point updates (p=0.861).

Table 8: Effect of Updates by Paid Status (NLLS Estimates)

	(1)	(2)	(3)
(Paid, Feature)	0.0553	0.0363	0.0500
	(0.0574)	(0.0576)	(0.0576)
(Paid, Bug Fix)	0.0478	0.0347	0.0471
	(0.0521)	(0.0524)	(0.0524)
(Free, Feature)	0.1022**	0.1014**	0.0968*
	(0.0506)	(0.0509)	(0.0509)
(Free, Bug Fix)	0.1393***	0.1355***	0.1337***
	(0.0424)	(0.0427)	(0.0427)
Price	-0.0264***		-0.0328***
	(0.0059)		(0.0049)
Paid		-0.1721***	-0.1317**
		(0.0534)	(0.0537)
IAP	0.3419**	0.3234*	0.2876
	(0.1436)	(0.1831)	(0.1794)
Ad	-0.1988***	-0.1908***	-0.1869***
	(0.0565)	(0.0574)	(0.0573)
Subscription	-0.1009***	-0.2067***	-0.1804***
	(0.0221)	(0.0487)	(0.0487)
$\ln(size)$	-0.1659***	-0.1949***	-0.1899***
	(0.0530)	(0.0568)	(0.0566)
ρ	0.8609***	0.8641***	0.8613***
	(0.0033)	(0.0033)	(0.0033)

Observations are app-weeks. Standard errors are in parentheses. All specifications include controls for apps' age appropriateness ratings, iOS platform version, and month.
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

case.[26]

Overall, the results show that consumers do not consider the content of an update when making a purchase decision. In conversations about this project with developers, a number of developers have made clear that they view updates focused on bug fixes to be less costly (from a production perspective) than updates that provide new features. Thus, it is surprising that, given the relatively similar effects on demand, developers engage in *both* Bug Fix and Feature updates. The fact that developers do 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 demand. For example, the more costly Feature updates may have a greater effect on maintaining user lock-in with an app's existing user base.

[26] A Wald test cannot reject the hypothesis that Feature updates have a larger effect than Bug Fix updates in either the Paid (p=0.970) or Free (p=0.562) case.

Table 9: NLTSLS Demand Estimates

	Update		VN Classification	
	(1)	(3)	(1)	(3)
Update	0.0967***	0.0917***		
	(0.0262)	(0.0262)		
Major			0.1907**	0.1862**
			(0.0925)	(0.0926)
Point			0.0884***	0.0840***
			(0.0271)	(0.0271)
Price	-0.2673***	-0.0552***	-0.2674***	-0.0552***
	(0.0543)	(0.0085)	(0.0544)	(0.0085)
Paid		-0.1555***		-0.1563***
		(0.0533)		(0.0533)
IAP	0.1355	0.2123	0.1326	0.2086
	(0.1509)	(0.1794)	(0.1509)	(0.1795)
Ad	-0.2549***	-0.2006***	-0.2547***	-0.2013***
	(0.0580)	(0.0573)	(0.0580)	(0.0573)
Subscription	-0.3010***	-0.2031***	-0.3007***	-0.2036***
	(0.0504)	(0.0484)	(0.0504)	(0.0484)
$\ln(size)$	-0.2833***	-0.2044***	-0.2836***	-0.2057***
	(0.0592)	(0.0566)	(0.0592)	(0.0566)
ρ	0.8600***	0.8609***	0.8601***	0.8609***
	(0.0033)	(0.0033)	(0.0033)	(0.0033)

Observations are app-weeks. Standard errors are in parentheses.
All specifications include controls for apps' age appropriateness ratings, iOS platform version, and month.
* $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Monetization Table 9 shows the results when the model is estimated using non-linear two-stage least squares (NLTSLS). The price coefficient in specification (1) is larger in magnitude compared to the NLLS results, but the disparity between the NLLS and NLTSLS results is smaller in specification (3), which suggests that controlling for whether or not an app is a paid app is the primary cause for bias in specification (1).

Noticeably, the coefficient on the paid indicator in specification (3) is much larger in magnitude than the coefficient on price. This suggests a non-linear effect of price on utility. In particular, consumers appear to be highly reticent to pay for an app, but conditional on paying, they are not particularly price sensitive. Using the results from specification (3), 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 \$6.99. Specification (3) is the preferred specification from these results, as it is able to

capture this non-linearity in price sensitivity.

Finally, I find that the inclusion of ads or subscriptions in an app is associated with lower utility. IAPs, on the other hand, do not have a statistically significant effect on utility, which may be due to the relatively limited information consumers receive about the inclusion of and content associated with IAPs.

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, 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. The topic of how developers choose how to monetize their apps is not currently well understood, and is left to future work. Since apps rarely change the form(s) of monetization they use, understanding this selection effect would require modeling apps' entry decisions into the available forms of monetization.

5.2 Robustness

The qualitative results presented above are robust to varying the constraints used when defining the sample, and to changing the scale parameter used when calculating sales for Free apps. Results showing this are forthcoming.

6 Conclusion

The digitization of consumer durable goods represents an ongoing paradigm shift in a number of industries. Digitization is enabling new forms of product monetization and is providing new methods for updating products. As this occurs in a number of industries, it is important to understand how these changes affect consumer behavior, competition, and product innovation.

This paper takes a first step in understanding the impact of the digitization of durable goods

by studying how consumers react to digital product updates in the smartphone application (app) industry. The app industry provides a unique opportunity to study consumer behavior under digitization without having to account for the specifics of a particular industry's non-digital technologies and costs.

In order to study consumers' preferences for updates, I first employ supervised and unsupervised machine learning methods to define markets and classify app updates, which is important given the large degree of heterogeneity in updates. To define markets, I cluster apps based on developer-provided descriptions of each app. In addition, I account for the fact that app marketplaces include a large number of products, many of which are no longer being actively developed.

I find that Major updates increase demand by more than smaller, "Point" updates, but I find no evidence that consumers respond differently based on whether the update adds new features or simply fixes bugs in the software. The fact that consumers do not react differently based on the content of an update, combined with the fact that developers produce both Bug Fix and Feature updates corroborates the notion of a paradigm shift in durable good industries. Bug Fix and/or Feature updates likely serve a purpose other than increasing the demand for new downloads. In particular, it is likely that Feature updates, which according to developers are more costly to produce, are used as a tool for maintaining user engagement, which is important when products are monetized through a means other than (or in addition to) an up-front price. Whether or not this is the case is left for future research.

Whether, and the degree to which, consumers respond to app updates is important, as digital updates are fundamentally different from traditional, non-digital product updates. Digital updates do not result in an entirely new product, and thus the salience of digital updates may differ from that of more traditional product updates. That said, of additional importance is how firms choose to monetize and update their products. This question is the focus of research that is currently in progress. Specifically, I am in the process of developing and estimating a dynamic supply-side model that endogenizes product updating, which will allow me to study how the changes brought on by digitization – the increased forms of monetization, and the ability to update a product long after a consumer has purchased it – affect innovation rates, both overall and by update type. This ongoing work also accounts for the differential effects of updates on new consumers (studied in this paper) and the existing user base, as owners of a particular app must continually choose whether

to engage with the product.

References

Aguiar, L., & Waldfogel, J. (2016, September). *Quality Predictability and the Welfare Benefits from New Products: Evidence from the Digitization of Recorded Music* (Tech. Rep.). Cambridge, MA: National Bureau of Economic Research, Cambridge, MA.

Apple. (2016, June). WWDC Keynote. In *World wide developer conference.* Apple.

Berry, S. T. (1994). Estimating discrete-choice models of product differentiation. *The RAND Journal of Economics*, *25*(2), 242.

Berry, S. T., Levinsohn, J., & Pakes, A. (1995, July). Automobile Prices in Market Equilibrium. *Econometrica*, *63*(4), 841–890.

Bresnahan, T., Li, X., & Yin, P.-L. (2016, May). *Paying Incumbents and Customers to Enter an Industry: Buying Downloads.*

Bresnahan, T., Orsini, J., & Yin, P.-L. (2014, February). *Platform Choice by Moible App Developers.*

Brynjolfsson, E., Hu, Y., & Smith, M. D. (2003, November). Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers. *Management Science*, *49*(11), 1580–1596.

Brynjolfsson, E., Hu, Y., & Smith, M. D. (2010, September). *The Longer Tail: The Changing Shape of Amazon's Sales Distribution Curve.*

Chevalier, J., & Goolsbee, A. (2003). Measuring Prices and Price Competition Online: Amazon.com and BarnesandNoble.com. *Quantitative Marketing and Economics*, *1*(2), 203–222.

Comino, S., Manenti, F. M., & Mariuzzo, F. (2016, May). Updates Management in Mobile Applications. iTunes vs Google Play. *Available at SSRN 2664463*.

comScore. (2016, March). comScore Reports January 2016 U.S. Smartphone Subscriber Market Share.

Davis, J. P., Muzyrya, Y., & Yin, P.-L. (2014, March). *Experimentation Strategies and Entrepreneurial Innovation: Inherited Market Differences in the iPhone Ecosystem.*

Doraszelski, U., Lewis, G., & Pakes, A. (2016, March). Just Starting Out: Learning and Equilibrium in a New Market. , 1–71.

Eizenberg, A. (2014, July). Upstream Innovation and Product Variety in the U.S. Home PC

Market. *The Review of Economic Studies*, *81*(3), 1003–1045.

Ershov, D. (2016, May). *The Effect of Search on Entry and Quality in the Mobile App Market.*

Gandhi, A., Lu, Z., & Shi, X. (2013, January). *Estimating Demand for Differentiated Products with Error in Market Shares .*

Gans, J. S. (2012, January). Mobile Application Pricing. , 1–24.

Garg, R., & Telang, R. (2012, October). *Inferring app demand from publicly available data .*

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

Goettler, R. L., & Gordon, B. R. (2011, December). Does AMD Spur Intel to Innovate More? *Journal of Political Economy*, *119*(6), 1141–1200.

Goode, L. (2016, September). *Apple's new subscription offerings are now available to App Store developers.* Retrieved from `http://www.theverge.com/2016/9/2/12774758/apple-developers-app-store-new-subscription-rules`

Gowrisankaran, G., & Rysman, M. (2012). Dynamics of Consumer Demand for New Durable Goods. *Journal of Political Economy*, *120*(6), 1173–1219.

Lee, R. S. (2013, December). Vertical Integration and Exclusivity in Platform and Two-Sided Markets. *American Economic Review*, *103*(7), 2960–3000.

Liu, Y. (2017, February). *Mobile App Platform Choice.*

Liu, Y., Nekipelov, D., & Park, M. (2013, December). *Timely versus Quality Innovation: The Case of Mobile Applications on iTunes and Google Play.*

Quan, T. W., & Williams, K. R. (2015, November). *Product Variety, Across-Market Demand Heterogeneity and the Value of Online Retail.*

Rousseeuw, P. J. (1987, November). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53–65.

Schiraldi, P. (2011, June). Automobile replacement: a dynamic structural approach. *The RAND Journal of Economics*, *42*(2), 266–291.

Spencer, G. (2015, September). *A Beginner's Guide to App Store Pricing Tiers.* Retrieved from `https://www.macstories.net/stories/a-beginners-guide-to-app-store-pricing-tiers/`

Yin, P.-L., Davis, J. P., & Muzyrya, Y. (2014, March). *Entrepreneurial Innovation: Killer Apps*

in the iPhone Ecosystem.

Zhao, Y. (2006, November). *Why are Prices Falling Fast? An Empirical Study of the US Digital Camera Market.*

Zhou, Y. (2016, April). *Failure to Launch in Two-Sided Markets: A Study of the U.S. Video Game Market.*

A Additional App Store Background

A.1 Consumers and the App Store

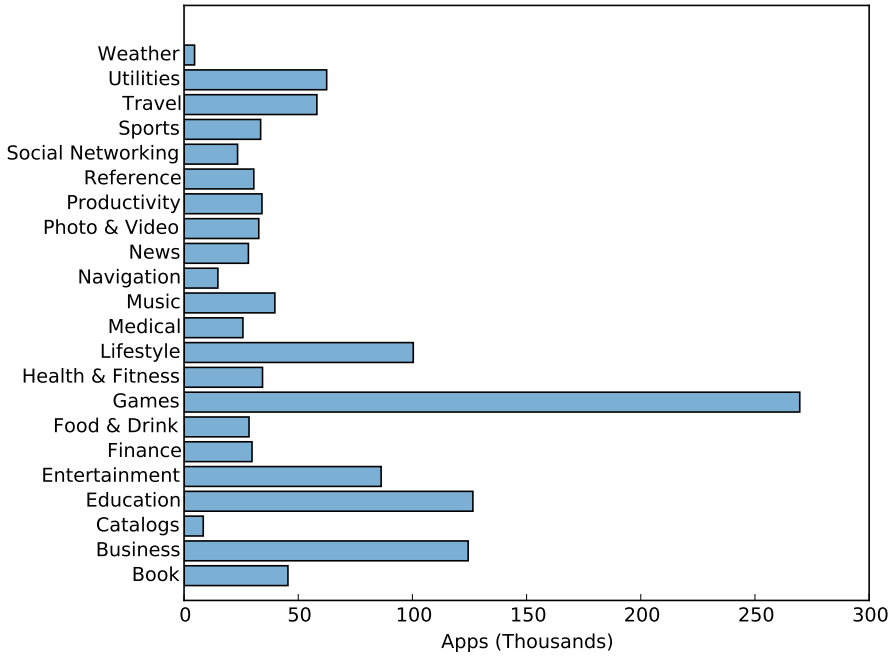
Hardware To access the iOS platform and its associated store, a consumer must first purchase either an iPhone, an iPad, an iPod Touch, or an Apple TV (Apple's mobile phone, tablet, touch-screen MP3 player, and television-based digital media player product lines, respectively).[27] These devices can be purchased either directly from Apple, through an electronics retailer (e.g., Best Buy or Amazon), or through a cellular service provider (e.g., AT&T or Verizon). iPhones must be purchased with a cellular service plan, while iPads come in two versions, one with and one without cellular service. The iPod Touch and Apple TV do not have cellular service. Unlike most other platforms, Apple exclusively designs and manufactures the devices compatible with its platform.

App Categories Given ownership of an iOS-compatible device, a consumer is able to directly access the App Store, Apple's mobile app marketplace. Within the store, apps are organized into 22 categories. Examples of categories include Games, Entertainment, and Productivity. Figure 12 shows a snapshot of the number of apps in each category.

Consumers register a credit card with the App Store, thus facilitating quick and easy purchases of apps. Having purchased an app, the consumer has access to that app for life, even if the app's developer chooses to remove the app from the App Store. However, without occasional product updates from the developer, a given app will likely not work with future versions of the platform. The iOS platform is typically given a major update once a year, with a variable, but generally small number of additional, incremental updates throughout the year.

[27] The App Store also includes apps for the Apple Watch product line, but the watch requires a connection to an iPhone or iPad in order to work.

Figure 12: Number of Apps by Category (12/31/2014)



Cross-device Use Apps in the App Store are clearly marked as either being for the iPhone, for the iPad, or they are marked as "Universal" apps, which run on either type of device.[28,29] For a number of years following the introduction of the iPad, consumers would have to purchase separate iPhone and iPad versions of a particular app (if available) if they wished to run the apps on both devices. This has become less common in recent years as Apple has changed the underlying structure of the iOS platform to better accommodate Universal apps.

A.2 App Development

To develop for the iOS platform, developers must join Apple's developer program, which costs $99/year per member. Developers create iOS apps primarily using some combination of the Objective-C and Swift programing languages. Any app characteristics the developer wishes to include in the product must be implemented via code. Developers are aided by a collection of public application programming interfaces (APIs), which are pre-written methods for interacting

[28] Developers are relatively limited in their ability to restrict their apps to particular devices within a given product line.

[29] Apps that run on iPhone are able to run on iPod Touch devices.

with the primary features of the platform and corresponding devices.[30] In addition, developers can have their app interface with other apps and web services, though limits exist on what types of interactions are permissible. These interactions are conducted through the use of that app's or service's API.

Given the fact that apps are fully digital products, and because Apple handles the requisite data center for hosting apps and propagating them to consumers' devices after purchase, the marginal cost of an additional sale is nearly zero, and in many cases, is in fact zero. Thus, app production is primarily characterized by the fixed costs of writing the app's code, and then additional fixed costs associated with writing the code for any future updates to the product. Indeed, as the capital requirements for producing apps are very low (all that is needed is a computer), entry costs are very low – which in part explains the large number of apps within the store.

[30] As an example, Apple provides a set of APIs for accessing the basic camera functionality. Thus, a developer does not have to write the code that make the device take a photo. Instead, a developer need only to "call" the relevant API and the phone will take and return a photo to the developer's application.