Quantitative indicators of a successful mobile application

PETER SKOGSBERG

KTH Information and Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Quantitative indicators of a successful mobile application

Peter Skogsberg

pesk@kth.se

Master of Science Thesis

Examiner: Professor Gerald Q. Maguire Jr.

Academic supervisor: Professor Gerald Q. Maguire Jr.
Industrial supervisor: Kenneth Andersson, The Mobile Life

Communication Systems
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

The smartphone industry has grown immensely in recent years. The two leading platforms, Google Android and Apple iOS, each feature marketplaces offering hundreds of thousands of software applications, or *apps*. The vast selection has facilitated a maturing industry, with new business and revenue models emerging. As an app developer, basic statistics and data for one's apps are available via the marketplace, but also via third-party data sources.

This report regards how mobile software is evaluated and rated *quantitatively* by both end-users and developers, and which metrics are relevant in this context. A selection of freely available third-party data sources and app monitoring tools is discussed, followed by introduction of several relevant statistical methods and data mining techniques. The main object of this thesis project is to investigate whether findings from app statistics can provide understanding in how to design more successful apps, that attract more downloads and/or more revenue.

After the theoretical background, a practical implementation is discussed, in the form of an in-house application statistics web platform. This was developed together with the app developer company The Mobile Life, who also provided access to app data for 16 of their published iOS and Android apps. The implementation utilizes automated download and import from online data sources, and provides a web based graphical user interface to display this data using tables and charts.

Using mathematical software, a number of statistical methods have been applied to the collected dataset. Analysis findings include different categories (clusters) of apps, the existence of correlations between metrics such as an app's market ranking and the number of downloads, a long-tailed distribution of keywords used in app reviews, regression analysis models for the distribution of downloads, and an experimental application of Pareto's 80-20 rule which was found relevant to the gathered dataset.

Recommendations to the app company include embedding session tracking libraries such as Google Analytics into future apps. This would allow collection of in-depth metrics such as session length and user retention, which would enable more interesting pattern discovery.

Keywords: mobile, smartphone, application, app, Android, iOS, statistics, data, metrics, quantitative, measure, downloads, rating, Pareto, successful, developer, publisher, data mining, R, ETL, API

Sammanfattning

Smartphonebranschen har växt kraftigt de senaste åren. De två ledande operativsystemen, Google Android och Apple iOS, har vardera distributionskanaler som erbjuder hundratusentals mjukvaruapplikationer, eller *appar*. Det breda utbudet har bidragit till en mognande bransch, med nya växande affärs- och intäktsmodeller. Som apputvecklare finns grundläggande statistik och data för ens egna appar att tillgå via distributionskanalerna, men även via datakällor från tredje part.

Den här rapporten behandlar hur mobil mjukvara utvärderas och bedöms *kvantitativt* av båda slutanvändare och utvecklare, samt vilka data och mått som är relevanta i sammanhanget. Ett urval av fritt tillgängliga tredjeparts datakällor och bevakningsverktyg presenteras, följt av en översikt av flertalet relevanta statistiska metoder och data mining-tekniker. Huvudsyftet med detta examensarbete är att utreda om fynd utifrån appstatistik kan ge förståelse för hur man utvecklar och utformar mer framgångsrika appar, som uppnår fler nedladdningar och/eller större intäkter.

Efter den teoretiska bakgrunden diskuteras en konkret implementation, i form av en intern webplattform för appstatistik. Denna plattform utvecklades i samarbete med apputvecklaren The Mobile Life, som också bistod med tillgång till appdata för 16 av deras publicerade iOS- och Android-appar. Implementationen nyttjar automatiserad nedladdning och import av data från datakällor online, samt utgör ett grafiskt gränssnitt för att åskådliggöra datan med bland annat tabeller och grafer.

Med hjälp av matematisk mjukvara har ett antal statistiska metoder tillämpats på det insamlade dataurvalet. Analysens omfattning inkluderar en kategorisering (klustring) av appar, existensen av en korrelation mellan mätvärden såsom appars ranking och dess antal nedladdningar, analys av vanligt förekommande ord ur apprecensioner, en regressionsanalysmodell för distributionen av nedladdningar samt en experimentell applicering av Paretos ”80-20”-regel som fanns lämplig för vår data.

Rekommendationer till appföretaget inkluderar att bädda in bibliotek för appsessionsspårning, såsom Google Analytics, i dess framtida appar. Detta skulle möjliggöra insamling av mer detaljerad data såsom att mäta sessionslängd och användarlojalitet, vilket skulle möjliggöra mer intressanta analyser.

Nyckelord: mobil, smartphone, applikation, app, Android, iOS, statistik, data, mätvärden, kvantitativ, mätning, nedladdningar, betyg, Pareto, framgångsrik, utvecklare, utgivare, data mining, R, ETL, API

Acknowledgements

Many thanks go to my academic supervisor and examiner at KTH Royal Institute of Technology, Professor Gerald Q. Maguire Jr., who has been very helpful with extensive feedback throughout the writing process. My academic opponent, Sarwarul Islam Rizvi, also contributed with helpful report feedback.

Thanks also go to my industrial supervisor Kenneth Andersson at the company The Mobile Life [129], whose ideas for implementation and functionality of the web platform has been highly useful.

Special thanks go to Professor Michael D. Smith, co-author of [30], and Jan Katrenic from [122], for taking the time to correspond with me via email to explain details of their respective works.

Finally, I wish to thank Remco van den Elzen at Distimo [126] and Christian Poppelreiter at Flurry [125] for their kind permission to use their graphs and illustrations in this thesis.

Table of Contents

List of Figures

List of Tables

List of abbreviations

AJAX	Asynchronous JavaScript and XML
ANR	Application Not Responding
API	Application Programming Interface.
APK	Application PacKage file format
ASCII	American Standard Code for Information Interchange
BB	BlackBerry
CEO	Chief Executive Officer
CSS	Cascading Style Sheets file format
CSV	Comma-Separated Value file format
DAU	Daily Active Users
DB	DataBase
DBMS	DataBase Management System
ETL	Extract, Transform, Load
GUI	Graphical User Interface
GZ	GNU Zipped Archive
HTML	HyperText Markup Language
IDE	Integrated Development Environment
iOS	A trade name for Apple's OS
iTC	iTunes Connect
JDE	Java Development Environment
JSON	JavaScript Object Notation
KDD	Knowledge Discovery in Databases
IDFA	IDentifier For Advertisers
IETF	Internet Engineering Task Force
IPA	IPhone Application file format
MAU	Monthly Active Users
MIME	Multipurpose Internet Mail Extensions
PC	Personal Computer
PCA	Principal Component Analysis
PDA	Personal Digital Assistant
PDF	Portable Document Format
PEAR	PHP Extension and Application Repository
PHP	Hypertext PreProcessor
PUL	PersonUppgiftsLagen
OLAP	OnLine Analytical Processing
OS	Operating System
QA	Quality Assurance
QWERTY	The layout of 'normal' full-size computer keyboards
R	The name of a widely used statistics program
REST	REpresentational State Transfer
RFC	Request For Comments
RIM	Research In Motion
RSS	Really Simple Syndication
SDK	Software Development Kit
SKU	Stock-Keeping Unit
SMM	Social Media Monitoring

SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer
SQL	Structured Query Language
T&C	Terms & Conditions
TLS	Transport Layer Security
TXT	TeXT file format
UDID	Unique Device Identifier
UNIX	UNiplexed Information and Computing System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAMP	Windows, Apache, MySQL, PHP
WAU	Weekly Active Users
WCSS	Within-Cluster Sum of Squares
WP	Windows Phone
XAP	Silverlight Application file format
XML.	EXtensible Markup Language
XNA	Xbox New Architecture

1 Introduction

The smartphone paradigm shift has caused an immense increase in the number and variety of mobile applications, or *apps* for short. The two leading mobile platforms, Google's Android OS and Apple's iOS, come with app catalogues – often called *markets* – each exceeding 700,000 apps [1]. In terms of numbers of downloads, the Google Play store has surpassed 25 billion app downloads (and probable installations[1]) [2].

Developers and publishers are becoming increasingly interested in gathering and exploiting data to improve their app's ratings and sales. There are clearly strong financial incentives for doing so. For example, according to financial analyst firms, a typical day on the Apple App Store yields in excess of US$15M in revenue [3], in a market that increased 47% since the same quarter last year [4]. *ABIresearch* projections that total mobile app revenues will be up to $46 billion in 2016 [20].

While both the Apple App Store and Google Play provide basic data regarding the number of downloads, user ratings, and demographics; more in-depth data (see chapters 6.1.8 through 6.1.12) that could be useful to publishers is often unavailable. Although there may be in aggregate a lot of data about apps, this data is not always publicly available or if it is available it may not be easily accessible. This gap has facilitated the rise of third-party analyst firms and tools, such as Flurry [125], Distimo [126], App Annie [127], and TestFlight [128], among others. (We will look at each of these in detail in sections 7.4, 7.5, 7.6, and 7.7; and the final sections of Chapter 7.)

These tools provide for access to more specific data about individual applications, but this data is still not sufficient from a developer's or publisher's perspective. There is also a sense of risk in not actually owning and controlling the data, when simply viewing this data via some third-party's website.

The introductory chapters will focus on the theoretical background needed to understand the rest of the thesis, including a brief market competition analysis; descriptions of the most commonly used & measured metrics for assessing apps, and a description of some of the existing third-party tools. Also a brief presentation of some data mining and statistical analysis tools that could be applied to the collected app data is given in Chapter 8.

Due to the immaturity of the smartphone industry, there exist only a small number of scientific papers on subject matter related to this thesis project. However, I have attempted to reference as much of the relevant recent material as possible when citing the relevant statistics. A thorough description of the current situation will be helpful when evaluating the implementation phase of this work, which consists of designing, implementing, and evaluating an in-house smartphone app statistics web platform.

[1] The app store can only really count downloads and does not know if the user has actually installed or continues to use any specific application.

2 Background

Official mobile app statistics are available from the marketplaces, such as Google Play and Apple App Store. However, these are often insufficiently detailed for an app publisher's needs. The interim solution of third-party data tools that gather and compile additional data are often based on the premise that app publishers are willing to share their official marketplace credentials with this third party, so that their software may act on the app publisher's behalf. These third party tools are generally web-based tools that restrict access to some of the raw data. Additionally, while these third party tools may be useful for generating simple charts and reports, they are often not suitable for spotting trends. In addition there are third party tools that require modifications of the app itself, for example to embed specific libraries for tracking purposes.

The optimal app statistics solution would be in-house in order to avoid dependence on external services that may or may not exist in the future or whose cost may increase. Storing all app metrics in a local database would also be an advantage over a third-party site where the data is read-only for the app developer. A single local data source offers better possibilities for statistical analysis and ensures traceability of the data. The size and system requirements should not be substantial, either. Normally, all data is also anonymized for personal integrity reasons.

What is conceptually very interesting is to combine many of the existing app statistics tools into a complete solution covering all aspects from demographics and device distribution, to number of downloads and rating version tracking, market ranking based on category and region, in-app usage patterns, app review analysis, and even automatic handling of bugs and software issues. Of course, for relative comparison of ones own apps versus competition some reliable third-party data sources must still be used. Nonetheless, having access to data for all these areas could potentially be a very powerful tool. Giving selected customers access to it would be even more powerful.

Controlling the database also creates the possibility for sharing this app data with others. This sharing might be done by using a specific API or by allowing read-only access to some portion of or the entire database by others. Another possibility is RSS feeds. For instance, an app publishing firm may share statistics for App A with Company B over a web service which ensures that only Company B can access the data about App A. Company B could embed or further publish this data on their own, subject to their agreement with the app publishing firm who provide this data. The statistics platform could provide semi-automated report generation, which currently requires considerable manual effort to compile the statistics obtained from several different sources.

3 Smartphone platforms

This chapter briefly reviews the dominant standards for smartphones, their history, and features. In addition, the chapter provides some statistics about these platforms.

As Android and iOS together account for approximately 86% of the total smartphone market [4], this chapter as well as the rest of this thesis will focus on these two platforms, but we will mention several of the smaller competing platforms, such as Windows Phone and BlackBerry with 2.4% and 5.3% market share (respectively) [4].

While Android is doing very well, now having 72% of the market up from 53% at the same quarter last year (see Table 1), it should be noted that some analysts forecast that iOS will regain some of the market. It is believed that many potential “iDevice” buyers were awaiting an upgrade until Apple’s iPhone 5 was released just recently, or they were awaiting the release of the iPad Mini.

Table 1 – Worldwide device sales by operating system 3Q12 [4]

	3Q12		3Q11	
Operating System	**thousand units**	**Market Share (%)**	**thousand units**	**Market Share (%)**
Android	122,480.0	72.4	60,490.4	52.5
iOS	23,550.3	13.9	17,295.3	15.0
Research In Motion	8,946.8	5.3	12,701.1	11.0
Bada	5,054.7	3.0	2,478.5	2.2
Symbian	4,404.9	2.6	19,500.1	16.9
Microsoft	4,058.2	2.4	1,701.9	1.5
Others	683.7	0.4	1,018.1	0.9
Total	**169,178.6**	**100.0**	**115,185.4**	**100.0**

3.1 Google Android

The Android platform is an open-source Linux-based operating system controlled largely by Google in a coalition with more than 300 partner companies [6]. It was first publicly released in 2008 and is at the time of publication at version 4.2, also called Jelly Bean. The Android Software Development Kit (SDK) is free and based on the Java programming language.

3.1.1 Devices

In addition to a plethora of different smartphones, Android is also a popular OS choice for vendors of tablets, media center systems, and other consumer electronics, as it does not come with substantial license costs. While Android’s versatility has given it a dominant position in the smartphone OS race, the diversity of devices that are running Android has led to *device fragmentation,* which is often criticized by developers, as they find it increasingly difficult to support all the different types of hardware, screen sizes, and screen resolutions. An app publisher recently collected user data from its ~700,000 users and found 3,997 different models of Android-powered devices [7]. OS version fragmentation is also a burden with regard to developers needing to consider backwards compatibility for their apps.

3.1.2 Marketplace – Google Play

Apps are officially distributed via the Google Play marketplace, previously known as Android Market. This marketplace is unregulated with regard to pricing, which allows the app publisher to decide which sales strategy they wish to employ: priced, free, subscriptions, *in-app purchases,* or combinations thereof. The app publisher keeps 70% of the revenue, with Google and its partners getting 30% [8]. The registration fee for this marketplace is a one-time cost of US$25. However, since the Android OS is open, another possibility is to distribute apps via third-party marketplaces or directly to the end-user via executable containers in the ".apk" file format. Apps published on Google Play are not specifically evaluated or verified by Google, so the responsibility for quality assurance lies solely with the app publisher.

3.2 Apple iOS

Apple released its first iPhone in 2007 [9], powered by the first version of iOS which can be seen as a light-weight version of the Mac OS. The latest version, released jointly with iPhone 5, is iOS 6 [27]. Unlike Android, iOS is proprietary software and has not been licensed to third-party hardware manufacturers.

Native apps developed for iOS are written in Objective-C with the Cocoa Touch framework, using the developer environment Xcode [28]. While Android development is open for all platforms, Xcode requires an Apple Macintosh computer. The SDK and other necessary software are free, but limited to testing of the app running on an iOS emulator. For testing on physical iDevices and to publish apps in Apple's App Store, a $99/year license is required.

3.2.1 Devices

Apple's product range is known for being relatively small, uniform, and compatible. The company ships four device series with iOS: iPod, iPhone, iPad (including the most recent iPad Mini), and Apple TV (a media center TV set top box). The tightly-controlled product family has ensured fewer problems with backwards compatibility and fragmentation than is the case with Android, at the expense of few choices available to the consumer. **Figure 3-1** shows the difference in iPad use as percentage of the total iOS use, by country.

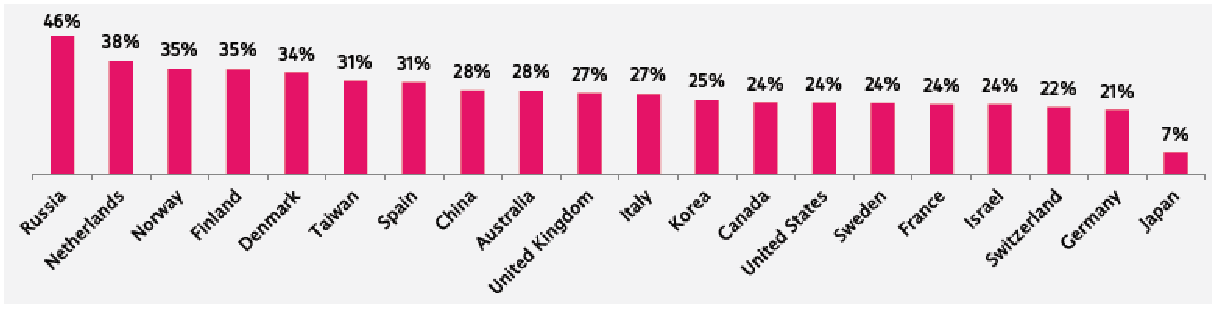


Figure 3-1 – iPad percentage of iOS app downloads, by country [3]

3.2.2 Marketplace – App Store

Each iOS app is verified by Apple before the app is accepted into the App Store. This evaluation is according to the guidelines and recommendations [33] that the developer had to accept when enrolling in the iOS Developer program. This process usually takes a week or so. Just as for Android, the revenue from sold apps are shared 70-30 between the app publisher and Apple [12]. There is support for in-app advertisements and in-app purchases as well.

However, app distribution via other channels is very limited, although Apple permits *ad hoc* distribution of an app in the executable container ".ipa" file format on up to 100 iOS devices per app per email or server. Apple also has a mechanism for within company distribution of an app via the Apple iOS Enterprise Distribution program [89].

While Google Play saw a big increase in numbers of app downloads and market share during 2012, the Apple App Store was far more successful financially for developers. For every $1 a developer made on an iOS app, he or she would make approximately $0.24 on the corresponding Android version of the app [21].

A simplified explanation for the great App Store advantage in profitability is the higher average app price for iOS [10] and that iDevices generally attract a higher-income customer base, as will be detailed in chapter 4 of this report. However, as we will see in following chapters, there are more profound explanations related to the difference in app business models.

3.3 Mobile web

While not a 'platform' per se, mobile websites are another way for developers and publishers to make their content available on smartphones. This approach has the inherent advantage of being cross-platform, assuring the largest possible user base by default; although studies show that thus far smartphone users prefer using an app over a mobile web site – in fact users prefer apps to surfing the web in general (see Figure 3-2) – both desktop and mobile editions of the web.

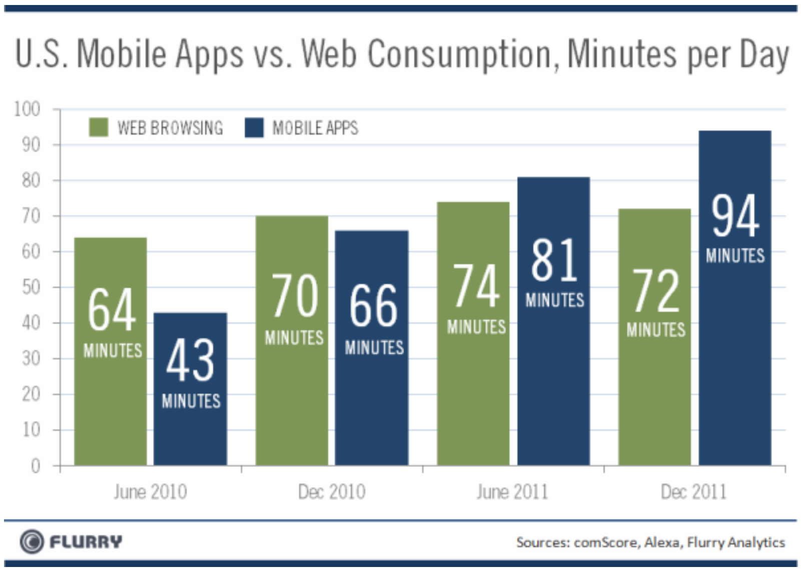


Figure 3-2 – Mobile apps versus Web consumption (US), minutes per day [36]

Another advantage of the mobile web is the relative maturity of the web and the number of available tools and the inherent possibilities of user session tracking (e.g. with cookies), as this enables powerful profiling using tools such as Google Analytics (see section 7.3 on page 27).

Some industry standards for the mobile web are settling. Additionally, there are techniques to bridge the gap between apps and mobile web, while ensuring platform-independence at the same time. Some of these techniques are described below.

3.3.1 jQuery Mobile

The jQuery JavaScript library that has been popular for the web for years and is available in a mobile version, jQuery Mobile. This is a “touch-optimized web framework for smartphones and tables” [37]. It is also compatible with HTML5. The library comes with extra tools, such as drag-and-drop interface design and CSS plug-ins.

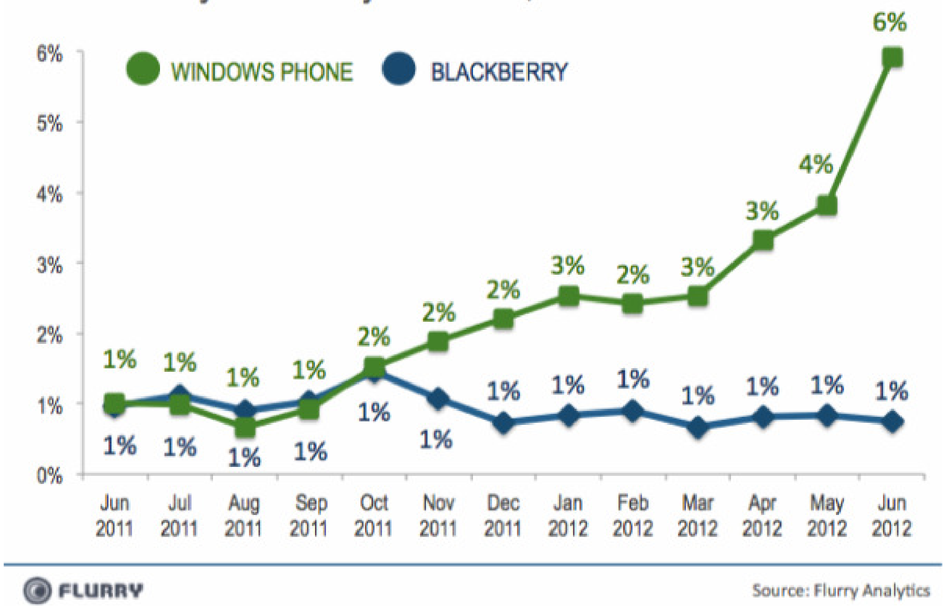
3.3.2 PhoneGap

For cross-platform mobile apps, PhoneGap has proven popular, with roughly 3.4% of the Android market [35]. This mobile development framework, purchased by Adobe, enables apps to be written in JavaScript, HTML5, and CSS; instead of the native programming languages (e.g. Java for Android and Objective-C for iOS). This makes a PhoneGap app platform agnostic. The difference from jQuery Mobile or other mobile web techniques is that the app is compiled and packaged as a native executable file – for instance, one for iOS and one for Android. Because of this, PhoneGap apps have access to the native OS libraries. The developer does not even have to have access to any device, but simply uploads their source code to a “cloud compiler” which will generate a platform executable for all the supported platforms.

3.4 Others

There are some other key smartphone players, although collectively they share 14% of the market (the remainder of the market left by Android and iOS). The primary contenders are Microsoft with its Windows Phone and the BlackBerry platform, owned by BlackBerry (formerly known as Research in Motion).

As shown in Table 1 (on page 5), BlackBerry controls 5.3% of the global smartphone market while Windows Phone had 2.4% [4]. Recent data concerning newly started app projects (see Figure 3-3) suggests that Windows Phone is reducing that gap.



	Jun 2011	Jul 2011	Aug 2011	Sep 2011	Oct 2011	Nov 2011	Dec 2011	Jan 2012	Feb 2012	Mar 2012	Apr 2012	May 2012	Jun 2012
WINDOWS PHONE	1%	1%	1%	1%	2%	2%	2%	3%	2%	3%	3%	4%	6%
BLACKBERRY	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%

Figure 3-3 – Flurry report on new project starts: RIM (BlackBerry) versus Microsoft Windows Phone [14]

3.4.1 Windows Phone

Windows Phone is Microsoft’s current platform (since 2010) [40] for the mobile market segment, replacing the earlier Windows Mobile OS. The new GUI for this OS is called Metro and is well aligned with the latest Windows for home computers (Windows 8) OS. Windows Phone is also at the time of this publication at version 8, succeeding 7.5 also known as Mango. Microsoft does not manufacture smartphone devices themselves, but have partnerships with multiple vendors, including Nokia, HTC, and Samsung.

Applications for Windows Phone are developed using Windows Phone Developer Tools which is an add-on to Microsoft Visual Studio 2010 or 2012. These tools provide an emulator for testing. Microsoft XNA (Xbox New Architecture) is also supported. The main programming language for non-game applications is C#.

The marketplace for Windows Phone apps is called Zune Marketplace and Windows Phone Store. The app revenue is shared 70-30 between Microsoft and the developer, exactly the same as for Apple's App Store and Google Play [41]. There is an annual registration fee of $99, the same amount as Apple charges iOS developers. Another feature in common with Apple is the verification process for each app before accepting it into the Zune Marketplace or Windows Phone Store.

3.4.2 BlackBerry

BlackBerry (formerly known as Research In Motion (RIM)) owns the BlackBerry brand. While many still see the BlackBerry devices as PDAs or Pocket PCs rather than a standard smartphone, BlackBerry products are very popular in certain niche markets. For example, in the Caribbean BlackBerry has 45% of market share [42]. BlackBerry manufactures their devices themselves. At the time of this publication, the BlackBerry OS is shipping version 7 and the new BlackBerry 10 products were just released [43]. BlackBerry devices are mostly non-touch screen with QWERTY mini keyboards, but there are touch-capable models too.

BlackBerry apps are distributed in the BlackBerry App World marketplace. Apps are written in Java using the BlackBerry JDE (Java Development Environment) or a corresponding plugin to an IDE such as Eclipse. App revenue is shared 70-30, as the general industry expectation dictates.

For BlackBerry tablets, powered by PlayBook OS 2.0, partial support has been launched for Android apps, but manual conversion is necessary and not all apps are compatible. Regardless, there have already been cases of pirated Android apps being resold as BlackBerry apps [44].

4 Demographics

Android users are generally a bit older than iPhone owners, as shown in Table 2 (data from August 2012) adapted from [18]. However, iPhone owners seem to generally have higher household incomes. This later aspect is not that surprising as the prices for the Apple products are generally higher compared to the prices for Android products.

Table 2 – Age and income distributions: Android and iPhone [18]

	Android	iPhone
Age span	% of audience	% of audience
13-17	5.4	6.5
18-24	17.2	19.9
25-34	25.1	26.4
35-44	21.0	18.7
45-54	17.1	14.8
55-64	9.3	7.7
>64	5.0	6.0
Household income span (pre-tax)		
<US$25k	17.2	8.1
US$25k to < US$50k	22.4	14.4
US$50k to < US$75k	20.0	19.6
US$75k to < US$100k	14.5	17.1
>US$100k	25.8	40.7

Narrowing down the demographics to the extremes – low app users and high app users, the Nielsen research firm found that the majority of iOS power users consist of people 25-44, while not surprisingly people 55 and older are more likely to be low app users. For further details see Figure 4-1.

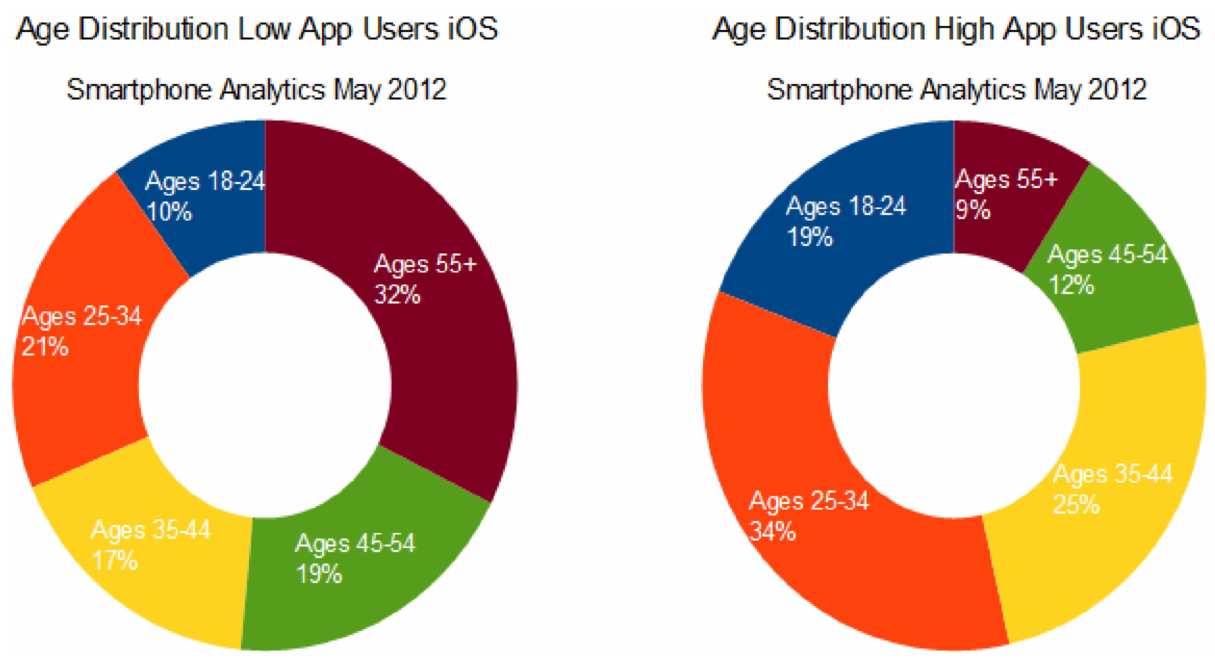


Figure 4-1 – Age distributions of high and low iOS app users, by age [19]

5 Business models

In addition to the traditional sales method of offering one's app for sale at a certain one-off price (see section 5.1) or for a periodic fee (see section 5.2), new business models have emerged in the app trade. The adoption of these other business models is growing with time. These new business models are described in the remainder of the chapter.

5.1 One-off payment

This is the most obvious app business model. Develop an app, set a price, and sell it to users. There is only a one-time cost for unlimited app use. Revenue is split 70-30 between the developer and the marketplace/platform owner.

One problem with this payment model is that the enormous selection of apps (700 000+ in both dominant marketplaces) has pushed prices down to levels that make it very hard to be profitable solely based upon the revenue stream from new downloads. The average selling price in Apple's App Store is ~US$2.15 [3].

However, one-off payments are popular among the top game publishers (such as Electronic Arts and Gameloft), where 35% of their revenues in 2012 came from one-off fees [3]. In fact, the list of the top 100 grossing apps on App Store is often 75% games [54]. I believe that the reason for this is that these well-recognized publishers have gained sufficient consumer trust so that the up-front purchase price is not perceived as a risky investment. Of course, setting a one-off price for an app or game does not prevent the publisher from also implementing a scheme for in-app purchases.

5.2 Subscriptions

Apps that provide access to something – a service or feature – are often suitable for the subscription payment model. An example is music streaming services, with apps that are free to download, but require monthly renewal of the account for a fee. Online multiplayer games may also fit this profile. Some newspapers and magazines also offer their premium material (not available on their public websites) through app channels, where access requires an active subscription. These subscriptions may be automatically renewed or prolonged at fixed time intervals.

5.3 Free or ad-sponsored

The free app business model for the purposes of this thesis includes both completely free apps that may be published as promotion or by amateurs, and apps that are free to download but derive income from embedded advertisements (ads). Such ad components are offered by Google through its Admob service [45] (replacing its earlier service Adsense for mobile), Apple through its iAd program [47], and by others, such as Crisp Wireless' platform [69], using a large variety of mobile ad types. Some of these different mobile ad types are shown in Table 3.

Table 3 – Different mobile ad types [69]

Ad type	Description
Full screen	An ad that takes up the entire screen for a limited period of time
Expandable	A smaller ad that expands upon user interaction
Location-based	Determined by geographical location (assuming the user opted in to be tracked) – for example directing the way to the nearest dealer or retailer or a product
Tap-to-video	Leads directly to a video
Tap-to-social network	Leads directly to a Social Network such as Facebook or Twitter
Commerce-enabled	Allows user to buy directly from a designated retailer, e.g. iTunes
Tap-to-call	An ad containing a phone number that can be directly called

5.4 Freemium

A variant of free is the business model *freemium,* a wordplay combination of free and premium coined by Jarid Lukin [26]. The idea is to release two versions of the app: a light-weight free version of the app that creates interest for the paid profession (pro) version. The model resembles the shareware or demonstration concept widely used in the PC world.

Gartner's research report "Market Trends: Mobile App Stores, Worldwide, 2012" [5] projects that 89% of app downloads are free, and that this percentage will continue to rise for years to come. See Table 4 for details.

Table 4 – Mobile App Store downloads, worldwide, 2010-2016 (millions of downloads) [5]

	2011	2012	2013	2014	2015	2016
Free Downloads	22,044	40,599	73,280	119,842	188,946	287,933
Paid-for Downloads	2,893	5,018	8,142	11,853	16,430	21,672
Total Downloads	24,936	45,617	81,422	131,695	205,376	309,606
Free Downloads%	88.4%	89.0%	90.0%	91.0%	92.0%	93.0%

5.5 In-app purchases

The freemium line of reasoning has led to a growing trend of in-app purchases, where the app is mostly a shell and its actual content is purchased separately. For example, games could be provided with just a few levels and offer an in-app store that sells additional content. Data from the Apple App Store suggest that this business model actually creates the major part of app revenue today (see Figure 5-1).

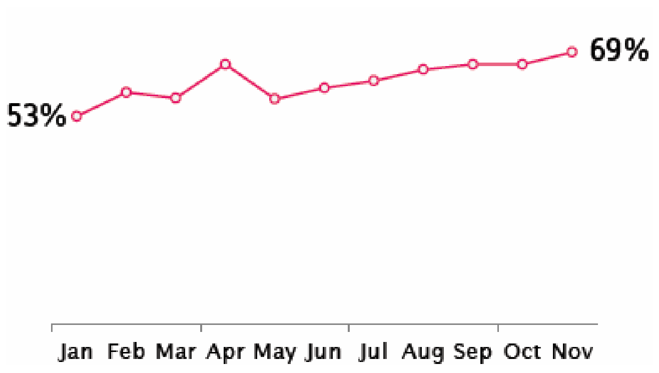


Figure 5-1 – 2012 development of in-app purchase revenue on App Store [3]

Analysts are concerned about the in-app purchase concentration in just the game app genre. Mark Beccue, senior analyst mobile services, has stated: *"The vast majority of current in-app revenue is being generated by a tiny percentage of people who are highly-committed mobile game players. We don't believe the percentage of mobile game players making in-app purchases will grow significantly, so for in-app purchase revenues to grow, mobile developers other than game developers must adopt it."* [20] His view is collaborated by findings that only 0.5 – 6% of players normally make in-app purchases in games [54].

Beccue adds that the in-app purchase revenue from Android apps could actually have been hindered by Google, as it did not introduce this feature until July 2011, and only for 17 countries as late as December 2011. This can be compared with Apple, who enabled in-app purchases starting with iOS version 3.0 (released in September 2009) [22].

Another positive aspect of in-app purchases is that a developer may put together a functional but very light version of the app and release it via the marketplace. If it attracts interest, they can quickly add additional content to keep up with the demand, while if the app should fail, the initial time investment is reduced as is any associated loss [48].

6 Metrics

As there are extremely many different aspects of app statistics (including understanding hardware segmentation, user demographics, payment models, and other aspects that are of interest from a developer or publisher point of view), this chapter describes some of the most common measurements within the app trade and elaborates with examples where applicable.

6.1 Quantitative

Quantitative indicators are numerical, measured values of specific metrics. They are easily interpreted and can be used as raw input data to data mining processes such as clustering. The later chapters will primarily analyze data with respect to quantitative indicators.

6.1.1 Downloads

This measure is perhaps the most obvious. How many people downloaded my app? This measure may also differentiate downloads per app version. Keep in mind that an average (US) smartphone has 41 apps installed [31], so app usage time is bound to be diversified.

Regardless of free or paid, a high percentage of Android and iOS apps actually attract very few downloads. Senior analyst Tim Shepherd stated *"We estimate that up to two-thirds of the apps in leading consumer app store catalogs receive fewer than 1,000 downloads in their first year, and a significant proportion of those get none at all"* [49]. These apps are unlikely to ever be profitable for the publisher. These apps are often grouped together and referred to as the 'long-tail', as the distribution curve of downloads flattens along the axis. However, it should be noted that the cumulative revenue of such long tailed distributions can be very profitable, see for example [90].

6.1.2 User rating

Both the Apple App Store and Google Play give app users the possibility to rate each app with a score on a scale of 1-5 with an optional comment. According to Brown [10], the average app rating is virtually identical on Google Play and Apple App Store – 3.58 as compared to 3.56.

Ratings seem to differ between different app categories. Educational apps seem to be rated the highest with average ratings over 3.8 while the News category averages less than 3.2.

There are also notable platform differences. For instance, Android has been widely criticized for its weaker game selection, as is reflected in the 3.18 average rating for games, while the Apple App Store average is 3.70 for games. On the other hand, the Tools category sees Android score 3.86 on average, perhaps due to better customization options and less strict user policies, while the Apple App Store average score is 3.40.

As pointed out by Hao et al. in [16], the marketplace owner receives 30% of the app revenue. This means that if higher ratings are correlated more sales, it is also in the marketplace owner's interest to drive up ratings. This mutual dependence is worth keeping in mind when reading the rest of this master's thesis.

6.1.3 Active users (numbers and percentages)

The recent remake of the Android Developer Console offers statistics for Device and User uninstalls, which means that a developer may track how many users are installing and uninstalling the app per day, version, country, language, device, or cell carrier. This is a very crude metric, perhaps the uninstall statistic is most suitable for setting up automatic alarm triggers so that if a new version causes an abnormally large number of uninstalls (drop-outs), then the developer or publisher should be alerted so that they might know that something has gone wrong. No similar statistics for uninstalls are officially available via the Apple App Store / iTunes Connect web. Note that "active users" refers to what in this thesis project we will call *user retention* or *loyalty* (this will be discussed further in section 6.1.8 starting on page 20).

6.1.4 Category and Market (rank and share)

There are many types of featured placements on Google Play: Featured, Staff Picks, Top Free, Top New Free, Top Paid, Top New Paid, Top Grossing, Trending Apps, Editor's Choice Apps, and Top Developer [38]. Apple App Store has a similar, although smaller set of promotional marketplace advertisement spots. So a top ranking app within a category or the overall marketplace not only means a top position in the listings – it also comes with free advertisements in these feature spots. This skew in exposure is reflected in the metrics.

As can be seen in Figure 6-1, the top five or ten applications make dramatically more money based on their top ranking alone, than those just below. However, the blue line (2012) is less steep than the green one (2010), indicating that the 'long-tail' is now getting a larger portion of the total app revenue. This could be explained by the much larger app catalogues in 2012 as compared to 2010. These findings are collaborated by the yearly report in [3]. These finding are also reinforced by results from [15] that found a 1st ranked app gets 150 times more downloads than an app ranked at position 200. However, at the very top of the 'head', things are more skewed as at the end of 2012 with just 7 apps accounting for 10% of the total Apple App Store revenue, compared to 11 apps in the beginning of 2012. For Google Play the figure is only 4 apps [3].

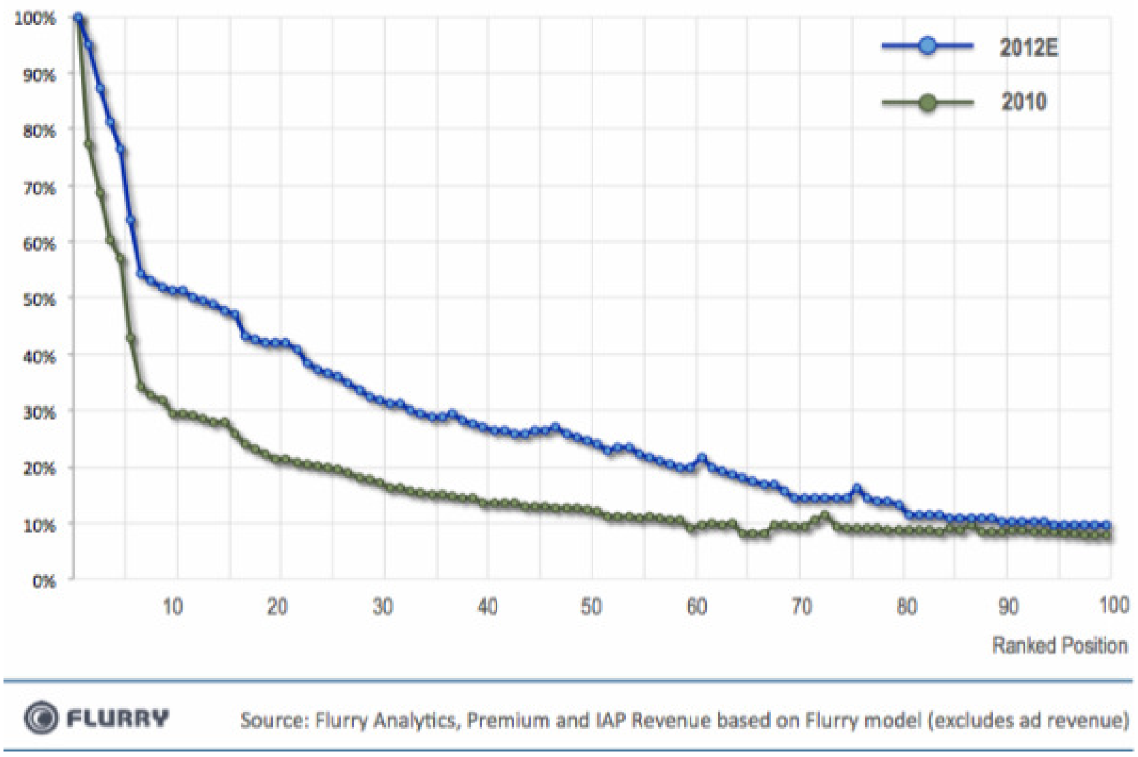


Figure 6-1 – Worldwide iOS & Android normalized revenue per rank [24]

In terms of revenue, it is clear that games represent the number-one revenue category (see Figure 6-2). In fact, games account for one third of revenue and an even larger share of downloads. This is largely due to in-app purchases [20] that seem to suit the game genre very well. Among the top 10 highest grossing cross-store publishers last year, 9 were game publishers [3].

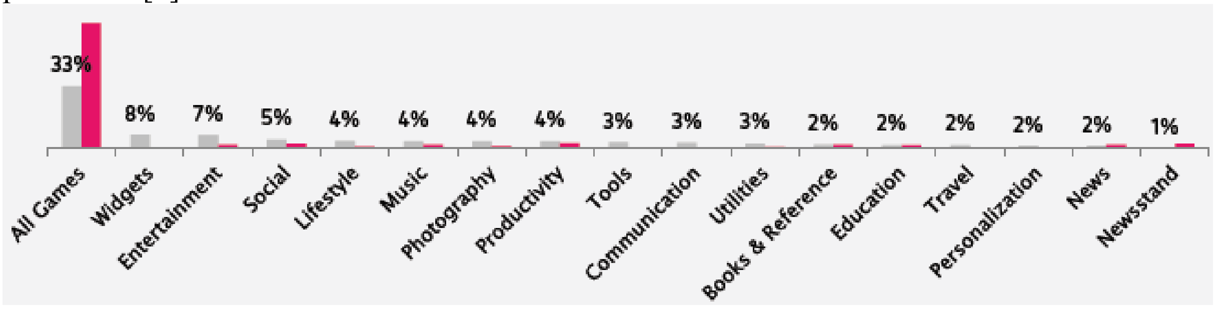


Figure 6-2 – Download percentages by app genre [3]

6.1.5 Geographic region (rank and market share)

Apple's iTunes Connect and the Google Play Developer Console both offer quite good possibilities to understand an app's user demographics. Not only are downloads tagged with the user's country, but also currency used for the purchase and the user's selected language. This information can be used to create targeted advertising, decide upon additional localization (i.e., translation into a given language and additional customization), and more. For these reasons it is important to have some general knowledge of the worldwide smartphone penetration and revenue shares.

App Annie Index November 2012 [25] states that for the preceding month, Japan overtook the USA for the first time, in terms of monthly revenue on Google Play. Japanese revenue on Google Play had increased ten-fold since January 2012. Japan accounted for 29% of revenue; US 26%, South Korea 18%, UK 4%, and the combined rest-of-world 23%. On the other platform, the US dominated iOS revenue for the same period with the US having 33%, Japan 14%, UK 7%, Australia 5%, and the rest-of-world 40%. Overall the US is the largest market by revenue, according to [3]. Notable is the lack of iPad sales in the Japanese market (see Figure 3-1 on page 6).

The smartphone penetration, e.g. the potential customer base, has of course increased considerably during 2012. According to the Google blog, referencing their Our Mobile Planet initiative [52], *"Today, we're releasing new 2012 research data, and the findings are clear—smartphone adoption has gone global. Today, Australia, U.K., Sweden, Norway, Saudi Arabia and UAE each have more than 50 percent of their population on smartphones. An additional seven countries—U.S., New Zealand, Denmark, Ireland, Netherlands, Spain and Switzerland—now have greater than 40 percent smartphone penetration."* - Dai Pham, Group Product Marketing Manager, Google Mobile Ads.

6.1.6 Demographic (rank and share)

Sensitive user data such as age, gender, and ethnicity is not readily available from either the Apple App Store or Google Play. Such data would likely be considered invasive to collect without asking permission[1]. However, Apple seems to track some user behaviour.

Apple previously used a unique hardware identifier a Unique Device Identifier (UDID) to allow mobile advertising companies to track individual users' behavior. However, after a hacker attack released 1,000,000 UDIDs in September 2012, Apple encouraged iOS developers to abandon functionality that requires a UDID [50]. Instead, starting with iOS 6

[1] In fact in some countries it might be illegal for a company to collect this data.

the company introduced Identifier For Advertising (IDFA), which is not device-specific but rather is more like a web cookie. The IDFA is deleted and a new one is created when the user resets their device. IDFA is active by default, but may be turned off [51].

Some generic demographic raw data is available for download through Google's Our Mobile Planet initiative [52] (see section 7.9 starting on page 33).

6.1.7 Paying users (share)

For freemium apps or apps offering in-app stores, the percentage of paying users measures how large a fraction of the users are paying customers versus those who settle for the free features. Note that this measurement may be skewed due to the fact that a relatively small, highly committed share of the user base account for the vast majority of the in-app purchase revenue [20].

For apps that are dually released in a free and a paid version, the share of paying users would be calculated as the number of downloads for the paid app, divided by the number of downloads for the free version plus the number of paying users. For apps with more complex business models, the calculation method for this metric is more complex.

6.1.8 Retention after a given time period

The retention time metric is sometime also referred to as loyalty. How many of the users that once downloaded the app still use it after a certain time period? Common thresholds include periodical use (Daily, Weekly, and Monthly Active Users – abbreviated as DAU, WAU, and MAU). Retention rate is highly dependent on app category and purpose (see Figure 6-3). Some apps are designed to be used rarely or even just once, while social media apps are likely intended to be used several times per day.

Mobile analyst firm Flurry has used a sample of apps used more than 1.7 billion times each week to see what patterns emerge in terms of loyalty (defined as having used the app during the last week). The overall figures (calculated as the mean over all categories) showed that an app that was installed in 2012 would have a 54% probability of still being used in 30 days, 43% after 60 days, and 35% after 90 days with an average of 3.7 app usage sessions per week. Interestingly enough, the 90-day retention rate was up significantly from 2009 when the corresponding probability was only 25%. This may be interpreted as a maturing app market that features apps that provide greater long-term value. The wider selection may also have caused the decrease in average app sessions per week from 6.7 uses in 2009 to 3.7 uses in 2012. As seen in Figure 3-2, however, overall time spent on apps is up considerably, which means that we now spend more total time on apps, but divide our attention across more apps.

Racherla, Furner, and Babb [17] speculate (along with others) that retention rate will become increasingly important as a metric, as smartphone penetration is projected to cover the majority of the potential customer base [52]. The idea is that in such a maturing market that it will be of greater importance to retain existing users than to attract newcomers, hence the business models are moving away from one-off purchases towards models that require user interaction: ad-sponsored apps, subscriptions, and in-app purchases.

A further contribution to these new app business models is piracy. Kharif [34] estimates that app sales would be 20-50% higher if it were not for piracy. Ad-sponsored apps, subscription-based apps, and those apps with in-app purchases are not as sensitive to piracy.

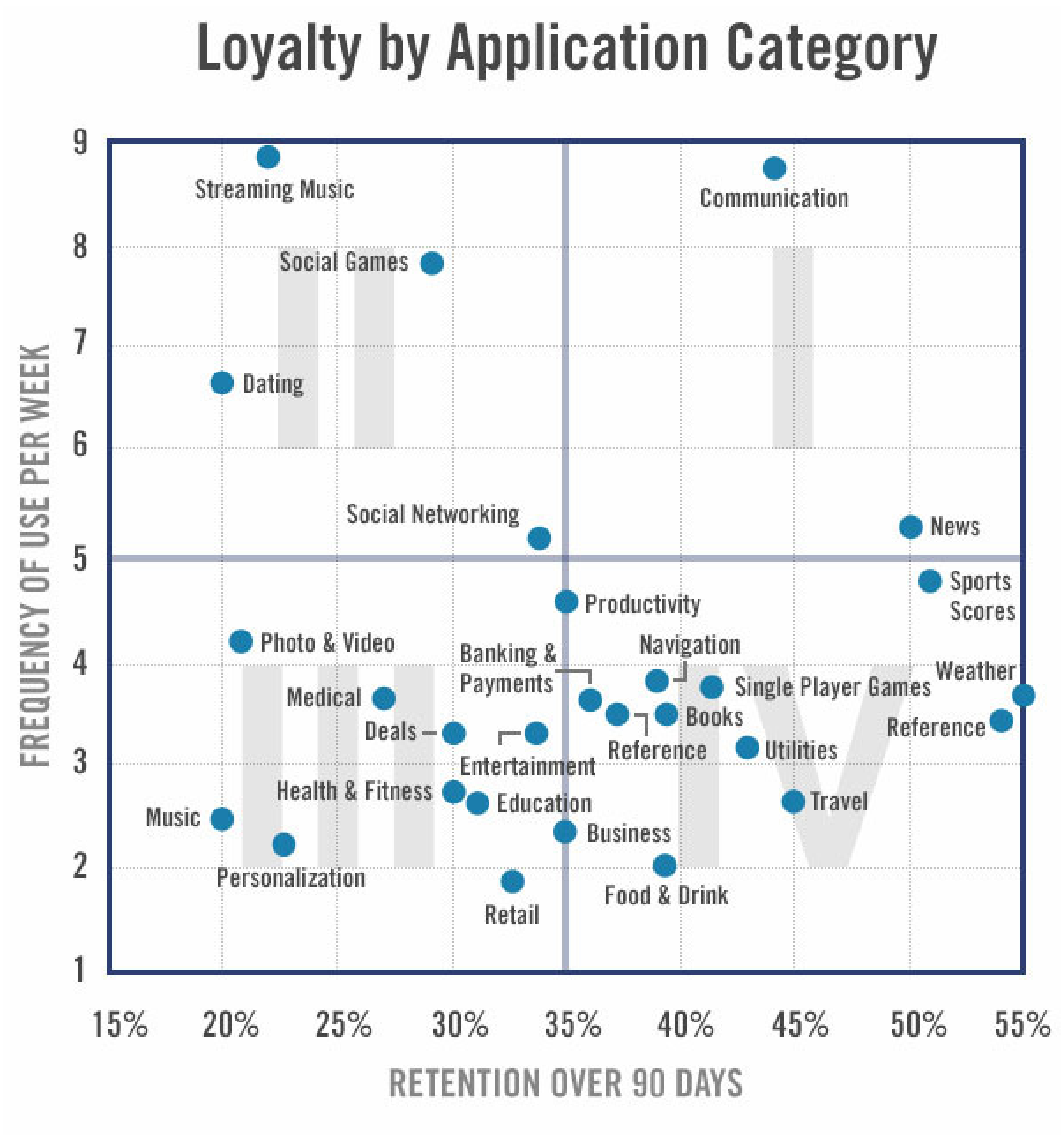


Figure 6-3 – Loyalty by app category [13]

6.1.9 Average time spent (session duration)

We have already noted (see Figure 3-2 on page 7) that an average smartphone owner spends 92 minutes daily using their apps. Obviously it would be interesting for an app maker to know how much time is spent within their app. However such metrics, on the individual level, are hard to gather without invading the user's privacy or modifying the app to include some method of tracking its usage (which the user might opt-in to use).

General results from analyst firm Flurry in September 2012 [32], suggested that the average app session on a smartphone is 4.1 minutes and on tablets exactly twice this, 8.2 minutes. The majority of time is spent on apps from the Games category; specifically 39% of smartphone app time and 67% on tablets [32].

6.1.10 In-app revenue

Statistics concerning in-app purchases seem to readily motivate the existence of this payment model. Even though only 0.5 – 6% of game players pay anything at all, when they do, they spend on average US$14 per transaction [55], with 51% of total revenue generated from transactions over US$20. This fact might be worth keeping in mind as a developer of an app with in-app purchases.

One reason for transaction sizes being larger than some might expect, is inventions such as in-app wallets or even internal currencies. An example of the latter is Electronic Arts popular game The Sims Freeplay, where a fictive currency called Simoleons can be purchased, along with other virtual objects [57].

As shown in Figure 6-4 there are also obvious differences between the two sexes and as a function of age group. Men spend more than women on average per transaction, US$15.6 compared to US$11.9 with the difference being most notable for people under 18. It can also be seen that the payment willingness seems to peak for the age group 25-34.

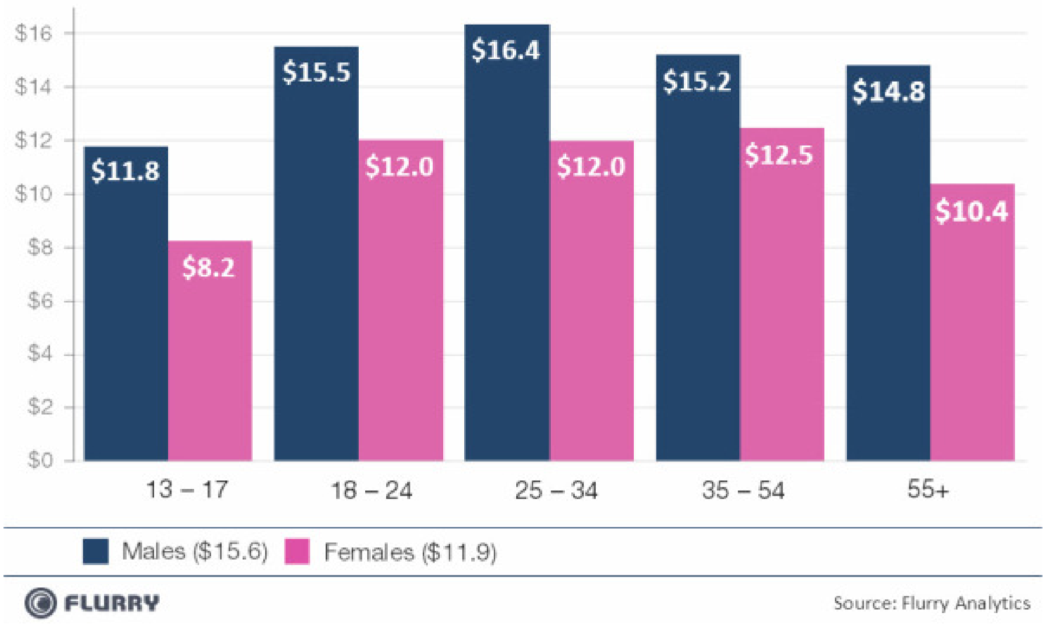
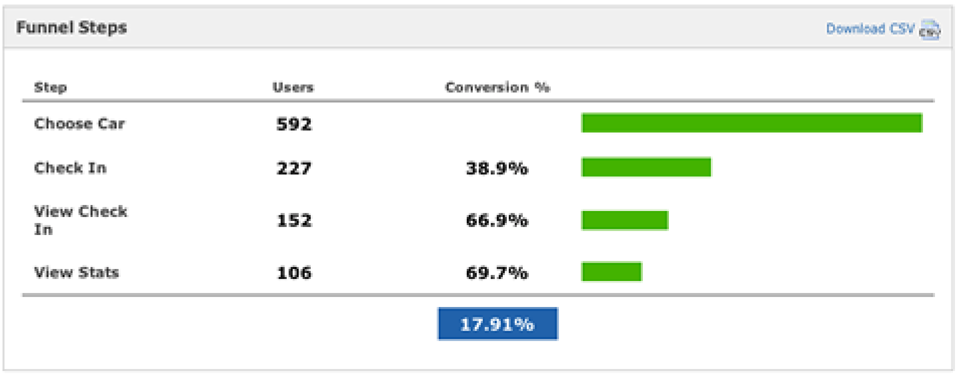


Figure 6-4 – Mobile Freemium Games: Average transaction size by age and sex (amounts in US Dollars) [56]

6.1.11 Funnel (conversion rate)

The funnel (conversion rate) metric can be defined as the percentage of customers that are browsing the app download page, who actually decide to download the app (converting from a potential customer into a user). This definition is used by the analyst firm Distimo for its App-Analytics platform, where they track web landings (views of the app's download page) versus the actual number of downloads [53].

A more general use of the funnel or conversion metric is to track an app session from start to a clearly defined goal. One could possibly measure the number of sessions that pass through defined checkpoints during a session. Typical examples include dividing booking procedures into steps, then keeping statistics for each step. Google Analytics is one of the most widely used tools within this area, but many others offer similar functionality. In the screenshot below from the web service Flurry Funnel Analysis, we can see multiple checkpoints which are being tracked for their conversion rates.


Funnel Steps

Download CSV

Step	Users	Conversion %
Choose Car	592	
Check In	227	38.9%
View Check In	152	66.9%
View Stats	106	69.7%
		17.91%

Figure 6-5 – Sample screenshot of Flurry Funnel Analysis

6.1.12 Return rate (regret and bounce)

The return rate metric should not be confused with uninstalls, since a user might uninstall for reasons that have nothing to do with the app, e.g. he or she might need more available space on the device. An app return is a claim for money back from the marketplace. Google Play has a once-per-app 15 minute return period which acts similar to a guarantee, so that if a customer purchases an obviously broken app or runs into compatibility issues before or after installing it, they may revert their purchase [60]. While Apple's App Store officially has the policy that "all sales are final" [61], there are discussions online that suggest that if you make an official complaint they may grant a return request. For mobile web sites, regret is interpreted as a session lasting shorter than a certain number of seconds or a session consisting of a single page view before leaving the site's domain.

6.2 Qualitative

The thesis will focus mainly on quantitative, measureable figures as interpreting human input manually is simply too cumbersome a task for this thesis project. However, there are many examples of automated web-crawling software that look for mentions of selected keywords to pinpoint words and topics in order to compute trends (referred to often as "trending" words and topics).

Applying this type of tool is often called Social Media Monitoring (SMM). The span of such tools ranges from the extremely simple Google Alerts that monitor certain keywords and compile daily email reports, to advanced dashboard tools such as Radian6 which starts at US$5,000 per month [58] and is used by enterprises such as Dell and Pepsi. The more advanced SMM tools can monitor most social networks in addition to the regular web, and allow for more detailed parameters. Similar features are also available in Google Analytics which is more extensively evaluated in section 7.3 starting on page 27.

The firm uTest [10] applied web-scraping SMM techniques to the particular case of app comments on the both major marketplaces, and found the issues shown in Figure 6-6 to be the most common complaints mentioned in app reviews. A brief justification of these results suggest that the installation process is more prone to errors on an Android handset, possibly because of compatibility issues (due to device fragmentation) and the need to approve each of the required application permission requests. Apps on the Apple App Store are generally much more expensive [10], as is reflected in complaints about pricing, but on the other hand Apple's rigorous app verification process seems to be reflected in fewer overall complaints about technical issues for iOS apps.

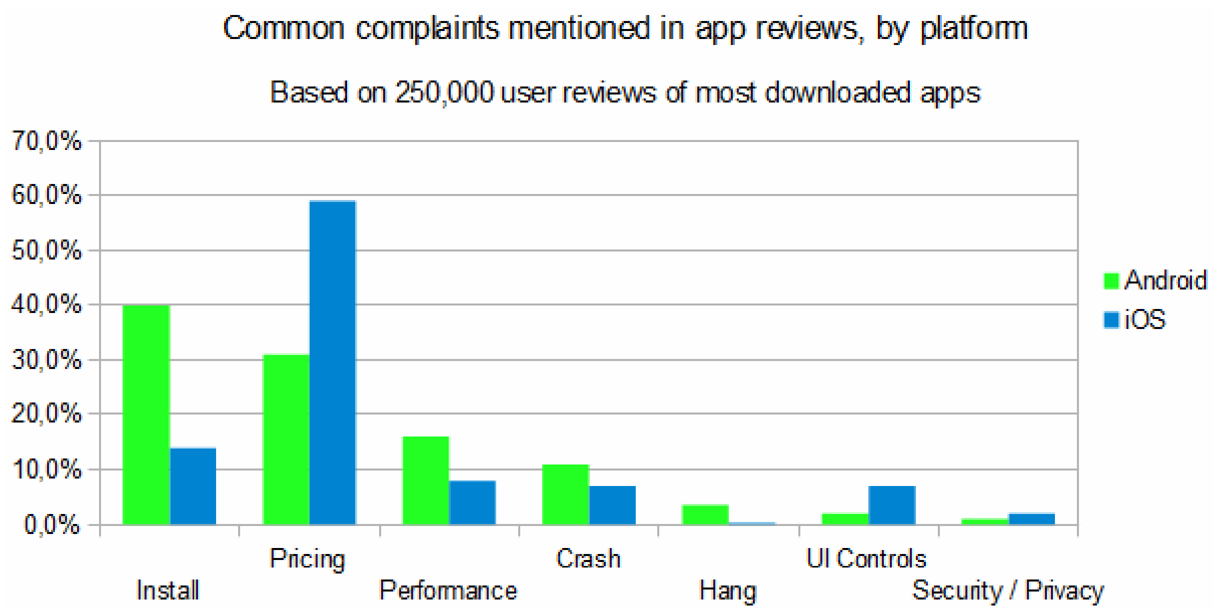


Figure 6-6 – Common complaints mentioned in app reviews, by platform [10]

The quantitative user reviews were likely mapped to separate categories using keyword filters, effectively turning the quantitative textual comments into a qualitative statistic. While this is a somewhat rough method and has considerable margins of error, it saves a lot of time and might be accompanied by manual follow-ups to verify the findings. I have attempted a similar approach of my own, which will be explained in more detail in section 10.2.8.

7 Web-based data collection and analysis tools

There are plenty of available tools for app statistics, almost all of which are online or web based. This chapter discusses some of the more popular alternatives and their features, strengths and weaknesses. The tools discussed in this section are evaluated briefly (after registering accounts and logging in whenever applicable) – not only from the perspective of the app company, but also as inspiration for the in-house app statistics solution to be developed. All third-party tools were free in the editions I have used.

7.1 Apple App Store

While the general public may only browse the Apple App Store catalogue, and judge ased upon a select few metrics, a few additional figures are available for developers after logging into their account. Overall, Apple's first-party solutions are still missing a few critical features however.

7.1.1 Publicly available

A potential app buyer browsing for apps (on the iTunes Preview website, in Apple´s iDevice App Store app or in their iTunes client) does not see the number of app downloads. He or she may only read ratings/reviews and see the average app rating (1-5 stars), the number of votes for the current app version, and aggregated results for all versions of the app.

As a rating is the only metric available to the browsing customer, the feature or category spots are likely to be of high importance, as will be discussed in-depth statistically in the following chapter. Listings such as "New and Noteworthy" and "10 Essentials" function as advertising spots for apps successful enough to rank high enough to be included in these listings. There are also "Top 10" categories for free and paid apps, separately.

7.1.2 Developer account – iTunes Connect

Unlike consumers, iOS app developers can log into a web platform called iTunes Connect (iTC) where they may view basic statistics, such as downloads and percentage trend change for their application on either a daily or weekly basis. An example of these statistics is shown in Figure 7-1. Surprisingly, iTC only displays statistics from the last 14 days or the last 13 weeks (a quarter of a year), respectively. This is a major limitation that prevents app publishers from understanding long-term trends (unless they themselves archive this data and process it themselves). This inhibits the app publisher from making basic comparisons of the current version of an app against previous app versions. Partly as a response to this, third-party solutions have arisen, such as the AppDailySales script described in section 7.8 starting on page 33.

In addition to statistics that can be view via the web platform, it is possible to request two types of reports, compiled as tab-delimited ".txt" files: apps and in-app purchases. These reports consist mostly of metadata about which regions and localizations the app is available in, along with basic information about its requirements (iPhone only or compatible with iPad, minimum iOS version, etc.). Although there are flags for whether the app supports the iPad or not, the developer may not opt to separate statistics between phone and tablet. This is another limitation that third-party sources have tried to rectify.

Alternatively, reports can be downloaded with a Apple provided Java class named Autoingestion [39]. This class takes parameters according to `java Autoingestion <username> <password> <vendorid> <report_type> <date_type> <report_subtype>`

`<date_yyyymmdd>` and downloads the export files just as if they were manually requested from iTunes Connect. However, the limited date ranges still apply.

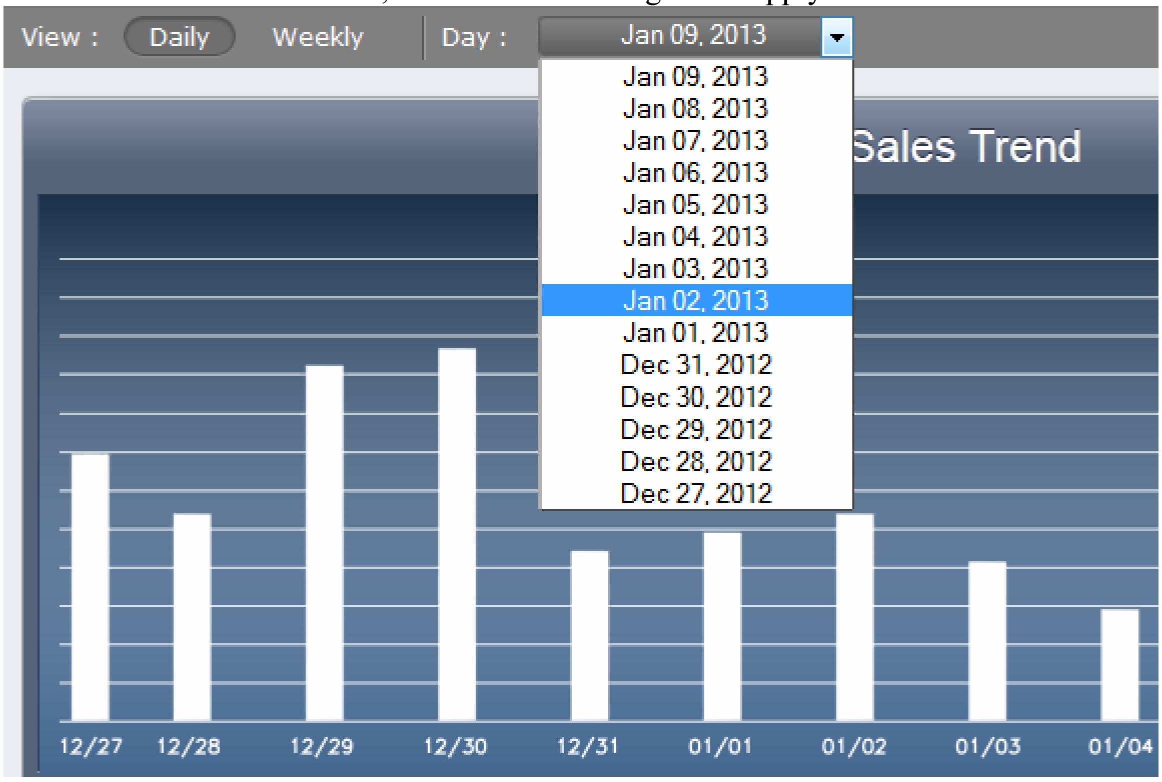


Figure 7-1 – Sample screenshot of iTunes Connect

7.2 *Google Play*

While Google offers a bit more in-depth detail in their statistics, both for the app-browsing potential customer and the logged in developer, compared to Apple – some elementary features are still absent. However, the export functionalities are significantly better than for Apple.

7.2.1 Publicly available

The number of downloads per app is not displayed when browsing Google Play (nor on the web or in-app). However, unlike Apple's App Store, anyone can see a rough estimate of the number of downloads and even view a trend chart. The number of downloads of the app will be presented as a range, such as "1 000 – 5 000" or "10 000 – 50 000". These gross statistics give the browsing customer a general idea of this app's popularity.

App ratings are listed along with review comments. These ratings give the average rating (on a scale of 1-5). Just as for Apple's App Store, you may flag another user's review as either helpful or spam.

Another similarity with Apple's App Store is the presence of 'feature spots'. Editor's choice and Top sellers are prominent advertising venues, and you can even find listings of "Recommended for you" which is not defined in detail, but is likely based on your previous app history.

7.2.2 Developer account – Developer Console

Even without logging into the developer's console, the publicly available download range data may be useful for app publishers for benchmarking their app against its competition.

Knowing that the competitor's app made the featured spot and that the number of downloads stack has reached six figures, may assist the app publisher in making a decision of whether or not to launch an ad campaign.

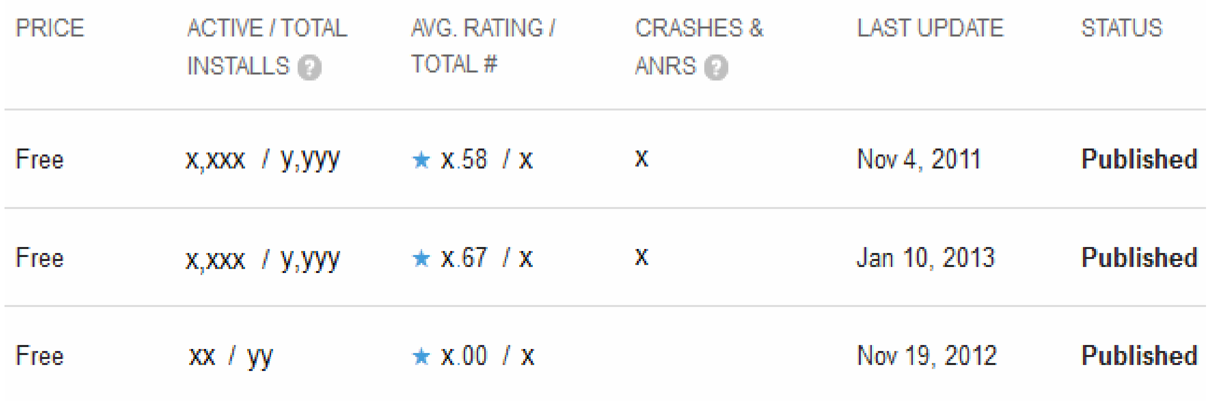
The following metrics are only available after logging into the Google Play Developer Console:

- Ratings statistics
  - Star ratings
  - Cumulative average rating
  - Daily average rating
- User statistics
  - Total user installs
  - Active user installs
  - Daily user installs
  - Daily user uninstalls
- Device statistics
  - Active device installs
  - Daily device installs
  - Daily device uninstalls
  - Daily device upgrades

All of these figures can be broken down by Android version, device, user's country, user's language, application version, and mobile carrier [38]. All of these statistics can be exportable to a CSV format file, where the complete history is available for every dimension (without the 13-week limitation of Apple's App Store). The division of data by dimension is an indicator that the Developer Console data may actually be seen as a tool compliant with Online Analytical Processing (OLAP), where the database is seen as a hypercube of facts available in n dimensions [78].

There are also automated crash log reports and Application Not Responding (ANR) reports. For crashes, these data are grouped together by type so that the developer may understand the frequency of different bugs. Complete stack traces are also available for debugging.

An example screenshot from Google Play Developer Console is shown in Figure 7-2.



PRICE	ACTIVE / TOTAL INSTALLS	AVG. RATING / TOTAL #	CRASHES & ANRS	LAST UPDATE	STATUS
Free	x,xxx / y,yyy	★ x.58 / x	x	Nov 4, 2011	**Published**
Free	x,xxx / y,yyy	★ x.67 / x	x	Jan 10, 2013	**Published**
Free	xx / yy	★ x.00 / x		Nov 19, 2012	**Published**

Figure 7-2 – Sample screenshot of Google Play Developer Console

7.3 Google Analytics

After dominating the full-fledged web, Google Analytics is also available for mobile use. Their library for Android is included in approximately 7.3% of Android apps, according to

[35]. The tool allows (among other things) for funnel/conversion rate analysis based on check-points and goals.

Google Analytics Mobile Apps SDK is at the time of this publication at version 2 and is available for both Android and iOS [59]. This SDK enables detailed data collection from app users. Many of the basic features such as OS version and device language can already be seen from the iTunes Connect and the Developer Console, respectively; but the difference with Analytics is that these statistics are available on a per user and per session level. Google Analytics is also available for use with the mobile web.

Examples of the detailed metrics include average session duration, number of screen views, percentages of new and returning users, as well as their languages and locations. The developer may also track the order of viewed screens (including which screens the user leaves the app from), how long app elements take to load, and errors and app crash logs. Since the developer may set triggers on any event, such as buttons being pressed on a screen, it is not necessary to change the current screen view to record an event. Therefore, sessions may be either manually closed – typically after completing a process – or because they timeout after a given period. Developers may also set custom variables that are transferred from the app to Analytics, and these variables may be cross-referenced against apps from different smartphone platforms to generate an overview.

Real-time data is available which allows following the movements of users who are presently using the app. There are also specially tailored funnel functions for the Ecommerce category. Funnels can be visualized as flows, as shown in Figure 7-3. The thicker the connecting line is, the more traffic flows via that connection. Red markers indicate a user quit.

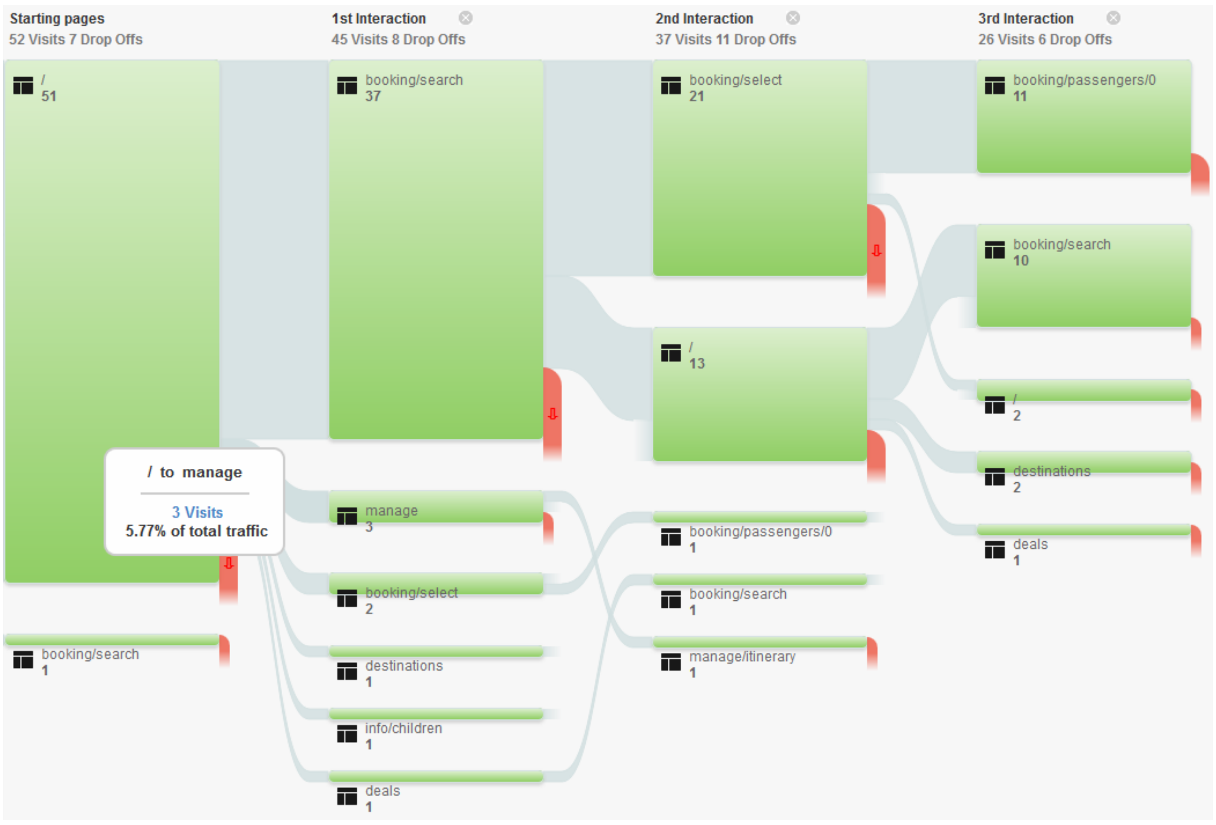


Figure 7-3 – Sample screenshot of Google Analytics: Visitors Flow

There are also a vast set of functions for tasks related to statistics, such as advertising (with the Admob package), search engine optimization, and tracking of social networks conversions.

7.4 Flurry

Flurry is primarily a tool for gathering app session data. It is available for iOS, Android, BlackBerry, and Windows Phone. According to its CEO, Simon Khalaf in [23], Flurry has stored more than one trillion unique events performed within more than 250 000 apps. It has gained a very strong position in the app insight field; according to [35] around 6.7% of Android apps have the Flurry Analytics library included. This is very close to Google Analytics' share, and much of the functionality is similar.

When logged in as a developer, for each application listed, one can study the app use sessions in more detail. The total number of sessions, median session length, and average active users are some statistics that are available. All data are available for export in to comma-separated value ".csv" file format.

Session duration is a very useful metric, as it shows the distribution of sessions among users. For instance, a developer considering including in-app advertisements can calculate how many ads would be rotated into the display and thus estimate the potential ad revenue. In Figure 7-4, we see that most users stay within the app for 1-3 minutes. We could infer that users in the 0-3 seconds span could be chalked up in the bounce rate (involuntary app launches or crashes at start-up). If these session lengths would rise to, let us say, 10% of total sessions, and then we should create an alarm that sends an email alerting the developer to investigate the reason for abnormally many sessions shorter than three seconds.

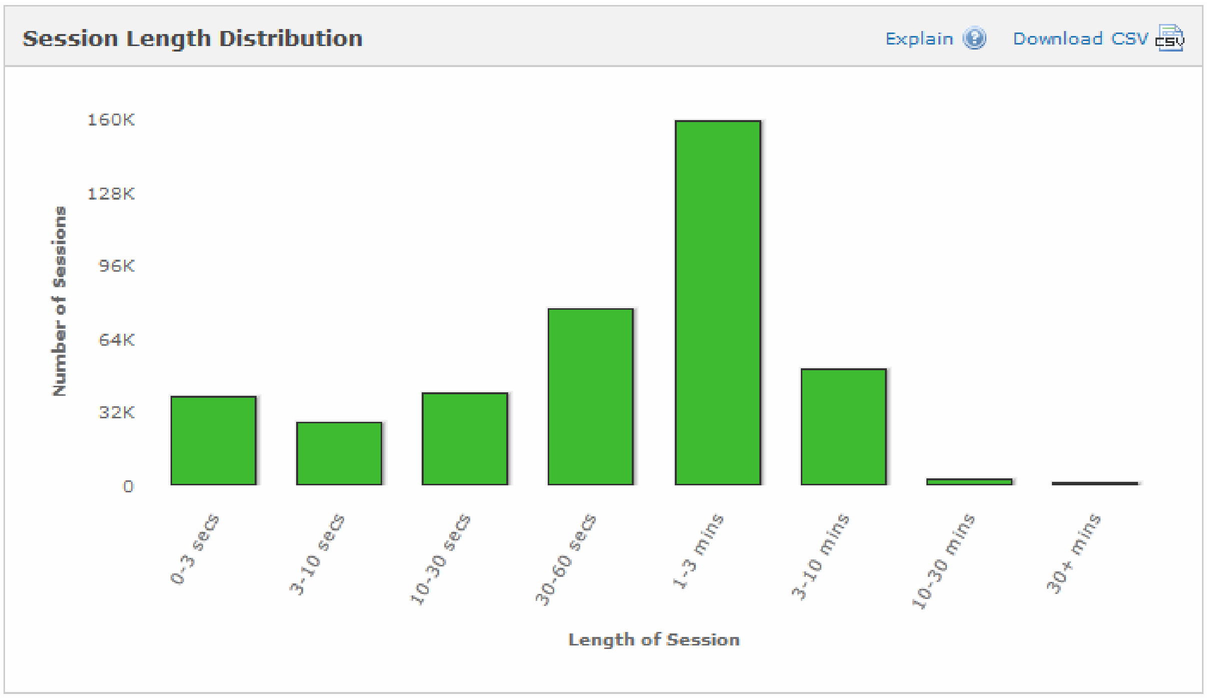
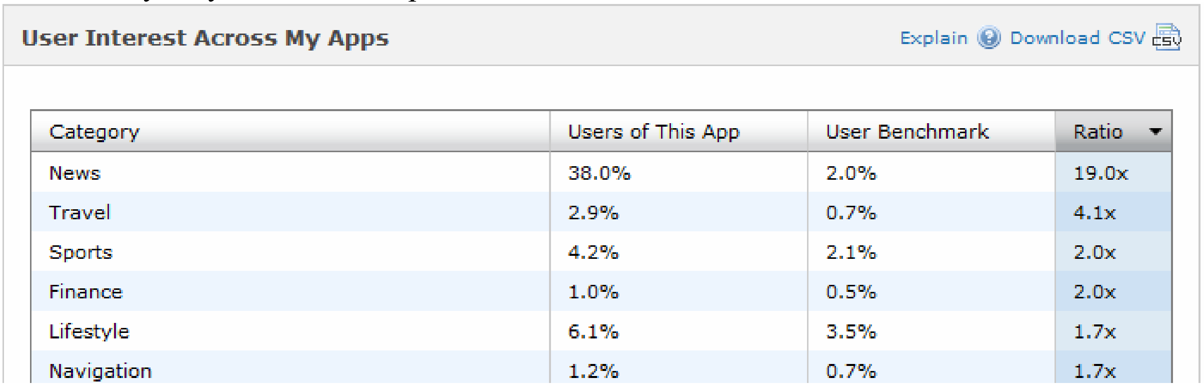


Figure 7-4 – Sample screenshot of a Flurry chart [63]

Metrics for frequency of use are also available: sessions per day, week, or month per user. There are also statistics of how many of the users spent only one session or only one day using the app (bounces) and how many users returned the day after installing the app. Retention rate (daily, weekly, or monthly) can also be displayed, telling us how many users remain active after the stated time period. Returning users can be plotted versus new users to further understand the app's user turnover.

Gauging user interest is an extremely powerful feature. Based on previously gathered data about a user's, Flurry can present their dominant app categories. In Figure 7-5, we see that a user of the current app is 19 times more interested in news than the average user. Such insight is obviously very valuable for optimization of advertisements.

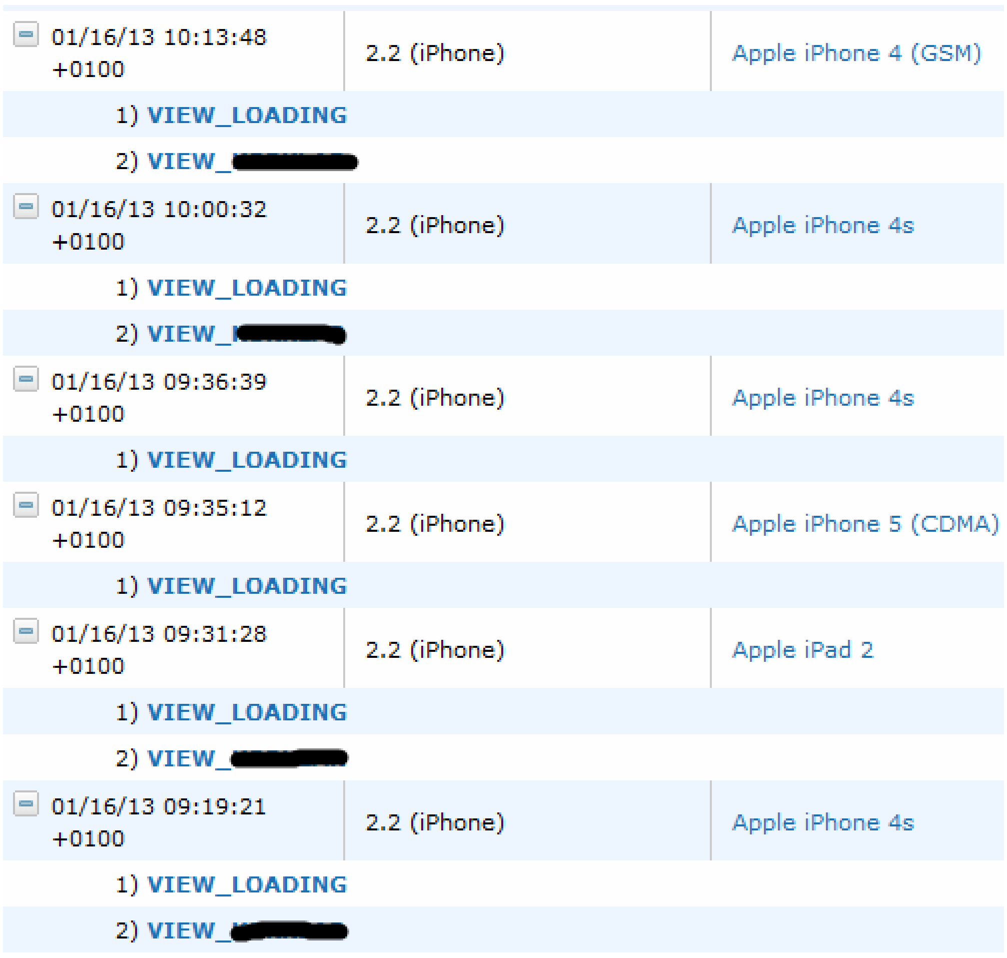


Category	Users of This App	User Benchmark	Ratio
News	38.0%	2.0%	19.0x
Travel	2.9%	0.7%	4.1x
Sports	4.2%	2.1%	2.0x
Finance	1.0%	0.5%	2.0x
Lifestyle	6.1%	3.5%	1.7x
Navigation	1.2%	0.7%	1.7x

Figure 7-5 – User interest in Flurry [63]

Flurry's feature set includes benchmarking against rival applications based on Flurry's estimates from similar data. It is also possible to manually enter statistics that are not collected within the app, such as age and gender (if you have some other means of knowing or estimating this data).

Finally, if the app has defined events (checkpoints), it is possible to track each and every event separately. This tells us which paths users take through the app, which might also be useful when trouble-shooting. In Figure 7-6, we see that no user within the last hour actually began to use the app for its intended purpose, but rather terminated the app at the loading screen or from the menu.

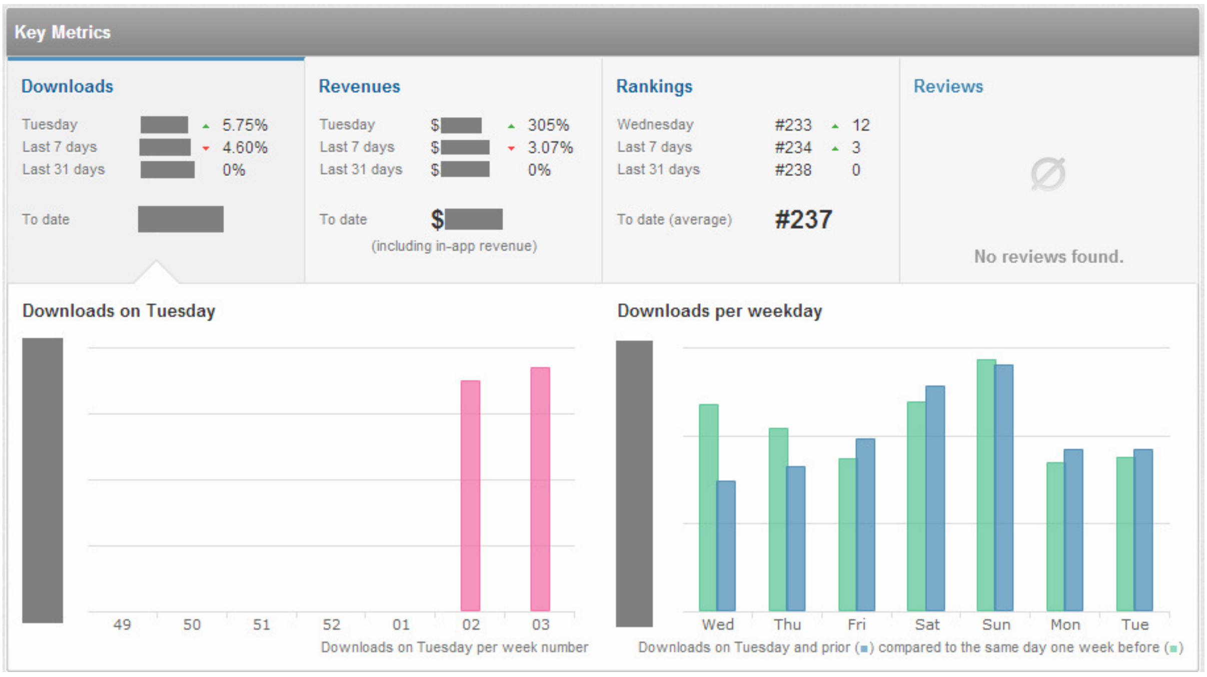


⊟ 01/16/13 10:13:48 +0100	2.2 (iPhone)	Apple iPhone 4 (GSM)
1) VIEW_LOADING		
2) VIEW_[redacted]		
⊟ 01/16/13 10:00:32 +0100	2.2 (iPhone)	Apple iPhone 4s
1) VIEW_LOADING		
2) VIEW_[redacted]		
⊟ 01/16/13 09:36:39 +0100	2.2 (iPhone)	Apple iPhone 4s
1) VIEW_LOADING		
⊟ 01/16/13 09:35:12 +0100	2.2 (iPhone)	Apple iPhone 5 (CDMA)
1) VIEW_LOADING		
⊟ 01/16/13 09:31:28 +0100	2.2 (iPhone)	Apple iPad 2
1) VIEW_LOADING		
2) VIEW_[redacted]		
⊟ 01/16/13 09:19:21 +0100	2.2 (iPhone)	Apple iPhone 4s
1) VIEW_LOADING		
2) VIEW_[redacted]		

Figure 7-6 – Event log in Flurry [63]

7.5 Distimo

A Dutch mobile-specialist analyst firm stated that their first mission was making the app market transparent. This company was born out of frustration over lack of insights [3]. In addition, to blogging and publishing yearly reports of its findings, they offer a (cross-OS) web platform called App Analytics that collects data regarding sales, rankings, and reviews of the developer's own apps. An optional premium tool called AppIQ allows developers to estimate the same figures for any app, not only their own. Figure 7-7 shows the standard dashboard of their platform.


Key Metrics

Downloads			Revenues			Rankings			Reviews
Tuesday	[redacted]	▲ 5.75%	Tuesday	$[redacted]	▲ 305%	Wednesday	#233	▲ 12	
Last 7 days	[redacted]	▼ 4.60%	Last 7 days	$[redacted]	▼ 3.07%	Last 7 days	#234	▲ 3	
Last 31 days	[redacted]	0%	Last 31 days	$[redacted]	0%	Last 31 days	#238	0	
To date	[redacted]		To date	$[redacted] (including in-app revenue)		To date (average)	#237		No reviews found.

Downloads on Tuesday
49
50
51
52
01
02
03
Downloads on Tuesday per week number
Downloads per weekday
Wed
Thu
Fri
Sat
Sun
Mon
Tue
Downloads on Tuesday and prior (■) compared to the same day one week before (■)

Figure 7-7 – Sample screenshot of Distimo Analytics

7.6 App Annie

App Annie is a web-based tool that can track statistics from Apple's iTunes Connect and Google Play accounts as a third-party agent acting on a developer's or publisher's behalf (assuming that you provide it with your login credentials). It may be setup to send a daily email report or you can export data manually to comma-separated value files (i.e., ".csv" file format).

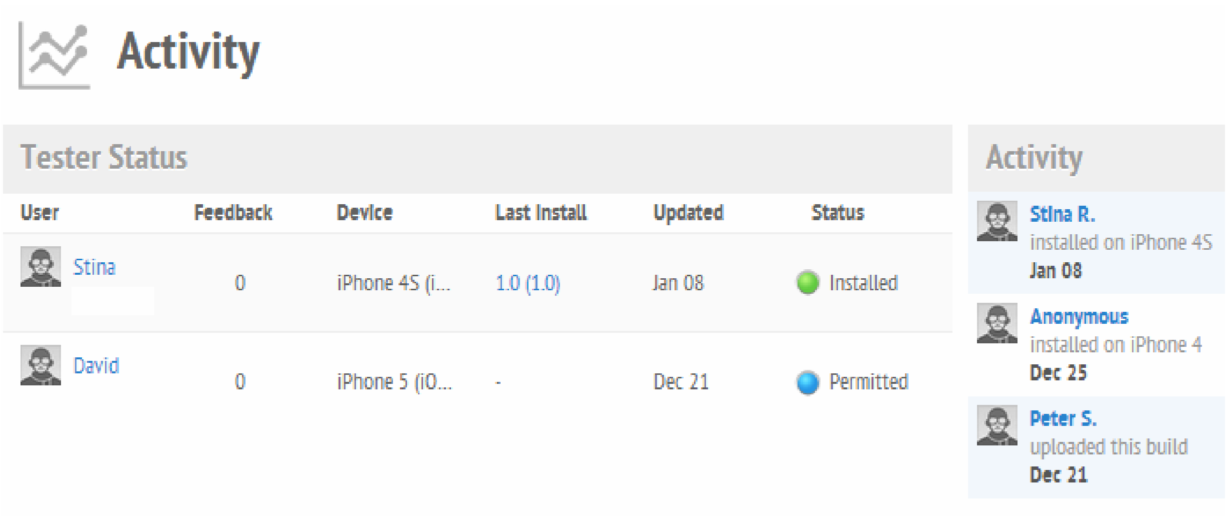
According to their own description, App Annie indexes 150 000+ apps and their daily downloads, revenues, rankings, and reviews. It has to date tracked over 11 billion downloads and US$1.5+ billion in publisher revenues [25]. It also offers a monthly report compiling interesting app statistics and trends.

At the time of this writing, the company has just launched an invitation-only beta test of an upcoming API. However, this was not available to me for use in my implementation.

7.7 TestFlight

While most other tools described earlier in this chapter are multi-platform, TestFlight only deals with iOS apps. It is primarily used for beta testing and processing crash log reports of iOS apps. One key function is the collaboration setup enabling iDevices, listed with their UDIDs, to install an app directly from a shared URL rather than having to go the official route via iTunes and ".ipa" files. This can be used to internally beta test or release an iOS application, without having to release it publicly on the App Store marketplace.

If the TestFlight SDK is included as a library into the app, crash logs, Over-the-air (OTA) updates, live session tracking (including funnels / conversion), and in-app questionnaires for beta testers are also available. Results are sent back to the TestFlight web platform after each app session finishes [62]. An example of the TestFlight user interface is shown in Figure 7-8.


Activity

Tester Status

User	Feedback	Device	Last Install	Updated	Status
Stina	0	iPhone 4S (i...	1.0 (1.0)	Jan 08	● Installed
David	0	iPhone 5 (iO...	-	Dec 21	● Permitted

Activity

Stina R.
installed on iPhone 4S
Jan 08

Anonymous
installed on iPhone 4
Dec 25

Peter S.
uploaded this build
Dec 21

Figure 7-8 – Sample screenshot of TestFlight

7.8 AppDailySales and Autoingestion

Since iTunes Connect only keeps track of a limited subset of all app sales data, developer Kirby Turner published an open source Python script that screen-scrapes (i.e., extracts relevant data from HTML elements) and can automate and schedule report fetching [46].

Since this script was published as source code rather than a web service, users did not have to provide a third party with their iTunes Connect credentials and could modify the code according to own needs. The script was updated and maintained by the author over multiple versions, but subsequently retired after Apple offered its Autoingestion Java class that more or less provides the same functionality [39]. However, although Autoingestion is well documented, it is not open-source and is only provided as an executable file. Therefore it is not customizable.

7.9 Google's Our Mobile Planet

Google's Our Mobile Planet is a web platform offering yearly statistics and charts of smartphone statistics and usage trends. This initiative was due to Google and its partners. It is built around surveys that 1,000 people from each of the 41 representative countries have answered. This survey contains questions about sex, age, marital status, education, employment, and living conditions (rural or city) [52]. The uniqueness of this website is that it provides all of the raw data as Excel spreadsheets for downloading, thus this is an interesting source for app creators who are interested in cross-referencing these data against their own gathered metrics. The site explicitly encourages site visitors to download the raw data and "to dig deeper".

8 Data mining and statistical analysis

This chapter discusses some basic mathematical theory regarding distributions, classification, and trend spotting in data sets. The relevance to app statistics is mainly to find out whether correlations hold between metrics such as ratings or rankings and the actual number of downloads or purchases. If so, how strong are these correlations and which metrics have the greatest impact? Which data entries are similar and how could they be classified? Can trends be discovered programmatically using data from a given database? This chapter will discuss formulae and methods to solve these problems.

8.1 Data mining phases and practices

Data mining is a subfield of computer science that utilizes computations to discover patterns and trends in large data sets [76]. In doing so, techniques from artificial intelligence, machine learning, statistics, and database systems may be used. For the purposes of this master's thesis, the latter two are the most applicable.

Data mining is also known as Knowledge Discovery in Databases (KDD) [77]. KDD is usually divided into three main phases: pre-processing, actual data mining, and results validation. The following sections describe them in this order.

8.1.1 Pre-processing – Extract, Transform, and Load (ETL)

Pre-processing data is the first step. This step consists of three tasks: extraction, transformation, and loading.

Data must first be either generated or obtained from their original sources. Techniques such as automatic web indexing and screen-scraping (text recognition or extraction from source code) may be used to do this. The AppDailySales Python script is one example of screen scraping. An initial validation of data is also done in this step, to reject data that is obviously faulty or insufficient.

The transformation step is necessary to convert the data into a uniform format. This includes possible translation between languages and character encodings, conforming to database schema constraints or query language, removing duplicates and null values, cropping data according to scope, sorting datasets, and many other related tasks.

After a successful data transformation, the cleaned dataset may be loaded into the database. The loading process may be a one-off import or a continuous process. In our case with app statistics, an initial manual import of historic data is planned, while future data will be automatically imported daily and be appended to the database.

8.1.2 Data mining – Task classes

Guidelines from the ACM SIGKDD Curriculum Committee [76] suggest that there are six common classes of data mining tasks:

1. Anomaly detection – identification of unusual data records which may indicate errors
2. Association rule learning – learning which data often appears together
3. Clustering – discovering underlying groups and structures
4. Classification – labelling data entries with tags or sorting them into categories
5. Regression – training a model to best-fit the data, enabling projection
6. Summarization – compression and visualization of the data set

I foresee that all of these tasks will be useful for the purposes of mobile app statistics. Some of these tasks are described in more detail in following sub-sections.

8.1.3 Results validation

Depending on which of the six data mining tasks are implemented, the validation of results may differ. However, a common method for validation is overfitting, which means applying the models and patterns that have been found to a new, not previously used dataset, and manually verifying the outcome. E-mail filtering is often used as an example, where models for detecting spam messages are derived from a training dataset, and subsequently evaluated on another dataset. The criteria for whether to accept a model's accuracy or not can be based on the percentage of false positives or false negatives. One technique for visualizing these results is a confusion matrix, described in section 8.6.4.2 starting on page 43.

8.2 Pearson correlation (r-value)

In determining whether two sets of variables are correlated (linearly dependant), the standard measure is the Pearson product-moment correlation coefficient which is usually just referred to as the r-value. The value ranges from -1 to +1 inclusive, where +1 is a perfect correlation and -1 is a perfect inverse correlation.

The r-value is calculated formally as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}$$

where X and Y are variable sets with mean averages of $\overline{X}$ and $\overline{Y}$.

For the purposes of this master's thesis, an application of this might be to investigate if the ranking of an app is correlated with its number of downloads. Presumably, if there is a relation between the two, then a lower rank (#1 being the best position), should be correlated with more downloads than the apps ranked below it. This is an example of an inverse correlation where we expect a negative r-value (as increasing rank should be correlated with decreasing numbers of downloads). Conversely, the correlation between the app's customer rating and its number of downloads should be positively correlated, as presumably higher ratings indicate that more people were convinced to download the app. My intention is to evaluate these expected relations.

8.3 Regression analysis and R^2

For a given set of measured data in a scatter plot (x-y coordinate pairs), the trend line or approximated function may be found through regression analysis. Normally this is done in the form of a minimized least-square sum for linear models.

A concrete measure of the model's goodness of fit is the R^2 value, which lays within the interval $0 \leq R^2 \leq 1$. The R^2 value is in fact the square of the r-value described above, as the letter implies. The closer to 1, the better the model fits the data set. Figure 8-1 shows a weak fit to the data used in the section 8.3.3.

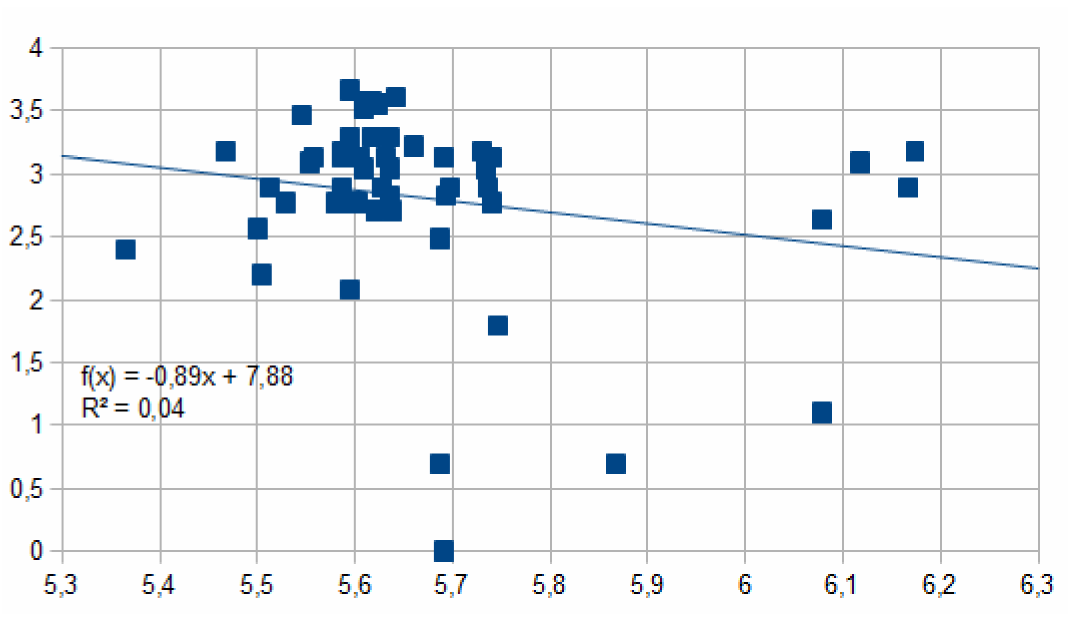


Figure 8-1 – Example of a poorly fitted linear regression in OpenOffice Calc

Regression analysis with a good fit may be useful to extrapolate the function in order to project values outside of the current data set. If projections are made based on the model, then good scientific practice stipulates that the uncertainty (margin of error) should be stated and that this margin of error should be within the stated confidence interval.

8.4 Pareto principle and distribution

The now-famous rule of thumb that 80% of the effect comes from 20% of the contributing factors is known as the 80-20 rule, or the law of the vital few or the Pareto principle after its author, the Italian economist Vilfredo Pareto who noticed an unequal wealth distribution in his country [29]. The same goes for purchases: as a general rule of thumb, 20% of the products account for 80% of the sales, while the 'long-tail' of obscure products only contributes a total of 20% of revenue.

This same rule-of-thumb led to a log-linear data distribution which also bears his name. Obviously, the ratio does not have to be exactly 80-20 since there are both a scale and a shape parameter. These parameters can be found by plotting log(x) and log(y) to find a reasonably straight line. Regressing this line to the y-intercept and finding the slope enables us to approximate either x or y [64].

The Pareto cumulative distribution function (CDF) is defined as:

$$1-\left(\frac{x_m}{x}\right)^{\alpha} \quad where\ x \geq x_m$$

This produces a typical Pareto curve. For the 80-20 rule to apply literally, the shape parameter (α) should be approximately 1.161.

8.4.1 Experiment by Brynjolfsson, Smith, and Hu

Why is the Pareto distribution interesting for mobile apps and their statistics? Amazon.com, a major online store for primarily books, does not display the actual number of sales for their books. However, they display relative ranking, just as the app marketplaces do. In [30], Brynjolfsson, Smith, and Hu performed several clever experiments and concluded that a Pareto distribution can satisfactory estimate the number of purchases based on ranking data. Their estimation formula is:

$$\log(\text{Quantity}) = \beta_1 + \beta_2 * \log(\text{Rank}) + \varepsilon$$ [30]

which is in the form of the generic linear parameter:

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_{i,j} + \varepsilon_i$$

They found the values for Amazon.com to be $\beta_1 = 10.526$, $\beta_2 = -0.871$, and $R^2 = 0.8008$. Note that β_2 is negative, as a lower ranking indicates a better position and presumably more purchases. By finding the intersection of the above equation, the model enables us to infer that, for instance, a book ranked 10th on Amazon.com has about 5,000 sales per week.

8.4.2 Experiment by Garg and Telang

Inspired by Brynjolfsson, Smith, and Hu's results, Garg and Telang [15] conducted a similar experiment but applied it to the mobile app market. However, they used a different method, not requiring actual demand data as used by Brynjolfsson, Smith, and Hu. Instead they collected public marketplace data from Apple's App Store for rankings and price over a period of time and inferred the Pareto distribution models from their calculations. The end result was not so far off, with a shape parameter of 0.86 for iPhone apps. The authors proceeded to calculate the relative strength of ranking positions. The formula for doing so used the estimated shape parameter α, also called β_2:

$$\text{downloads}_1/\text{downloads}_2 = (\text{ranking}_1/\text{ranking}_2)^{-0.86}$$ [15]

Using this estimate, the top-ranked application would attract almost twice as many downloads as the second [$(1/2)^{-0.86} = 1.82$].

8.4.3 Attempt to replicate on my own dataset

I set out to replicate the experiment, to see if there was correlation to be found in my own app data. Fortunately, I had access to all data for a specific app published on Google Play. This app was not in the overall marketplace's top-500 list, but in the subcategory top-500 Tools which might make the implications weaker. For a period of about a month, the app's ranking data was collected from App Annie and actual download figures were collected directly using the Google Play Developer Console. These data were entered into an OpenOffice Calc spreadsheet and plotted just as in Brynjolfsson, Smith, and Hu's experiment.

I took an intermediate step not considered in the reference experiment and calculated the correlation between rank and downloads. This function was available in OpenOffice as correl(). The correlation value was found to be -0.18, indicating a relation. Again, the negative sign is because the proportion is inverse; as lower rank is better – since it implies more downloads.

My calculations found different values for the parameter estimates ($\beta_1 = 7.88$, $\beta_2 = -0.8936$, and $R^2 = 0.04$), the function generated plausible results, as shown in Figure 8-2, even despite the fact that the R^2 value is very weak. The β_2 value is also reasonably in line with what others have found (see for example [30]) and to the iPhone-specific value of -0.86 found by [15].

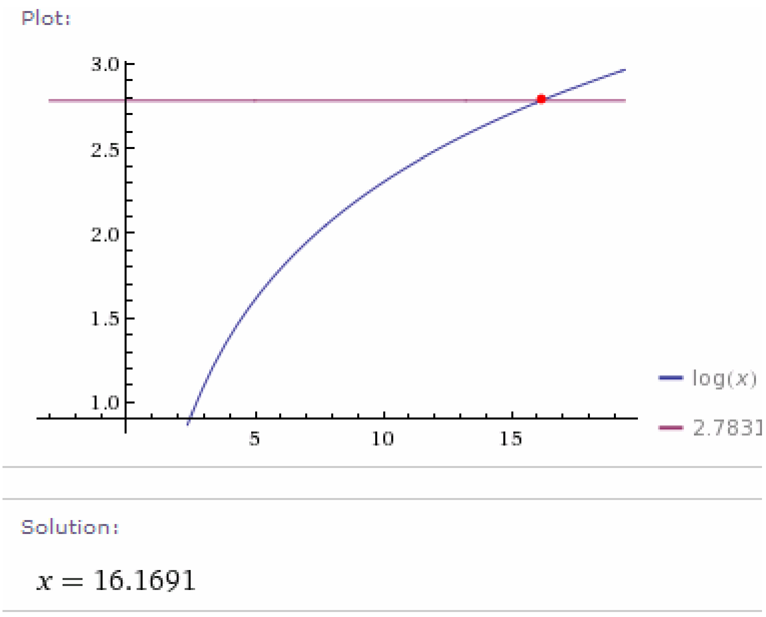


Figure 8-2 – Estimate of daily downloads at rank 300

However, as discussed in [15], models based on calibration at low baseline sales might diverge for top sellers, since the *"distribution of app demand is top-heavy (even within the 200 top apps)"*. The top-heavy effect was also found by Valadares who writes *"The results also show that the rank-dependent values decline steeply with rank for the top 10 products. The effect of product rank on demand remains economically significant for the apps in the top 50, but for apps ranked 50 or higher this effect is negligible"* [65].

Using my equation to estimate how many daily downloads the number-one ranked app in the category receives, the answer was roughly 2,600 – far too few to be accurate. A proposed reason for this could be (following the authors of [15] in their method) that their model only fits within the range $1 \leq x \leq 200$ since they use data from that interval, however considering my dataset the interval for my equation only holds for $200 \leq x \leq 500$.

During the implementation phase I attempted to apply this analysis to larger data sets and for more than just one app. As pointed out by my academic thesis supervisor, however, it is not certain that this model can accurately predict downloads in even the later range since the long-tail distribution may also be present in this case. This particular calculation has not been performed, but a distributional chart supporting the long-tailed distribution is to be found in figure 11-8.

8.5 Principal Component Analysis

Principal component analysis (PCA) is a technique that can reduce the dimensions of a dataset. For instance, if for all apps for which we have data on ranking, price, rating, number of reviews, percentage of active users, generated revenue, average session length, and so forth, we might want to determine which of these factors causes the most variance in the number of downloads. That is to say: Which factor is most important for the end result of predicting the number of downloads?

This might be very unclear from just looking at the multidimensional raw data, even though the underlying pattern may be simple and mainly caused by a few factors. PCA performs a linear orthogonal transformation of the data set, via the intermediate step of the co-variance matrix [70]. This transform is defined as follows:

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\ \mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)] & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix}.$$

PCA can now reduce the dimensions of the dataset by finding a linear transformation of the above matrix so that the vectors become orthogonal [70]. This means that the vectors are **not** co-dependent. This procedure is called a diagonalization of the matrix. A subset of the eigenvectors that are found is used as axes for the transformed data. We may even select the size of this subset to contain a certain amount of the total data. Effectively, the first component now represents the biggest factor for the variance of the data, and the others follow in descending order.

PCA is popular partly because it is parameter-less (besides the dataset), and therefore it is very simple to apply in a black-box manner. It does not require in-depth understanding of the algorithms behind it nor does it require *a priori* knowledge about what the most important components are. A common application for PCA is compression of images because it keeps the most important data, while filtering out data that has little effect on the overall result [70].

Finding out which quantitative indicators contribute most to an app's success (for free apps this is the number of downloads, while for paid apps this is the revenue from the app) should be of interest to the app publishing company at which this thesis project is being undertaken. I will perform PCA on the data sets gathered to find out what these indicators are for the app(s) that will be considered.

There are useful functionalities in mathematical software such as MATLAB, including the functions `CovMat` for the covariance matrix and `eigs` for picking out the eigenvectors [71]. These will no doubt be helpful in the analysis process.

8.6 Clustering

While PCA filters out the most important components of data, clustering is another technique for prioritizing and simplifying data. Clustering attempts to group similar data together into clusters. While normal clustering algorithms are discrete (i.e. a data entry either belongs to a cluster or it does not), fuzzy clustering allows for nuances via partial belonging – membership degree ranges from 0 to 1 [72].

A useful example of (discrete) clustering adopted from [73], regarding vehicles is shown in Table 5. We can cluster these in a scatter plot where the x-axis is top speed and y-axis weight. As can be seen in Figure 8-3, this results in three distinct vehicle classes: Lorries (heavy and slow), Medium market cars (medium weight, medium top speed), and Sports cars (light and fast). Note that not all of the variables are used, for example, color and air resistance are not in the plot. When clustering we should focus on selected aspects. Therefore a PCA step before the clustering might be useful to help focus on the important properties.

Table 5 – Sample data suitable for clustering

Vehicle	Top speed (km/h)	Colour	Air resistance	Weight (kg)
V1	220	Red	0.30	1300
V2	230	Black	0.32	1400
V3	260	Red	0.29	1500
V4	140	Gray	0.35	800
V5	155	Blue	0.33	950
V6	130	White	0.40	600
V7	100	Black	0.50	3000
V8	105	Red	0.60	2500
V9	110	Gray	0.55	3500

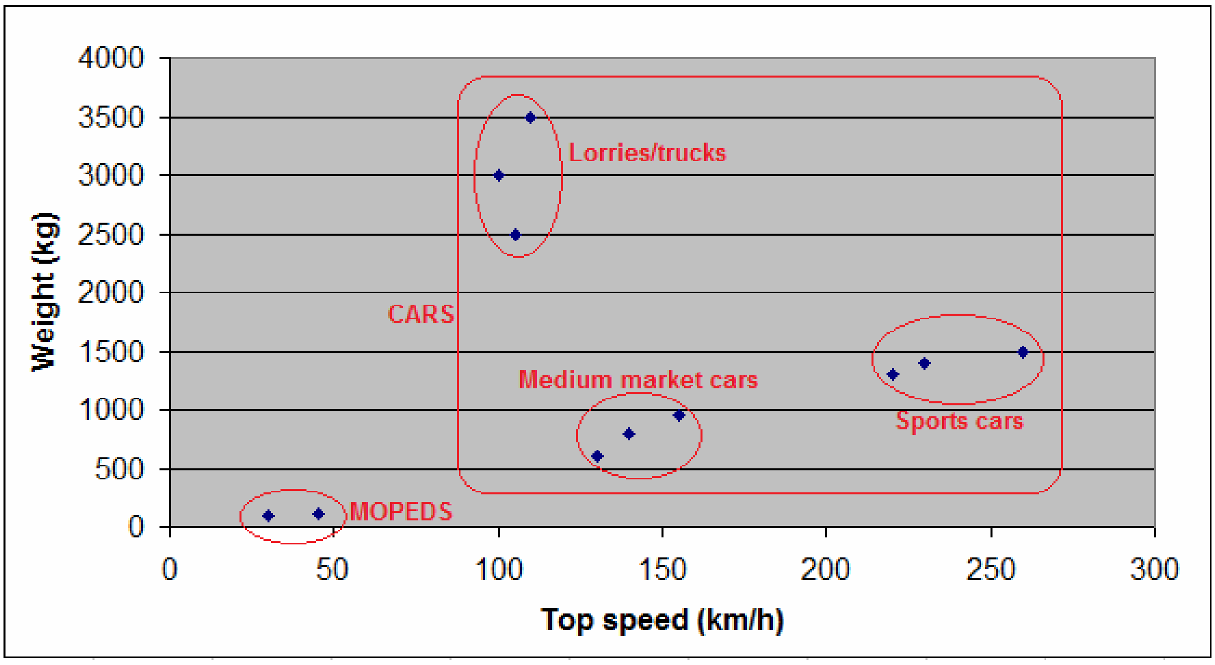


Figure 8-3 – Clustering of similar data entries

8.6.1 Fuzzy clustering

In the above scenario discrete clustering is likely to be adequate, but should we add a car with top speed of about 180 km/h and weight around 2,000 kg, it would end up in the middle of all our defined classes. In such a case it could be useful to say that it partially belongs to each class, rather than excluding it from all classes. The degree to which it could be considered that the entry belongs to the respective cluster can be viewed as the "likelihood" of it being in that class [74].

There are also situations of nested clusters. Consider what would happen if we added some mopeds into the vehicle data set. Their weight of around 100 kg and top speeds of 45 km/h would make them ideal for a class and cluster of their own. After this change it would be useful to group clusters together by vehicle type. This establishes that a data entry for the purposes of clustering may inherit classes; a vehicle may be an instance of a sports car, which in turn is an instance of a car.

8.6.2 Dendrograms and cophenetic correlation

The above classifications are useful in the computation of clustering. Using the same vehicle example as before, we plot the clusters as classes. This is normally done in a dendrogram – as dendro means tree in Greek. The dendrogram displays inheritance and relations hierarchically in a tree structure. The vehicle example is modelled in Figure 8-4.

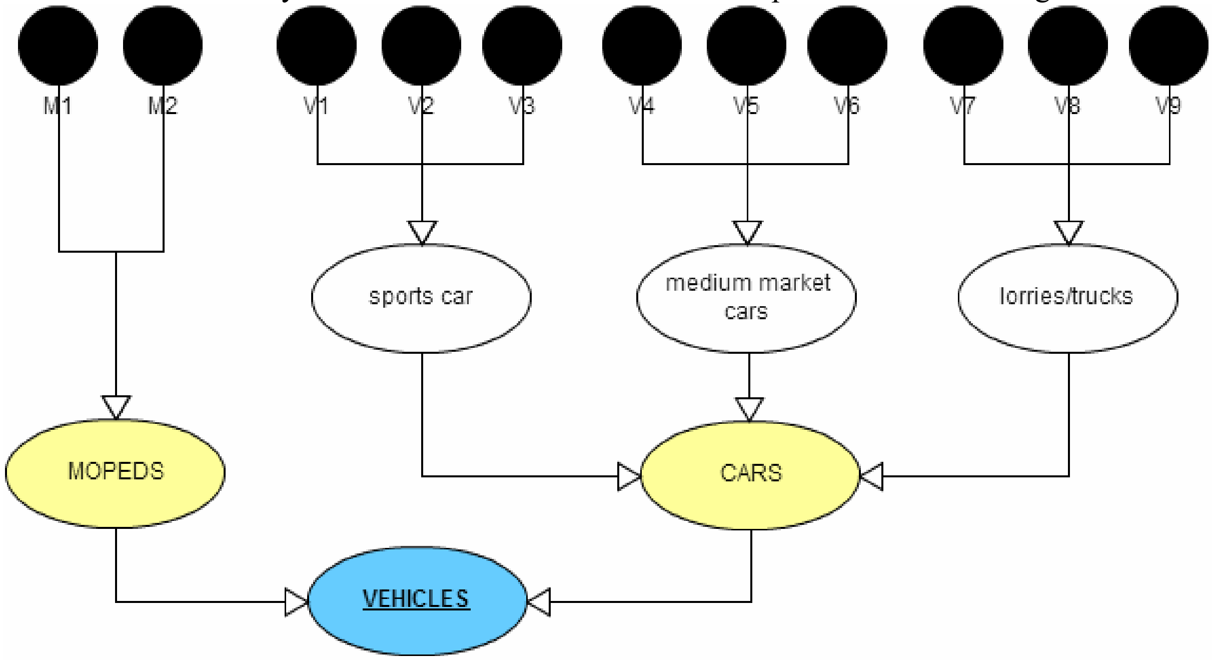


Figure 8-4 – Sample dendrogram for classes of vehicles

There are measurements of how well a dendrogram $\{T_i\}$ represents the distances of the raw data entries $\{X_i\}$. One such metric is cophenetic correlation coefficient [74] which is calculated based on a comparison between dendrogrammatic distance between tree nodes T_i and T_j, and the Euclidean distances between observations $X_i - X_j$. The correlation efficient c is then given by:

$$c = \frac{\sum_{i<j}(x(i,j) - x)(t(i,j) - t)}{\sqrt{\left[\sum_{i<j}(x(i,j)-x)^2\right]\left[\sum_{i<j}(t(i,j)-t)^2\right]}} \quad [75]$$

where x is the mean average of x(i,j) and t the mean average of t(i,j).

8.6.3 k-means clustering

Mathematically computing clusters can be done in a variety of ways. One famous and intuitively natural algorithm is k-means clustering. This algorithm belongs to the centroid-based clustering algorithm family.

The idea is to divide n observations into k clusters using an iterative process, so that the Within-Cluster Sum of Squares (WCSS) is minimized. For this reason, k-means clustering can be seen as an optimization problem. A generalization of [73] yields the steps:

1. Place k cluster centers at random.
2. Assign each data point to the nearest cluster centre.
3. Compute the new center of each class (the centroid).
4. Move the cluster centers accordingly.
5. Iterate until no “visible change” happens after the center is moved.

After this algorithm terminates every data point n_i is assigned to the closest of the k clusters. A membership matrix M (of dimension n*k) now holds Boolean values for whether or not each observation n_i belongs to cluster k_j.

Fuzzy clustering using the fuzzy c-means algorithm works very much the same, but the membership matrix M instead contains floating point values for each observation n_i indicating its likeliness of belonging to each cluster k_j [73].

8.6.4 Analysing clusters – Jaccard index and Confusion matrix

After observations are assigned to clusters it might be interesting to compare clusters against other clusters, especially if our constraints allow observations to belong to multiple clusters. Two methods for doing this are described in the paragraphs below.

8.6.4.1 Jaccard index

How similar are the clusters? One very simple metric that answers this question is the Jaccard Index, which varies from 0 to 1; where 0 means no observations are common among the two clusters and 1 means that the clusters contain exactly the same observations. The function takes parameters A and B - the two clusters (sets of observations). The Jaccard Index, or coefficient, is then calculated as the number of total unique observations in A and B divided by the total number of elements. This quotient indicates how disjunctive the clusters are.

The Jaccard Index formula is:

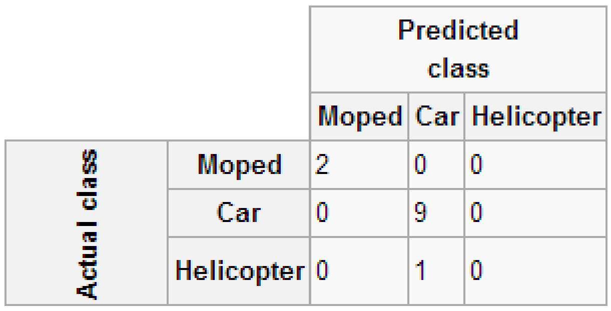
$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}$$

8.6.4.2 Confusion matrix

Another method for visualizing the results of a classification or clustering algorithm is to create a confusion matrix. This matrix is useful for identifying cases where observations may have been incorrectly labeled and therefore classified into suboptimal clusters.

For the vehicle example, a likely confusion might occur if we add a helicopter to the vehicle set. It weighs about 1,000 kg and has a top speed of around 250 km/h. Our clustering algorithm would therefore be likely to cluster helicopters with cars (in the sports cars sub-cluster). Visualizing this in a confusion matrix as shown in Table 6, it is easy to spot errors as all correct identifications are aligned along the diagonal. We immediately discover that the helicopter was mislabeled.

Table 6 – Sample confusion matrix



		Predicted class		
		Moped	Car	Helicopter
Actual class	Moped	2	0	0
	Car	0	9	0
	Helicopter	0	1	0

Incorrectly labeled data or data that is erroneous in itself is sometimes called statistical noise, which may weaken the accuracy of adaptive models. It is therefore desirable to remove such anomalies as early as possible in the analysis, or to devise a scheme to handle them separately.

8.7 Software for data mining

Programmatically performing data mining is facilitated by many software products. There are modules for query languages and databases, plugins for spreadsheet programs (such as Microsoft Excel and OpenOffice Calc), packages in mathematical software (such as MATLAB and R [74, 75], and standalone programs (such as ELKI [79], KNIME [80], and RapidMiner [81]).

Each type of tool has its advantages. Database export might be easiest to spreadsheet programs, MATLAB and R are perhaps the most well-documented tools, and standalone programs come pre-packaged with modules for most the clustering and PCA techniques described above. For the practical purposes of this master's thesis project, I will attempt to use MATLAB and possibly R for manual operations and the standalone programs experimentally due to their complexity.

9 Methodology

There are so many statistics available from publicly available reports that one might fit together these puzzle pieces and form a bigger picture of the app market situation. For instance, take the fact that Android has been criticized for a weak game selection. This is reflected in a lower overall rating for Android games, compared to games on iOS (with scores of 3.2 versus 3.7 respectively out of 5). Ratings and reviews are the only metrics available for a potential app buyer when browsing. We also know that games account for 1/3 of the total marketplace revenue and that in-app purchases are most used in the game genre [20]. Add to this the fact that the majority of app revenue today comes from in-app purchases (Figure 5-1). Summing up all these facts and figures, the almost 4:1 revenue advantage of iOS over Android (shown in Figure 6-1) is suddenly not so surprising.

From an industrial business perspective, the changing trends in app business models are of great interest. We have seen that the traditional one-off purchase payment model is losing ground to models that require user interaction: ad-sponsored apps, subscriptions, and in-app purchases. What this means in terms of app statistics, is that the traditionally dominant metric "number of downloads" will be of decreasing importance, while "user retention/loyalty" will likely take on a more prominent role.

While the above reasoning represents only some examples of what could be done given suitable statistics and analysis of them, it is exactly these types of patterns that would be useful to track using internally stored data from a database. Ideally these trends should be found programmatically without requiring much human interaction by using statistical analysis. This is where this thesis expects to contribute, along with an implementation that will be evaluated.

9.1 Summary

From the statistics in the earlier chapters we have learned a lot about the smartphone market, in terms of platform specifics and differences, user demographics, payment models, and trends. Mobile app statistics have been discussed, not only which metrics are of importance and why, but also which first- and third-party app data tools are currently on the market and what data they collect. Some theories about statistics and data mining have also been discussed. Having studied all this material, the goals for the master's thesis project may be specified (as in section 9.2).

9.2 Goals of Master Thesis

During the literature study phase it became increasingly clear that suitable sets of marketplace data from Apple App Store and Google Play will not be accessible for importing into a local database as most of the relevant data is **not** publicly disclosed. This is why the statistical phase of this thesis project will be done through a case study, using the data sets from apps published by The Mobile Life [129], the company at which I am conducting this thesis project. Basic statistical analysis will be done on the available data sets. There will also be a recommendations section. This section will review whether a developer or publisher could utilize some combination of the publicly available tools and private data related to their application. This is done in order to derive satisfactory app statistics. The best suitable tools for this purpose will also be reviewed in this section. Most importantly, an in-house app statistics platform will be developed, implemented, and evaluated.

9.2.1 Limitations

The literature study phase of this thesis project focused mostly on the Android and iOS platforms, as does the final master's thesis – both the implementation of the tool and the data analysis. The statistical techniques described in the previous sections should only be viewed as a preliminary set of suitable tools, some of which will be used in this very report and some that are recommended for future use cases.

9.2.2 Primary goals

There are three clearly defined main goals for the app statistics implementation: storing data locally, presenting it suitably, and making it accessible.

1. First and foremost, available app data from third-party tools should, whenever possible, be transformed into a uniform format and then imported into a local database. This corresponds to the ETL phase. Additionally, manual entries must be possible since some customers of the app publisher may not want to disclose all of the data that they might want to include in their copy of the database. Hence it should be possible to add some data manually, possibly at irregular intervals. A few benefits of local data are direct access to data for SQL queries and other statistical tools, local backup/redundancy and one single data source which facilitates easier report writing. The relatively small data sizes that will be imported should not require much resources and could be deployed onto an existing server, mainly dedicated to another purpose.
2. This database should be accompanied by a front-end offering some visual representation of this data, at the very least displaying in tables; but preferably also as diagrams or charts.
3. This front-end should be made available to external viewers so that their visualization could be done on a mobile device. This was an especially desired feature for the company at which this thesis project was undertaken, but which technique to use remains open, i.e., it could be a native app or mobile web site.

9.2.3 Secondary goals

The functions listed in this section are things that are **not** required, but would be considered nice additions to the system.

1. The front-end should not only connect directly to the database, but should also offer an API with user access control management, so that third-parties may be given access the data of interest to them, without having full system access.
2. Some type of primitive SMM tool for mapping app reviews into category buckets using keyword lists would be useful, if review texts are available.
3. Alarm or trigger system. For instance, if the overall rating for App A drops below 2 out of 5, then automatically send an email to the relevant person (or persons) to encourage investigation.
4. Automatic or semi-automatic monthly reports could be generated based on the trends that are found. Such reports are already sent to many customers of the app company, and currently require all data to be manually compiled and written.

9.3 Implementation outline

This section discusses a rough estimate of which techniques and data sets the implementation phase utilized for its design and implementation. Finally a blueprint of the system architecture is provided.

9.3.1 Data collection

The company at which this thesis project is undertaken is an app publisher, which enables me to use actual live data and statistics from their own published applications. I will use this data, but to preserve confidentially I will anonymize the actual results and findings. The majority of the published apps are free to download, so my statistics will not focus on price and revenue metrics. A few of the company's apps have experimentally included either Google Analytics SDK or Flurry SDK, although these are used in a minority of the apps that will be studied. However, the statistics from these systems may be compared to the results that I find using the in-house data.

Whenever possible, scripts for importing data from marketplaces and third-party APIs should be scheduled to execute every n days, so that data is always retrieved and stored safely into the in-house local database. Such scheduling can be done on a Unix-based server via the crontab command [82] or in Windows through the scheduling utility.

Data sets can be imported from following sources:

iOS App Annie will be used rather than directly connecting to iTunes Connect, since App Annie has stored data ranging further back than the 13 weeks available on iTunes Connect. Therefore the initial import of historic data should be done through App Annie export files.

The Apple Autoingest Java class or the still-functioning AppDailySales Python script are probably useful for fetching current stats, since one of the goals of the implementation is to have as little third-party dependency as possible.

Android Google Play Developer Console API is surprisingly limited, and currently only provides subscription statistics. Actual app data is not available over the API [67]. There are some reverse-engineered scripts, but most were developed for the previous version of the marketplace [68], when it was called Android Market. Furthermore, their screen-scraping nature also makes them unreliable, as any small change to the marketplace layout could break their functionality.

Even though the Developer Console lacks a good API, it offers manual data export in the form of multiple ".csv" files available for download, and it is possible to request a ".zip" file containing all of these files. The file names shown in Figure 9-1 are mostly self-explanatory.

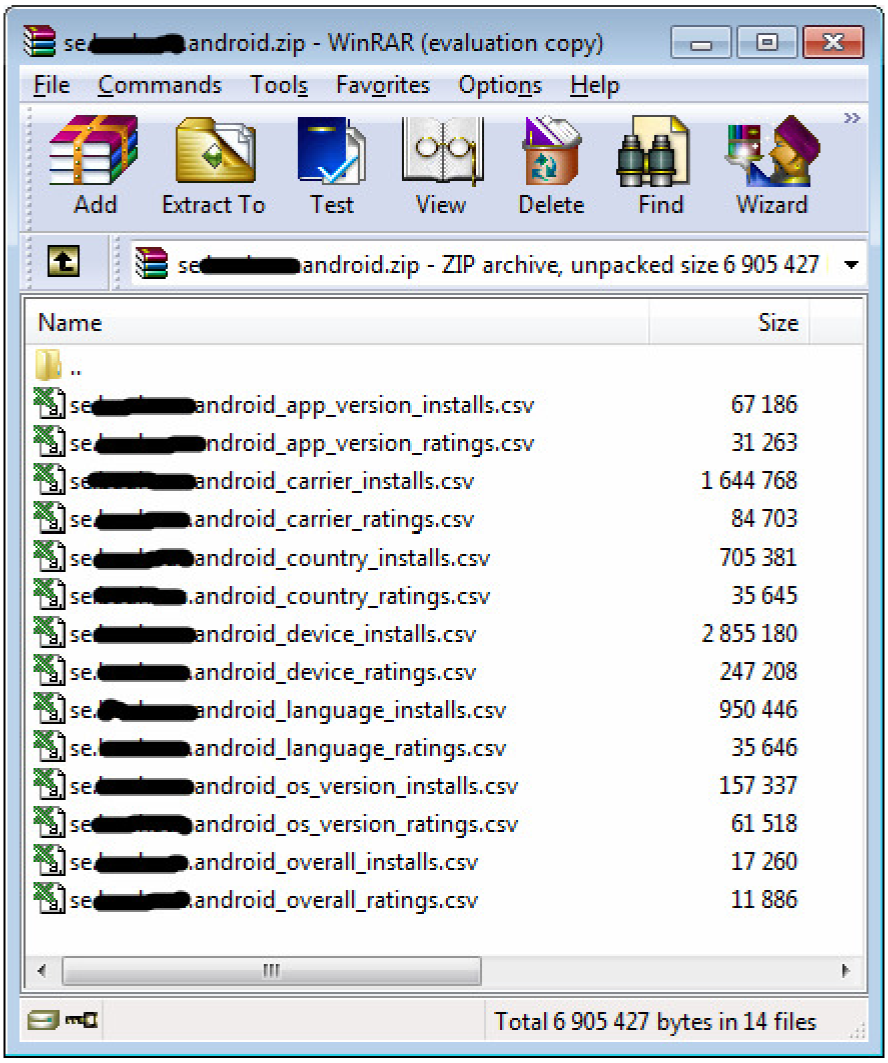
The solution to this problem is to either write a script that logs into the Developer Console, downloads the ".csv" export file and imports it to the database, or to rely on a trustworthy third-party data source such as Distimo's API. A more robust approach for the future might be to include the Google Analytics library into each app (both Android and iOS). This would be useful for multiple reasons, including access to more mature API.

Cross-platform

- Distimo's API [83] is very well-documented and is a popular third-party source for app statistics. It appears after some initial testing to be suitable for integration into the final local implementation.
- Google Analytics session data might be explored experimentally, since only a small fraction of the company's apps have included the necessary library as of yet.
- Flurry offers an API over XML or JSON that I may also attempt to utilize for session data.

Generic / OurMobilePlanet Excel spreadsheets may be of interest for cross-referencing

cross-reference material	demographics and user behaviour.



se[redacted]android.zip - WinRAR (evaluation copy)

se[redacted]android.zip - ZIP archive, unpacked size 6 905 427

Name	Size
..	
se[redacted]android_app_version_installs.csv	67 186
se[redacted]ndroid_app_version_ratings.csv	31 263
se[redacted]android_carrier_installs.csv	1 644 768
se[redacted]android_carrier_ratings.csv	84 703
se[redacted]android_country_installs.csv	705 381
se[redacted].android_country_ratings.csv	35 645
se[redacted]android_device_installs.csv	2 855 180
se.[redacted].android_device_ratings.csv	247 208
se.[redacted]android_language_installs.csv	950 446
se.[redacted].android_language_ratings.csv	35 646
se[redacted]android_os_version_installs.csv	157 337
se[redacted]android_os_version_ratings.csv	61 518
se[redacted].android_overall_installs.csv	17 260
se[redacted].android_overall_ratings.csv	11 886

Total 6 905 427 bytes in 14 files

Figure 9-1 – The archive file containing Android .csv export data

9.3.2 Back-end – database structure

As the company has previously used MySQL databases for their back-ends, I have also used a MySQL database for this project. The server will likely be hosted in the cloud, but during development I will run a local MySQL server on my laptop. I will utilize Windows, Apache, MySQL, and PHP (the WAMP stack) during development, as it is well-documented and I have used it before. Many useful plug-ins and modules are also prepackaged with it.

For compatibility reasons, at least the iOS statistics collected will follow the same database architecture as the official app data from iTunes Connect. These were documented by Apple in [39]. This structure was adapted in Table 7.

Table 7 – iOS report database structure

Report Field	Report Data Type	Notes
Provider	CHAR(5)	Typically Apple
Provider Country	CHAR(2)	Typically US
SKU	VARCHAR(100)	Product identifier
Developer	VARCHAR(4000)	
Title	VARCHAR(600)	
Version	VARCHAR(100)	
Product Type Identifier	VARCHAR(20)	Initial download / update, etc.
Units	DECIMAL(18,2)	Aggregated number of units
Developer Proceeds (per item)	DECIMAL(18,2)	
Begin Date	Date	Start date of report
End Date	Date	End date of report
Customer Currency	CHAR(3)	Three-character ISO code
Country Code	CHAR(2)	Two-character ISO code
Currency of Proceeds	CHAR(3)	
Apple Identifier	DECIMAL(18,0)	Apple ID for the app
Customer Price	DECIMAL(18,2)	Price displayed in App Store
Promo Code	VARCHAR(10)	Optional promotion code
Parent Identifier	VARCHAR(100)	For in-app purchases
Subscription	VARCHAR(10)	Renewal or new
Period	VARCHAR(30)	Duration of subscription

For Android, the most interesting of the .csv files in Figure 9-1 on page 48 would probably be overall_ratings and overall_installs. The formats for these files are:

overall_installs.csv	`date,daily_device_installs,active_device_installs, daily_user_installs,total_user_installs,active_user_installs, daily_device_uninstalls,daily_user_uninstalls, daily_device_upgrades`
	
	`Example:`
	`20130112,30,2898,25,8685,2782,18,18,0`

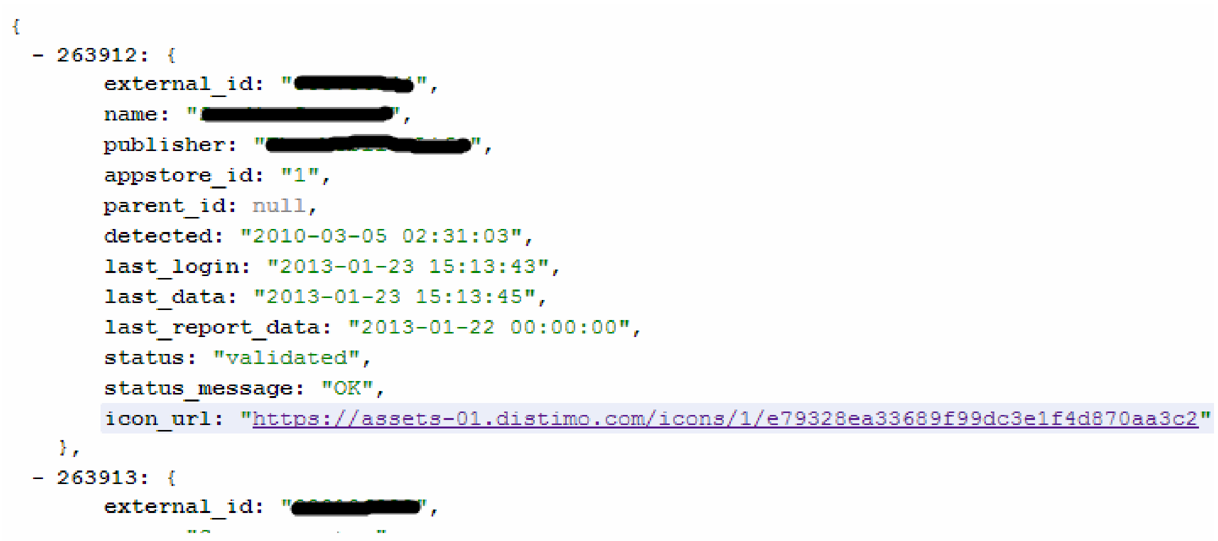
overall_ratings.csv	`date,daily_avg_rating,total_avg_rating`
	
	`Example:`
	`20130112,0.0,3.4705882352941178`

The database will be set up with specific tables dedicated to this data and will utilize a very similar table layout.

In addition to the platform specific data, whenever possible category and marketplace ranking will also be collected, as well as app reviews.

9.3.3 Back-end – API specification

As third parties must be able to access at least the most vital app statistics via a public interface, the back-end will host an API for this purpose. It will likely have a RESTful API written in PHP that accepts queries in the form of a Uniform Resource Identifier (URI) and responds with generated JSON or XML structured data. An example of a JSON response from Distimo's API is shown in Figure 9-2. This sample might be useful for inspiration.



```
{
  - 263912: {
        external_id: "[redacted]",
        name: "[redacted]",
        publisher: "[redacted]",
        appstore_id: "1",
        parent_id: null,
        detected: "2010-03-05 02:31:03",
        last_login: "2013-01-23 15:13:43",
        last_data: "2013-01-23 15:13:45",
        last_report_data: "2013-01-22 00:00:00",
        status: "validated",
        status_message: "OK",
        icon_url: "https://assets-01.distimo.com/icons/1/e79328ea33689f99dc3e1f4d870aa3c2"
    },
  - 263913: {
        external_id: "[redacted]",
```

Figure 9-2 – Sample JSON response output

The URI format is well-documented in IETF's RFC 3986 [84] and fits well with the purposes described in that RFC: *"a source of information with a consistent purpose (e.g., "today's weather report for Los Angeles")"*. The app statistics equivalent would then be "today's downloads for App A".

The final format of the URI-scheme used will be described in section 10.3.7, but parameters to be included are to include an identifier for which app or publisher data is being requested and some type of password, token, or nonce for determining whether the request shall be granted. As all accounts will still require manual activation from the app company, this security scheme shall suffice, but a future approach for authentication and authorization might be securing the connection using TLS or SSL.

The easiest approach to a format would be sending parameters as URL variables, as in the following very preliminary draft of an API format:

http://server/appstats/api.php?appID=[appID]&statType=[statType]&period=[period]&token=[token]

Another possibility is to employ a "cleaner" URI form with more hierarchical structure:

http://server/appstats/appID/statType/period?token=[string]

thus an example request might look like:

http://123.123.123.123/appstats/1/downloads/lastMonth?token=password123

The latter format can be achieved by altering the .htaccess file of the Apache server, using the mod_rewrite module [85]. This effectively redirects requests, enabling an arbitrary URL format.

9.3.4 Front-end and visualization

Considering the requirement that the statistics platform must be viewable via a smartphone, there are two options. Either the statistics platform must be developed in dual versions: for desktop computers and for smartphones; or a cross-platform hybrid developed based on web techniques. The advantage with the first approach is the possibility of developing a native app for one or more smartphone OS's. However, as the app publishing company creates apps for all major smartphone platforms, it seems counterintuitive to write native apps for every platform, or to restrict access to only a target selected subset. This suggests initially developing a web-based platform that uses responsive design – depending on the size of the viewport, the layout will be adapted for a nice user experience.

One popular established front-end framework is Twitter Bootstrap [86], which is open-source and free to use. It comes with good legacy compatibility with older browsers and has a responsive design, thanks to a modular grid layout that enables grid boxes to be rearranged based on the available space. It is also easily customizable and well-documented. For all these reasons, I will likely make use of it as part of the front-end implementation. Figure 9-3 shows a work-in-progress mock-up displaying the same page on a laptop and a smartphone side by side.

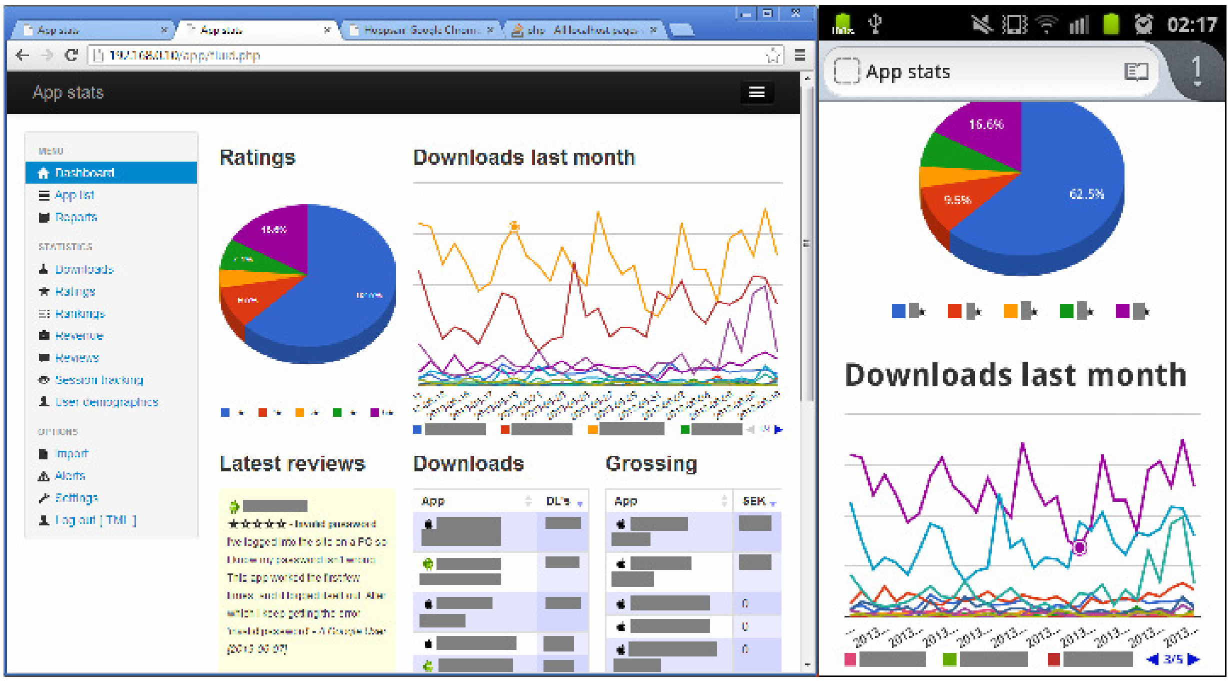


Figure 9-3 – Responsive design with Twitter Bootstrap
(laptop version on the left and smartphone version on the right)

Another advantage of the web technique is that there are APIs and plug-ins for visualization, such as Google Charts API [87] and DataTables for jQuery [88], which may be useful when presenting the data. The pie chart in the above figure was in fact generated with Google Charts. It is also legacy compatible: "*Charts are rendered using HTML5/SVG technology to provide cross-browser compatibility (including VML for older IE versions) and cross platform portability to iPhones, iPads and Android. No plugins are needed*" [87].

9.3.5 Abstract system architecture

Figure 9-4 shows a preliminary blueprint of the internal architecture of the proposed system. This shows schematically how the server will acquire data from external sources over their APIs and how data will be made available from our server via an API of our own.

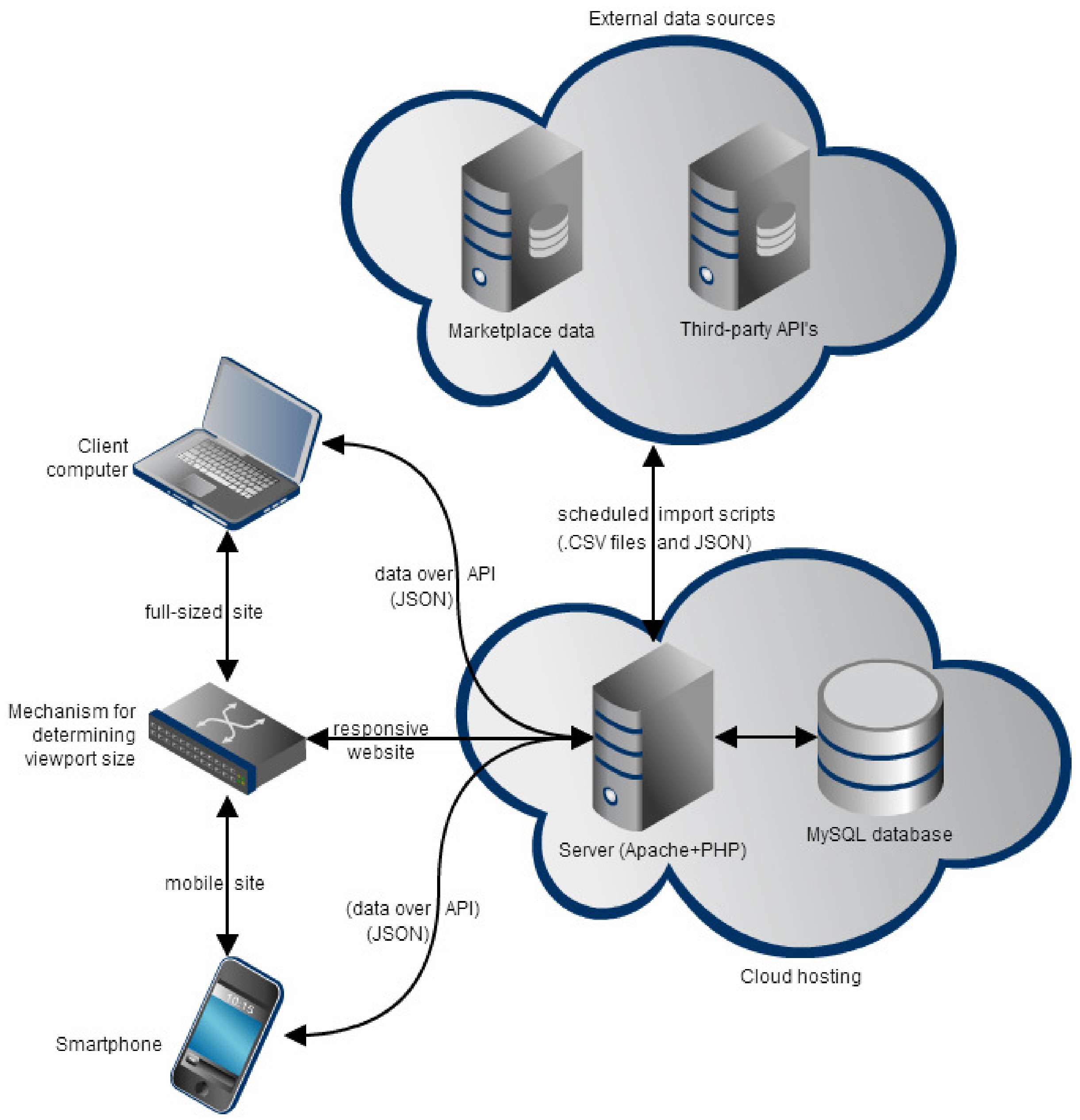


Figure 9-4 – Abstract architecture of the app statistics platform and external services

10 Implementation phase

This chapter describes the actual implementation of the in-house app statistics web platform, how it is designed and the details of some of its inner workings. For a more generic and architectural overview, see the previous chapter.

10.1 Login and authentication

The aforementioned Twitter Bootstrap templates include a basic login page, shown in Figure 10-1, which was modified somewhat but mostly used intact. Some form markup was added, making both fields required before submission, and in case of a failed login, a response message is displayed.

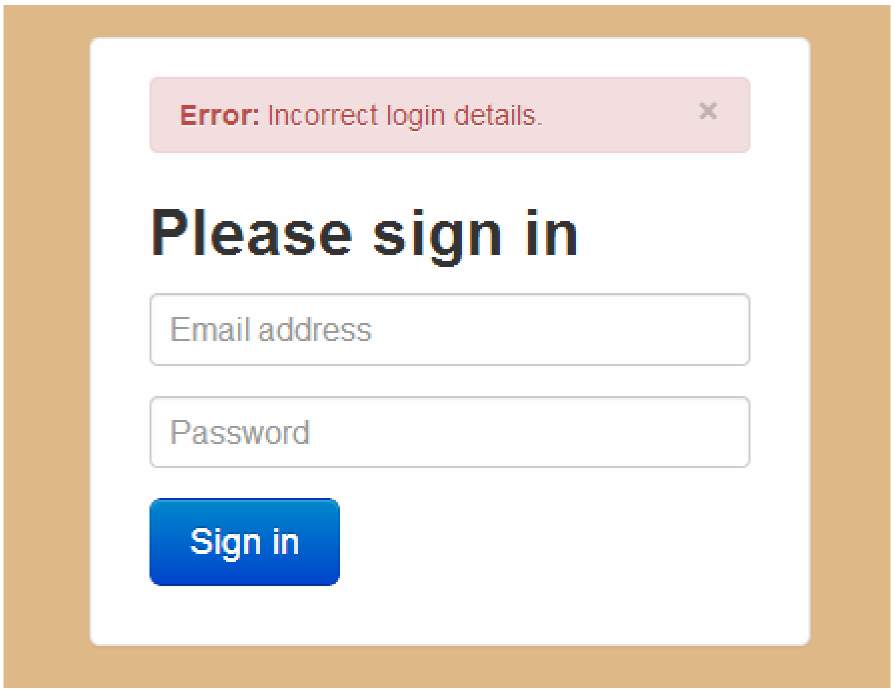

Error: Incorrect login details.

Figure 10-1 – Login page after failed login attempt

The credentials are verified against a table in the MySQL database called login, where all user accounts are stored. Accounts may have different privileges (both in the sense that they may be given access to certain apps, but also on what level data is made available). A certain column in the database specifies a clearance level on a scale from 0-2, as explained below:

Table 8 – Clearance levels for user accounts

Clearance level	Privileges and access
0	Read-only (child account): view rights of all data for certain apps. May not share data further, nor create child accounts of its own.
1	Customer: view rights of all data for certain apps. Can import raw data from files manually, and change account settings. May create, edit and delete child read-only accounts and share data for select apps with them.
2	Admin: view rights of all data for all apps for all customers. Can import raw data from files manually, and change account settings. May create and delete accounts of lower clearance; both customer

	accounts and read-only child accounts.

Accounts of clearance levels 0 and 1 always have to have a "parent" account, to identify who their superior is. Accounts of level 1 default to having Admin as their parent. However, the admin account itself has no parent.

10.2 Menu – Structure and pages

The clearance levels reflect directly on which pages and views are available and viewable to the logged in user. Figure 10-2 is an illustration of the left-hand side menu of the web platform – on the left are the options available to a user of clearance 1 or 2, while the read-only account of level 0 are restricted as can be seen on the right.

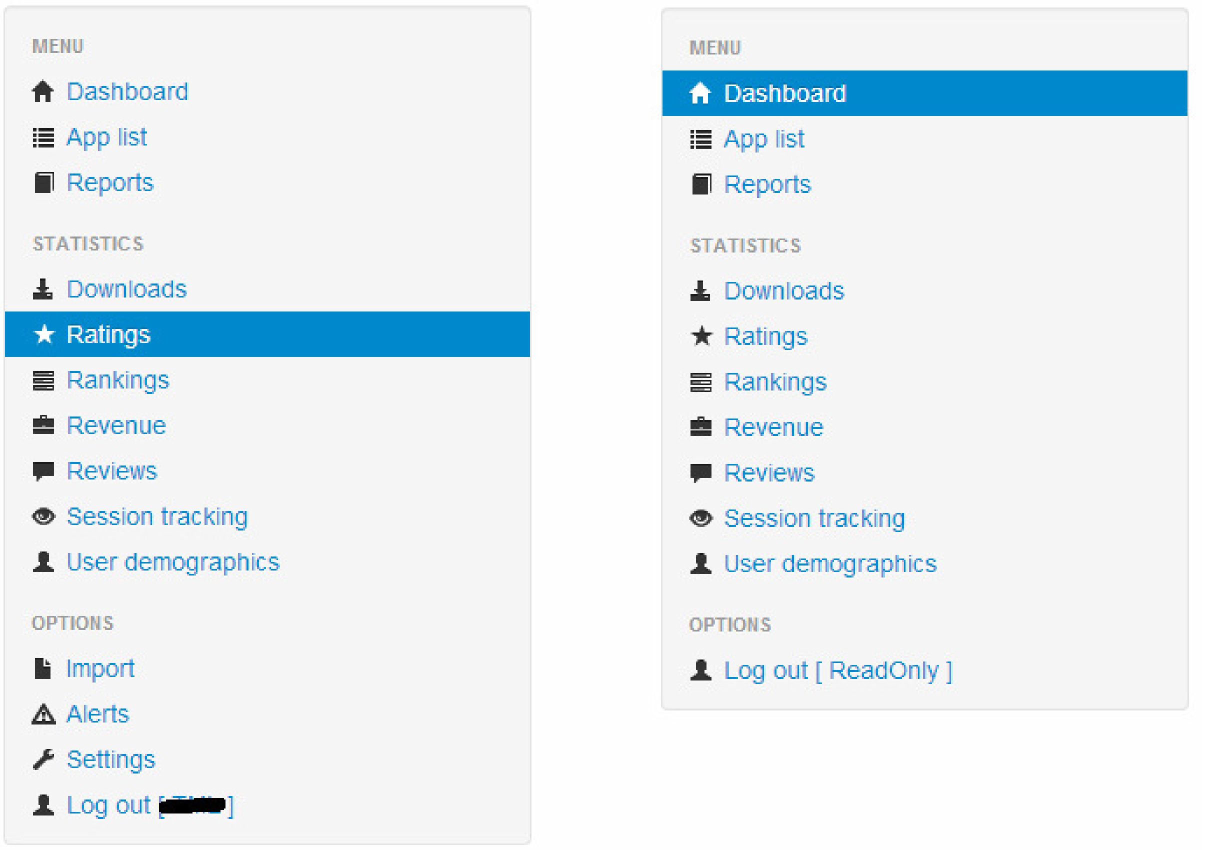


Figure 10-2 – Menu seen with different account privileges

All the options from the "full" menu on the left-hand side will be described in further sub-sections below.

10.2.1 Dashboard

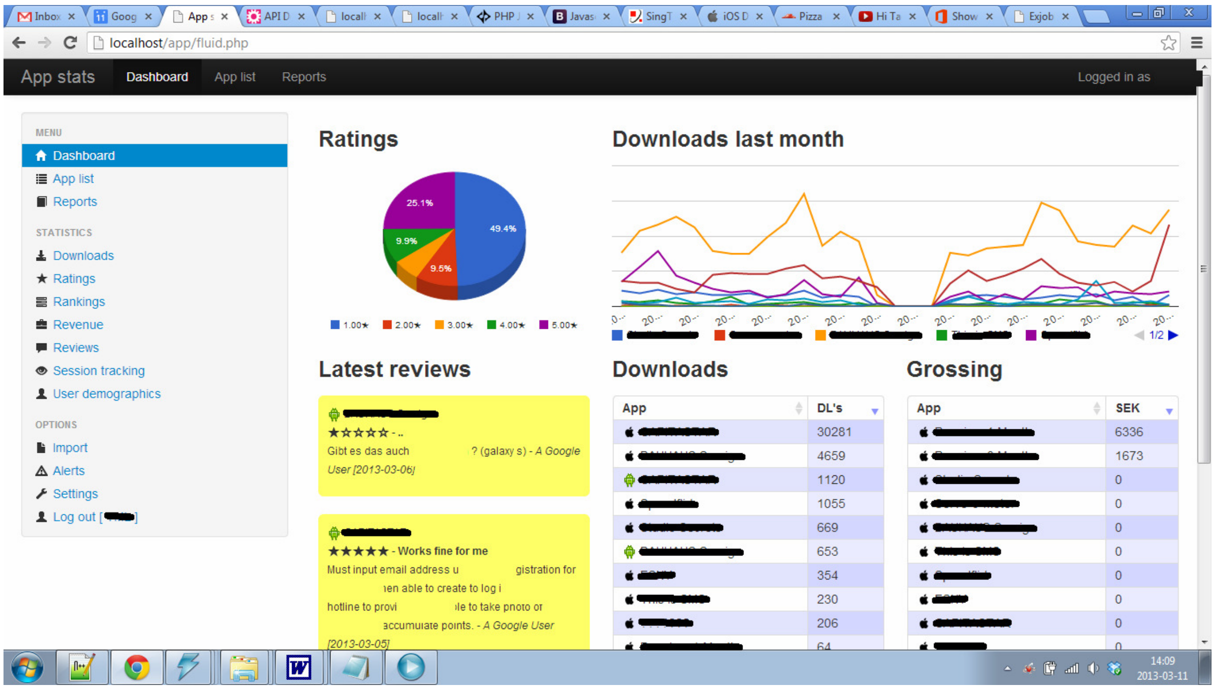
The first page seen after a successful login is the front page or dashboard. The idea of this page is to give a quick overview of vital statistics for all your apps, without having to specify any criteria. As all other pages, the design is made using a responsive CSS template so that the layout depends on the viewport size. The full-fledged site is displayed for screens that have at least 768 pixels of width, and columns are instead made fluid (adjusted for mobile devices) if the width should be less [91]. Refer to figure 9-3 for an example of this.

The PHP function include [96] enables a very modular design in combination with the Twitter Bootstrap grid layout. In the dashboard page, each of the containers is injected into the page source using include calls, as follows:

```
<div class="span4">
   <h3>Latest reviews</h3>
   <?php include 'commentlist.php'; ?>
</div><!--/span-->

<div class="span4">
   <h3>Downloads</h3>
   <?php include 'applist.php'; ?>
</div><!--/span-->
```

And so forth. This setup opens the door to a future customization of the dashboard where each logged in user may choose which "widgets" should be displayed for each box, and the dashboard page could simply include these pages conditionally. Such a personalization of the dashboard is not prioritized at the present stage, but the web platform is built with these options in mind. Some other calls are made using jQuery AJAX capabilities to prevent page reloading. I have on the other hand refrained from using iframes as this is considered by some to be bad practice.

As the pie chart (showing app rating distribution) and line chart (showing downloads per day over the last month) are rendered using Google Charts [87], some custom JavaScripts were written by me and bound to the window.resize event to ensure that the charts would always make use of the full grid space made available to them. A screenshot of the full-sized dashboard page is displayed in Figure 10-3.



[redacted]
★☆☆☆☆ - ..
Gibt es das auch ? (galaxy s) - *A Google User [2013-03-06]*

[redacted]
★★★★★ - Works fine for me
Must input email address u gistration for ıen able to create to log i hotline to provi ıle to take pnoto or accumuıate points. - *A Google User [2013-03-05]*

App	DL's
[redacted]	30281
[redacted]	4659
[redacted]	1120
[redacted]	1055
[redacted]	669
[redacted]	653
[redacted]	354
[redacted]	230
[redacted]	206
[redacted]	64

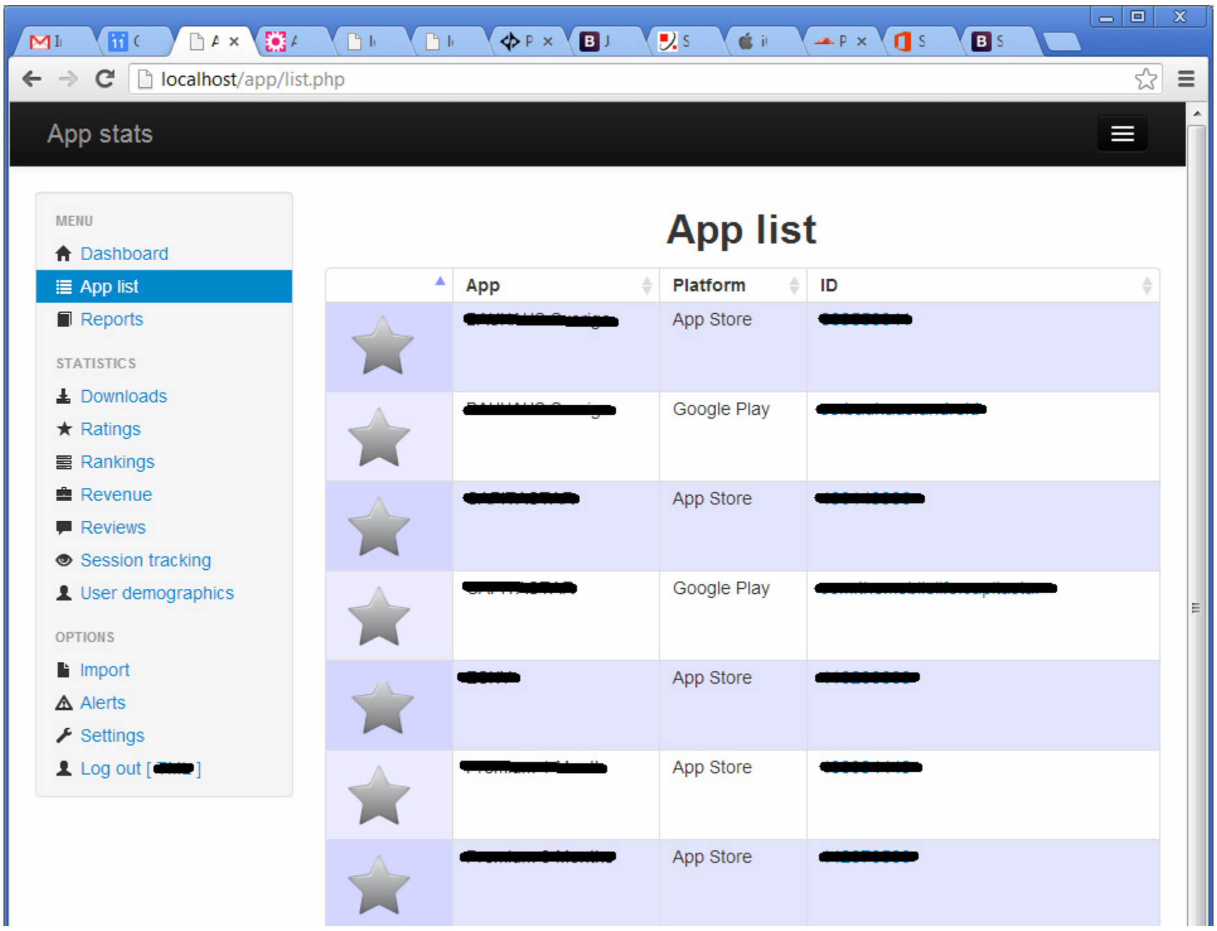
App	SEK
[redacted]	6336
[redacted]	1673
[redacted]	0
[redacted]	0
[redacted]	0
[redacted]	0
[redacted]	0
[redacted]	0
[redacted]	0
[redacted]	0

Figure 10-3 – The dashboard page

Other components on the dashboard include top lists over the most downloaded apps and top grossing apps all-time and a select portion of the latest user reviews. App names are whenever possible accompanied by a small icon to indicate whether it is an Apple App Store app or a Google Android app.

10.2.2 App list

The app list page is self-explanatory – it is a list over all the apps that the currently logged in account may view statistics for. It can be seen in Figure 10-4. In all screenshots below, icons are censored, but are otherwise fetched from the app store by means of an application ID. The ID column also links to the public landing page of the respective marketplaces product page. This table, along with many others in my implementation, use the DataTables jQuery plugin [88] in order to provide sorting of tables along with other functionality.



App list

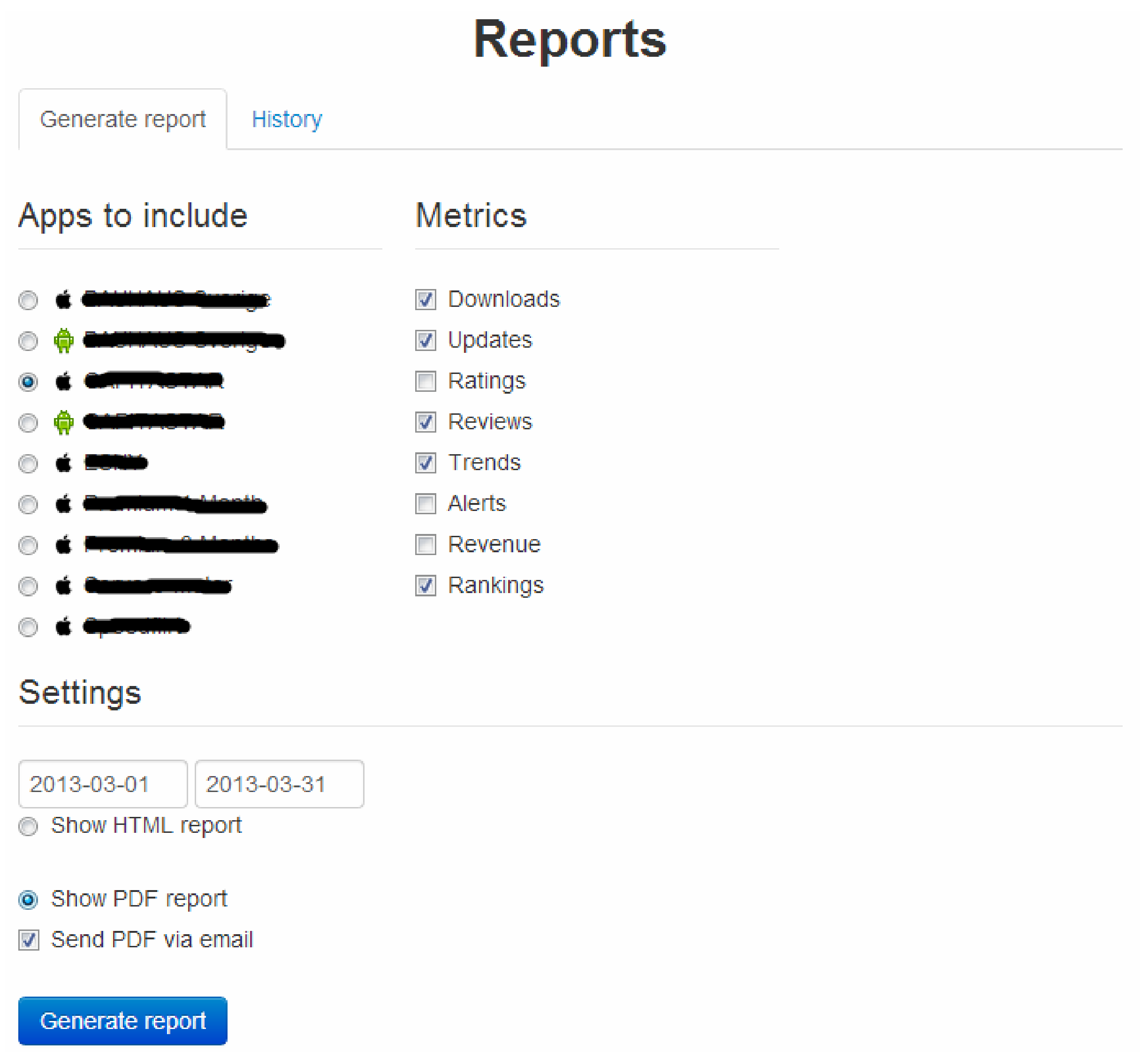
	App	Platform	ID
★	[redacted]	App Store	[redacted]
★	[redacted]	Google Play	[redacted]
★	[redacted]	App Store	[redacted]
★	[redacted]	Google Play	[redacted]
★	[redacted]	App Store	[redacted]
★	[redacted]	App Store	[redacted]
★	[redacted]	App Store	[redacted]

Figure 10-4 – App list page

10.2.3 Reports

The idea behind the report page is to allow dynamic generation of reports on a per-app basis. In other words, after selecting one or more apps and a defined time period, all relevant and available statistics should be fetched and presented in a report-friendly design. This should also be exportable to PDF.

The settings for the output are selected through a simple form, and the user may choose to view a report as a web page (in HTML), PDF in the browser, or have a PDF sent to his/her email address as an attachment.



Reports

Generate report | History

Apps to include

- [redacted]
- [redacted]
- [✓ [redacted]]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

Metrics

- [✓ Downloads]
- [✓ Updates]
- Ratings
- [✓ Reviews]
- [✓ Trends]
- Alerts
- Revenue
- [✓ Rankings]

Settings

2013-03-01 | 2013-03-31

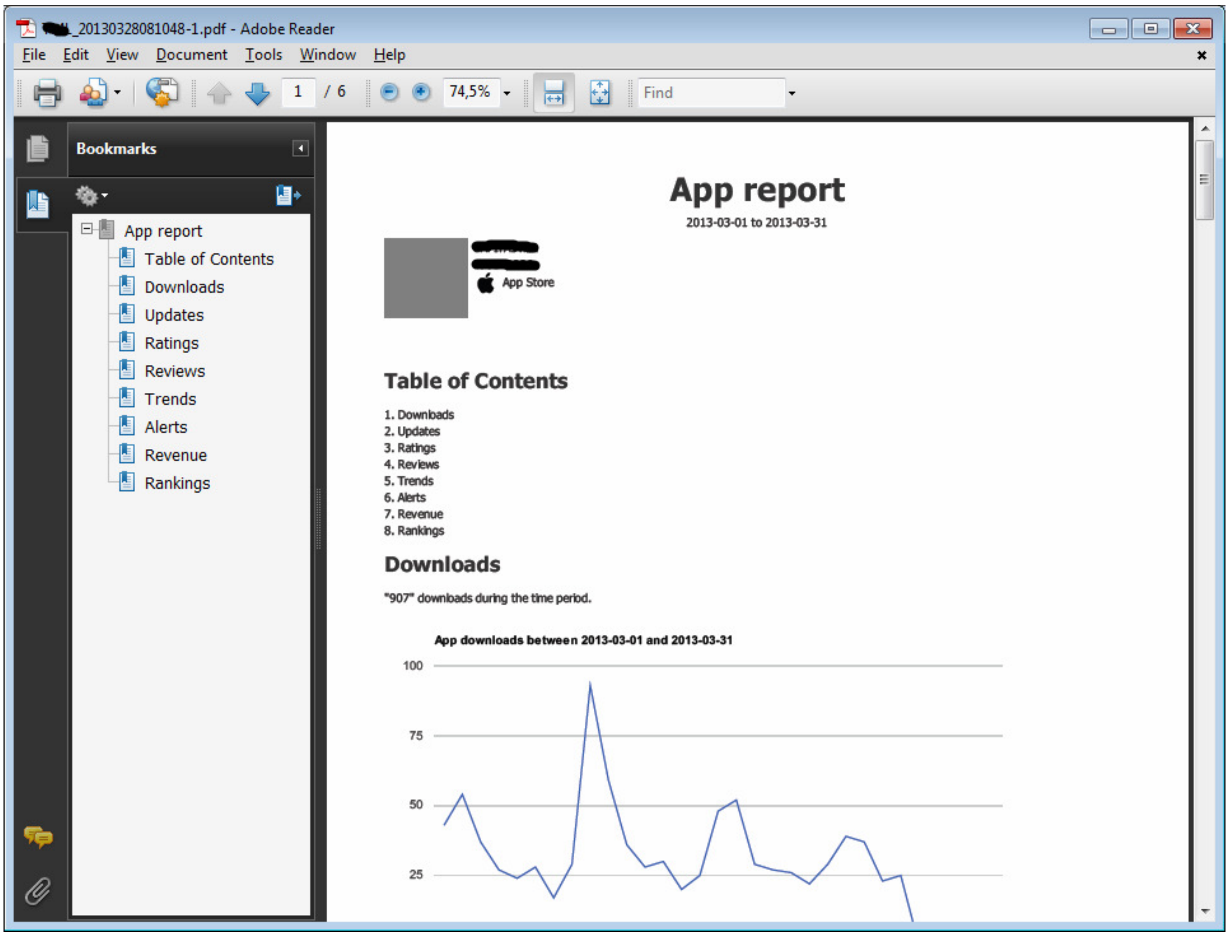
- Show HTML report
- [✓ Show PDF report]
- [✓ Send PDF via email]

Generate report

Figure 10-5 – Report generation

Upon clicking the button, the user is met with an AJAX spinner icon from [107] indicating the report generation process has started. Only the metrics selected by the user are included in the report. Had the user opted to receive or view a PDF, then the generation is done using wkhtmltopdf [108], a relatively new PDF generation library built upon WebKit, which is the web page rendering engine used by browsers such as Chrome and Safari [109]. This is a reliable way to create PDF, as it is more likely to resemble the original HTML page even in its PDF form.

The package also creates a proper Table of Contents (seen on the left-hand side in Figure 10-6) by means of parsing header tags in the HTML code and turning these into chapters, sub-chapters and sections.


[redacted]_20130328081048-1.pdf - Adobe Reader

File Edit View Document Tools Window Help

1 / 6 74,5% Find

Bookmarks

- App report
  - Table of Contents
  - Downloads
  - Updates
  - Ratings
  - Reviews
  - Trends
  - Alerts
  - Revenue
  - Rankings

App report

2013-03-01 to 2013-03-31

[redacted]

App Store

Table of Contents

1. Downloads
2. Updates
3. Ratings
4. Reviews
5. Trends
6. Alerts
7. Revenue
8. Rankings

Downloads

"907" downloads during the time period.

App downloads between 2013-03-01 and 2013-03-31

Figure 10-6 – Generated PDF report

The report is optionally sent to the user’s email address, using PEAR Mail [98] and its MIME package for attaching the newly generated PDF.

10.2.4 Downloads

Based on a proof-of-concept by Karl Monaghan [92], this simple Google Chart displays downloads of App Store apps over the last 30 days. I have done several modifications of his example code, however, attempting to make it compatible with the data structures for Android data as well. The layout is seen in Figure 10-7.

The options set a time period, selecting a single app (allowing the chart to auto-adjust axis scales accordingly) and changing the metric between either downloads or updates. Data is fetched via the jQuery .getJSON() method which retrieves data asynchronously from a PHP script which responds with JSON output.

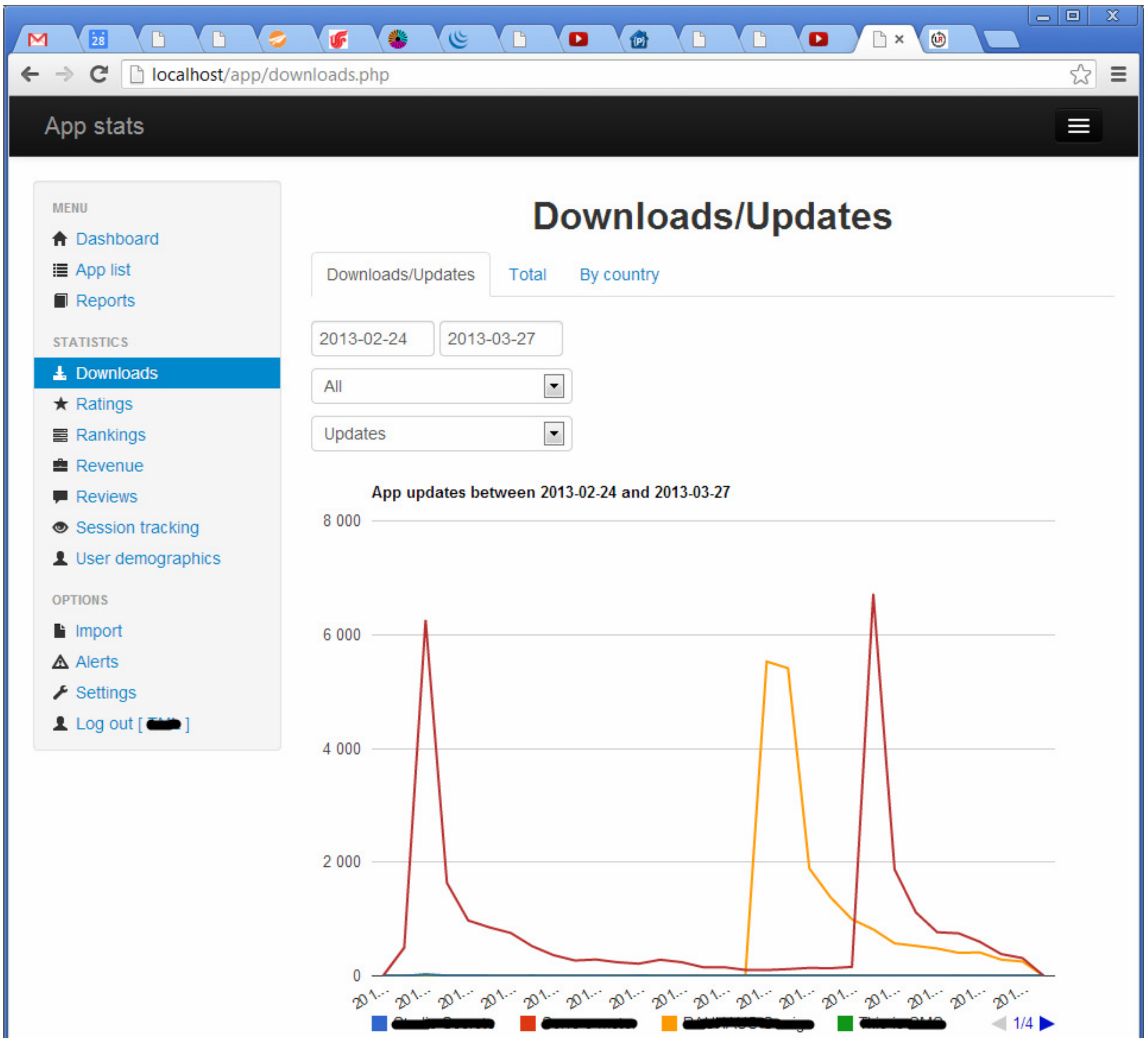
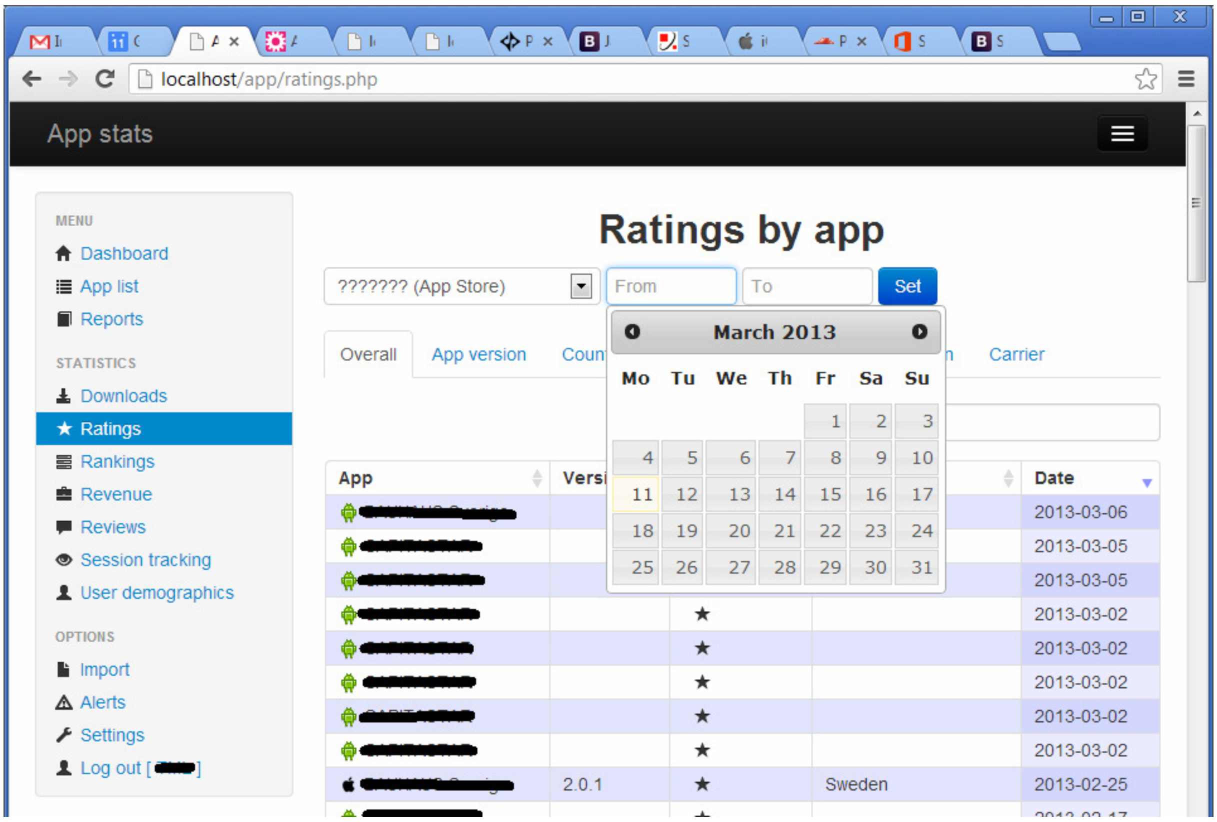


Figure 10-7 – Downloads page

10.2.5 Ratings

All app ratings, as indicated with 1-5 stars, are listed in a table as seen in Figure 10-8. Apps can be selected individually and ratings may be filtered with time periods based on starting and/or ending date. The search text field above the table enables free-text search, and the table can be sorted based on any column. Dates are chosen either by typing them manually with the keyboard or by clicking on the popup calendar component provided by the standard jQuery UI library.



localhost/app/ratings.php

App stats

MENU
- Dashboard
- App list
- Reports

STATISTICS
- Downloads
- Ratings
- Rankings
- Revenue
- Reviews
- Session tracking
- User demographics

OPTIONS
- Import
- Alerts
- Settings
- Log out [[redacted]]

Ratings by app

??????? (App Store) | From | To | Set

Overall | App version | Coun | n | Carrier

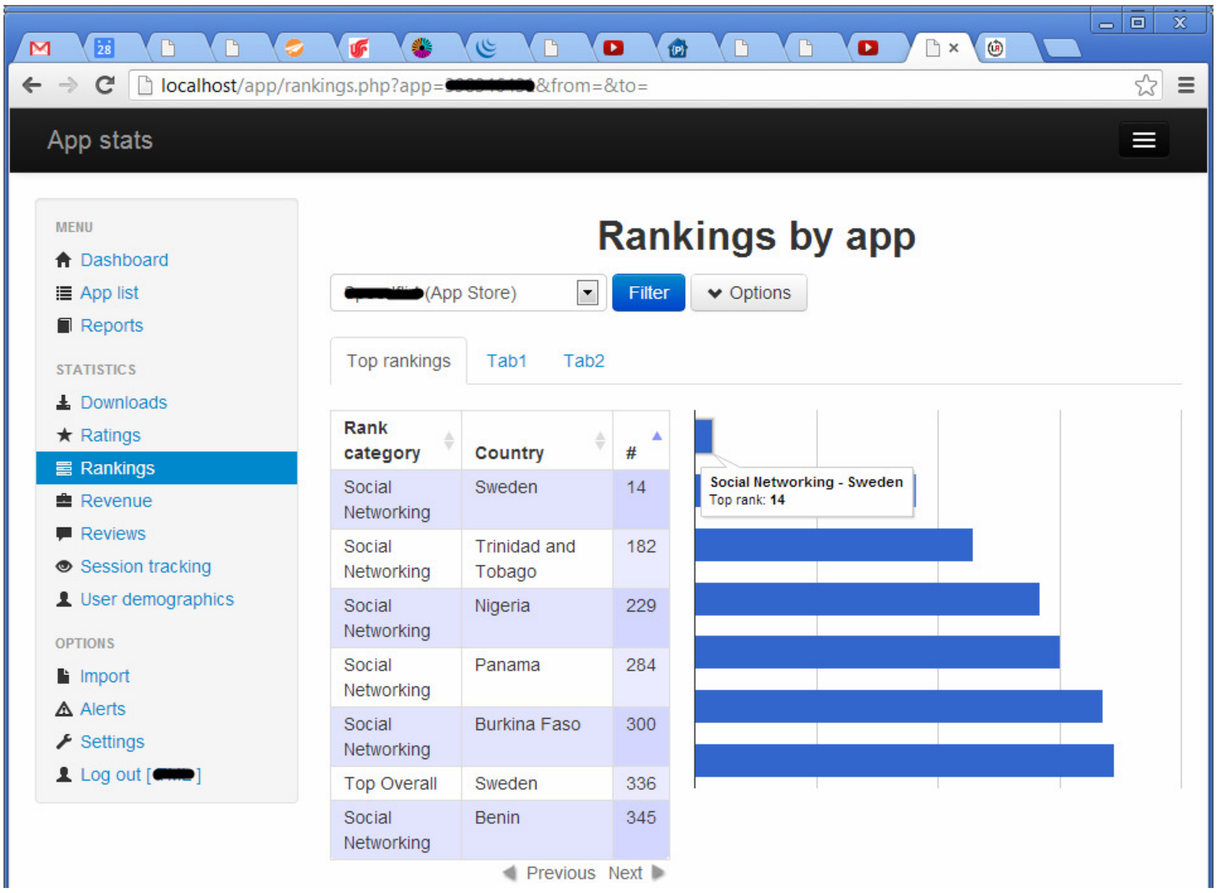
March 2013						
Mo	Tu	We	Th	Fr	Sa	Su
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

App	Versi			Date
[redacted]				2013-03-06
[redacted]				2013-03-05
[redacted]				2013-03-05
[redacted]		★		2013-03-02
[redacted]		★		2013-03-02
[redacted]		★		2013-03-02
[redacted]		★		2013-03-02
[redacted]		★		2013-03-02
[redacted]	2.0.1	★	Sweden	2013-02-25

Figure 10-8 – Ratings page, by app

10.2.6 Rankings

This page is still under development although functional. The view still needs functionality to track each of the displayed categories over time. In its current state, this page lists the highest achieved top position for each category the app belongs to. An example of the page as it is at this time is shown in Figure 10-9.



Rankings by app

Rank category	Country	#
Social Networking	Sweden	14
Social Networking	Trinidad and Tobago	182
Social Networking	Nigeria	229
Social Networking	Panama	284
Social Networking	Burkina Faso	300
Top Overall	Sweden	336
Social Networking	Benin	345

Figure 10-9 – Rankings by category

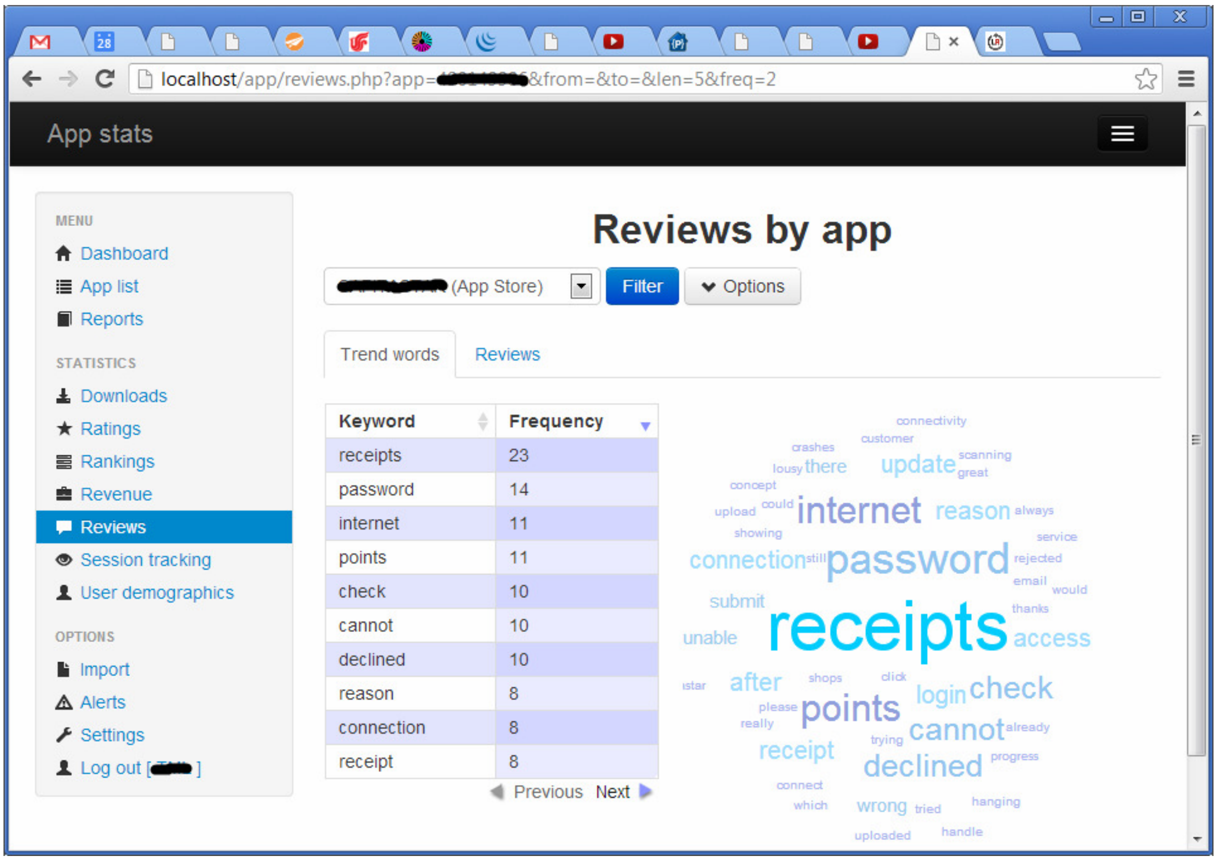
10.2.7 Revenue

This page is yet to be developed. It was not prioritized – mostly because almost all of the apps within the set which we worked with for this project are released free of charge – but such a page would be a nice feature. As the automatically imported data from the iTunes Connect webpage features app prices and number of units purchased, the total revenue per app is already calculated and displayed in a widget box on the dashboard. However, the corresponding data is not available for Google Play. Should a third-party API prove adequate, a multi-platform simplified view over app revenue might be included, otherwise this view will only work for iOS apps.

10.2.8 Reviews

Much work has gone into this metric, as current third-party statistics services leave much to desire. The first tab of this view (see Figure 10-10) contains a toplist of frequent words used in the reviews of the selected app for the filtered period of time (calendar dates). This is done by parsing headlines and review texts from all imported reviews and counting the frequencies of all words of a length greater than n characters, using array_count_values in PHP [99]. The most frequent words are rendered into a *word cloud* using the jQuery plugin jQCloud [100] which helps to visualize the keyword trends. The more frequent a keyword is among reviews, the bigger font size in the word cloud. The following illustration is a practical example of an app where the login service was having issues, also preventing the users from uploading their receipts which was a core functionality of the particular app. The problems at hand are very easy to spot thanks to the table and word cloud.

However, common everyday words, often referred to as *stop words* [113], should probably be pruned using some kind of dictionary word list to increase precision. Some techniques for pruning and filtering out unwanted words are discussed in chapter 2.2.2 of [113]. As the authors also point out, if we were to expand the trend words functionality to allow phrases, we must not over-prune, as meaningful phrases could be built from a combination of stop words.

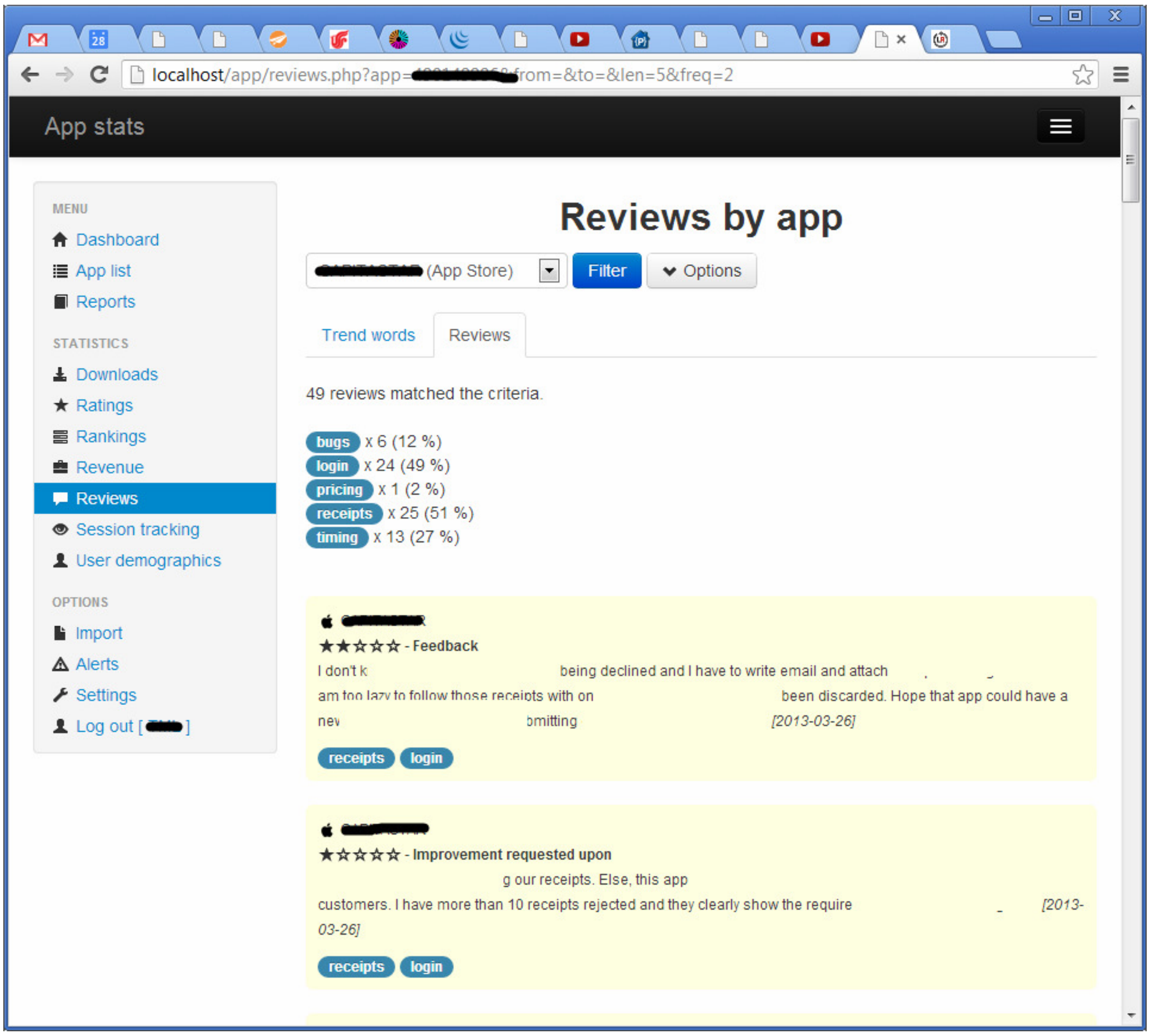


Keyword	Frequency
receipts	23
password	14
internet	11
points	11
check	10
cannot	10
declined	10
reason	8
connection	8
receipt	8

Figure 10-10 – Trending review keywords and word cloud

The second tab lists all reviews in reverse chronological order and may be filtered per app and date period (see Figure 10-11). Each review states the app (including version info when available), platform, rating (1-5 stars), review title and text, author and review date.

Each review may also be tagged with labels, which are applied should the review match filters set up by the user. These labels are counted and presented with numbers and percentages at the top of the page. How these tags work will be presented in more detail in section 10.2.13.


localhost/app/reviews.php?app=[redacted]from=&to=&len=5&freq=2

App stats

MENU
- Dashboard
- App list
- Reports

STATISTICS
- Downloads
- Ratings
- Rankings
- Revenue
- Reviews
- Session tracking
- User demographics

OPTIONS
- Import
- Alerts
- Settings
- Log out [[redacted]]

Reviews by app

[redacted] (App Store) | Filter | Options

Trend words | Reviews

49 reviews matched the criteria.

bugs x 6 (12 %)
login x 24 (49 %)
pricing x 1 (2 %)
receipts x 25 (51 %)
timing x 13 (27 %)

[redacted]
★★☆☆☆ - Feedback
I don't k being declined and I have to write email and attach
am too lazy to follow those receipts with on been discarded. Hope that app could have a
nev omitting [2013-03-26]
receipts login

[redacted]
★☆☆☆☆ - Improvement requested upon
g our receipts. Else, this app
customers. I have more than 10 receipts rejected and they clearly show the require [2013-03-26]
receipts login

Figure 10-11 – Review page with tags applied

10.2.9 Session tracking

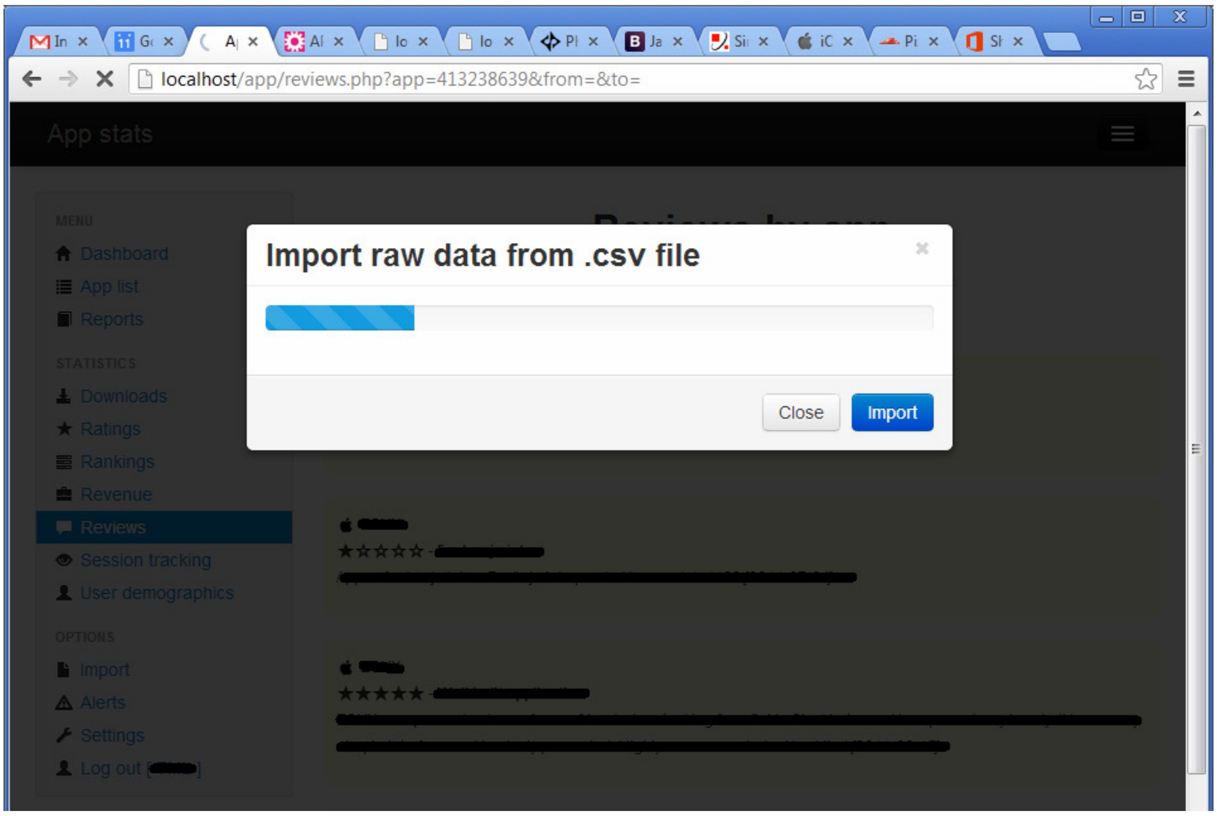
This and the following page are mostly placeholders for upcoming integration with the session-tracking packages such as Google Analytics and Flurry SDK. Such functionality could not be incorporated into the project within the time constraints of this master's thesis project, but this data is highly useful and a suitable topic for future development.

10.2.10 User demographics

This page is presently not implemented, but simply a placeholder menu link for future addition.

10.2.11 Import

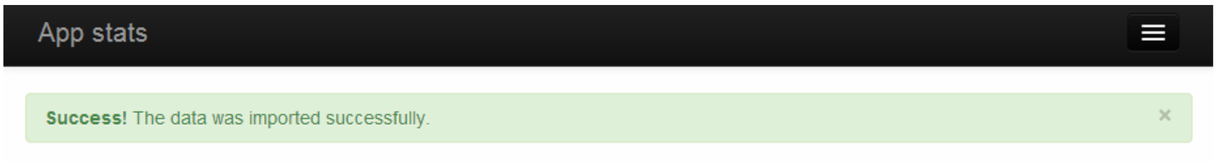
As per the requirement from the company at which this thesis project was undertaken, customers must be able to manually import raw data that was exported from each marketplace. In order to realize this functionality, the well-documented CSV-functionality of PHP was used. Upon clicking the menu link – only available to accounts of clearance levels 1 or higher – a popup dialog appears through which a CSV format file may be selected. This dialog is shown in Figure 10-12, with the progress bar in current display.



[redacted]

Figure 10-12 – Manual import of data

Then an animated progress bar indicates the file upload to server and the database insertion, upon which either a failure or success message is displayed to the user at the top of the dashboard start page (see Figure 10-13). These messages are always dismissable with a click on the X icon.



Success! The data was imported successfully.

Figure 10-13 – Dismissable user notices

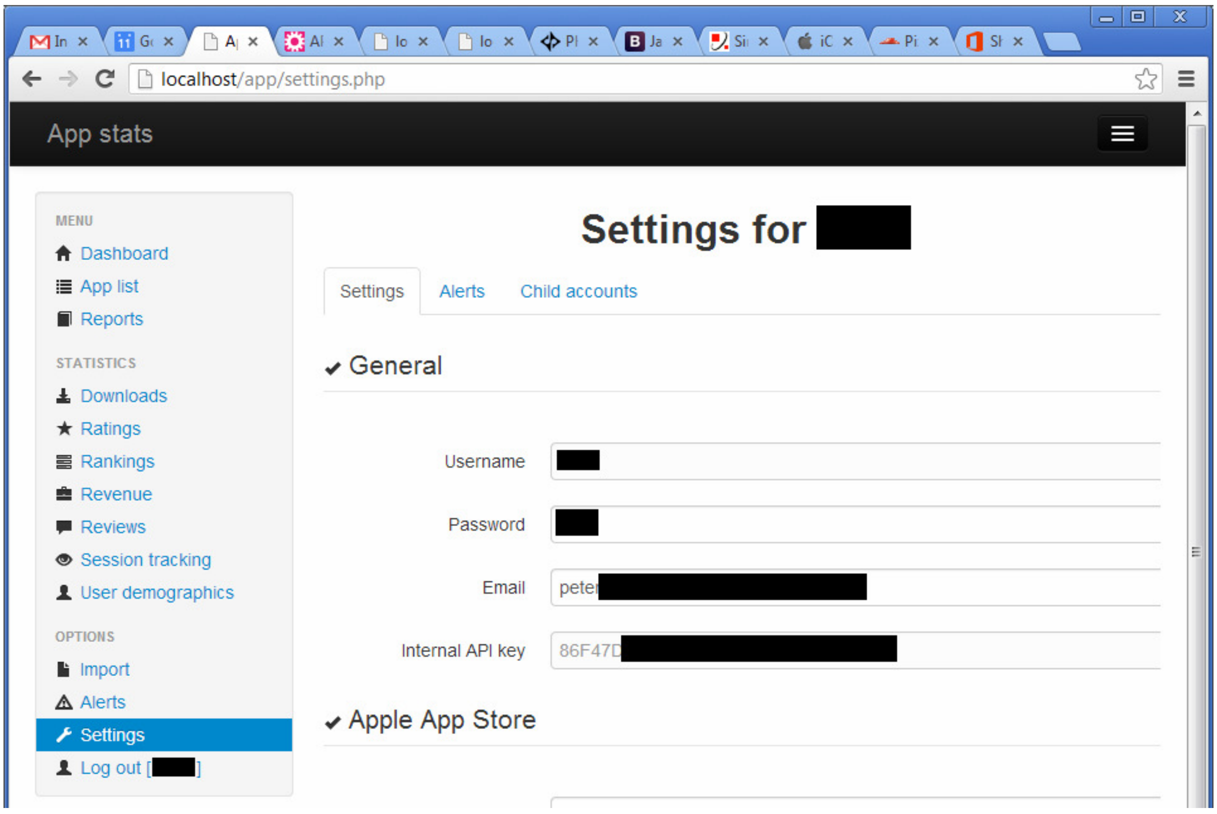
10.2.12 Alerts

This view was experimentally implemented, see section 10.4 for a further description. The idea is to construct a form-based view, enabling any metric stored in the local database to be monitored, and upon satisfaction of criteria, emails should be sent out when these alerts are triggered. The form for setting up such an alert would ideally use components to form full sentences explaining the filter. A few scenarios are shown below:

[E-mail me] if [App A] - [Average rating] [Gets higher than] ______
or
[Notify me upon login] if [App B] - [Receives new review] [] ______

10.2.13 Settings

The settings page, only available to accounts of clearance level 1 or above, is divided content-wise into tabs and groups. On the main page, the user may input their login details to their iTunes Connect account, Google Play Developer Console, and Distimo (third-party API) details, in addition to their internal username, password, and email for the web platform itself. For each section, a checkbox or cross will indicate that details are either sufficiently and correctly entered or that they are not yet provided by the user.



Settings for [redacted]

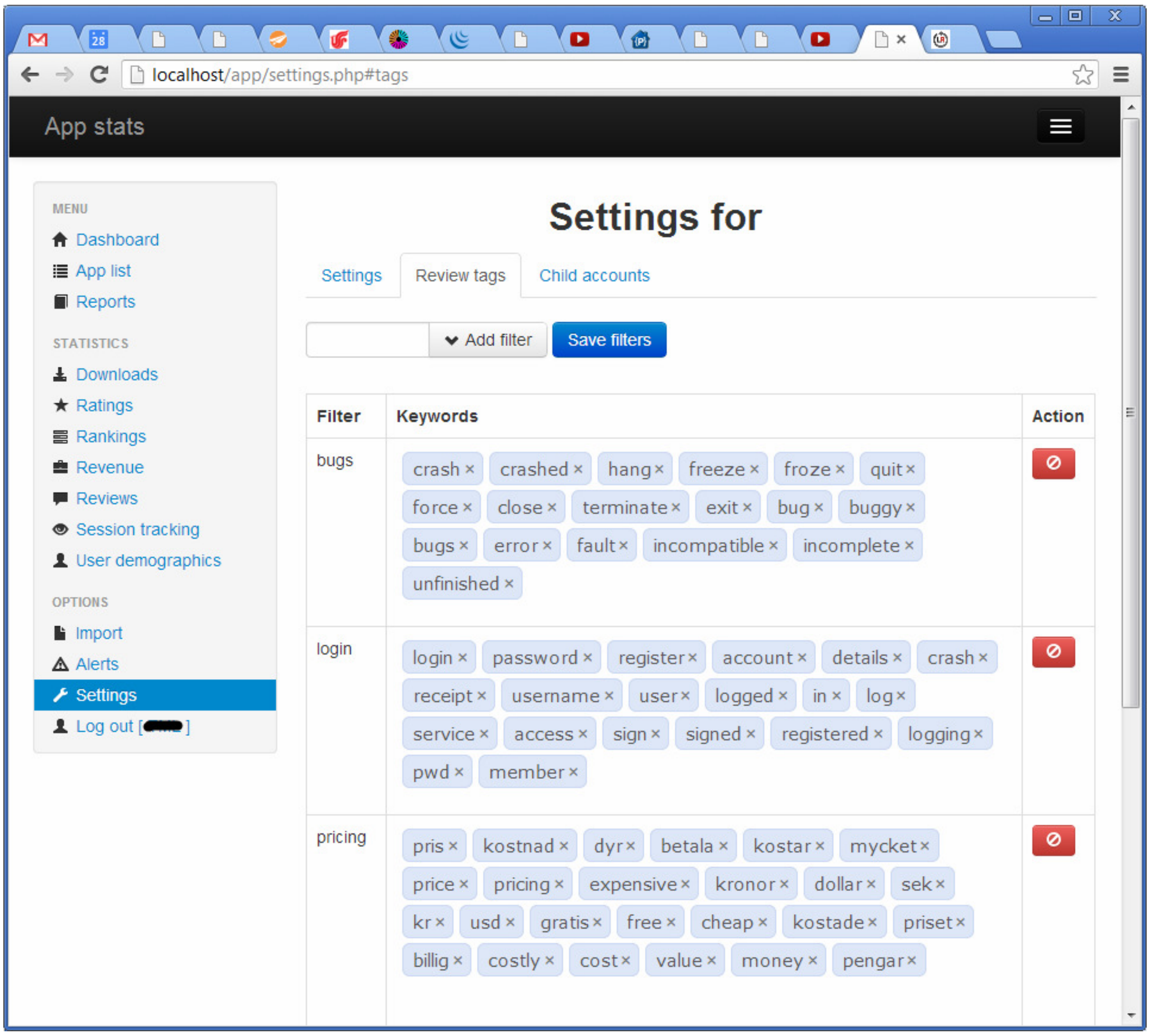
✓ General

Username	[redacted]
Password	[redacted]
Email	pete[redacted]
Internal API key	86F47D[redacted]

✓ Apple App Store

Figure 10-14 – Settings page: first tab

The second clickable tab allows for creating review tags, by entering keywords and saving them as filters. Figure 10-15 displays some of these tags found in reviews, based on the filters already created by the user. The input fields are connected by a jQuery plugin called Tag-it [110], with a customized CSS style for turning a comma-separated word list into individually editable word tags.


localhost/app/settings.php#tags

App stats

MENU
- Dashboard
- App list
- Reports

STATISTICS
- Downloads
- Ratings
- Rankings
- Revenue
- Reviews
- Session tracking
- User demographics

OPTIONS
- Import
- Alerts
- Settings
- Log out [[redacted]]

Settings for

Settings | Review tags | Child accounts

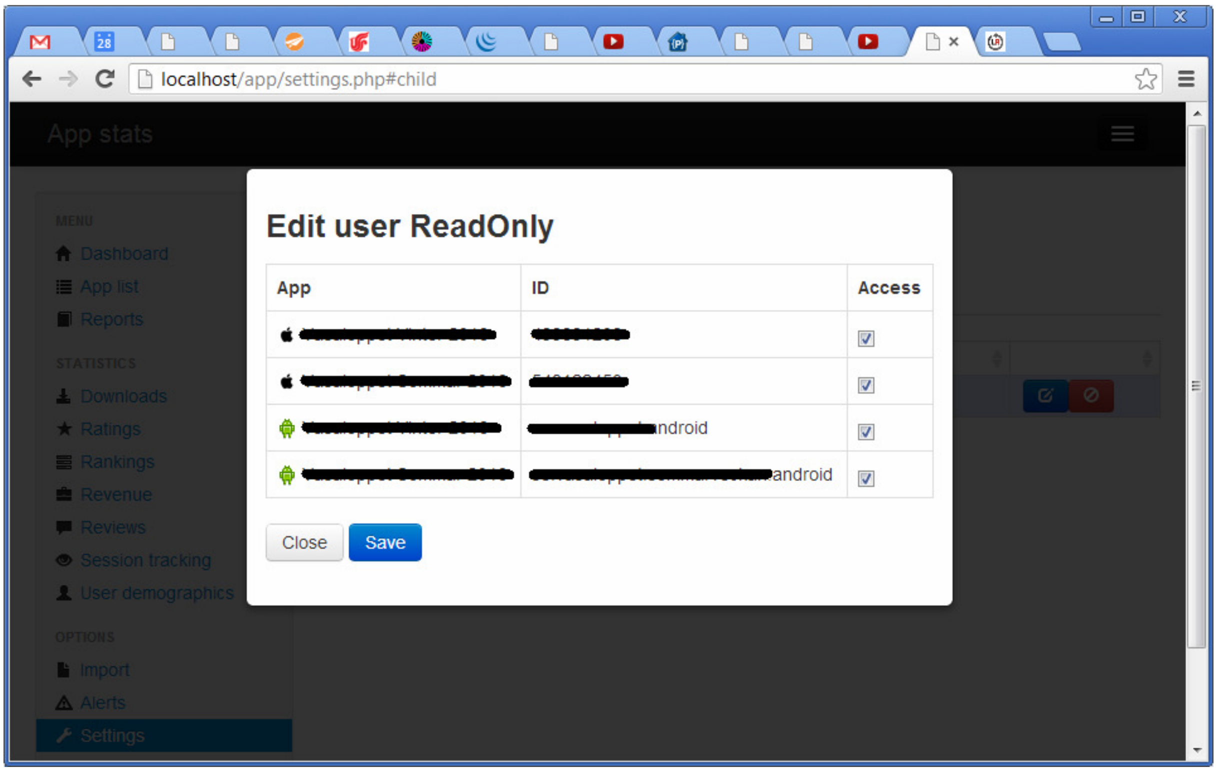
Add filter | Save filters

Filter	Keywords	Action
bugs	crash × crashed × hang × freeze × froze × quit × force × close × terminate × exit × bug × buggy × bugs × error × fault × incompatible × incomplete × unfinished ×	⊘
login	login × password × register × account × details × crash × receipt × username × user × logged × in × log × service × access × sign × signed × registered × logging × pwd × member ×	⊘
pricing	pris × kostnad × dyr × betala × kostar × mycket × price × pricing × expensive × kronor × dollar × sek × kr × usd × gratis × free × cheap × kostade × priset × billig × costly × cost × value × money × pengar ×	⊘

Figure 10-15 – Review tag filters

Hopefully, an upcoming version of the app stats web platform will enable alerts to be triggered when a specified percentage of recent reviews attract a specific label.

The Child accounts tab lists any already created child accounts (with read-only permission) and allows for creating new ones. This tab is shown in Figure 10-16.



Edit user ReadOnly

App	ID	Access
[redacted]	[redacted]	✓
[redacted]	[redacted]	✓
[redacted]	[redacted]ndroid	✓
[redacted]	[redacted]android	✓

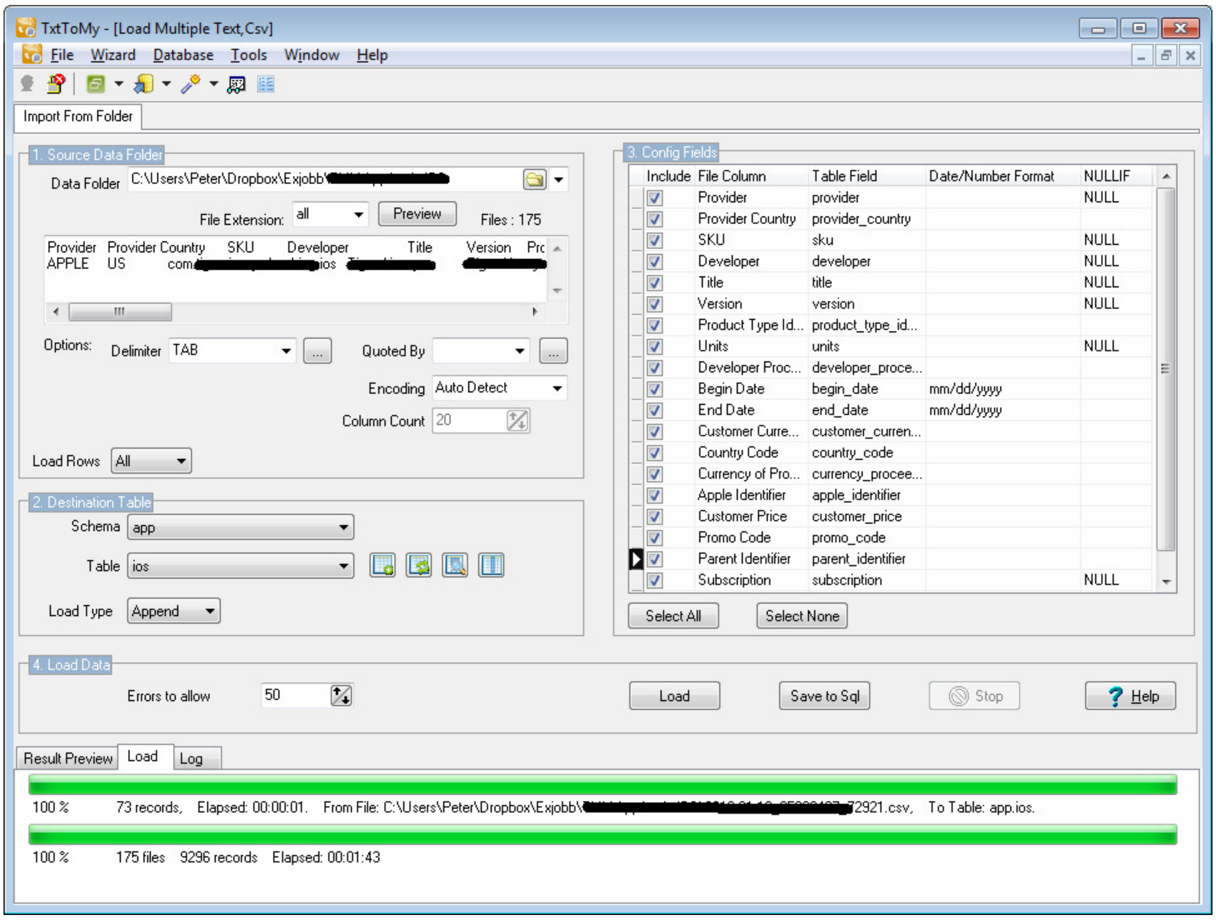
Figure 10-16 – Editing user permissions

10.3 Data and database

This section describes some of the underlying database schemas, how manual data import is done, and how external data sources are polled regularly for automated import. The outlines of the internal API are also discussed.

10.3.1 Import – Initial backlog

For current and future statistics, all metrics should be automatically fetched and imported from external data sources. However, statistics from the past may need to be manually imported. As mentioned in chapter 9, App Annie was a good source for this backlog data as it provides exported CSV files. However, as one such file is generated per day, this could easily result in hundreds of CSV files. A small utility called TxtToMy [97] allows for batch importation of these files, by mapping CSV column names to MySQL table column names. Figure 10-17 shows a sample screenshot from one such import operation.


TxtToMy - [Load Multiple Text,Csv]

Import From Folder

1. Source Data Folder

Data Folder: C:\Users\Peter\Dropbox\Exjobb\[redacted]

File Extension: all — Files : 175

Provider	Provider Country	SKU	Developer	Title	Version	Prc
APPLE	US	com[redacted]ios	[redacted]		[redacted]	

Options: Delimiter: TAB — Quoted By: — Encoding: Auto Detect — Column Count: 20

Load Rows: All

2. Destination Table

Schema: app

Table: ios

Load Type: Append

3. Config Fields

Include	File Column	Table Field	Date/Number Format	NULLIF
✓	Provider	provider		NULL
✓	Provider Country	provider_country		
✓	SKU	sku		NULL
✓	Developer	developer		NULL
✓	Title	title		NULL
✓	Version	version		NULL
✓	Product Type Id...	product_type_id...		
✓	Units	units		NULL
✓	Developer Proc...	developer_proce...		
✓	Begin Date	begin_date	mm/dd/yyyy	
✓	End Date	end_date	mm/dd/yyyy	
✓	Customer Curre...	customer_curren...		
✓	Country Code	country_code		
✓	Currency of Pro...	currency_procee...		
✓	Apple Identifier	apple_identifier		
✓	Customer Price	customer_price		
✓	Promo Code	promo_code		
✓	Parent Identifier	parent_identifier		
✓	Subscription	subscription		NULL

4. Load Data

Errors to allow: 50

```
100 %      73 records,   Elapsed: 00:00:01.   From File: C:\Users\Peter\Dropbox\Exjobb\[redacted]2921.csv,   To Table: app.ios.
100 %      175 files   9296 records   Elapsed: 00:01:43
```

Figure 10-17 – Batch import of CSV data with TxtToMy

10.3.2 Import – Manual

The code for the web platform import function (as described in section 10.2.11) contains a constant, associative array which can be seen as a key-value pair that allows array data. If we import a file with a name ending in 'android_overall_ratings', a function columnFormat($importtype) will return the corresponding column structure for this particular database table: (appID, date, daily_avg_rating, total_avg_rating). This allows us to dynamically construct our SQL INSERT statement. These mappings are applied for each different CSV file type allowed for import.

The actual handling of the CSV data is facilitated by functions in PHP, such as fgetcsv [95]. This function takes as parameters the data set, maximum length per line, delimiter character, and escape characters. Extracting actual data into a workable format is quite user-friendly and easy with this function.

Should we attempt to import a file of unknown format, we are returned to the start page with a dismissible error notice. The actual file upload code snippet was based on a sample from W3 Schools [94].

10.3.3 Import – Scheduled and automated

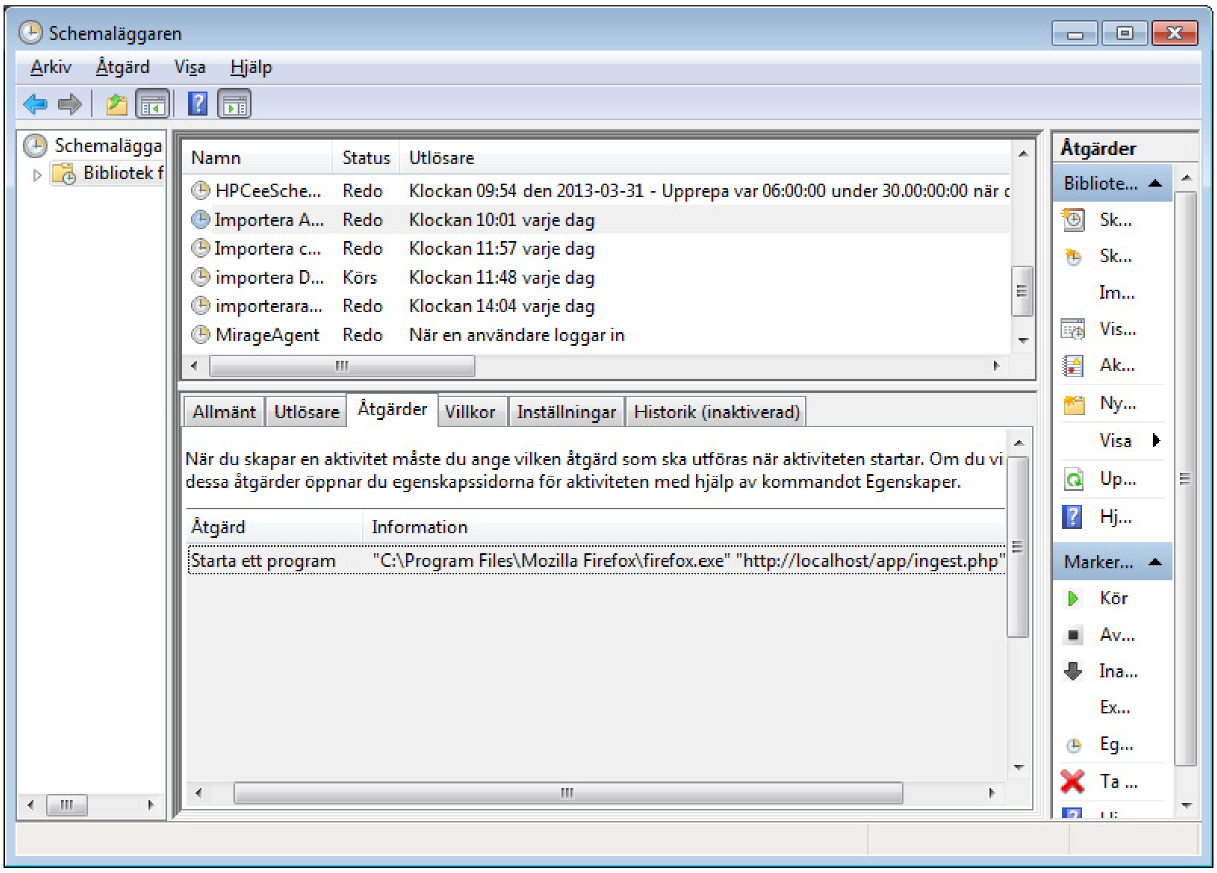
Two different sources are used for automated data import.

The first is the URL set up by Apple to serve its Autoingestion Java class requests. This is unofficially available for other clients as well, as long as the request format is correct. The script from [92] is a base for this, but many modifications have been made, such as supporting a dynamic number of App Store accounts. Hopefully, since Apple's own Java class uses the same format, this data source should remain available for a long period of time.

The second data source is via Distimo APIs for ranking, downloads, and reviews. The returned JSON is decomposed and stored into the database. My code ensures the import of all possible data by executing a data poll for each distinct user account with Distimo credentials, and for each app belonging to those accounts. As these imports may contain thousands of entries it is beneficial to use prepared statements to reduce execution time.

Both of these import types are automated and scheduled to run daily at a certain time of day by opening the respective PHP file in a web browser. Should the computer not be turned on at the time, lack a network connectivity or fail for some other reason, then the scripts are scheduled to recover as soon as possible. Below is a screenshot of how these scripts are scheduled in my development environment (using Windows). The crontab command [82] has the corresponding functionality for a UNIX or Linux environment.

One option for this scheduling is to wrap all scripts into a single call (see Figure 10-18), from a single scheduling PHP file. This is more convenient than scheduling multiple scripts. However, separating each import type into its own scheduled script enables spreading the load evenly over the day and enables clear log traces for all executions. It also enables a more robust setup compared to scheduling a single PHP file which would represent a single point-of-failure.



Namn	Status	Utlösare
HPCeeSche...	Redo	Klockan 09:54 den 2013-03-31 - Upprepa var 06:00:00 under 30.00:00:00 när c
Importera A...	Redo	Klockan 10:01 varje dag
Importera c...	Redo	Klockan 11:57 varje dag
importera D...	Körs	Klockan 11:48 varje dag
importerara...	Redo	Klockan 14:04 varje dag
MirageAgent	Redo	När en användare loggar in

När du skapar en aktivitet måste du ange vilken åtgärd som ska utföras när aktiviteten startar. Om du vi
dessa åtgärder öppnar du egenskapssidorna för aktiviteten med hjälp av kommandot Egenskaper.

Åtgärd	Information
Starta ett program	"C:\Program Files\Mozilla Firefox\firefox.exe" "http://localhost/app/ingest.php"

Figure 10-18 – Scheduling import scripts in Windows

10.3.4 MySQL database queries and constraints

The database employs some standard constraints such as foreign keys between tables to ensure correctly entered data. For instance, appID must be consistent with an existing application for comments, downloads, and the other data to be imported. Constraints for uniqueness are also used to avoid duplicate data. More details about this are given in the following section regarding data principles.

10.3.5 Principles

The development process and particularly the data import scripting has been done to maintain a few constraints to ensure consistency and confidentiality of data.

No timeout – Regardless of which PHP script is executing, manual or automated/scheduled, they need to override the normal server setting for a timeout which is 30 seconds. Otherwise, should the amounts of data to insert be too large, some data could be accidentally excluded. The solution is to write at the top of each PHP file "set_time_limit(0);" where the zero may actually be changed to a non-infinite, but still generous value before deploying the implementation onto a production server, to protect against non-terminating code.

No duplicates – Either by defining aggregate primary keys over all columns which may not be repeated in the table, or by adding a unique index, we ensure that duplicate data is not present in our tables. An example of the latter is ALTER IGNORE TABLE \`app\`.\`testtabell\` ADD UNIQUE INDEX idx_all (column1, column2, … columnN);

No unnecessary polling – By ordering each table by date, finding the latest date of a successful import, we may set this date as the "from" parameter for our next data polling. This ensures that we do not have to filter through duplicates of data already imported. An example of a query that finds this date is SELECT begin_date FROM ios ORDER BY begin_date DESC LIMIT 1;.

No manipulation – Even by changing user data such as HTTP GET variables in URLs that are editable by the user, no unauthorized access should be possible – as the PHP session is monitored server-side at every page load and all SQL queries include subqueries to the effect of …WHERE \`app\`.\`comments\`.appid IN (SELECT DISTINCT appid FROM \`app\`.\`apps\` WHERE publisher = 'loggedInUser') which effectively guarantees that only data from permitted apps is returned.

Similarly, should there be a MySQL error, instead of the common practice $result = mysql_query($query) or die(mysql_error()); which details the database error and prints details to the browser, I have opted to simply redirect the user, using for example header('Location: login.php?msg=FAIL'); which does not reveal information about the underlying database, or its table structure.

10.3.6 Server requirements

Besides the obvious software requirements such as reasonably recent versions of PHP and MySQL, some add-ons were required server-side before deploying the implementation from my development environment (laptop) onto a corporate server.

For instance, the ingestion of Apple App Store data fetches data in the form of compressed .gz archive files. As these must be programmatically unzipped server-side before the import can commence, we need permission to run system or exec commands in PHP which is not always enabled by default. Safe mode must be turned off, file upload and cURL modules must be enabled, and there needs to be an unzip library present. Compared to the example from [92] which did not work on my test server, I instead installed GnuWin [93] on the server and changed the call from gunzip to gzip –d (the last flag meaning deflate). 7zip is another option.

10.3.7 API outlines

It was decided that only the most basic data should be made available through the internal API – partly to minimize support issues, and partly to focus on the core statistics. Statistics will be possible to break down by app, date period, and by type of metric. An aggregate metric for all the apps connected to the account should also be available. A table over the valid request types is shown in table 9:

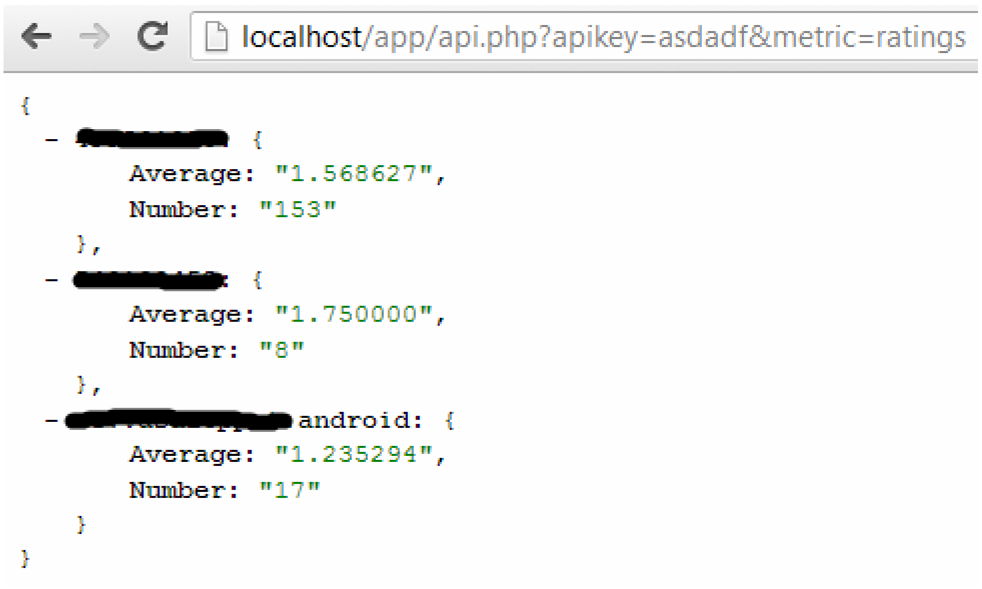
Table 9 – Valid API request types

Metric requested	Required parameters	Optional parameters	Output (JSON response structure)
Downloads per app	API key, App ID	From date, to date	date > metric
Downloads total	API key	From date, to date	appID > date > metric
Ratings/Reviews per app	API key, App ID	From date, to date	date > metric
Ratings/Reviews total	API key	From date, to date	appID > date > metric
Rankings per app	API key, App ID	From date, to date	date > category name > metric
Rankings total	API key	From date, to date	appID > date > category name > metric

The above schematic is a first version of the API. It is likely to be developed further at the company at a later time. Optional parameters such as platform, app version, or country are likely additions. Revenue and other metrics would also be useful.

A table in the database schema was created for logging purposes, storing queries, user (via a lookup of the API key) and request timestamp – also the IP address from which the request was made, by the PHP system variable $_SERVER['REMOTE_ADDR']. Such logs will be useful in the unlikely event of abuse.

A sample API request is shown in Figure 10-19, with the request URI and the resulting JSON response output. Note that faulty API requests produce a JSON compliant error message response and for security purposes are logged in the system logs (including requesting IP address) under a separate error category. For an actual code sample from the API function, see the Appendix A of this report.


localhost/app/api.php?apikey=asdadf&metric=ratings

```
{
  - [redacted] {
        Average: "1.568627",
        Number: "153"
    },
  - [redacted] {
        Average: "1.750000",
        Number: "8"
    },
  - [redacted] android: {
        Average: "1.235294",
        Number: "17"
    }
}
```

Figure 10-19 – Internal API request

10.4 Milestone progress evaluation

Comparing the implementation maturity against the primary and secondary goals listed in sections 9.2.2 and 9.2.3, we find that while the focus and priority has been correct (emphasis on the primary goals deliverables), much progress has been made on the secondary goals as well. However, statistical analysis had been hard to perform until more data was imported. This is probably one area that would benefit from more attention in the future – refer to chapter 11 for more details on the analysis. In table 10 is a brief summary on development progress, compared to the stated primary and secondary goals.

Table 10 – Milestone progress of primary and secondary goals

Goal	Current status	Remaining
Primary goal 1: First and foremost, available app data from third-party tools should, whenever possible, be transformed into a uniform format, and (automatically) imported into a local database. This corresponds to the ETL phase. Additionally, manual entries must be possible since some customers of the app publisher may not want to disclose all of the data that they might want to include in their copy of the database. Hence it should be possible to add some data manually.	- Downloads, user ratings/reviews and rankings are automatically fetched from Distimo's API as JSON and imported into local database tables. - Apple's App Store data are also scheduled and auto-imported from iTunes Connect. - Manual imports of CSV-files from Android export data is working, but not yet for *all* dimensions (such as data broken down by country).	- Add support for all import file types for Android - Add support for iOS manual import

Primary goal 2: This database should be accompanied by a front-end offering some visual representation of this data, at the very least in tables; but preferably also as diagrams or charts.	A front-end is designed and data displayed in both tables and charts.	Improve designs
Primary goal 3: This front-end should be made available for external viewers so that visualization could be done on a mobile device. This was an especially desired feature for the company at which this thesis project is undertaken, but which technique to use remains open, i.e., it could be a native app or mobile web site.	The technique was designed to be accessible as a mobile web using a responsive CSS template. It is presentable in both full-sized web version and mobile edition. This goal is on the right track, but still needs final testing after the end of the master's thesis project.	Acceptance testing – preferably in multiple phases; internal, beta and public release. Internal testing has begun at the time of this publication, but shall continue for some period of time.
Secondary goal 1: The front-end should not only connect directly to the database, but also offer an API that will allow user access control management, so that third-parties may access the data of interest to them, without having full system access.	The API has been built and is working, generated JSON responses verified with JSONlint [111]. – Database logging of all API requests is in place – User management is in place and working (i.e. child accounts with read-only rights)	
Secondary goal 2: Some type of primitive SMM tool for mapping app reviews into category buckets using keyword lists would be useful, if review texts are available.	– Review comment texts are imported automatically using Distimo API for most apps – Users can create filters (tags) that are automatically applied to reviews. Frequencies and percentages of these labels are calculated and displayed. – Trending review keywords are drawn in a *word cloud* and listed in a table.	These filters should be incorporated into the future development of Secondary goal 3, refer to the table row below.

Secondary goal 3: Alarm or trigger system. For instance, if the overall rating for App A drops below 2 out of 5, then automatically send an email to the relevant person (or persons) to encourage investigation.	– Not implemented as of yet. Such a control script could be scheduled to run just after the last of the daily imports is finished. – The PEAR::Mail component in PHP [98] seems suitable for sending out the alert mails via any SMTP mail account. The component is already used for sending out report emails.	Not prioritized, but a likely future addition
Secondary goal 4: Automatic or semi-automatic monthly reports could be generated based on the trends found. Such reports are already sent to many customers of the app company, but currently require everything to be manually compiled and written.	Users may generate reports, either in HTML format for direct viewing in the browser, exported to PDF using wkhtmltopdf [108], or as PDF attachments sent via email using PEAR::Mail [98].	Users should be able to set up rules for regularly automated reports being sent out via email.

In addition to the primary and secondary goals, some comments regarding the implementation are:

- I have a maximum of 16 apps (11 for iOS, 5 for Android) to gather statistics for. These are a select subset of the company's app catalogue, based on data sensitivity. Since most of the Android apps have also been released in iOS versions, this data should in my opinion not skew the results too much.
- The implementation is somewhat dependant on Distimo's third-party API. However, the back-end is quite flexible so that another data source could easily append its data to the same database tables for interchangeability. An approach for the future might be to include Google Analytics SDK into every app (iOS and Android) just to get access to a better first-party API for data. If this is not feasible, another option is to investigate the upcoming API from App Annie (at the time of this publication in private invitation-only beta phase) so as to collect data from multiple sources.
- An aggregated view of all available data on a per-app-basis was designed. This summary is the foundation of the reports view (also enabling the PDF report generation, requiring only a defined time period of what the report should cover).
- Many additional modifications of the current code are necessary to make everything multi-platform compatible. For instance, as Apple's App Store data includes data about app pricing whereas Google Play does not, revenue cannot be calculated automatically even from a manual import of Google Play data. Third-party API's such as Distimo may have to be used for this purpose as well.

- For internal versioning of the app stats web platform, I will follow the Semantic Versioning 2.0.0-rc.1 guidelines (formed by Tom Preston-Werner, one of the co-founders of GitHub), available from [112]. The format is <major>.<minor>.<patch> where each field's increase resets the fields to the right to zero. For instance, a bugfix patch release would increase version number 1.0.0 to 1.0.1, and the following version with new features would be either 1.1.0 or 2.0.0 depending on the size of the update.

11 Analysis

Using the manually and automatically imported app data, we performed some statistical analysis. One approach is to tailor specific queries that return very specific data – in this scenario we might write SQL queries that are executed directly on the back-end database. Another method is to use the capabilities of mathematical software such as R [115] and its many add-on packages. The latter also facilitates drawing statistical findings in charts and diagrams.

11.1 SQL query analysis

An initial manual analysis phase might be done via writing SQL queries to execute on the local database, as it now contains substantial amounts of imported raw data. A few example queries, their use cases, and results (in some cases slightly censored) are given below.

Which weekday is the most successful in terms of downloads?
 (i.e. which day might be optimal for an upcoming app launch?)

```
SELECT
DATE_FORMAT(statdate, '%W') as weekday,
sum(`value`) as downloads
FROM `app`.`downloads `
GROUP BY weekday
ORDER BY downloads DESC
```

Weekday, number of downloads

```
'Sunday', xx061
'Saturday', xx773
'Friday', xx153
'Thursday', xx863
'Tuesday', xx599
'Monday', x472
'Wednesday', x276
```

Which platform is most successful in terms of average ratings, Android or iOS?
(i.e. which platform should we focus on for the future, or which platform needs improvement?)

```
SELECT
case
   WHEN appid IN (SELECT appid FROM `app`.`apps` where platform = 'Google Play') then 'Google Play'
   WHEN appid IN (SELECT appid FROM `app`.`apps` where platform = 'App Store') then 'App Store'
   else "Unknown platform"
end
as platform,
avg(rating) as avg_rating,
count(rating) as no_ratings
FROM `app`.`comments`
GROUP BY platform
```

Platform, average rating, # ratings

```
'App Store', x.x81586, x91
'Google Play', x.x58065, x1
```

Is there any correlation between the number of app versions and the app's average rating?
(i.e. does it pay to keep releasing new versions and fixing bugs / adding features?)

```
SELECT application_name,
  AVG(rating) as avg_rating,
  COUNT(rating) as no_ratings,
  COUNT(DISTINCT version) as no_versions
FROM `app`.`comments`
GROUP BY appid
ORDER BY avg_rating DESC
```

App, avg. rating, # ratings, # ver

```
'App A', x.285714, 7, 2
'App B', x.500000, 12, 3
'App C', x.722222, 36, 6
'App D', x.700935, 107, 9
'App E', x.448276, 29, 8
'App F', x.333333, 3, 1
'App G', x.750000, 8, 2
'App H', x.722222, 36, 6
'App I', x.568627, 153, 4
'App J', x.285714, 14, 0
'App K', x.250000, 16, 0
'App L', x.000000, 1, 0
```

The output from this last query was imported into OpenOffice Calc (and adjusted for those apps with “0” app versions). By means of the correl() function mentioned in section 8.2, we find a positive Pearson correlation of 0.53 between the number of app versions and the number of reviews (not surprisingly); but what is interesting is a positive correlation of 0.25 was also found between the number of app versions and a higher average rating. It seems as if most of the time, app updates reflect positively on the aggregated rating.

Even though this type of query based analysis should not be underestimated for future use, the idea is that those questions that we might want to track regularly should be displayed in the web GUI through tables and charts, both for convenience and to easily include the data into reports. An intermediate step to this effect is to draw and plot findings manually at first.

11.2 Statistical analysis

Apps may be developed for vastly different purposes. Within the app portfolio made available to me, there are apps that are intended for use on one single day per year (for a particular event), as well as social apps, intended for daily use. Most apps are either completely free of charge or a one-off purchase, but one app uses a subscription scheme with monthly or quarterly renewal, and another app uses premium add-on charges for SMS notification services.

Extracting some key metrics for each app available to me, I generated a data set that would be suitable for statistical computations. The table includes for each app its smartphone platform, number of downloads and updates, the number of app versions, number of ratings and the aggregated average rating. I have also applied *ad-hoc* labels indicating whether the app would be considered either ‘promotional/event based’ or ‘commercial/professional’, and either ‘free’ or ‘paid (any business model)’. Revenue would have been an interesting addition under other circumstances, but too little data was available (shared from clients and collected from app stores) to make it meaningful in this particular case. Table 11 demonstrates the data set, although most sensitive data has been removed or modified for anonymity, instead indicating the figure sizes in digits with the respective number of letters ‘x’.

Table 11 – Compiled data and metrics for in-house apps

AppID	Platform	no_DL	no_UPD	no_ver	no_ratings	avg_rating	pr/com	free/paid
1	App Store	xxx	x	1	x	x,333333	pr	free
2	App Store	xx	x	2	x	x,285714	pr	free
3	App Store	xxxx	xxxxx	6	xx	x,722222	pr	free
4	App Store	xxx	x	8	xx	x,448276	com	
5	App Store	xxxx	xx	9	xxx	x,700935	com	paid
6	App Store	xxx	X	3	xx	x,5	pr	free
7	App Store	xxxx	xxxxx	8	xx	x,632653	com	free
8	App Store	xxxxx	xxxxx	4	xxx	x,568627	com	paid
9	App Store	xxx	xx	2	x		pr	free
10	App Store	xxxx	xxx	2	x	x,75	com	paid
11	App Store	xxx	xxx	3	x		com	free
12	Google Play	xxxx	xxxx	7	xx	x,266667	com	free
13	Google Play	xx	x	3	x		com	free
14	Google Play	xxxx	xxxx	6	x	x	pr	free
15	Google Play	xxxxx	xxxx	5	xx	x,235294	com	paid
16	Google Play	xxx	xxx	2	x		com	paid

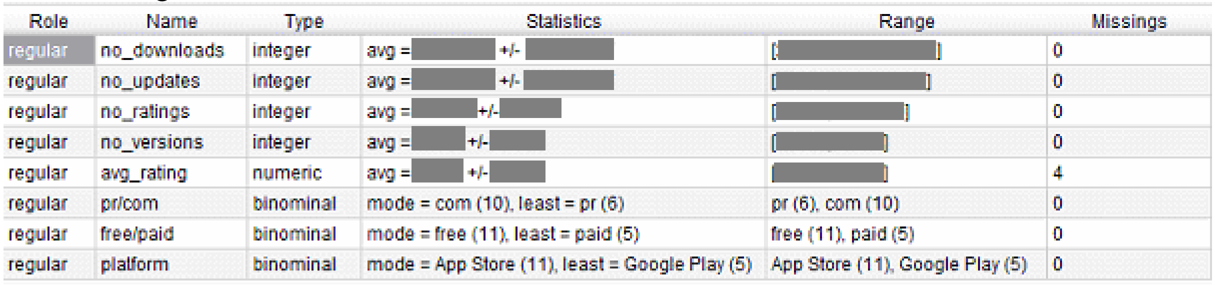
The overview data table (as shown above) was compiled using a handful of SQL queries, and their respective outputs were manually compiled into a spreadsheet table. A more programmatic approach would have been to write one single huge SQL query, possibly using SQL commands such as LEFT JOIN [105] and UNION [106], creating either result sets or virtual tables containing each type of metric, per app, per date. If this analysis is to be expanded in the future, the data set compilation should be automated.

There are roughly three steps (whether manual or automated) to consider before the analysis, and they correspond closely with the three ETL phases. First, we import the data – for example from CSV files containing raw data. Secondly we need to prepare the data, including removing data that is not sufficiently detailed. This can be done by applying the R function na.omit() which simply cuts off entries where data is not available (N/A), such as app numbers 9,11, 13, and 16 in this case. Before plotting data, a realignment of the scale may be a good idea. The third and final step is to apply the statistical method upon the cleaned dataset.

Caveat: the manually applied labels for free/paid and PR/commercial are omitted in most analysis methods. The reason is that they are *nominal* text values with unmeasurable distances between them, thus they are neither *ordinal* or *interval* values. The scale for levels of measurement has been further described by S. S. Stevens as early as 1946 in [120].

There are ways to include nominal values into the analysis, such as assigning each label to a dummy numerical value (called Filmer and Prichet's method), or using polychoric correlation to estimate ordinal variables as suggested by Kolenikov and Angeles, the authors of [118]. However, these transformations fall outside the scope of PCA. That also means, regrettably, that I cannot apply k-means cluster apps based upon the type of smartphone platform (i.e., Android or iOS). A binominal value could have been assigned, but this approach would not work for more platforms than two.

The software RapidMiner [81] provides for a nice overview over the dataset and some core statistics, available directly after data import. An example of output from this software is shown in Figure 11-1.



Role	Name	Type	Statistics	Range	Missings
regular	no_downloads	integer	avg = [redacted] +/- [redacted]	[[redacted]]	0
regular	no_updates	integer	avg = [redacted] +/- [redacted]	[[redacted]]	0
regular	no_ratings	integer	avg = [redacted] +/- [redacted]	[[redacted]]	0
regular	no_versions	integer	avg = [redacted] +/- [redacted]	[[redacted]]	0
regular	avg_rating	numeric	avg = [redacted] +/- [redacted]	[[redacted]]	4
regular	pr/com	binominal	mode = com (10), least = pr (6)	pr (6), com (10)	0
regular	free/paid	binominal	mode = free (11), least = paid (5)	free (11), paid (5)	0
regular	platform	binominal	mode = App Store (11), least = Google Play (5)	App Store (11), Google Play (5)	0

Figure 11-1 – Metadata in RapidMiner

11.2.1 Pearson correlations

Instinctively, the first thing we might want to find out is which metrics are correlated (or inversely correlated). Manual calculation of this is tedious – a much better way is to use R [115], mentioned in section 8.7.

The R function cor() produces a correlation matrix between all components as follows:

```
> tabell <- read.csv ("csvfile.csv", header=TRUE)
> cormatrix <- cor(tabell, use="complete.obs", method="pearson")
> cormatrix
             no_downloads no_updates  no_ratings no_versions  avg_rating
no_downloads   1.00000000  0.1818684  0.73463351 -0.08399037 -0.31796936
```

```
no_updates      0.18186844  1.0000000  0.26792291  0.36452805 -0.21879013
no_ratings      0.73463351  0.2679229  1.00000000  0.33140755 -0.09957761
no_versions    -0.08399037  0.3645280  0.33140755  1.00000000 -0.30510898
avg_rating     -0.31796936 -0.2187901 -0.09957761 -0.30510898  1.00000000
```

As the matrix is symmetric along the diagonal, and each component obviously is perfectly correlated to itself, we only need to read values either above or below the diagonal. We see that the number of app versions is correlated to both the number of downloads and the number of updates, and that the number of ratings correlates strongly with downloads as well. In the case of the negative correlation between avg_rating and no_downloads/no_updates, the explanation is probably skewed data, as one particular, very popular app had back-end server connection issues that lowered its rating significantly.

The above matrix may be illustrated in a *correlogram,* indicating correlation using colors, symbols, texts or shapes. The below Figure 11-2 is one such correlogram using the same dataset. The darker the shade of blue, the stronger correlation; while the darker the shade of red, the stronger the inverse correlation.

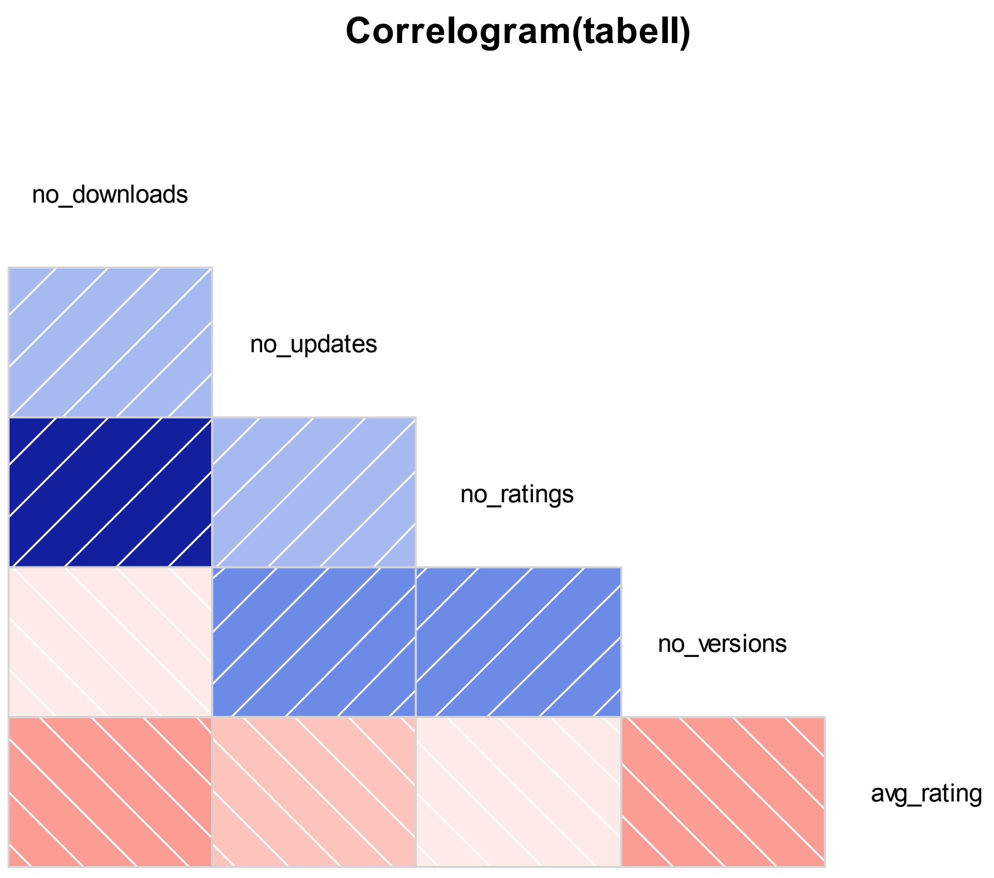


Figure 11-2 – Correlogram for components

11.2.2 k-means clustering

Considering the data for all available apps, could there be some hidden categories or groups of apps based on these or other variables? This can be learned by applying the k-means clustering as described in section 8.6, using the following sample code snippets from [116].

```
> tabell <- read.csv ("csvfile.csv", header=TRUE)
> tabell <- na.omit(tabell)
> tabell <- scale(tabell)
> fit <- kmeans(tabell, 4)
> tabell <- data.frame(tabell, fit$cluster)
tabell
   no_downloads no_updates   no_ratings no_versions avg_rating fit.cluster
1   -0.43384912 -0.5457723 -0.711521399  -1.5244978  0.1308371           4
2   -0.50366831 -0.5453849 -0.626351755  -1.1511514  2.1004928           4
3    0.01008318  1.0034960 -0.008871838   0.3422342  0.5231670           3
4   -0.48535508 -0.5454624 -0.157918714   1.0889270  0.2467971           1
5   -0.39893953 -0.5437576  1.502889338   1.4622734  0.5016916           1
6   -0.46753239 -0.5457723 -0.519889700  -0.7778050  1.3078268           4
7   -0.24875470  2.7770995  0.267929504   1.0889270 -0.5760426           3
8    2.97445543  0.4315384  2.482340241  -0.4044586 -0.6406351           2
10  -0.41684398 -0.5023001 -0.605059344  -1.1511514 -0.4576573           4
12  -0.36705489 -0.1793196 -0.456012468   0.7155806 -0.9452668           1
14  -0.41005828 -0.4107063 -0.754106221   0.3422342 -1.2142933           1
15   0.74751768 -0.3936584 -0.413427646  -0.0311122 -0.9769174           1
```

As can bee seen in the output above, the appended rightmost column indicates which cluster each app belongs to. All numbers have at this stage been rescaled for fitting purposes and should no longer be read literally. Using the clusplot() function in the R package cluster, we may show all apps along the axes of the two primary components (the following section will describe the principal component analysis, PCA, in more detail).

```
> library(cluster)
> clusplot(tabell, fit$cluster, color=TRUE, shade=TRUE, labels=2, lines=1)
>
```

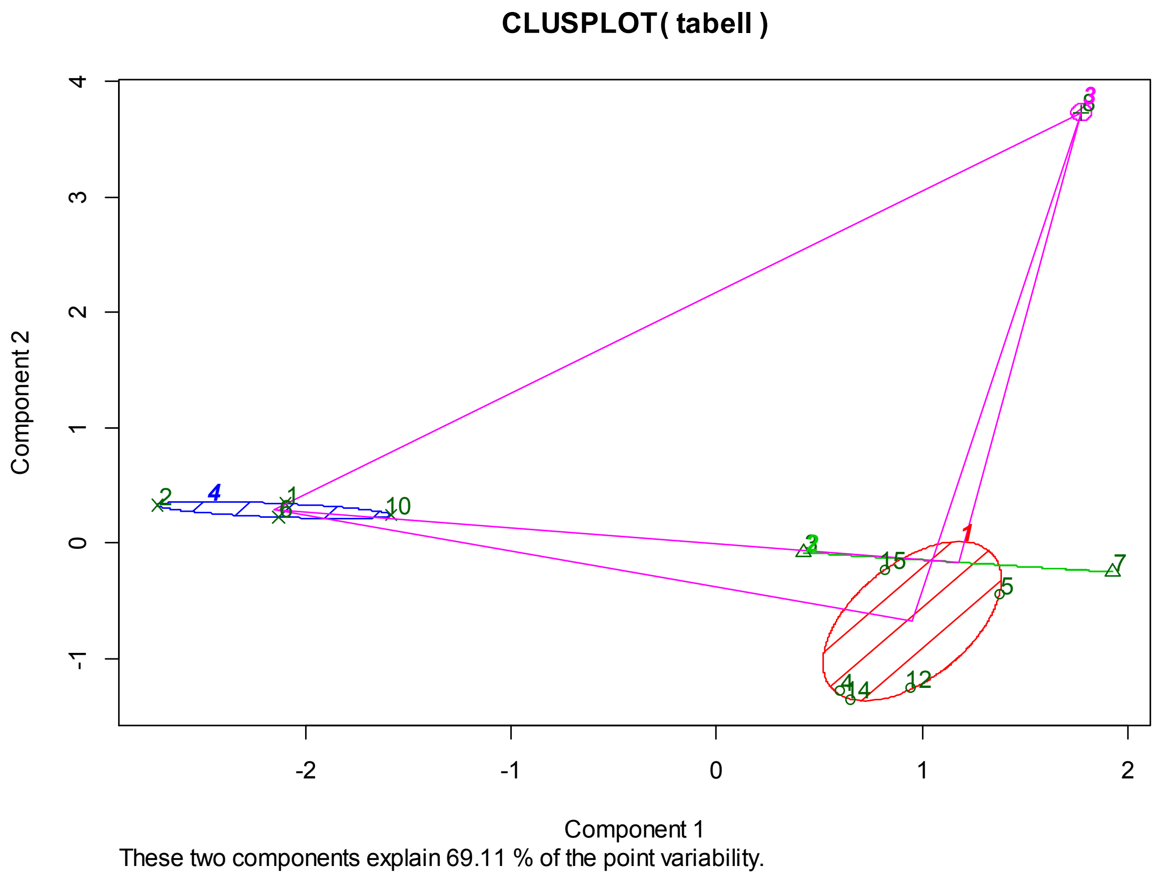


Figure 11-3 – Clusplot in R: Four app clusters

As for the app data, component 1 is the number of downloads and component 2 is the number of updates. We can identify our four (4) clusters in Figure 11-3, printed in different symbols and colouring. But perhaps three (3) clusters would have sufficed seeing as the centre points of the green and red clusters are relatively close. An alternate clusplot using three clusters is shown in Figure 11-4.

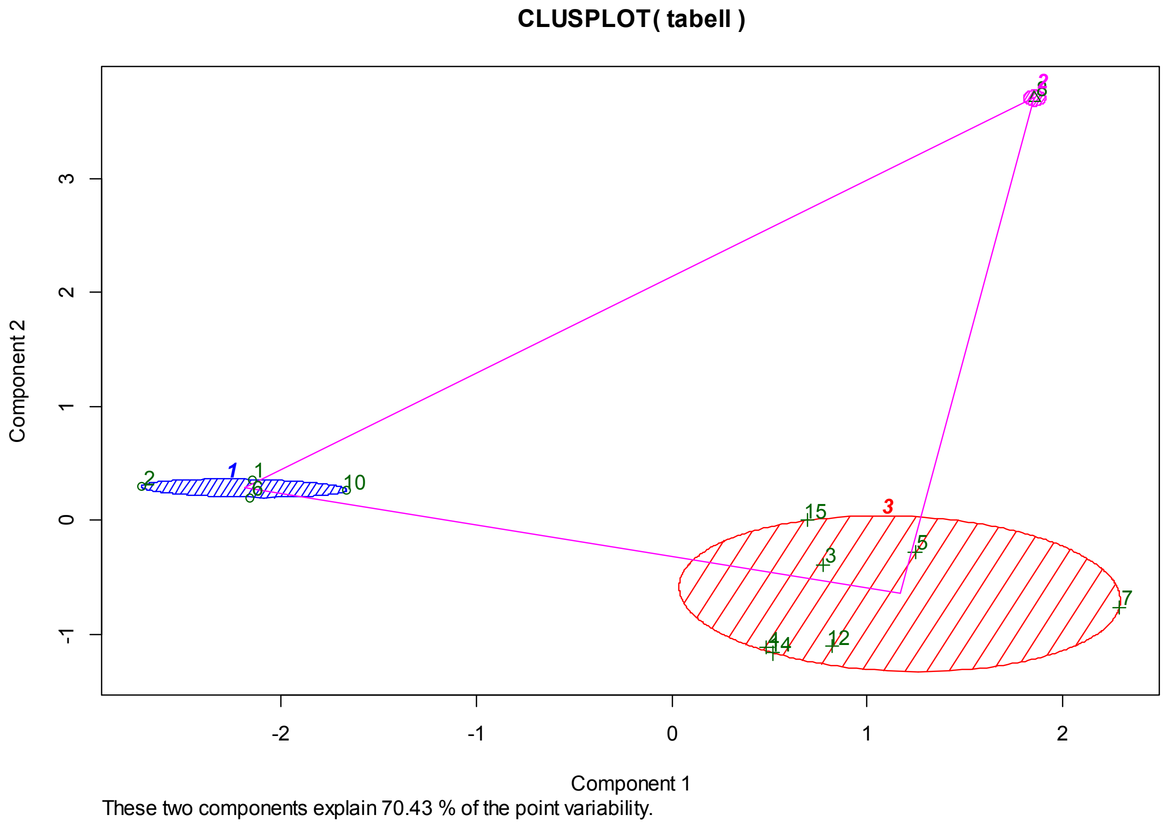


Figure 11-4 – Clusplot in R: Three app clusters

This is easier for a human to interpret and to use for labelling the apps into categories.

Roughly, one might label the clusters as following:

1. (Blue, Left) – Low-volume apps, both in terms of downloads and updates
   App(s): 1, 2, 6, 10
2. (Purple, Top right) – Extreme *outlier* in terms of many downloads
   App(s): 8
3. (Red, Bottom right) – “Normal” app group
   App(s): 3, 4, 5, 7, 12, 14, 15

An observation is of course that cluster 2 is an extreme outlier. There are many tests for outliers; one is Grubb’s test (available in the ‘outliers’ R package [124]) which on the α = 0.01 significance level guarantees that app number 8 is an extreme outlier in terms of downloads. However, Grubb’s test requires normal distribution, and as the data better fits a long-tailed distribution, we can not trust the test completely but simply use the result as an indicator.

We might also want to draw the cluster memberships in a hierarchical agglomerate structure, such as a dendrogram which was described in section 8.6.2. Comparing the dendrogram in Figure 11-5 with the cluster plot, we identify roughly the same classifications, only illustrated in a different way.

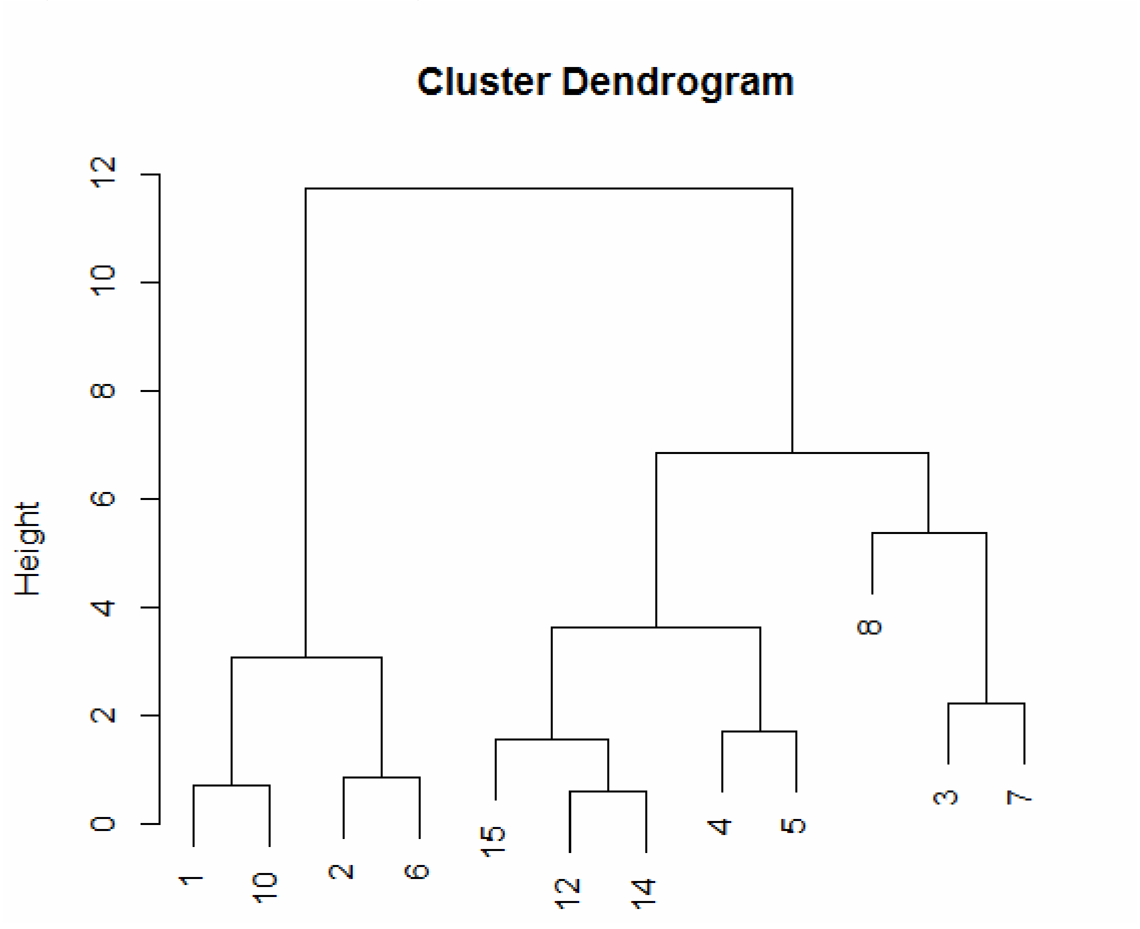


Figure 11-5 – Cluster dendrogram in R

11.2.3 Principal component analysis (PCA)

The above clustering and its plots only display the two primary components (number of downloads and updates) along the x and y axes. It is often interesting when doing principal component analysis (PCA) to identify these components, and order them by variance impact. This can be done programmatically in R as follows, using the princomp() and summary() functions:

```
> tabell <- read.csv("csvfile.csv", header=TRUE)
> tabell <- na.omit(tabell)
> fit <- princomp(tabell, cor=TRUE, na.action = na.exclude)
> summary(fit)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp.5
Standard deviation     1.4590274 1.1174517 0.9297381 0.8173957 0.29998694
Proportion of Variance 0.4257522 0.2497396 0.1728826 0.1336271 0.01799843
Cumulative Proportion  0.4257522 0.6754918 0.8483744 0.9820016 1.00000000
```

The cumulative proportion of variance can be plotted, as shown in Figure 11-6, by:

```
> plot(fit,type="lines")
```

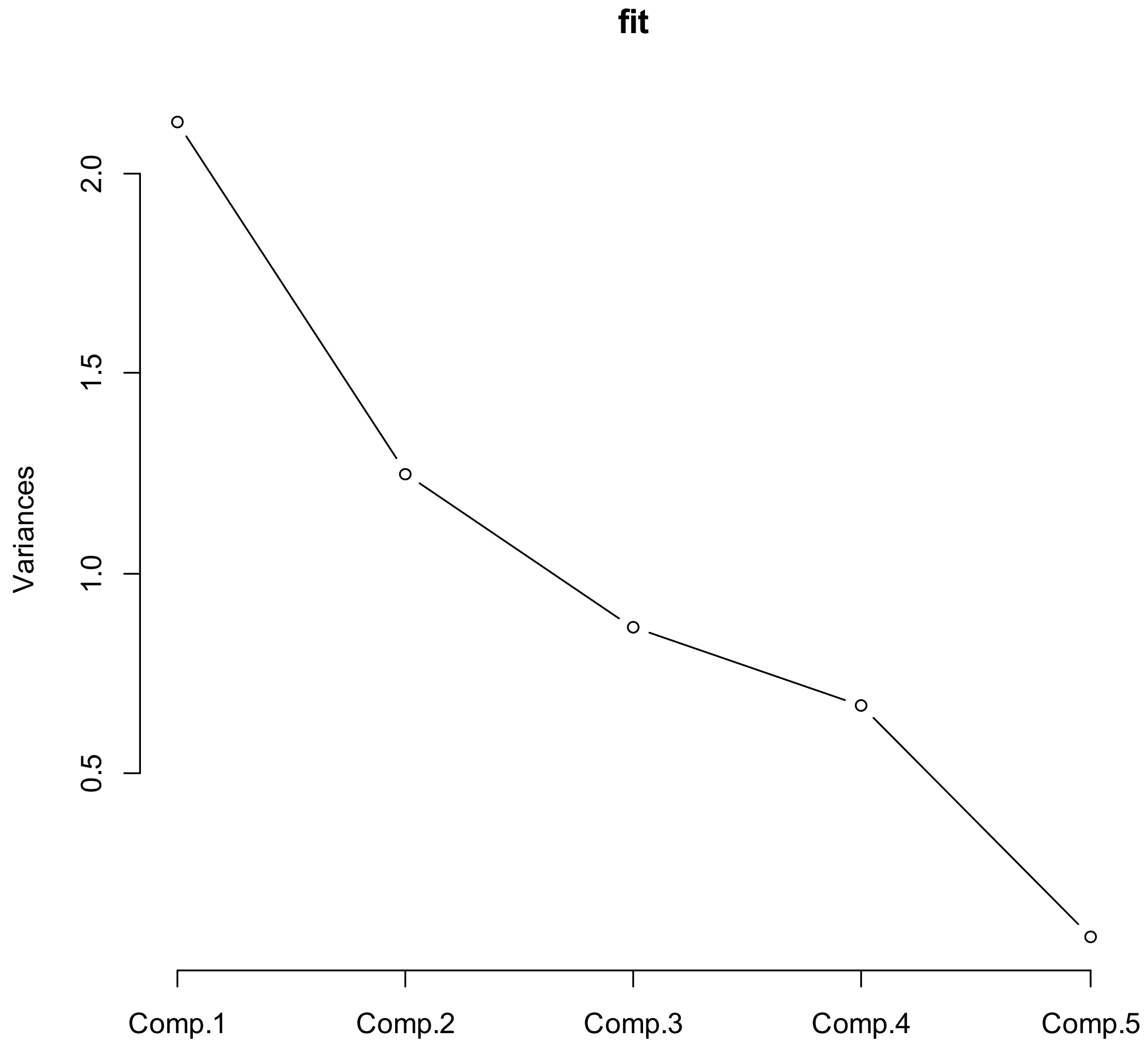


Figure 11-6 – Aggregate variance by component

It is also possible to render the PCA in a biplot, such as Figure 11-7, a double-purpose plot type designed by Gabriel in 1971 [117]. It plots both data entries and the components in the same plot, also indicating how components interwork. In R, it is invoked as follows:

```
> biplot(fit)
```

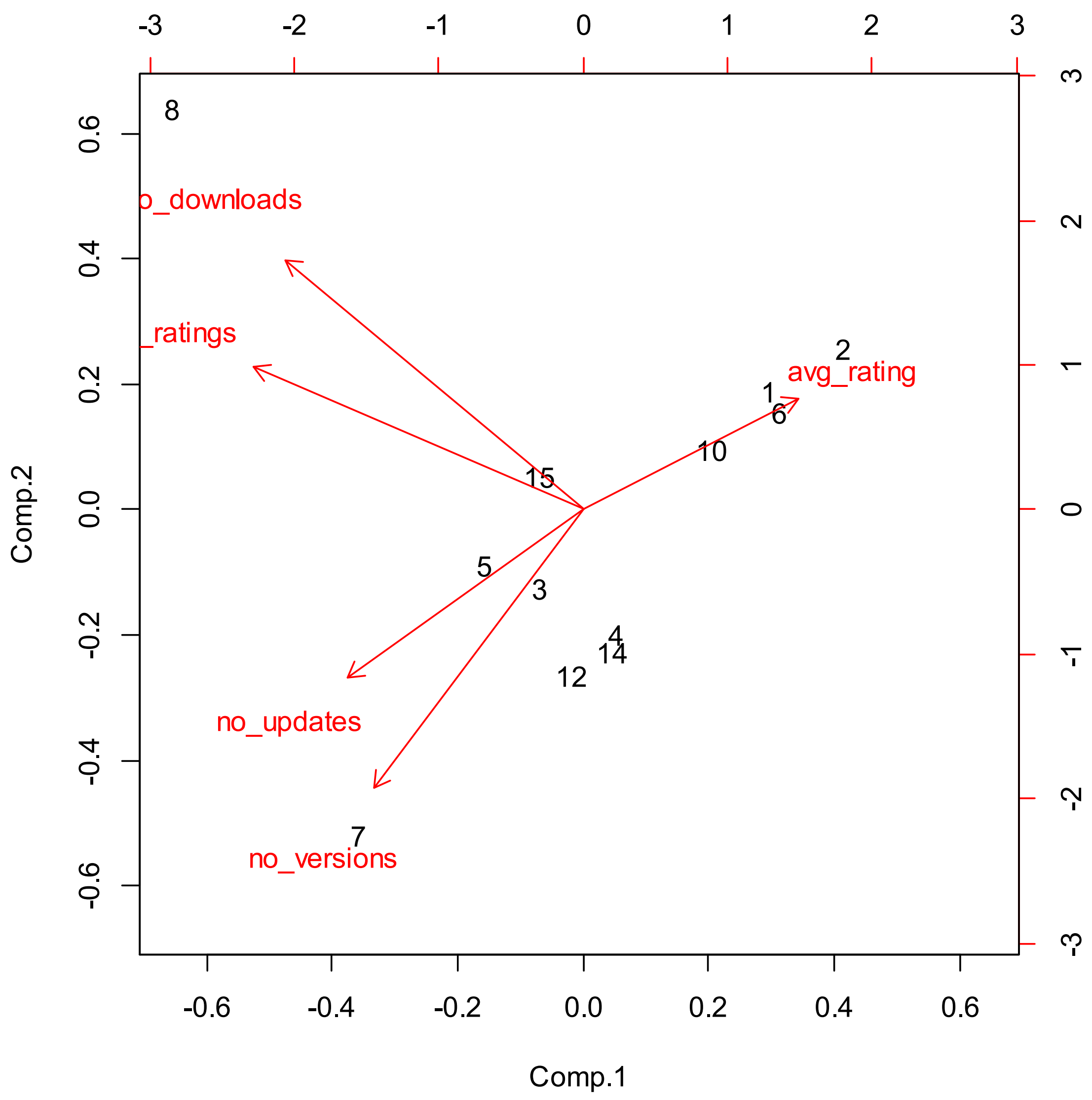


Figure 11-7 – Biplot over components and observations

As for interpretation of a biplot, some guidelines are described in [119]:

- *The cosine of the angle between a vector and an axis indicates the importance of the contribution of the corresponding variable to the axis' dimension.*
- *The cosine of the angle between vectors indicates correlation between variables. Highly correlated variables point in the same direction; uncorrelated variables are at right angles to each other.*
- *Points that are close to each other in the biplot represent observations with similar values.*
- *You can approximate the coordinates of an observation by projecting the point onto the variable vectors within the biplot.*

We can, for instance, again see that the number of ratings follows the number of downloads (quite logically) as the angle between their vectors is small. Similarly, the number of updates follows the number of released app versions which is also instinctively true. Using the axes and data points, projection onto the axes of two primary components displayed approximates their variables. As an example, app number 8 has attracted a relatively high number of both downloads and updates, while app number 7 has fewer downloads than updates. App number 2 is on the other hand very light-weight both in terms of downloads and updates. Also note how the biplot follows the correlation matrix from section 11.2.1.

11.3 Analysis results

Which trends and findings can we extract from all of the above? Is the data reliable? Is it in line with what others have found? And can the theoretical models described in this thesis report be applied to our dataset? This chapter breaks down the key discoveries of our statistical analysis.

11.3.1 Findings

First off, we may describe the variance impact per component. In our sample, the number of downloads account for about 43% of the variance, and the number of updates for an additional 25%. The two first principal components, then, account for an aggregated 68% of variance, which means they are the two metrics that vary the most.

As for distribution, an interesting finding is that Pareto's 80-20 rule (law of the vital few), described in section 8.4, is relevant for our dataset. Out of the 16 apps used in the analysis, 3 of them (= 18.75%) account for 85% of the downloads, and similarly, 3 of the 16 apps account for 68% of the reviews.

As for an approximate distribution of downloads, using regression analysis in OpenOffice Calc, we find a function with 0.91 in R^2 value (goodness of fit), which should be considered to be very accurate. The long-tailed distribution is present also in this dataset, as can be seen in Figure 11-8.

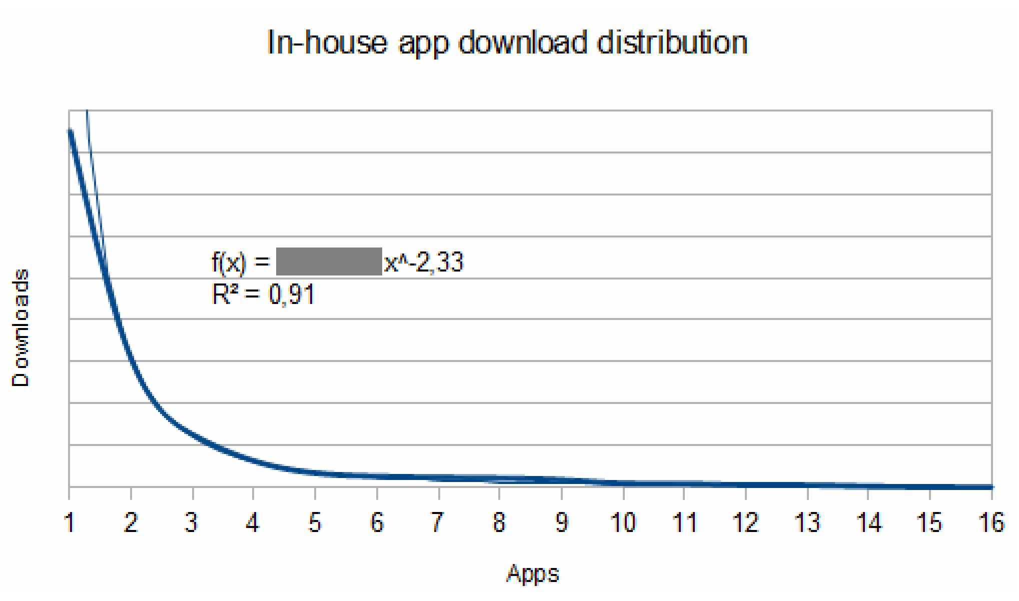


Figure 11-8 – In-house app download distribution and regression analysis model

Another application somewhat relevant to Pareto's 80-20 rule is the rating distribution. It seems as if most app users would either rate an app as 'very good' (5/5) or 'very bad' (1/5). Basically, a user does not generally make an effort to review an app that was perceived to be average – hence strong satisfaction or dissatisfaction is required to motivate the user to write a review.

The rating distribution in our dataset had an aggregated share of 'either 1/5 or 5/5' of about 79%. The distribution per app is displayed below in Figure 11-9 as a bar chart:

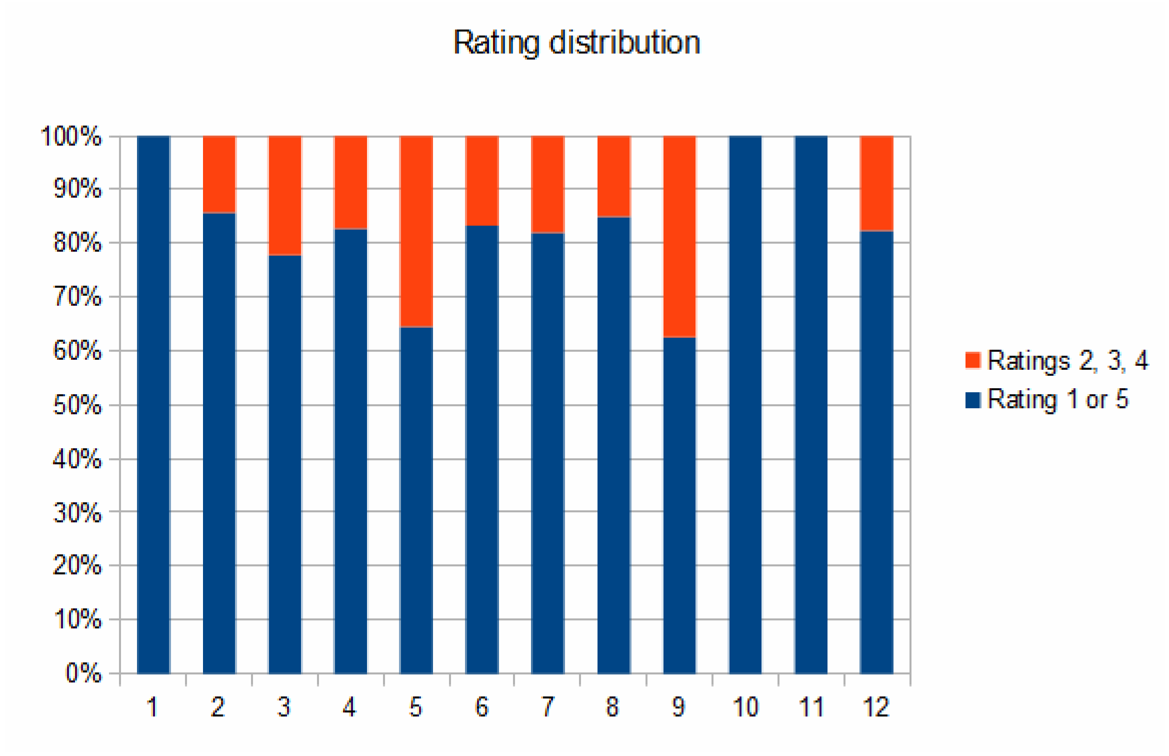


Figure 11-9 – Bar chart over rating distribution: 1 or 5 versus 2, 3 or 4

In further analyzing the ratings, there is a strong correlation between the average rating of an app, and the percentage of users who have rated it. On average, each app in the dataset was rated by 2% of its downloaders. For the top rated apps this figure was in two instances over 10%. These results gave a Pearson correlation of 0.68 between the percentage of reviewers and the average rating – although it should be stated that some calculations rely on very few observations. Regardless, this correlation shows the importance of encouraging power-users of an app to actually write a review. This goes especially for paid apps – even though only 5 out of 16 apps were 'paid', they accounted for 65% of total reviews.

Analyzing actual review content, I used my web platform to filter out the trending keywords from all reviews. Again, the keywords form a long-tail like distribution, which could be very close estimated by a power distribution function ($R^2 > 0.98$). This long-tail is apparent in Figure 11-10.

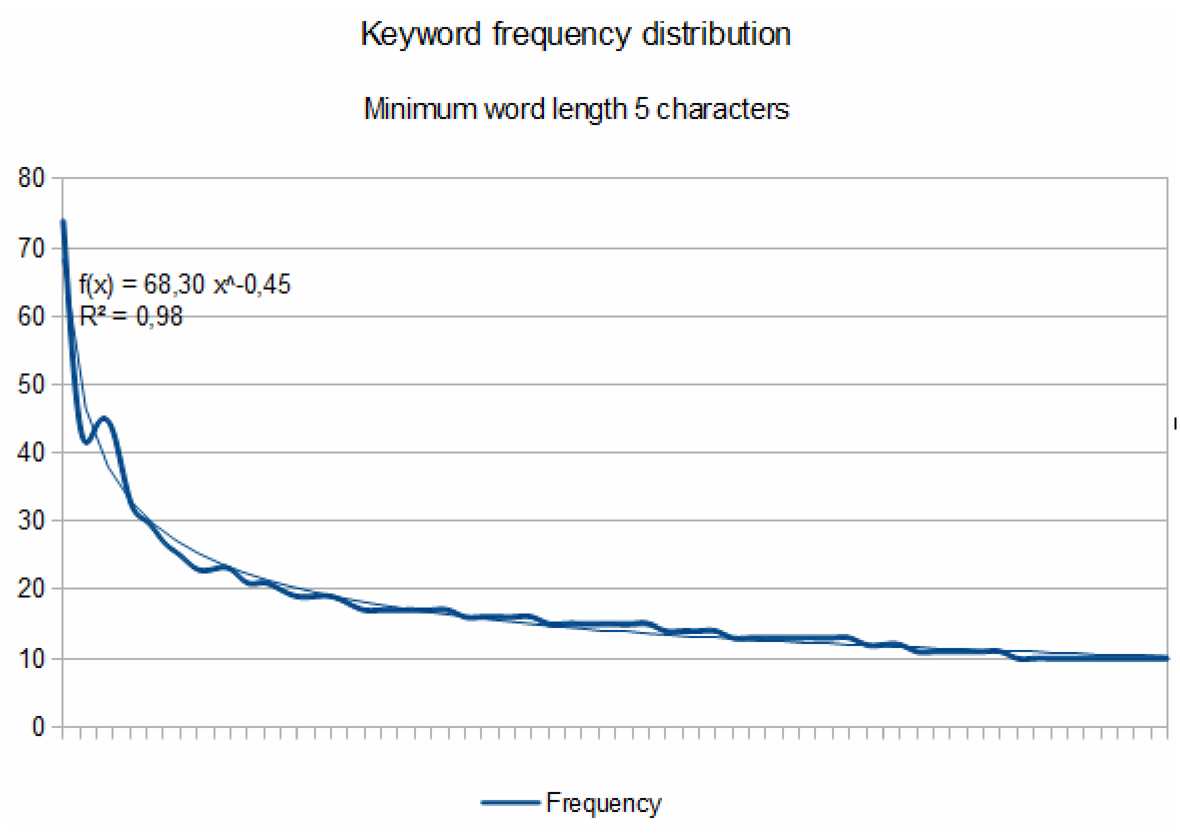


Figure 11-10 – Keyword frequency distribution (words common in reviews)

As previously mentioned, many of these words were adjectives, names, or other words of little significance, which is why the words should first be filtered against a compiled list of common *stop words* before put this analysis into real-world use. Another method is to just filter out nouns. Regardless, it is still conceptually very interesting that most reviewers seem to be discussing the same issues or features. This suggests that correctly aligned updates can heavily increase the customer's satisfaction. Hopefully the keyword mining tool will be of assistance in this pursuit.

The nominal values (app platform) and labels applied by me onto the app set could not be used in computational statistical methods mentioned in this chapter. However, using RapidMiner [81] they could be plotted for a visual overview. For instance, in Figure 11-11 (with logarithmic x axis for readability), we see download figures indicated with their type (free/paid) and platform. A general summarization might be that the most downloaded apps are paid, and that iOS apps attract more downloads than those on Android.

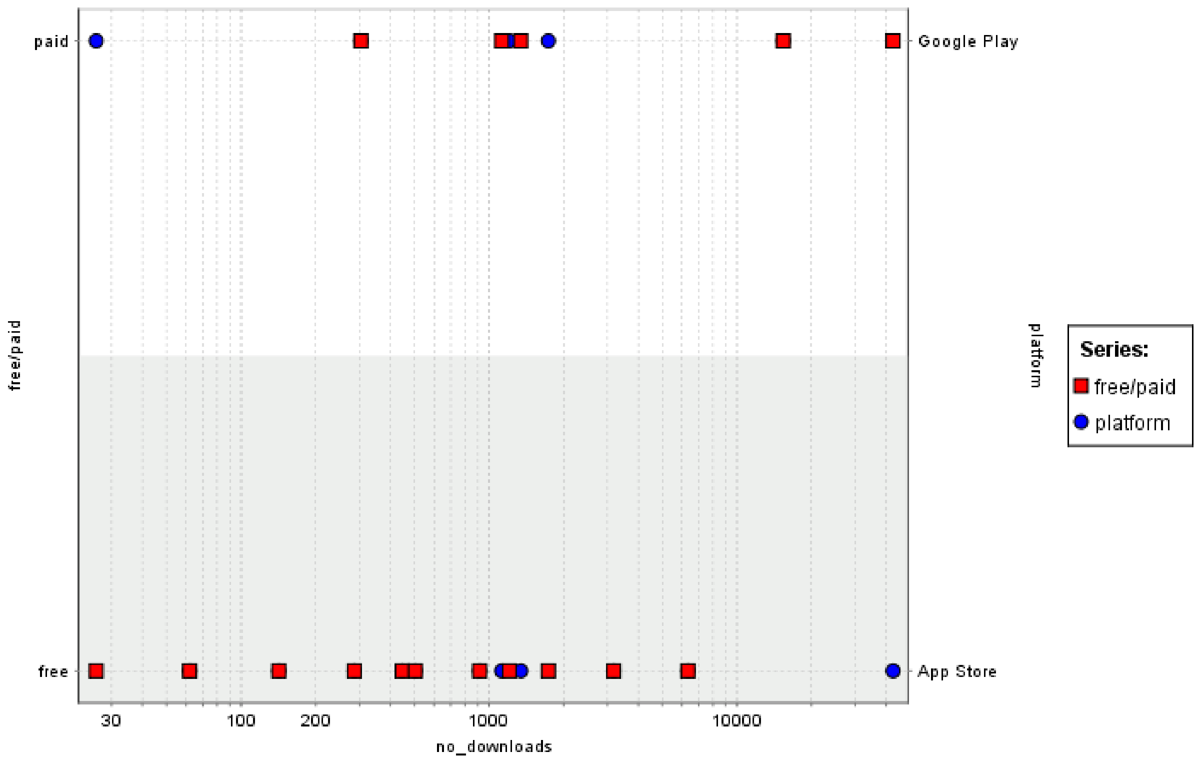


Figure 11-11 – Dual-axis binominal chart: free/paid and Android/iOS apps

In fact, RapidMiner provides for a very useful 'scatter matrix' that plots all components pairwise against each other. This gives a top-down view with just a few clicks. As we can see in the cropped-out excerpt shown in Figure 11-12, obviously below the diagonal all charts are simply mirrored in that the two axes have switched places.

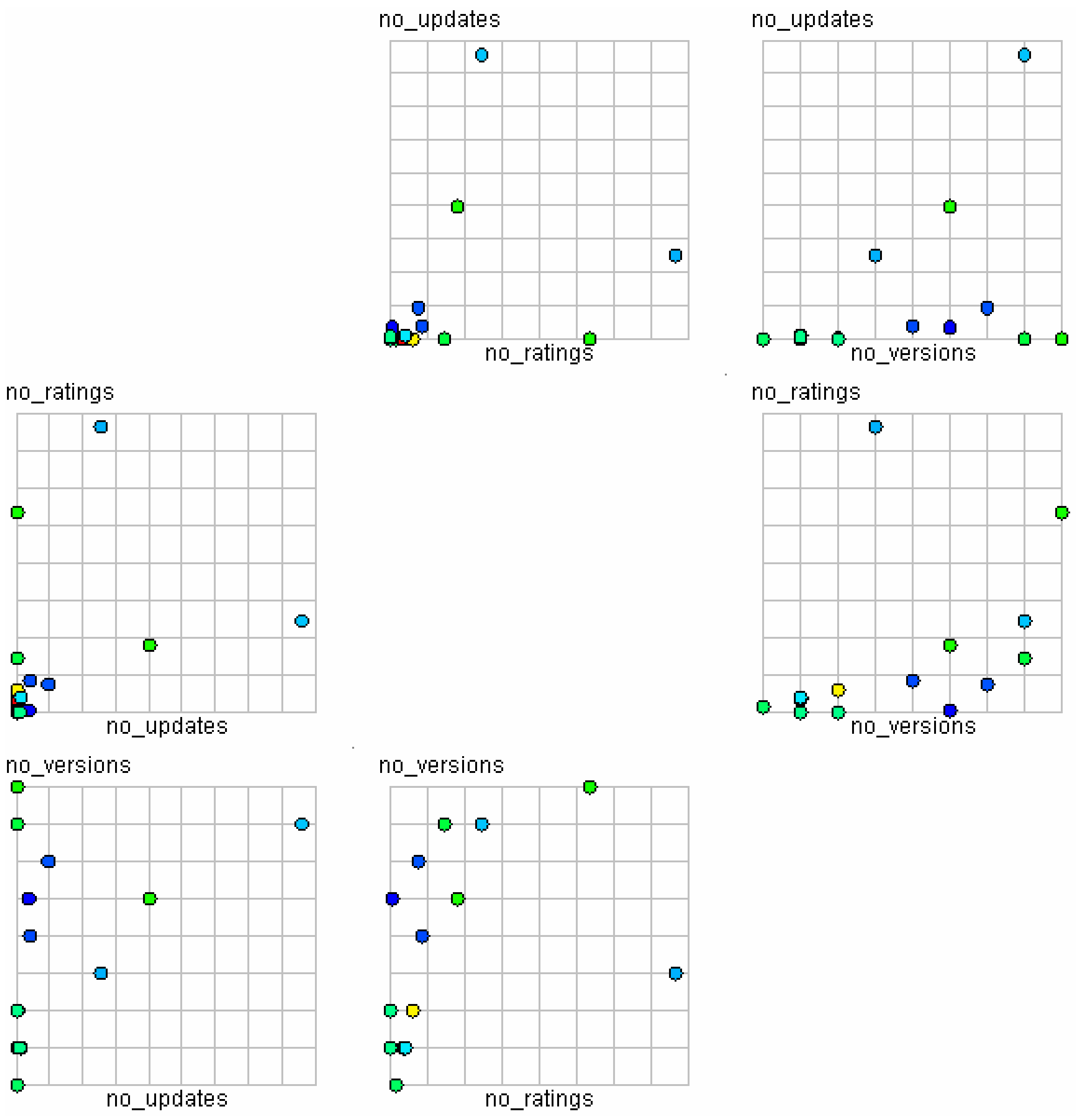


Figure 11-12 – Excerpt from scatter matrix in RapidMiner

11.3.2 Comparison against third-party findings

Some of the above findings can be compared to publicly available figures. For instance, our finding that the two ‘extreme’ ratings (1/5 and 5/5) account for 79% of reviews can be compared against ratings of the top-25 Android apps [122]. While naturally the averages are higher in the toplist than for our own dataset, the percentage of downloaders who wrote a review was upwards of 6%. This reveals that the top apps were three times more likely to attract a review than the apps in our dataset.

The aggregated share of ratings ‘1 or 5’ were also even bigger for these 25 apps than for our dataset, with the average being more than 86%. As the very top of the rankings may be skewed, I attempted to find another source for comparison – as the Android figures generally seem more publicly available, I used them once again as benchmark. In [123], the author scraped more than 5.6 million app reviews and their ratings, and found the distribution shown in Figure 11-13.

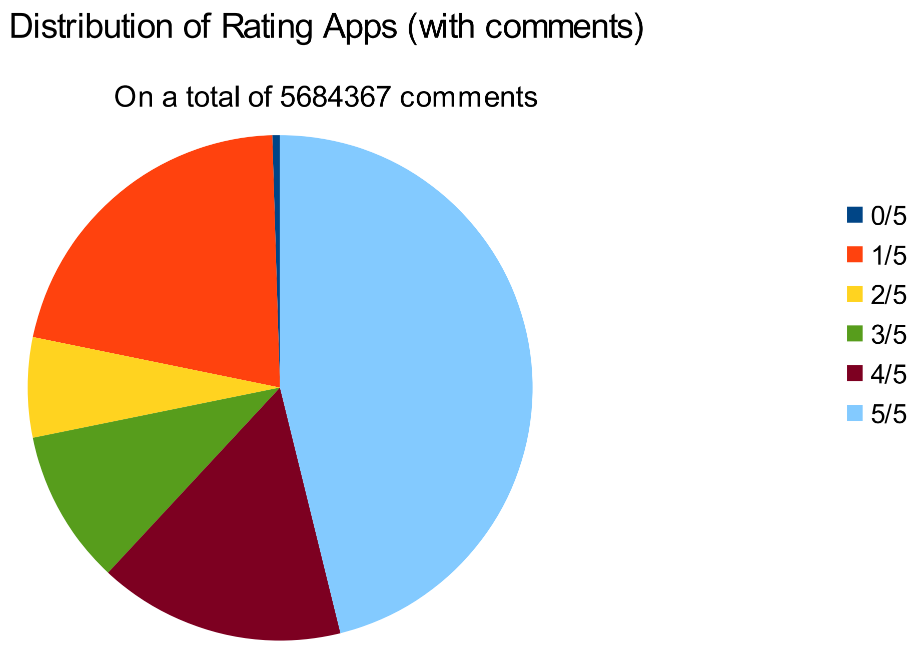


Figure 11-13 – Distribution of Android Market/Google Play apps [adopted from 123]

Summing up the shares for 1/5 and 5/5 once again, the aggregate is 67.5%, slightly less but still within what could be considered applicable for the “law of the vital few”.

11.4 Recommendations for further analysis

Lack of support for nominal values is an obvious drawback in the analysis that was performed. It would be very useful to be able to cluster apps by the type of their smartphone platform, whether they are promotional or commercial, and so forth. Either an alternative statistical method that can handle nominal values could be used – the *polycor* R package [121] looks promising but was omitted for lack of time – or manual conversion into ordinal scales could be attempted.

Clustering and PCA are inherently limited in that we only choose to focus on select components. As in this case, the number of downloads and the number of updates are the two principal components, the clustering based on these components might not be very meaningful for the developer. The Pearson correlations are also limited in that they do not distinguish between cause and effect – does the number of ratings increase downloads, or is it simply that an app with many downloads has more users and therefore get more reviews? A larger dataset could possibly assist in differentiating cause from effect.

A more interesting approach would probably be to embed session tracking libraries such as Google Analytics [59] or Flurry [63], and then cluster apps by average app session lengths, retention rates, and other in-depth metrics. As these libraries have to be included in the app executable before publication on the marketplace, this is a company policy decision for the future. There are clearly strong financial incentives for using such libraries, as they could help to decide which business models would generate most revenue – should apps use in-app purchases, rely on embedded advertisements’ or simply be sold as one-time purchases? The answer is of course highly dependant on each app’s usage pattern.

Additionally, including this tracking could enable many new metrics to be available, such as OS version, screen size, and other technical specifications. Some of the latter figures are available in the Developer Console, but not offered over their API. Finally, one very powerful benefit of embedding tracking libraries into apps is the option to send arbitrary variables back to the developer. Anything can be recorded. As an example, for location based apps, knowing the GPS signal strength and (coarse) location could be extremely useful. For commercial apps, measuring the conversion rate and in-app sales figures are both obviously highly interesting.

RapidMiner would be a good choice for these future, more in-depth analyses, based on my initial tests. This program enables abstract model generation as a process flow; importing, formatting, filtering, and analyzing data as building blocks. Below is an example process flow in RapidMiner, where two data sets (purple blocks) are imported, of which one is the training dataset and the other is the set we want to make estimations on. The training data is filtered for missing values (pink) and other anomalies before a polynomial regression model is created (green), and applied directly onto the second dataset.

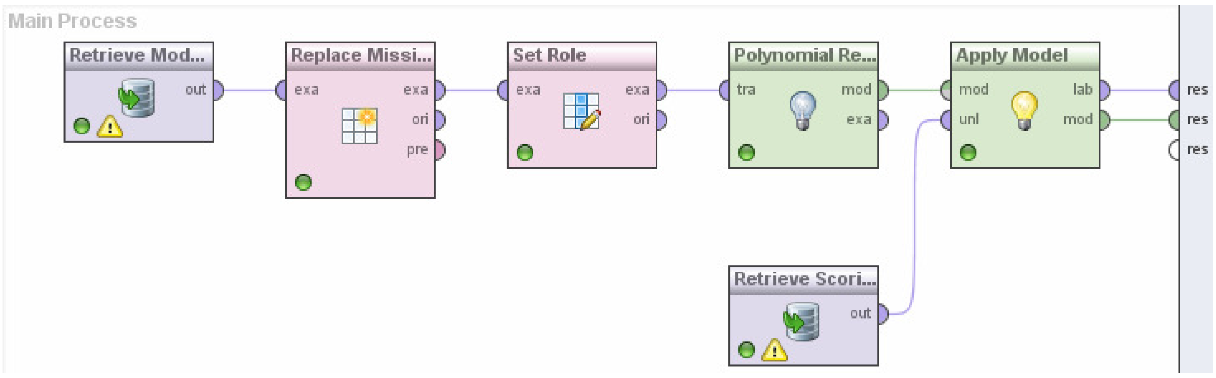


Figure 11-14 – Process development view in RapidMiner

The most interesting aspect of these modelling capabilities, is that if an app company decides to include session tracking libraries into their future apps, this could potentially generate enough data to develop a model that could estimate the corresponding values for older apps that lack the session based tracking.

Even for manual comparison, session based metrics can be compared against public data sources, such as the OurMobilePlanet initiative [52] where populations of recorded demographics have responded to questions regarding their mobile use and behaviour.

12 Conclusions

This chapter summarizes the entire thesis project by making a few remarks and stating some conclusions. The chapter covers both the theoretical side of app statistics and the implementation of the web platform, ending with an outline of suggestions for future work and some reflections around the possible privacy issues that could ensue, if and when the developed platform starts to track app sessions on an individual level.

12.1 Conclusion

Limitations in first-party API's are unfortunate, as these limitations prevent access to some highly interesting data. For instance, when doing a manual import of CSV files from Google Play Developer Console, we may separate app ratings and downloads per device (manufacturer and model). This is to my knowledge currently not available even through use of any third-party API. Even though this is available when logging in to the Developer Console, it would be very useful for automating bug tracking if we could find trends automatically. A concrete example from the app company involved in this thesis, is a recent scenario with an Android app that had a bug only present in Sony Ericsson phones of a certain model. This particular bug was found manually, but had the Google Play API allowed for extraction of app crashes per device model, it could have been found programmatically in an automated fashion. Hopefully, it is only a matter of time and API maturity before this data can be readily accessed by developers.

The necessity to use third-party APIs has taught me the importance of designing software in a modular fashion, so that a data source or back-end may be easily replaced by another. Who knows which services might be available tomorrow? The best advice I could give someone who wanted to make their own smartphone app tool is to *collect all data* – enable the database schema to store all possible data, from all possible data sources, and start gathering data as soon as possible. Determining how to use and display the data is a later task, but the data might not be there to import tomorrow. However, a solid database schema facilitates tasks such as writing SQL queries against the database. Other points to consider in the design phase, is to separate front- and back-ends (sending data formatted as JSON, XML, or a similar format) which makes it easier to port the tool onto another platform in the future. Because of the 14-day history limitation for App Store data, the locally stored data has already proved helpful.

The web platform implementation came to revolve more than originally planned around the *qualitative* metric: review comments. As no third-party APIs seem to extract detailed information from reviews, this seems to be a useful function for the in-house platform. Bucketing keywords transforms the qualitative texts into quantitative frequencies, enabling ranking of words or tags based on their popularity. The resulting 'word cloud' and 'trending review keywords' have already proven useful to the company, with actual live use cases for the company's clients.

The statistical analysis was initially expected to use a multitude of software tools - but in the end OpenOffice Calc and R with some additional packages were able to cover all major statistical methods discussed in the theory section. RapidMiner was also used briefly. The analysis itself follows the same theme as the web platform implementation – all the charts and R scripts are there, ready to be applied with more and better data. It has to be said that clustering is not particularly relevant in its present stage, but will instantly become more useful if the company distributes apps with session tracking capabilities.

Evaluating the final implementation outcome: all three primary goals were achieved, with the exception of some manual import file formats. As data import was automated, it is not as important however. The secondary goals were in most cases taken to a proof-of-concept level, with the API as the notably most interesting functionalty.

Even at its current stage, the statistical findings are already helpful to the app developer. Some examples include the tendency that apps attract most downloads during weekends, the correlation between more app versions and higher average rating, and the fit of the Pareto 80-20 rule upon both review keyword distribution and the aggregated share of 1/5 and 5/5 ratings. The latter indicates that a focused effort towards a few critical bugs or feature requests (specifically those that most reviewers mention), could dramatically improve the app average rating. In turn, it could be speculated that this effort would prevent many of the 1/5 reviews from ever being submitted, which could be another contributing factor to a higher average rating. The apps in the case study data set could also be divided into at least three different basic clusters based on their statistical profiles.

12.2 Future work

The ideal next addition to the app statistics platform would indeed be for all the company's apps to include a session tracking library, such as either Google Analytics [59] or Flurry [63]. This would enable more in-depth reports, more useful clustering and trend finding possibilities, and provide a good way of collecting demographics and vendor/handset model statistics. These libraries support sending app-specific variables from the app source, which makes the data collection possibilities nearly unlimited. As mentioned, session tracking could also prove useful for decision-making in terms of choosing the optimal app business model.

As for statistical modelling, it would be interesting to try and model an app's interest over time; the number of daily downloads as a function of days passed since release. There are complications when developing such a model, as for instance app version updates will of course influence the app's interest. A simplified model that assumes one single app version without updates could be a starting point.

Hopefully, Google and Apple will in a not too distant future improve their APIs so that the data importation scripts that have been developed can migrate to using first-hand data sources, rather than depending on third-party APIs. In such an event, it would probably be wise to collect and save data from all sources in parallel during a transition period, to verify consistency. Even if neither Google nor Apple provide better data access in the future, there are some other data sources available if one is willing to pay for the service, such as Applyzer [101] and Appfigures [102]. All the tools evaluated in this thesis were free of charge in the versions that were tested. The additional libraries mentioned were either left for future work because of time limitations or the fact that they were associated with costly subscriptions.

Feature-wise, the most desired new feature would be to implement the alerts/trigger system. Setting up filters for automatic statistics monitoring, with alert emails sent out when applicable, would reduce the current workload of having to periodically watch each app manually. General code clean-up and refactoring should then be undertaken, and a thorough documentation must be produced in the event that a programmer other than myself should overtake responsibility for the in-house app statistics tool.

After deploying the implementation from my development laptop onto a production server – and verifying that functionality is still intact – the web platform can gradually be put into use. During the first stage it might be only used internally for the company's purposes, such as tracking and generating PDF reports for the company's own use. Gradually, select customers of the app publisher may then be invited and be given their own login accounts to try out the platform – at first they may receive read-only accounts; but if everything works as intended their accounts could be upgraded to full privileges. It is my belief that this gradual roll-out will help to locate existing bugs or issues.

12.3 Reflections

Besides the technical requirements on the server environment that is necessary before deployment to production, there are some legal and ethical aspects to address before making the web platform publicly accessible.

As the planned additions listed in section 12.2 include integrating live per-user session data for tracking purposes, the question of personal integrity becomes important, not only from a company's goodwill perspective but also legally.

SDKs for measuring user session data are normally clear about their terms. For instance, Google Analytics clearly state in their Terms & Conditions (T&C) that *"You will not (and will not allow any third party to) use the Service to track, collect or upload any data that personally identifies an individual (such as a name, email address or billing information), or other data which can be reasonably linked to such information by Google."* [104]. Normally, these app sessions are identified by a anonymized nonce or number rather than actually transferring personal data such as name or email. This setup is very much similar to the previously mentioned IDFA used by Apple on their iDevices [51].

This aligns well with the Swedish personal data integrity laws, Personuppgiftslagen 1998:204 (PUL) [114], which states that any gathered data in a *structured setup* must abide by certain directives according to Swedish law. As the supervising government agency Datainspektionen clarifies [103], any database storage of data would automatically be categorized as a structured setup. Related policies also require any site owner to inform their users whenever cookies are used for storing personal information. These integrity and legal matters must be addressed before deploying the app statistics platform onto a production server.

In order to understand the importance of carefully designing any user identifiers, one just has to look at mistakes done by large online companies such as Netflix [130]. In an attempt to crowdsource a 10% more effective algorithm for calculating movie recommendations, Netflix released (what they believed to be) anonymized data sets over movie rentals. Two students from Texas University soon were able to reverse engineer and thus reveal the unobfuscated data. After settling lawsuits from users who claimed privacy damages, Netflix eventually called off their second contest [130]. Such scenarios highly stresses why thought must go into the privacy and legal issues that arise when handling personal and sensitive information.

References

[1] Brian Womack, "Google Says 700,000 Applications Available for Android", *BusinessWeek,* [Online] October 29, 2012, Available at: http://www.businessweek.com/news/2012-10-29/google-says-700-000-applications-available-for-android-devices (Accessed January 10, 2013).

[2] Dan Rowinski, "25B Downloads And Google Play Is Still Growing Fast", (ReadWrite), [Online] September 26, 2012. Available at: http://readwrite.com/2012/09/26/25-billion-app-downloads-later-google-play-is-growing-fast (Accessed January 10, 2013).

[3] Gert Jan Spriensma, "2012 – Year in Review", (Distimo), [Online] 2012, Available at: http://www.distimo.com/publications/archive/Distimo Publication - Full Year 2012.pdf (Accessed February 3, 2013).

[4] Christy Pettey and Rob van der Meulen, "Gartner Says Worldwide Sales of Mobile Phones Declined 3 Percent in Third Quarter of 2012; Smartphone Sales Increased 47 Percent", (Gartner), [Online] 2012, Available at: http://www.gartner.com/it/page.jsp?id=2237315 (Accessed January 8, 2013).

[5] Christy Pettey and Rob van der Meulen, "Gartner Says Free Apps Will Account for Nearly 90 Percent of Total Mobile App Store Downloads in 2012", (Gartner*),* [Online] September 11, 2012, Available at: http://www.gartner.com/it/page.jsp?id=2153215 (Accessed January 8, 2013).

[6] "Android, the world's most popular mobile platform", Google, [Online], Available at: http://developer.android.com/about/index.html (Accessed February 3, 2013).

[7] "Android Fragmentation Visualized", OpenSignal, [Online], Available at: http://opensignal.com/reports/fragmentation.php (Accessed February 3, 2013).

[8] "Transaction Fees – Google Play for Developers Help", Google, [Online], Available at: http://support.google.com/googleplay/android-developer/bin/answer.py?hl=en&answer=112622 (Accessed February 3, 2013).

[9] Natalie Kerris and Steve Dowling, "Apple Reinvents the Phone with iPhone", (Apple), [Online] January 9, 2007, Available at: http://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html (Accessed February 3, 2013).

[10] Mike Brown, "[Infographic] The State of Mobile App Quality: Android vs. iOS", (Software Testing Blog*),* [Online] February 2012, Available at: http://blog.utest.com/infographic-the-state-of-mobile-app-quality-android-vs-ios/2012/02/ (Accessed January 10, 2013).

[11] "Comparison of free and paid Android apps", (AppBrain), [Online], Available at: http://www.appbrain.com/stats/free-and-paid-android-applications (Accessed February 3, 2013).

[12] “Distribute your app – iOS Developer Program – Apple Developer”, (Apple), [Online], Available at: https://developer.apple.com/programs/ios/distribute.html (Accessed February 3, 2013).

[13] Peter Farago, “App Engagement: The Matrix Reloaded”, (Flurry), [Online], Available at: http://blog.flurry.com/bid/90743/App-Engagement-The-Matrix-Reloaded (Accessed January 10, 2013).

[14] Peter Farago, “Microsoft May Be Closer Than It Appears in Android’s Rearview Mirror”, (Flurry), [Online], Available at: http://blog.flurry.com/bid/86277/Microsoft-May-Be-Closer-Than-It-Appears-in-Android-s-Rearview-Mirror (Accessed January 10, 2013).

[15] Rajiv Garg and Rahul Telang, “Estimating App Demand from Publicly Available Data”. (SSRN Electronic Journal), [Online] 2012, Available at: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1924044 (Accessed January 10, 2013).

[16] Lin Hao et al, “The Economic Value of Ratings in App Market”, (SSRN Electronic Journal), [Online] 2011, Available at: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1892584 (Accessed January 10, 2013).

[17] Pradeep Racherla, Christopher Furner, and Jeffry Babb, “Conceptualizing the Implications of Mobile App Usage and Stickiness: A Research Agenda”, (SSRN Electronic Journal), [Online] 2012, Available at: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2187056 (Accessed January 10, 2013).

[18] Dan Rowinski, “Sorry, Samsung, iPhone Is Not Your Mother’s Smartphone”, (ReadWrite), [Online] October 8, 2012, Available at: http://readwrite.com/2012/10/08/sorry-samsung-iphone-is-not-your-mothers-smartphone (Accessed January 11, 2013).

[19] Dan Lee, “Courting Today’s Mobile Consumer”, (Nielsen), [Online] 2012, Available at: http://nielsen.com/content/dam/corporate/us/en/reports-downloads/2012-Webinars/courting-the-mobile-consumer-7-18-final.pdf (Accessed February 3, 2013).

[20] “In-App Purchases to Outpace Pay-Per-Download Revenues in 2012”, (ABIresearch technology market intelligence), [Online] 2012, Available at: http://www.abiresearch.com/press/in-app-purchases-to-outpace-pay-per-download-reven (Accessed February 3, 2013).

[21] Peter Farago, “App Developers Signal Apple Allegiance Ahead of WWDC and Google I/O”, (Flurry), [Online] June 7, 2012, Available at: http://blog.flurry.com/bid/85911/App-Developers-Signal-Apple-Allegiance-Ahead-of-WWDC-and-Google-I-O (Accessed January 11, 2013).

[22] “iTunes Store: About In-App Purchases”, Apple, [Online], Available at: http://support.apple.com/kb/ht4009 (Accessed February 3, 2013).

[23] Simon Khalaf, “Mobile Apps: We Interrupt This Broadcast”, (Flurry), [Online] December 5, 2012, Available at: http://blog.flurry.com/bid/92105/Mobile-Apps-We-Interrupt-This-Broadcast (Accessed January 11, 2013).

[24] Peter Farago, “The Great Distribution of Wealth Across iOS and Android Apps”, (Flurry), [Online] July 31, 2012, Available at: http://blog.flurry.com/bid/88014/The-Great-Distribution-of-Wealth-Across-iOS-and-Android-Apps (Accessed January 11, 2013).

[25] “App Annie Index Nov 2012”, (App Annie), [Online] November 2012, Available at: http://blog.appannie.com/app-annie-index-november-2012/ (Accessed February 3, 2013).

[26] Fred Wilson, “The Freemium Business Model”, [Online] March 2006, Available at: http://www.avc.com/a_vc/2006/03/the_freemium_bu.html (Accessed February 3, 2013).

[27] Nat Harrison and Teresa Brewer, “Apple Introduces iPhone 5”, (Apple), [Online] September 12, 2012, Available at: http://www.apple.com/pr/library/2012/09/12Apple-Introduces-iPhone-5.html (Accessed February 3, 2013).

[28] “Start Developing iOS Apps Today: Introduction”, (Apple), [Online], Available at: https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/Introduction.html (Accessed February 3, 2013).

[29] F. John Reh., “Pareto’s Principle - The 80-20 Rule”, (About.com Management), [Online] Available at: http://management.about.com/cs/generalmanagement/a/Pareto081202.htm (Accessed January 14, 2013).

[30] Erik Brynjolfsson, Michael D. Smith, and Yu Jeffrey Hu, “Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers”, (SSRN Electronic Journal), [Online] 2003, Available at: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=400940 (Accessed January 14, 2013).

[31] “State of the Appnation – A Year of Change and Growth in U.S. Smartphones”, (nielsenwire), [Online] May 16, 2012, Available at: http://blog.nielsen.com/nielsenwire/online_mobile/state-of-the-appnation-%E2%80%93-a-year-of-change-and-growth-in-u-s-smartphones (Accessed February 3, 2013).

[32] Peter Farago, “The Truth About Cats and Dogs: Smartphone vs Tablet Usage Differences”, (Flurry), [Online] October 29, 2012, Available at: http://blog.flurry.com/bid/90987/The-Truth-About-Cats-and-Dogs-Smartphone-vs-Tablet-Usage-Differences (Accessed January 14, 2013).

[33] “App Review Guidelines – Apple Developer”, Apple, Available at: https://developer.apple.com/appstore/guidelines.html (Accessed February 3, 2013).

[34] Olga Kharif, “Piracy Cuts Into Paid App Sales”, (BusinessWeek: technology), [Online] November 1, 2012, Available at: http://www.businessweek.com/articles/2012-11-01/piracy-cuts-into-paid-app-sales (Accessed January 14, 2013).

[35] “AppBrain, Developer tools - Android library statistics | AppBrain.com”, (AppBrain.com), [Online], Available at: http://www.appbrain.com/stats/libraries/dev (Accessed January 14, 2013).

[36] Peter Farago, “Mobile App Usage Further Dominates Web, Spurred by Facebook”, (Flurry), [Online] January 9, 2012, Available at: http://blog.flurry.com/bid/80241/Mobile-App-Usage-Further-Dominates-Web-Spurred-by-Facebook (Accessed January 14, 2013).

[37] “jQuery Mobile: Touch-Optimized Web Framework for Smartphones & Tables”, (jQuery), [Online], Available at: http://jquerymobile.com/ (Accessed February 3, 2013).

[38] “Google Play for Developers Help”, Google, Available at: http://support.google.com/googleplay/android-developer/?hl=en (Accessed February 3, 2013).

[39] “iTunes Connect Sales and Trends Guide: App Store v7”, Apple, [Online], Available at: http://www.apple.com/itunesnews/docs/AppStoreReportingInstructions.pdf (Accessed February 3, 2013).

[40] “Windows Phone 7: A Fresh Start for the Smartphone”, Microsoft, [Online] October 10, 2012, Available at: http://www.microsoft.com/en-us/news/features/2010/oct10/10-11wp7main.aspx (Accessed February 3, 2013).

[41] Brandon Watson, “New Policies for Next Gen Windows Phone Marketplace”, (Windows Phone Developer Blog), [Online] June 7, 2010*,* Available at: http://blogs.windows.com/windows_phone/b/wpdev/archive/2010/06/07/new-policies-for-next-gen-windows-phone-marketplace.aspx (Accessed February 3, 2013).

[42] “BlackBerry ‘leading’ the smartphone charge”, (NATIONNews*),* [Online] November 23, 2011, Available at: http://www.nationnews.com/articles/view/blackberry-leading-the-smartphone-charge/ (Accessed February 3, 2013).

[43] Ben Bryant, “BlackBerry 10 handsets confirmed for January 30 launch”, (Telegraph.co.uk), [Online] November 12, 2012, Available at: http://www.telegraph.co.uk/technology/blackberry/9672758/BlackBerry-10-handsets-confirmed-for-January-30-launch.html (Accessed January 14, 2013).

[44] Dara Kerr, “BlackBerry App World said to hawk pirated Android apps”, (CNET), [Online] January 8, 2013, Available at: http://news.cnet.com/8301-1035_3-57562905-94/blackberry-app-world-said-to-hawk-pirated-android-apps/ (Accessed January 14, 2013).

[45] “AdMob – Monetize and promote your apps with ads”, Google, [Online], Available at: http://www.google.com/ads/admob/ (Accessed February 3, 2013).

[46] Kirby Turner, “appdailysales”, (Github), [Online], Available at: https://github.com/kirbyt/appdailysales (Accessed February 3, 2013).

[47] “iAd – Apple Developer”, Apple, [Online], Available at: https://developer.apple.com/iad/ (Accessed February 3, 2013).

[48] Tero Kuittinen, “Missing the dazzling profit potential of free apps”, (BGR*),* [Online] November 19, 2012, Available at: http://bgr.com/2012/11/19/mobile-app-market-analysis-free-apps/ (Accessed January 15, 2013).

[49] "Top app stores risk losing control of app discovery", (Canalys), [Online] June 11, 2012, Available at: http://www.canalys.com/newsroom/top-app-stores-risk-losing-control-app-discovery (Accessed February 3, 2013).

[50] Joe Aimonetti, "Apple bolstering privacy by ending developer access to UDIDs", (CNET), [Online] March 28, 2012, Available at: http://news.cnet.com/8301-13579_3-57406058-37/apple-bolstering-privacy-by-ending-developer-access-to-udids/ (Accessed January 15, 2013).

[51] Laura Stampler, "Here's Everything We Know About IFA, The iPhone Tracking Technology In Apple's iOS 6", (Business Insider), [Online] October 15, 2012, Available at: http://www.businessinsider.com/everything-we-know-about-ifa-and-tracking-in-apples-ios-6-2012-10 (Accessed January 15, 2013).

[52] "Our Mobile Planet", Google, Ipsos, Mobile Marketing Association, [Online], Available at: http://www.thinkwithgoogle.com/mobileplanet/sv/ (Accessed January 15, 2013).

[53] "App Analytics – Conversion Tracking", (Distimo), [Online], Available at: http://www.distimo.com/app-analytics#conversion-tracking (Accessed February 3, 2013).

[54] Jeferson Valadares, "Free-to-play Revenue Overtakes Premium Revenue in the App Store", (Flurry), [Online] July 7, 2011, Available at: http://blog.flurry.com/bid/65656/Free-to-play-Revenue-Overtakes-Premium-Revenue-in-the-App-Store (Accessed January 15, 2013).

[55] Jeferson Valadares, "Consumers Spend Average of $14 per Transaction in iOS and Android Freemium Games", (Flurry), [Online] July 25, 2011, Available at: http://blog.flurry.com/bid/67748/Consumers-Spend-Average-of-14-per-Transaction-in-iOS-and-Android-Freemium-Games (Accessed January 15, 2013).

[56] Jeferson Valadares, "Mobile Freemium Games: Women Thrifty, Men Binge", (Flurry), [Online] September 26, 2011, Available at: http://blog.flurry.com/bid/72755/Mobile-Freemium-Games-Women-Thrifty-Men-Binge (Accessed January 15, 2013).

[57] "Electronic Arts, The Sims Freeplay", (App Store), [Online], Available at: https://itunes.apple.com/us/app/the-sims-freeplay/id466965151?mt=8 (Accessed January 15, 2013).

[58] "Pricing packages to meet your needs – Salesforce marketing cloud", (Radian6), [Online], Available at: http://www.radian6.com/what-we-sell/radian6-pricing/ (Accessed February 3, 2013).

[59] "Google Analytics Mobile Analytics and Reporting", Google, [Online], Available at: http://www.google.com/analytics/features/mobile.html (Accessed February 3, 2013).

[60] "Returning apps – Google Play Help", Google, [Online], Available at: http://support.google.com/googleplay/bin/answer.py?hl=en&answer=134336 (Accessed February 3, 2013).

[61] "iTunes Store – Terms and Conditions", Apple, [Online], Available at: http://www.apple.com/legal/itunes/us/terms.html#APPS (Accessed February 3, 2013).

[62] "TestFlight: Beta Testing On The Fly", (TestFlight), [Online], Available at: https://testflightapp.com/sdk/ (Accessed February 3, 2013).

[63] "Flurry | App Advertising and Analytics", (Flurry), [Online], Available at: https://dev.flurry.com/ (Accessed February 3, 2013).

[64] Allen B. Downey, *Think stats*: *Probability and statistics for programmers*, 1st ed. O'Reilly Media*,* 2011.

[65] Octavian Carare, "The Impact of Bestseller Rank on Demand: Evidence from the App Market", (SSRN Electronic Journal), [Online] 2010, Available at: http://papers.ssrn.com/abstract=1564314 (Accessed January 17, 2013).

[66] "Configuration and Reporting API Limits and Quotas – Google Analytics – Google Developers", Google, [Online], Available at: https://developers.google.com/analytics/devguides/reporting/core/v3/limits-quotas (Accessed February 3, 2013).

[67] "Google Play Android Developer API", Google*,* [Online], Available at: https://developers.google.com/android-publisher/quotas (Accessed February 3, 2013).

[68] Alexandre Thiel, "android-market-api – Android Market for all developers!", [Online], Available at: http://code.google.com/p/android-market-api/source/browse/trunk/AndroidMarketApi/ (Accessed February 3, 2013).

[69] Chuck Martin, *The Third Screen: Marketing to Your Customers in a World Gone Mobile*, 1st ed. Nicholas Brealey Publishing, 2011. ISBN: 978-1857885644.

[70] Jonathon Shlens, "A Tutorial on Principal Component Analysis", Carnegie Mellon University, Computer Science Department, [Online] December 10, 2005, Available at: http://www.cs.cmu.edu/~elaw/papers/pca.pdf (Accessed January 30, 2013).

[71] Jerome Lecoq, "PCA demystified", (Matlabtips.com), [Online] March 29, 2012, Available at: http://www.matlabtips.com/pca-demystified (Accessed January 30, 2013).

[72] Frank Höppner, "What is (fuzzy) cluster analysis?", Ostfalia University of Applied Sciences, [Online] March 21, 2013, Available at: http://public.fh-wolfenbuettel.de/~hoeppnef/clustering.html (Accessed April 2, 2013).

[73] Jan Jantzen, "Tutorial On Fuzzy Clustering", Technical University of Denmark, [Online] August 2001, Available at: http://www.csee.usf.edu/~manohar/Papers/Pancreas/Clustering.pdf (Accessed January 30, 2013).

[74] Jari Oksanen, "Cluster Analysis: Tutorial with R", University of Oulu, [Online] February 2, 2012, Available at: http://cc.oulu.fi/~jarioksa/opetus/metodi/sessio3.pdf (Accessed January 31, 2013).

[75] “Cophenetic correlation coefficient”, (MathWorks), [Online], Available at: http://www.mathworks.se/help/stats/cophenet.html (Accessed February 3, 2013).

[76] “Data Mining Curriculum: A Proposal (Version 1.0)”, *(*Intensive Working Group of ACM SIGKDD Curriculum Committee)*,* [Online] April 30, 2006, Available at: http://www.sigkdd.org/curriculum/CURMay06.pdf (Accessed January 31, 2013).

[77] Usama Fayyad, Gregory Piatetsky-Shapiro and Padhraic Smyth, “From data mining to knowledge discovery in databases,” in *Association for the Advancement of Artificial Intelligence (AAAI) 97,* 1997, pp. 37-54. Available at: http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf (Accessed February 1, 2013).

[78] “OLAP Council Whilte Paper”, Symcorp*,* [Online] 1997, Available at: http://www.symcorp.com/downloads/OLAP_CouncilWhitePaper.pdf (Accessed February 1, 2013).

[79] “ELKI: Environment for Developing KDD-Applications Supported by Index-Structures”, Ludwig-Maximilian University of Munich, [Online], Available at: http://elki.dbs.ifi.lmu.de/ (Accessed February 1, 2013).

[80] “KNIME | Konstanz Information Miner”, KNIME, [Online], Available at: http://www.knime.org/ (Accessed February 1, 2013).

[81] “RapidMiner -- Data Mining, ETL, OLAP, BI”, SourceForge, [Online], Available at: http://sourceforge.net/projects/rapidminer/ (Accessed February 1, 2013).

[82] “The Open Group Base Specifications Issue 7: IEEE Std 1003.1-2008”, The IEEE and The Open group, [Online] 2008, Available at: http://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html (Accessed February 1, 2013).

[83] “API Documentation”, Distimo, [Online], Available at: https://analytics.distimo.com/support/api/v2/examples (Accessed January 24, 2013).

[84] L. Masinter, T. Berners-Lee, and R.T. Fielding, “IETF RFC 3986: Uniform Resource Identifier (URI): Generic Syntax”, IETF, [Online] January 2005, Available at: http://tools.ietf.org/html/rfc3986 (Accessed February 1, 2013).

[85] “Apache HTTP Server Version 2.4 – Apache Module mod_rewrite”, The Apache Software Foundation, [Online], Available at http://httpd.apache.org/docs/2.4/mod/mod_rewrite.html (Accessed February 1, 2013).

[86] “Bootstrap – Sleek, intuitive, and powerful front-end framework for faster and easier web development”, Twitter, [Online], Available at: http://twitter.github.com/bootstrap/index.html (Accessed February 1, 2013).

[87] “Google Chart Tools – Google Developers”, Google, [Online], Available at https://developers.google.com/chart/ (Accessed February 1, 2013).

[88] “DataTables (table plug-in for jQuery)”, DataTables, [Online], Available at http://www.datatables.net/index (Accessed February 1, 2013).

[89] “iOS Developer Enterprise Program”, Apple, [Online], Available at https://developer.apple.com/programs/ios/enterprise/ (Accessed March 8, 2013).

[90] Chris Anderson, *The Long Tail: Why the Future of Business is Selling Less of More*. Hyperion, 2006, ISBN: 1401302378.

[91] “Scaffolding – Bootstrap”, Twitter, [Online], Available at http://twitter.github.com/bootstrap/scaffolding.html (Accessed March 12, 2013).

[92] Karl Monaghan, “A Sunday afternoon project: Providing App Store stats via JSON”, [Online] September 18, 2011, Available at http://www.karlmonaghan.com/2011/09/18/a-sunday-afternoon-project-providing-app-store-stats-via-json/ (Accessed March 12, 2013).

[93] “GnuWin”, SourceForge, [Online], Available at http://sourceforge.net/projects/gnuwin32 (Accessed March 12, 2013).

[94] “PHP File Upload”, W3schools, [Online], Available at http://www.w3schools.com/php/php_file_upload.asp (Accessed March 12, 2013).

[95] “PHP: fgetcsv – Manual”, PHP.net, [Online], Available at http://php.net/manual/en/function.fgetcsv.php (Accessed March 12, 2013).

[96] “PHP: include – Manual”, PHP.net, [Online], Available at http://php.net/manual/en/function.include.php (Accessed March 12, 2013).

[97] “TxtToMy – Save time on Importing Csv (Txt) to Mysql”, WithData, [Online], Available at http://www.withdata.com/txttomy.html (Accessed March 12, 2013).

[98] “Manual :: Mail”, PHP.net, [Online], Available at https://pear.php.net/manual/en/package.mail.mail.php (Accessed March 12, 2013).

[99] “PHP: array_count_values – Manual”, PHP.net, [Online], Available at http://php.net/manual/en/function.array-count-values.php (Accessed March 13, 2013).

[100] Luca Ongaro, “jQCloud – jQuery plugin for drawing neat word clouds that actually look like clouds”, GitHub, [Online], Available at https://github.com/lucaong/jQCloud (Accessed March 13, 2013).

[101] “APPlyzer – iOS & Mac App Rankings, Reviews”, [Online], Available at http://www.applyzer.com/ (Accessed March 18, 2013).

[102] “appFigures Plans & Pricing”, [Online], Available at http://www.appfigures.com/pricing (Accessed March 18, 2013).

[103] “Personuppgiftslagen – Datainspektionen”, Datainspektionen, [Online], Available at http://www.datainspektionen.se/lagar-och-regler/personuppgiftslagen/ (Accessed March 18, 2013).

[104] “Google Analytics Terms of Service”, Google, [Online], Available at http://www.google.com/analytics/terms/us.html (Accessed March 18, 2013).

[105] “MySQL :: MySQL 5.0 Reference Manual :: 13.2.8.2 JOIN Syntax”, MySQL, [Online], Available at http://dev.mysql.com/doc/refman/5.0/en/join.html (Accessed March 18, 2013).

[106] “MySQL :: MySQL 5.0 Reference Manual :: 13.2.8.4 UNION Syntax”, MySQL, [Online], Available at http://dev.mysql.com/doc/refman/5.0/en/union.html (Accessed March 18, 2013).

[107] “Ajaxload – Ajax loading gif generator”, [Online], Available at http://www.ajaxload.info (Accessed March 28, 2013).

[108] Jakob Truelsen, “wkhtmltopdf – Convert html to pdf using webkit (qtwebkit)”, [Online], Available at https://code.google.com/p/wkhtmltopdf/ (Accessed March 28, 2013).

[109] “The WebKit Open Source Project”, [Online], Available at http://www.webkit.org/ (Accessed 28 March, 2013).

[110] Levy Carneiro Jr, Alex Ehlke et al, “jQuery Tag-it! Simple and configurable tag editing widget with autocompete support”, GitHub, [Online], Available at http://aehlke.github.com/tag-it/ (Accessed March 28, 2013).

[111] “JSONLint – The JSON Validator”, [Online], Available at http://jsonlint.com/ (Accessed March 28, 2013).

[112] Tom Preston-Werner, “Semantic Versioning 2.0.0-rc.1”, [Online], Available from http://semver.org/ (Accessed March 28, 2013).

[113] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to information retrieval*, New York: Cambridge University Press, ISBN: 0521865719.

[114] Personuppgiftslag (1998:204). 20 May, 1998.

[115] “The R Project for Statistical Computing”, [Online], Available at http://www.r-project.org/ (Accessed April 9, 2013).

[116] “Quick-R: Cluster Analysis”, Statmethods, [Online], Available at http://www.statmethods.net/advstats/cluster.html (Accessed April 9, 2013).

[117] K. Ruben Gabriel, “The biplot graphic display of matrices with application to principal component analysis”, The Hebrew University, Jerusalem, *Biometrika* 58 (3) pp. 453-467.

[118] Stanislav Kolenikov and Gustavo Angeles, “The Use of Discrete Data in Principal Component Analysis for Socio-Economic Status Evaluation”, University of North Carolina at Chapel Hill, Carolina, NC. [Online] February 2005, Available at http://www.unc.edu/~skolenik/talks/Gustavo-Stas-PCA-generic.pdf (Accessed April 9, 2013).

[119] “Multivariate Analysis: Principal Component Analysis: Biplots :: SAS/IML® Studio 3.2 User’s Guide”, SAS Institute Inc, [Online], Available at http://support.sas.com/documentation/cdl/en/imlsug/62558/HTML/default/viewer.htm#ugmultpca_sect2.htm (Accessed April 9, 2013).

[120] S. S. Stevens, “On the theory of scales of measurement,” *Science*, vol. 103 (no 2684), pp. 677-680, June 1946. DOI: 10.1126/science.103.2684.677.

[121] John Fox, “Polychoric and Polyserial Correlations”, R-project.org, [Online] April 3, 2010, Available at http://cran.r-project.org/web/packages/polycor/polycor.pdf (Accessed April 12, 2013).

[122] Jan Katrenic, “Android Market App Ranklist”, Androidrank.org, [Online], Available at http://www.androidrank.org/listcategory?category=&start=1&sort=0&price=all (Accessed April 15, 2013).

[123] Nicolas Sorel, “Rating Statistics of the Android Market”, AndroLib.com, [Online], Available at http://www.androlib.com/appstatsratings.aspx (Accessed April 16, 2013).

[124] Lukasz Komsta, “Tests for outliers”, R-project.org, [Online] February 15, 2013, Available at http://cran.r-project.org/web/packages/outliers/outliers.pdf (Accessed April 19, 2013).

[125] “Flurry”, Flurry, [Online], Available at http://www.flurry.com (Accessed June 20, 2013).

[126] “App Analytics, Conversion Tracking &amp; Market Data | Distimo”, Distimo, [Online], Available at http://www.distimo.com (Accessed June 20, 2013).

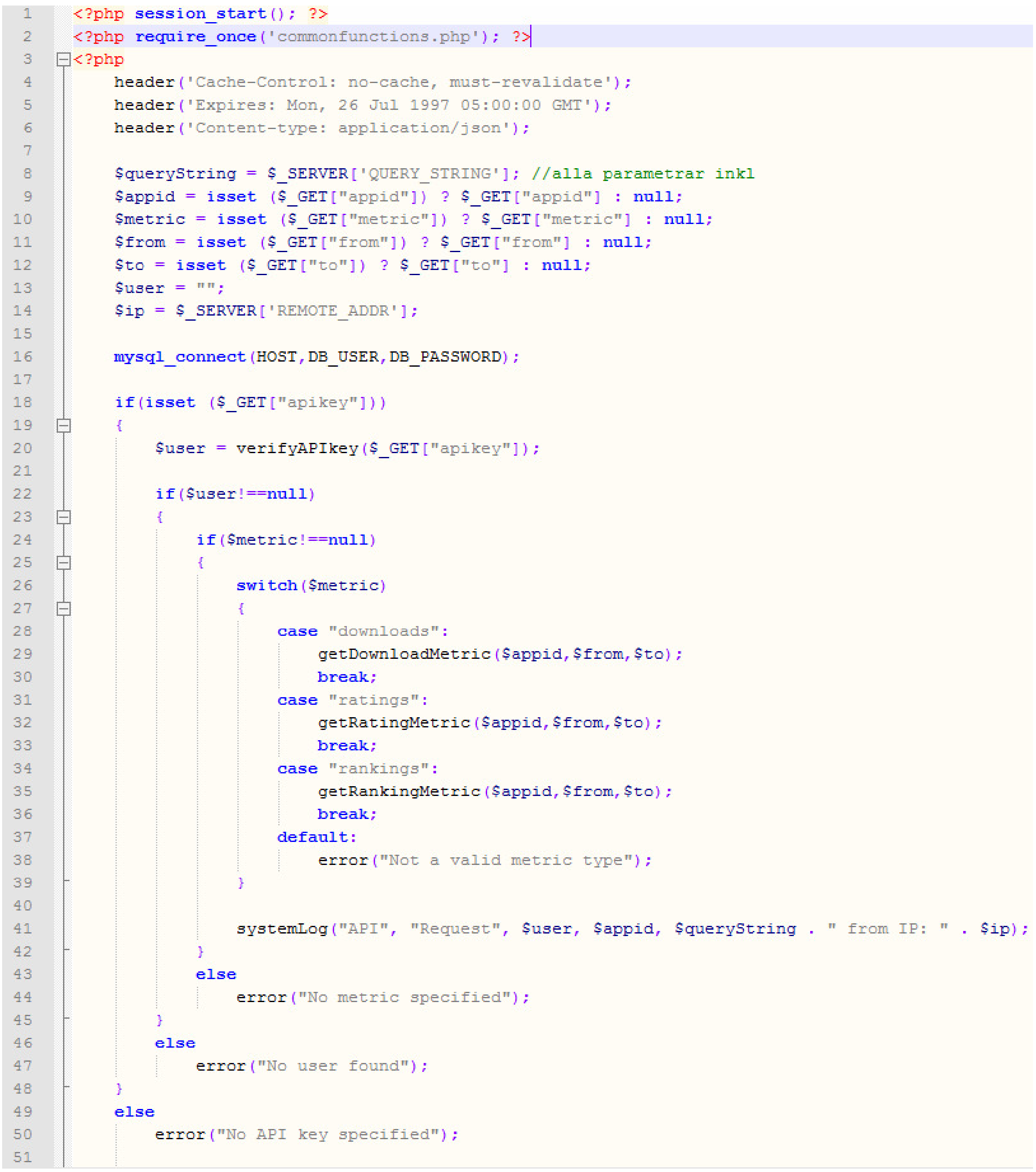
[127] “App Annie - App Ranking, Analytics, Market Intelligence”, App Annie, [Online], Available at http://www.appannie.com (Accessed June 20, 2013).

[128] “TestFlight - Beta Testing On The Fly”, TestFlight, [Online], Available at http://www.testflightapp.com (Accessed June 20, 2013).

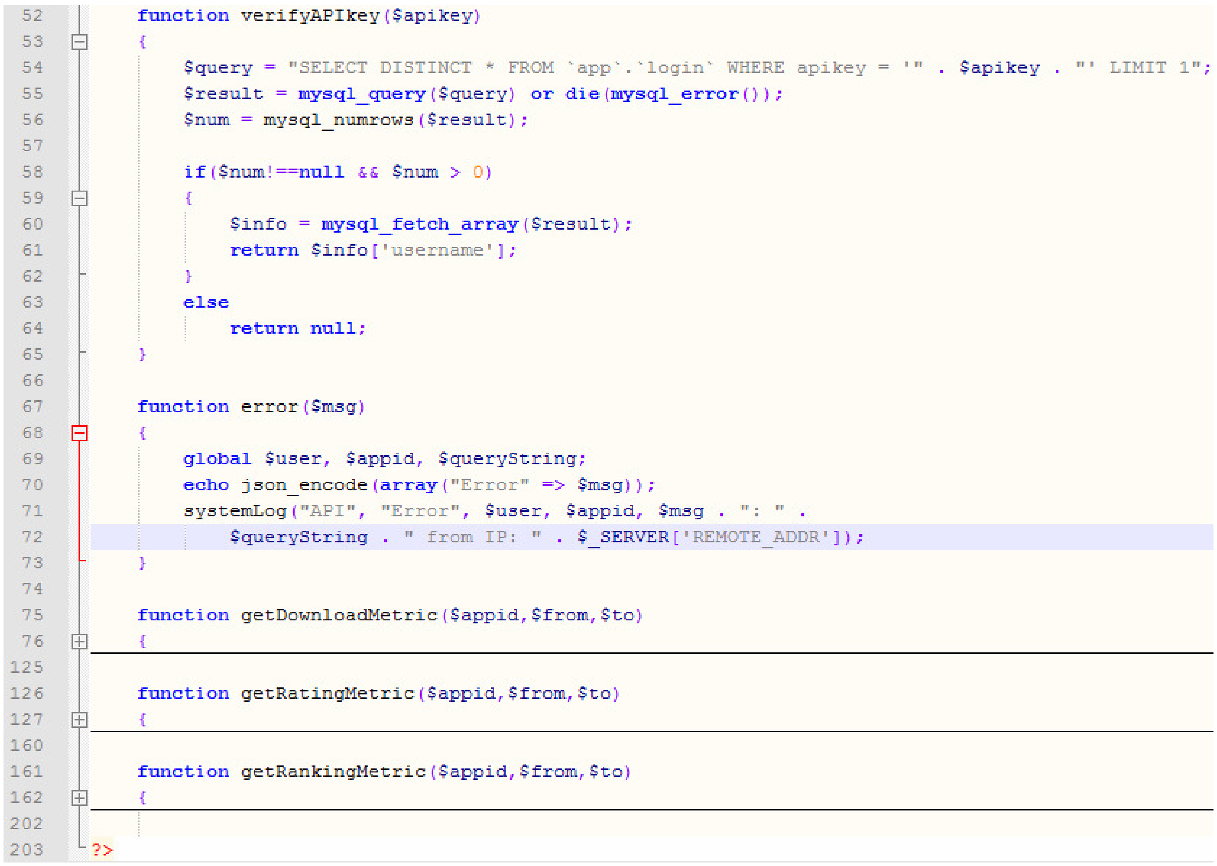
[129] “The Mobile Life”, The Mobile Life, [Online], Available at http://www.themobilelife.com (Accessed June 20, 2013).

[130] Julianne Pepitone, “5 data breaches: From embarrassing to deadly - Netflix accidentally reveals rental histories”, CNNMoney, [Online] December 14, 2010, Available at http://money.cnn.com/galleries/2010/technology/1012/gallery.5_data_breaches/index.html (Accessed June 21, 2013).

Appendix A – Sample API code



```php
<?php session_start(); ?>
<?php require_once('commonfunctions.php'); ?>
<?php
    header('Cache-Control: no-cache, must-revalidate');
    header('Expires: Mon, 26 Jul 1997 05:00:00 GMT');
    header('Content-type: application/json');

    $queryString = $_SERVER['QUERY_STRING']; //alla parametrar inkl
    $appid = isset ($_GET["appid"]) ? $_GET["appid"] : null;
    $metric = isset ($_GET["metric"]) ? $_GET["metric"] : null;
    $from = isset ($_GET["from"]) ? $_GET["from"] : null;
    $to = isset ($_GET["to"]) ? $_GET["to"] : null;
    $user = "";
    $ip = $_SERVER['REMOTE_ADDR'];

    mysql_connect(HOST,DB_USER,DB_PASSWORD);

    if(isset ($_GET["apikey"]))
    {
        $user = verifyAPIkey($_GET["apikey"]);

        if($user!==null)
        {
            if($metric!==null)
            {
                switch($metric)
                {
                    case "downloads":
                        getDownloadMetric($appid,$from,$to);
                        break;
                    case "ratings":
                        getRatingMetric($appid,$from,$to);
                        break;
                    case "rankings":
                        getRankingMetric($appid,$from,$to);
                        break;
                    default:
                        error("Not a valid metric type");
                }

                systemLog("API", "Request", $user, $appid, $queryString . " from IP: " . $ip);
            }
            else
                error("No metric specified");
        }
        else
            error("No user found");
    }
    else
        error("No API key specified");

```



```php
    function verifyAPIkey($apikey)
    {
        $query = "SELECT DISTINCT * FROM `app`.`login` WHERE apikey = '" . $apikey . "' LIMIT 1";
        $result = mysql_query($query) or die(mysql_error());
        $num = mysql_numrows($result);

        if($num!==null && $num > 0)
        {
            $info = mysql_fetch_array($result);
            return $info['username'];
        }
        else
            return null;
    }

    function error($msg)
    {
        global $user, $appid, $queryString;
        echo json_encode(array("Error" => $msg));
        systemLog("API", "Error", $user, $appid, $msg . ": " .
            $queryString . " from IP: " . $_SERVER['REMOTE_ADDR']);
    }

    function getDownloadMetric($appid,$from,$to)
    {

    function getRatingMetric($appid,$from,$to)
    {

    function getRankingMetric($appid,$from,$to)
    {

?>
```

TRITA-ICT-EX-2013:124