Causal Impact for App Store Analysis

William Martin
University College London, United Kingdom
w.martin@ucl.ac.uk

ABSTRACT

App developers naturally want to know which of their releases are successful and which are unsuccessful. Such information can help with release planning and requirements prioritisation and elicitation. To address this problem, I performed causal analysis on 52 weeks of popular app releases from Google Play and Windows Phone Store. The results reveal properties of successful releases in multiple app stores, and showcase causal analysis as a useful technique for developers seeking to know more about their software releases.

Keywords

App Store Mining and Analysis, Causal Impact

1. INTRODUCTION

App developers are often motivated to adopt high frequency release strategies [3, 6, 7, 8, 10, 12, 16, 17]. Updates can be made for reasons beneficial to users such as for fixes, improvements and new features, and they can also be made for reasons not beneficial to users, such as updating advertisement libraries [8]. Sometimes updates are made simply to try and stimulate an app's relative performance in the store [3], although high code churn can lead to decreased ratings [7]. Popular apps have been found to have high update frequencies, which do not correlate with their ratings [17].

Developers are privy to information about their own apps, such as sales and cash flow; as outsiders we do not have this information. However, app stores provide a method of measuring app performance, in the form of download ranks, ratings and reviews. I recorded metadata about the most popular apps in Google Play and Windows Phone Store over a 52 week period, in order to track their relative performance. Causal Impact Analysis was performed on 1,547 releases to identify those which may have caused significant changes in app performance.

ACM Student Research Competition (SRC) '16 Austin, Texas USA

ACM ISBN 978-1-4503-4205-6/16/05.
DOI: http://dx.doi.org/10.1145/2889160.2891033

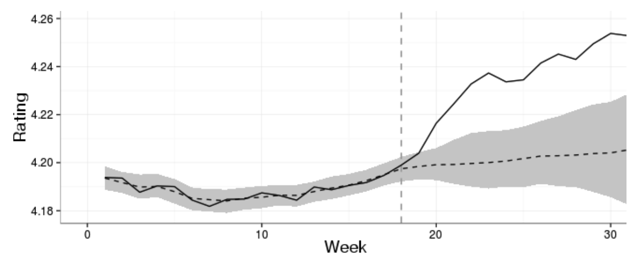

Figure 1: Causal impact graph for Carp Fishing Simulator. The solid line shows the observed vector, the dotted line shows the counter-factual prediction, and the shaded region indicates the 95% confidence interval. After the release (vertical dotted line) the observed rating is shown to deviate significantly from the prediction.

2. CAUSAL IMPACT ANALYSIS

Causal inference [11, 13] is a technique used for exploring events of significant impact on time series data. It is used in economic forecasting, and also, recently, in software defect prediction [4, 5, 26]. Causal Impact Analysis [1, 2] is a form of causal inference, that works by training on a pre-event data vector in order to make a counter-factual prediction. This prediction tells us the most likely course of the vector after the event, from which we can determine whether a significant change has occurred in the observed data vector.

The counter-factual predictor has three components: i) local trend, which works by sampling a normal distribution for noise between time points, based on the variance in the pre-event time period; ii) (optional) seasonal trend, which applies a repeating bias that sums to zero over its time period; iii) control trend, which applies coefficients to a set of data vectors from external unrelated objects, and serves to account for global variance in the model.

Fig. 1 shows an example significant release, identified by my analysis, for the "Carp Fishing Simulator".

3. STUDY DESIGN

I collected information on the rating, download rank, number of ratings and number of ratings in the last week from 307 Google Play apps and 726 Windows Phone Store apps over 52 weeks.

I applied Causal Impact Analysis using the `CausalImpact` framework, with no seasonal trend, and using the set of apps with no releases over the 52 week time period for the control trend. This set of apps consists of 97 apps in the Google Play dataset, and 397 apps in the Windows Phone Store dataset.

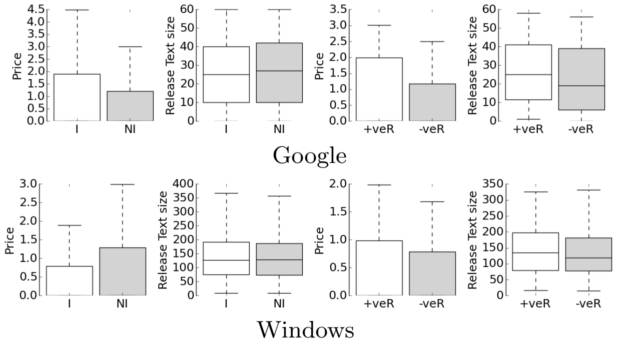

Figure 2: Box plots for Price and Release Text size for Google and Windows, comparing (I)mpactful against (NI) non-impactful releases, and (+veR) releases that positively impact rating against (-veR) releases that negatively impact rating.

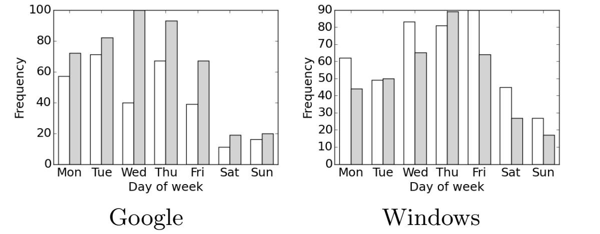

Figure 3: Histograms for day of release comparing impactful (white) and non-impactful (grey) releases, for Google and Windows.

The set of apps with potentially impactful releases consisted of 210 Google Play, and 539 Windows Phone apps; the set of releases for which there was sufficient prior and posterior information to perform causal impact analysis was 1,547.

I set a 99% confidence interval on results, meaning that only deviations with a less than 1% chance of being observed, if there had been no impact, were deemed significant. A separate experiment needed to be run for each performance metric and release, in order to train a separate predictor in each case. Each experiment then returned a p-value indicating the likelihood that the cumulative difference between the observed vector and the prediction could have occurred.

4. RESULTS

After applying causal impact analysis to releases in Google Play and Windows Phone Store, 301 out of 754 releases (40%) were found to be impactful in Google Play, and 437 out of 793 (55%) were found to be impactful in Windows Phone Store. Each impactful release was found to affect at least one of the four performance metrics positively or negatively. In both stores, approximately half of the releases impacted rating in some way: 20% positively impacting rating and 30% negatively impacting rating.

I grouped the impactful releases together in each store, in order to compare against releases which were not found to significantly impact app performance. I also compared the set of impactful releases that positively affected rating, against those that negatively impacted rating, in order to identify ways in which developers might be able to increase their ratings. The following candidate causes were considered, for the differences between these groups of releases: price, day of release, and length of release description text.

As shown in Fig. 2, across both stores, impactful releases were *more* likely to positively impact rating if they were more expensive. The length of release text also plays a factor: releases with longer (presumably more descriptive) release text, are more likely to be impactful, and to positively impact rating. Fig. 3 shows histograms for the releases on each day of the week, comparing impactful against non-impactful releases. We can see that releases are more likely to be impactful if released between Saturday and Tuesday in both app stores.

More results can be found in the technical report [15].

5. RELATED WORK

Past studies on app store release planning have looked into day of release [6, 10], potential reasons for updates [3, 8] and update frequency [17]. I built on this work and looked at releases which sigificantly impact app performance in their app stores, using empirical data mined from app stores as is commonly used in app store analysis [9, 14, 18, 22, 24, 25]. Previous studies have compared Blackberry and Android stores [23], and compared multiple Android stores [20, 21]; I compared properties between releases in Google Play and Windows Phone Stores. Past studies such as the one by Ruiz et al. [19] have used longitudinal data to monitor changes in rating; this study monitors such metrics, and looks for releases which may have caused significant deviations to them.

6. THREATS TO VALIDITY

In order to imply a causal relationship between releases and subsequent app performance, we need to apply a very large assumption indeed: that no external events may have caused the observed changes to an app's relative performance. Clearly, this assumption cannot be made, and so we cannot assume true causality between releases and the changes observed. However, we can use causal impact analysis to identify those releases after which a performance metric changes significantly; then, by grouping such 'impactful' releases together, we can ameliorate risk of external factors as causes for the observed changes, and can identify useful properties and heuristics for developers.

7. CONCLUSIONS AND FUTURE WORK

I have collected app metadata for the consistently most popular apps in Google Play and Windows Phone Store over a 52 week period. By performing causal impact analysis on individual releases, I have identified a subset of releases after which their app's performance changed significantly. I have grouped together these 'significant releases' in order to identify candidate causes for their impact, and found that price, day of release and descriptions of release content are factors in a release's likelihood to be impactful, and to positively impact app rating.

This study has shown that causal impact analysis is a useful method for identifying individual releases which may have affected an app's performance. This is particularly useful for developers, who may wish to run the tool on their own releases to work out specific successes or failures, in order to adapt their releasing or requirements development approach going forward.

8. REFERENCES

[1] K. H. Brodersen. CausalImpact. https://google.github.io/CausalImpact/CausalImpact.html. Retrieved 28th May 2015.

[2] K. H. Brodersen, F. Gallusser, J. Koehler, N. Remy, and S. L. Scott. Inferring causal impact using bayesian structural time-series models. *Annals of Applied Statistics*, 9:247–274, 2015.

[3] S. Comino, F. M. Manenti, and F. Mariuzzo. Updates Management in Mobile Applications. iTunes vs Google Play. *Centre for Competition Policy (CCP), University of East Anglia*, 2015.

[4] C. Couto, P. Pires, M. T. Valente, R. S. Bigonha, and N. Anquetil. Predicting software defects with causality tests. *Journal of Systems and Software*, 93:24–41, 2014.

[5] C. Couto, C. Silva, M. T. Valente, R. Bigonha, and N. Anquetil. Uncovering causal relationships between software metrics and bugs. In *Proceedings of the 16th European Conference on Software Maintenance and Reengineering (CSMR'12)*, pages 223–232, 2012.

[6] D. Datta and S. Kajanan. Do app launch times impact their subsequent commercial success? an analytical approach. In *International Conference on Cloud Computing and Big Data (CloudCom-Asia)*, pages 205–210. IEEE, 2013.

[7] L. Guerrouj, S. Azad, and P. C. Rigby. The influence of App churn on App success and StackOverflow discussions. In *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER'15)*, pages 321–330, 2015.

[8] J. Gui, S. Mcilroy, M. Nagappan, and W. G. Halfond. Truth in advertising: The hidden cost of mobile ads for software developers. In *Proceedings of the 37th International Conference on Software Engineering (ICSE'15)*, 2015.

[9] M. Harman, Y. Jia, and Y. Zhang. App Store Mining and Analysis: MSR for App Stores. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR'12)*, pages 108–111, 2012.

[10] N. Henze and S. Boll. Release your app on sunday eve: Finding the best time to deploy apps. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI'11)*, pages 581–586, 2011.

[11] P. W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):pp. 945–960, 1986.

[12] F. Khomh, T. Dhaliwal, Y. Zou, and B. Adams. Do faster releases improve software quality? an empirical case study of Mozilla Firefox. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR'12)*, pages 179–188, 2012.

[13] M. H. Maathuis and P. Nandy. A review of some recent advances in causal inference. *arXiv preprint arXiv:1506.07669*, 2015.

[14] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang. The app sampling problem for app store mining. In *Proceedings of the 12th IEEE Working Conference on Mining Software Repositories (MSR'15)*, pages 123–133, 2015.

[15] W. Martin, F. Sarro, and M. Harman. Causal impact analysis applied to app releases in Google Play and Windows Phone Store. Technical report, University College London, 2015.

[16] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman. A survey of app store analysis for software engineering. Technical report, University College London, 2016.

[17] S. McIlroy, N. Ali, and A. E. Hassan. Fresh apps: an empirical study of frequently-updated mobile apps in the google play store. *Empirical Software Engineering*, pages 1–25, 2015.

[18] R. Minelli and M. Lanza. Samoa – a visual software analytics platform for mobile applications. In *Proceedings of 29th International Conference on Software Maintenance (ICSM'13)*, 2013.

[19] I. J. Mojica, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan. An examination of the current rating system used in mobile app stores. *IEEE Software*, 2015. To appear.

[20] Y. Y. Ng, H. Zhou, Z. Ji, H. Luo, and Y. Dong. Which android app store can be trusted in china? In *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference*, COMPSAC '14, pages 509–518, Washington, DC, USA, 2014. IEEE Computer Society.

[21] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. P. Markatos, and T. Karagiannis. Rise of the planet of the apps: A systematic study of the mobile app ecosystem. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 277–290, New York, NY, USA, 2013. ACM.

[22] F. Sarro, A. A. Al-Subaihin, M. Harman, Y. Jia, W. Martin, and Y. Zhang. Feature lifecycles as they spread, migrate, remain and die in app stores. In *Proceedings of the Requirements Engineering Conference, 23rd IEEE International (RE'15)*. IEEE, 2015.

[23] M. D. Syer, B. Adams, Y. Zou, and A. E. Hassan. Exploring the development of micro-apps: A case study on the blackberry and Android platforms. In *Proceedings of the 2011 IEEE 11th International Working Conference on Source Code Analysis and Manipulation (SCAM'11)*, pages 55–64, 2011.

[24] M. D. Syer, M. Nagappan, A. E. Hassan, and B. Adams. Revisiting prior empirical findings for mobile apps: An empirical case study on the 15 most popular open-source Android apps. In *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research (CASCON'13)*, pages 283–297, 2013.

[25] Y. Yang, J. Stella Sun, and M. W. Berry. APPIC: Finding The Hidden Scene Behind Description Files for Android Apps. Technical report, Dept. of Electrical Engineering and Computer Science University of Tennessee, 2014.

[26] P. Zheng, Y. Zhou, M. R. Lyu, and Y. Qi. Granger causality-aware prediction and diagnosis of software degradation. In *IEEE International Conference on Services Computing (SCC'14)*, pages 528–535, 2014.