Causal Impact Analysis for App Releases in Google Play

William Martin, Federica Sarro and Mark Harman
University College London, London, United Kingdom
{w.martin, f.sarro, mark.harman}@ucl.ac.uk

ABSTRACT

App developers would like to understand the impact of their own and their competitors' software releases. To address this we introduce Causal Impact Release Analysis for app stores, and our tool, CIRA, that implements this analysis. We mined 38,858 popular Google Play apps, over a period of 12 months. For these apps, we identified 26,339 releases for which there was adequate prior and posterior time series data to facilitate causal impact analysis. We found that 33% of these releases caused a statistically significant change in user ratings. We use our approach to reveal important characteristics that distinguish causal significance in Google Play. To explore the actionability of causal impact analysis, we elicited the opinions of app developers: 56 companies responded, 78% concurred with the causal assessment, of which 33% claimed that their company would consider changing its app release strategy as a result of our findings.

CCS Concepts

•Software and its engineering → Software creation and management;

Keywords

App Store Mining and Analysis, Causal Impact

1. INTRODUCTION

Rapid release strategies can offer significant benefits to both developers and end users [18], but high code churn in releases can correlate with decreased ratings [14]. Releases occur for a number of reasons [1], such as updating libraries [15], or stimulating downloads and ratings [9], in addition to more traditional bug fixes, feature additions and improvements. McIlroy et al. [27] studied update frequencies in Google Play, finding that 14% of their studied apps were updated in a two-week period. Nayebi et al. [29] found that half of surveyed developers had a clear release strategy, and experienced developers believed it affects user feedback.

FSE'16, November 13-19, 2016, Seattle, WA, USA

DOI: http://dx.doi.org/10.1145/2950290.2950320

In this paper we introduce an approach to causal impact analysis to help app developers understand the impact of their releases. We mine and record time-series information about apps, in order to identify how releases can affect an app's rating and rating frequency. Causal Impact Analysis [6] is a form of causal inference [17, 21], which we use in this paper to identify sets of app releases that have caused a statistically significant ($p \leq 0.01$) change in subsequent user rating or rating frequency. Causal inference has previously been used primarily in economic forecasting, but it has also seen use for software defect prediction [10, 11, 44]. We introduce our "Causal Impact Release Analysis" tool, `CIRA`, which runs causal impact analysis for app store datasets.

We follow up on the causal impact analysis with more traditionally familiar (frequentist) inferential statistical analysis to further investigate the probabilistic evidence for potential causes. We investigate the influence of properties that are under developers' control such as price and release text, on both the positive and negative statistically significant releases. We use information retrieval to investigate the top terms and topics that occur in the release text of statistically significant releases, and also investigate the overall effects of release frequency on user rating behaviour. Furthermore, we contacted the developers of releases that were identified by `CIRA` as significant, to ask whether causal impact analysis is useful to them. To facilitate our study, we mined data, weekly, from Google Play between February 2015 and 2016.

Our contributions are as follows:

1. We introduce our tool, `CIRA`, for performing causal impact analysis on app store data[1].

2. We contacted developers of significant releases: 52 developers responded, 78% of whom agree with `CIRA`'s assessment, of which 33% claimed that their company would consider changing their release strategy.

3. We study Google Play app releases using `CIRA`, finding:

3.1. Paid app releases have a greater chance of affecting subsequent user ratings (a chance of 40% for paid apps, compared with 31% for free apps).

3.2. Paid apps with releases that had significant positive effects have higher prices.

3.3. Free apps with significant releases have a greater chance for their effects to be positive (37% for paid apps compared with 59% for free apps).

3.4. Releases that positively affected user ratings had more mentions of bug fixes and new features.

3.5. Releases that affected subsequent user ratings were more descriptive of changes.

[1] Available at http://www0.cs.ucl.ac.uk/staff/w.martin/cira

2. DEFINITIONS

In this section we define the app metrics recorded and the terminology used to refer to our data throughout the study.

2.1 Success Metrics

In order to assess app success, the following app-level metrics are used[2]:

(R) Rating: The average of user ratings made for the app since its first release on the store.

(N) Number of ratings: The total number of ratings that the app has received.

(NW) Number of ratings per week: The number of ratings that the app has received since the previous snapshot, which is taken a week earlier.

2.2 Developer-controlled Properties

The following observable app store properties are under developers' control:

(P) Price: The amount a user pays for the app (in GBP) in order to download it. This value does not take into account in-app-purchases and subscription fees, thus it is the 'up front' price of the app.

(L) Description length: The length in characters of the description, after first processing the text as detailed in Section 5.3, and further removing whitespace in order to count meaningful characters only.

(RT) Release text: The app's *description* and *what's new* sections from its app store page, in each case included only when they have changed from the app's previous release. An app's *description* and *what's new* sections can change at any point in time, but changes typically coincide with releases.

Version identifier: We determine a 'release' to have occurred if and only if the version identifier changes.

2.3 Data

We mine app data from Google Play between February 2015 and February 2016, as detailed in Section 5.1.

Full set: All apps mined in the time period. Some apps drop out of the store (for unknown reasons), but they are included in the full set for the duration in which they appear in the store. This full set consists of 38,858 apps.

Control set: The set of apps that have no new releases over the studied time period. We refer to this as the control set, as it is the benchmark by which we can measure changes in the releasing apps. Apps that drop out of the store are not included in the control set because consistency is required for a reliable control set. This control set consists of 680 apps.

Target set: The set of app releases that occur in the studied time period and occur at least 3 weeks after the previous releases, and at least 3 weeks before the next release. The target releases have some longevity which suggests they are more than 'hotfixes' for recently introduced bugs. They also have a non-trivial window of data on either side so that we can observe any effect the release may have had on the app's success. This ensures sufficient data availability to perform causal impact analysis. Apps that drop out of the store are included in the target set if they include adequate prior and posterior information as defined above. This target set consists of 14,592 apps and 26,339 releases.

[2]Number of downloads can be used as a metric, however most app stores (including the one analysed in this work) do not make this information publicly available.

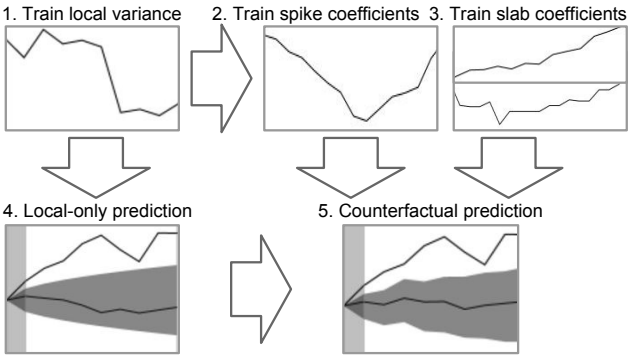

Figure 1: CIRA workflow showing the local and global variance components. The resultant counterfactual predicition is compared with the observed vector after the release in order to compute the probability that the observed vector would have been predicted. If this probability is low (≤ 0.01) it indicates that there was a significant impact at the time of the release.

3. CAUSAL IMPACT RELEASE ANALYSIS: CIRA

Causal inference is a method used to determine the causal significance of an event or events, by evaluating the post-event changes using observational data. A traditional approach to causal inference is differences-in-differences analysis [4], which compares the differences between the prior and posterior vectors (i.e. observations) of a group (e.g. mobile apps in a store) that receives a given intervention (which in our case is a release), against a group that does not receive this intervention (i.e. control set). Conversely, the Causal Impact Analysis method [6], based on state-space models, treats each vector separately, in each case using the full control set to provide global variance. Multiple apps typically do not receive the same intervention (release) at the same time, which makes the former method difficult to apply in our case. This motivates our use of Causal Impact Analysis, which we apply to app releases using our tool, `CIRA`.

`CIRA` trains a Bayesian structural time-series model [17, 39] on the data vector for each target release, using a set of unaffected data vectors known as the control set (defined in Section 2.3). This enables the model to make a prediction of the data vector in the posterior time period accounting for local and global variations. Each metric (R, N, NW) and each release, requires an individual experiment, as the method works each time on a single data vector.

Fig. 1 shows the overall `CIRA` workflow:

1. Train the local trend parameters using the deviation of the observed vector in the prior time period. This is used to sample changes and compute the confidence interval.

2. Compute the 'spike' [39] for the observed vector from the set of controls, and assign coefficients for them, in order to help us make an accurate prediction.

3. Use the rest of the control set as the 'slab' [39], assigning equal coefficients for them, in order to account for global variations in the dataset.

4. Make a set of predictions using the local trend trained earlier, sampling for changes and noise.

5. Combine local-only prediction with control changes mutliplied by their coefficients. This is the counterfactual prediction [6].

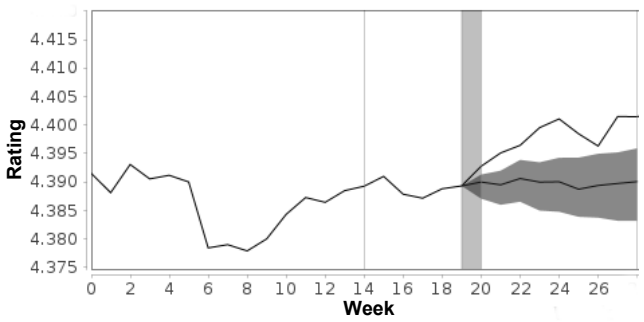

Figure 2: Causal impact analysis for the rating of an invoice management application in Google Play. After the target release (shaded vertical bar), the observed vector (solid line plotted throughout) clearly moves outside the confidence interval (shaded region) and deviates significantly from the counterfactual prediction (solid line inside shaded region).

We then compute the cumulative pointwise difference between the observed vector and our prediction, normalised to the length of time at which each deviation occurs, and compute the cumulative probability from our local trend parameters. A low value ($p \leq 0.01$) indicates that the success metric changed significantly; see for example Fig. 2, which shows that the rating of an invoice management application deviates significantly from the predicted vector. By setting our threshold to be 0.01, we have a 0.01 probability of claiming a significant change where one does not exist, and therefore expect roughly 1% false positive rate.

We use CIRA to identify the set of releases (see RQ3), for which there is evidence of a statistically significant effect on the metrics we collect. Our subsequent inferential statistical analysis complements this, pointing to the set of potential properties which may play a role in this causal significance. Of course, the degree to which we can assume causality relies on the strength of our causal assumptions, that the control set is unaffected by the release, and the relationship of the control to the released app is unchanged.

In any and all causal impact analyses, it is of course impossible to identify external properties that might have played a role in the changes observed. In the case of app stores, our current analysis cannot, for example, identify advertising campaigns, timed to coincide with the new release. However, if such an external factor does have a significant effect, then our causal impact analysis may detect it. We ask developers in RQ5 if they are aware of external factors which may have caused the observed changes, in order to determine how often this may be the case.

4. RESEARCH QUESTIONS

This section explains the questions posed in our study and how we approach answering them.

RQ1: Do app metrics change over time?

Before performing a detailed analysis of the changes over time, we first set a baseline by establishing whether app success metrics change over time or between different snapshots. Using the metrics R, N, NW and L as defined in Section 2, we compute their standard deviation over 52 weeks. These distributions are drawn using box plots, which enables us to establish whether metrics change over time, and to what extent any such changes occur.

We determine whether app success changes more or less for those apps that have releases, by comparing plots for those apps that have no releases (the control set), with releasing apps (the target set). If the metrics for the target set change over time more than those in the control set, this motivates further analysis of releases that could *cause* the observed changes. It is expected that releasing apps would show greater deviation in description length, describing changes and features.

RQ2: Do release statistics have a correlation with app success?

We build on this analysis by measuring whether app success is correlated with the number of app releases in the time period studied, and whether it is correlated with the time interval between releases. This will show whether a large (or conversely, small) number of releases might lead to increased success, and likewise for release interval.

For both of these experiments, only apps in the target dataset are used. We perform correlation analysis between the number of releases of each app and their value for the metrics R and N at the end of the time period. We do not use NW because this number is set on a per-week basis, but instead use the change in R and N from the first snapshot to the last, denoted ΔR and ΔN respectively. Additionally we do not use L because this does not represent app success.

RQ2.1: Does the number of releases have a high correlation with app success? We perform correlation analysis between the number of releases in the studied time period and the metrics R, ΔR, N and ΔN at the end of the time period.

RQ2.2: Does the median time interval between releases have a correlation with app success? We perform correlation analysis between the median interval between releases of each app and the metrics used in RQ2.1.

RQ3: Do releases impact app success?

There are two limitations to correlation analysis. Firstly, correlation analysis seeks an overall statistical effect in which a large number of apps and their releases participate. However, it may also be interesting, from the developers' point of view, to identify *specific* releases after which an atypically high change occurs in subsequent success, accounting for 'background' app store behaviour; general correlation analysis is not well-suited to this more specific question. Secondly, of course, as is well-known, any correlation observed does not necessarily imply the presence of a cause (correlation is not causation). Therefore, even were we to find strong correlations, this would not, in itself, help us to identify causes. This motivates our use of causal impact analysis. We apply causal impact analysis on each target release to see whether it caused a statistically significant change in any of the metrics R, N or NW, defined in Section 2. Causal impact analysis is described in Section 3.

RQ3.1: What proportion of releases impact app success? We compute the proportion of apps whose releases have affected success, and the proportion of overall releases. We group results under the metrics affected: R, N, NW and the intersection of R and NW, overall and for positive or negative changes.

RQ3.2: How does the causal control set size affect results? Causal impact analysis uses a control set: a set of unaffected data vectors, which in our case is the set of apps that have zero releases in the period studied. As the set is used in two different ways ('spike' and 'slab' [39], as explained in Section 3), we test whether the set size makes a difference that could influence our results. Our approach is similar to the experiment using different control sets in the study by Brodersen et al. [6]. We compute the causal impact analysis results for each metric with the full target dataset, using repeatedly halved control sets of size 340, 170, 85, 42, 21, 10, 5, and 3 which are each randomly sampled from the maximal set of 680 non-releasing apps.

We compute the agreement between each set of results and the results used to answer RQ3.1. Agreement is defined as $(\frac{YY+NN}{total})$, where YY indicates a significant change as detected on both datasets, NN indicates no significant change detected on both datasets, and total is a count of all results (including disagreements). We also compute the Cohen's Kappa [8], which takes into account the chance for random agreement and is bounded above by the agreement. We compute a second set of results using 680 control vectors, which will show the expected difference between consecutive runs with the same control set, since there is a random component in the predictive model.

RQ4: What characterises impactful releases?

We use the causal impact analysis results from RQ3 to analyse significant and non-significant releases.

RQ4.1: What are the most prevalent terms in releases? We pre-process the release text from all releases in a given store as described in Section 5.3, then identify the 'top terms' (most prevalent) for each set of releases using two methods: **TF.IDF** [22] and **Topic Modelling** [5].

We train both methods on the release text corpus, treating each instance of release text as a document. For both methods, we sum the resultant scores (probabilities) for terms (topics) over each set of releases. We restrict ourselves to only the top three terms to avoid over specialisation.

The topic model is trained using 100 topics. We chose 100 topics for three reasons: i) the number of documents in the corpus (updated release text from target releases), each consisting of tens to hundreds of words, which is a large sized corpus; ii) we do not wish to over-generalise, nor over-fit; 100 topics will allow diversity without assigning the same topic to every document with a release; iii) 100 topics is a common selection for non-trivial datasets, serving as the default setting in `GibbsLDA++`[35] and `JGibbLDA`[36]. Intuitively, the choice of 100 topics allows for diversity in the trained topics, without unduly elevating the risk of training a topic that is overly specific to an app or release.

RQ4.2: How often do top terms and topics occur in each set of releases? We compute the counts in each set of releases that contain top terms that emerge from TF.IDF and topic modelling, as performed in RQ4.1. We apply a bag-of-words model, which ignores the ordering of the words in each document. This eliminates the need to check for multiple forms of text that discuss the same topic.

RQ4.3: What are the effects of each of the candidate causes? We select the sets of statistically significant and non-significant releases, as well as the sets of significant releases that increased and decreased rating, and compare the distributions of several developer-controlled properties. This will help us to establish potential causes that may have led to the releases being significant, or positively affecting an important success metric for developers: the rating.

We consider properties of the release that lie within the control of the developer: (P)rice, (RT_{size}) the size of release text (defined in Section 2.2) in words, and (RT_{change}) the change in RT_{size} on the week of release. For each of these properties, we use the Wilcoxon test and Vargha and Delaney's $\hat{A}_{12}$ effect size comparison [41], to identify statistically significant differences between causally significant releases and non-significant releases. We use the untransformed Vargha and Delaney's comparison [31] because we are only interested in the raw probability value it produces. In case a statistical difference is found for a given aspect, we use box plots to understand if this property may play a role in the changes observed (i.e. it may be a candidate cause).

At this point we will have identified a subset of releases that exhibit statistically significant success effects, and we will have identified further differences between sets of significant and non-significant releases, as well as the differences between positive and negative significant releases. To further explore the actionability potential of the tool and our findings, we ask the following research question.

RQ5: Is causal impact analysis useful to developers?

Because there is no ground truth, only different implications of any discovered statistical significance, we can only answer this question semi-quantitatively. To determine whether our results are useful we simply ask the developers of causally significant releases, as determined by our tool, `CIRA`, whether they agree with the classification. We email developers via the email addresses contained on their app store pages, informing them of the significant release and proposing to send a report detailing the tool's findings. We expect a large proportion of these emails may fail to reach a human, and can only confirm contact is established if we hear back from a developer. Once contact is established with the app's developers, we ask them the following questions[3]:

Agree with detected significance: we ask if the developers of the app agree with `CIRA`'s assessment that the release was significant.

External cause of changes: we ask if the company is aware of an external event which may have caused the detected significant change, such as advertising campaigns.

Would change strategy: we ask if the company would consider changing their release strategy based on findings.

Receiving further reports: we ask if the company is interested in receiving further reports for their app releases.

Learning contributing factors: we ask if the company is interested in learning more about the characteristics of significant releases, from the results of our study.

5. METHODOLOGY

This section describes the methods used in our study for extracting and analysing app store data.

5.1 Data Mining

We gathered a set of 38,858 apps from the Google Play store from February 2015 to February 2016, which had appeared in a Google Play free, paid, or any category's 'top 540' list at least once in the year before February 2015.

[3]Questionnaire available on the accompanying web page.

We used this list because it includes apps that are likely to have been downloaded and reviewed, and so have measurable success, yet enables a large dataset that is likely to consist of many app releases.

Only 1% of apps release software updates more frequently than once per week, and these have already been studied in detail by McIlroy et al. [27]. For this reason, we set the time interval between between data collection to one week.

Some apps dropped out, were removed or deleted from the store, or had other technical issues that prevented successful data mining on a particular week. Gaps in the data such as this are to be expected, and do not prevent causal impact analysis from running.

We extract app metrics from each of the 38,858 apps including price, rating, number of ratings, description, *what's new* and version. Google Play reports rounded app ratings (rounded to 1 decimal place), thereby creating a potential source of imprecision, which we would like to overcome. We therefore calculate the (more precise) Google Play average ratings, using the extracted numbers of ratings in each of the five star rating categories (from 1-5).

5.2 Explanation of the Frequentist Inferential Statistical Analysis Techniques Used

This subsection explains the role played, in our overall analysis, by traditional frequentist inferential statistical techniques (with which software engineers are most likely to be already familiar), while Section 3 explains causal impact analysis (which is comparatively less widely used in the domain of software engineering).

We use correlation analysis in order to understand correlations between the observed app metrics and both the quantity and interval of their releases. However, since it is well known that 'correlation does not imply causation', we further investigate the causal effect of releases using causal impact analysis (explained in more detail in Section 3). We follow up the causal impact analysis with a nonparametric inferential statistical analysis, to provide further evidence as to the relative likelihoods that each of the developer-controlled app properties that we observed plays a role in the effects detected by causal impact analysis.

In RQ2 and RQ4 we use Pearson and Spearman statistical correlation tests. Pearson introduced the measurement of linear correlation [34], and Spearman subsequently extended Pearson's work to include rank-based correlation [40]. Each correlation metric reports a rho value and a p value. The p value denotes the probability that a rho value is different to zero (no correlation). A rho value of 1 indicates perfect correlation, while -1 indicates perfect inverse correlation, and 0 indicates no correlation. Values between 0 and 1 (-1) indicate the degree of correlation (inverse correlation, respectively) present.

In RQ3 and RQ4 we also compare distributions using a two-tailed unpaired non-parametric Wilcoxon test [42], that tests the Null-hypothesis that the result sets are sampled from the same distribution. We also compare the result sets using Vargha and Delaney's $\hat{A}_{12}$ effect size comparison test [41], which results in a value between 0 and 1, indicating the likelihood that one measure will yield a greater value than the other.

In our case, we apply these inferential statistical tests to examine differences in properties between sets of releases. We compare the set of releases that have caused a significant change (according to the previous causal impact analysis) to those which have not, and compare the set of releases that positively affected rating to those that negatively affected rating. The p value is the probability that we would observe the difference in median values we find, or one more extreme, given that there is, in fact, no difference in the two distributions from which the releases are drawn. It is a conditional probability, usually used to reject the Null-hypothesis (that there is no difference).

However, in our case, a prior causal impact analysis has revealed that there is a significant causal difference between the two sets, yet it remains unknown what this cause is. Causal impact analysis does not fully overcome the problem that 'correlation is not causation', but it does provide evidence for causal significance (in our case causal effect of a release on subsequent success). Therefore, each p value from our subsequent Wilcoxon tests can be interpreted as an indication of a potential cause for the difference observed, which, if low ($p \leq 0.01$) warrants further investigation of the associated property.

Since we are computing multiple p values, the reader might expect some kind of correction, such as a Bonferroni or Benjamini-Hochberg [3] correction for multiple statistical testing at the (traditionally popular) 0.05 probability level (corresponding to the 95% confidence interval). However, since we are *not* using p values to test for significance; should a p value lie above this (corrected) threshold, then this *does not* necessarily indicate that the property does not contribute to the observed causal significance. Quite the contrary; since we have already observed that there exists a causal significance, then any property that exhibits a low p value ($p \leq 0.01$) remains that with a chance of having *some* influence on the causal effect, from amongst those properties assessed using inferential statistics.

5.3 Information Retrieval Techniques

To answer RQ4, we perform information retrieval analysis on release text, using both TF.IDF and Topic Modelling. We use two different techniques, to increase the confidence with which we can identify the top terms that occur in the release text of app releases that exhibit causal significance.

Filtering: Text is cast to lower case and filtered for punctuation and stopwords, using the English language `stopwords` from the Python NLTK data package[4].

Lemmatisation: Each word is processed by the Python NLTK `WordNetLemmatizer`, in order to be transformed into its 'lemma form', to homogenise singular/plural, gerund endings and other non-germane grammatical details.

TF.IDF: TF.IDF [22] finds 'top terms' in release text: each term in each document is given a score of TF (Term Frequency) multiplied by the IDF (Inverse Document Frequency). The IDF is equal to the log of the size of the corpus divided by the number of documents in which the word occurs.

Topic Modelling: We use topic modelling [5] to find the top topics in release text. Topic modelling is a generative technique that trains a probabilistic graphical model on a set of unstructured textual documents, under the assumption that they are generated from a set of latent topics.

[4] `nltk.corpus.stopwords.words('english')`

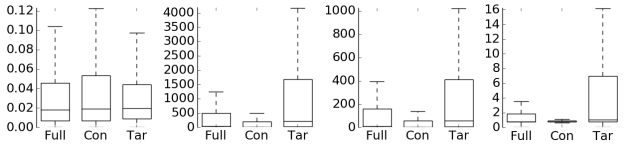

Figure 3: RQ1: Standard deviation box plots for (R)ating, (N)umber of ratings, (NW) number of ratings per week and (L)ength of description. Each plot shows the standard deviations for apps in the (Con)trol set, (Tar)get set and Full set.

6. RESULTS

This section answers the questions posed in Section 4.

RQ1: Do app metrics change over time? We can see from the box plots in Fig. 3 that the metrics (R)ating, (N)umber of ratings, (NW) number of ratings per week and (L)ength of description do change over time, because their median standard deviation is always positive. The deviation in number of ratings and number of ratings per week is high, but very low for rating. This is because, for apps with many ratings, even a small change corresponds to thousands of users rating higher or lower than the established mean.

We can see that the deviation is approximately even between the control and target datasets for rating deviation; this is a surprising result, and indicates that either a) ratings are unstable even for stable, established, non-releasing apps, or b) app releases have little effect over all, globally detectable, ratings. The finding of a low deviation in the target set means that a causally significant release (that affects rating), has a good chance to 'stand out from the crowd'.

The box plots show that deviations in number of ratings and rating frequency length are significantly higher for the target releasing dataset, than for all apps in the dataset and for the control set. This is expected, and shows some utility to app releases: to increase user activity in downloading and rating the apps, and perhaps to increase the user base.

The deviation in description length is higher for the target dataset as expected, suggesting that descriptions are updated to provide information about releases. This finding supports the intuition that descriptions may be used as a channel of communication between developers and users.

Answer to RQ1: Do app metrics change over time? The metrics (N)umber of ratings and (NW) number of ratings per week show a high standard deviation for apps between February 2015 and February 2016, but (R)ating shows only a small deviation. The deviation in user rating frequency is higher for the target releasing dataset, suggesting that app releases lead to user activity.

RQ2: Do release statistics have a correlation with app success? Having established a baseline, we now measure the correlations between the number and interval of releases and success metrics.

RQ2.1: Does the number of releases have a high correlation with app success? Table 1 presents the results of correlation analysis between release quantity and median interval, and app metrics for the target dataset.

Table 1: RQ2: Significant ($p \leq 0.01$) correlations between each of number of releases and median release interval, and success metrics (R)ating and (N)umber of ratings at the end of the time period studied, as well as the change in these metrics from first to last week.

Release statistic	Method	R	ΔR	N	ΔN
Quantity	Spearman	0.13	0.09	0.27	0.30
	Pearson	0.14	-	0.09	0.10
Median interval	Spearman	-0.12	-0.06	-0.19	-0.21
	Pearson	-0.09	-0.02	-0.04	-0.05

We only report correlation coefficients (the rho values) that are deemed significant ($p \leq 0.01$), i.e., where there is sufficient evidence that rho $\neq 0$.

The results in Table 1, indicate only weak significant correlations between the success metrics and their change over the time period studied. The strongest correlation, for ΔN where rho = 0.30, is still too weak to definitely suggest a strong relationship. We therefore conclude that there is no strong overall correlation between release frequency and the app metrics we collect, but there is evidence for a weak correlation between number of releases and number of reviews accrued over a year.

RQ2.2: Does the median time interval between releases have a correlation with app success? Table 1 presents the results of correlation analysis between release interval and app metrics. As these results revealed, there is little evidence for any strong correlation between the median inter-release time period and the app metrics we collect. Our findings corroborate and extend the recent findings by McIlroy et al. [27], who reported the rating was unaffected by release frequency in the Google app store. This is interesting because there is evidence that app developers release more frequently when an app is performing poorly [9]; our results indicate that this, perhaps rather desperate behaviour, is unproductive.

Answer to RQ2: Do release statistics have a correlation with app success? Neither higher numbers of releases nor shorter release intervals correlate strongly with changes in success.

RQ3: Do releases impact app success? The results from RQ1 and RQ2 have established that app rating metrics do vary over releases, but that the number of releases and time intervals between releases are not important factors in determining these changes. This makes causal impact analysis potentially attractive to developers. With it, developers can seek to identify the set of specific releases that had a higher effect on success, using evidence for significant changes in post-release success compared with the set of non-releasing apps. This is the analysis to which we now turn in RQ3.

RQ3.1: What proportion of releases impact app success? Table 2 presents overall summary statistics for the results of causal impact analysis. The 'Apps' row indicates the number of apps summarised as part of the target dataset, and the 'Target releases' row indicates the number of releases these apps underwent in the studied time period that were outside of a 3-week window of other releases.

Table 2: RQ3.1: Causal impact analysis results, indicating the number of casually significant releases over the target dataset, and the number of causally significant releases in each sub-group (as determined by a release's effects on subsequent app success).

Details of target releases (percentages reported over 26,339 target releases)			
Type	**Total (of target)**	**Apps**	14,592
Non-significant	17,639 (67.0%)	**Target releases**	26,339
Significant	8,700 (33.0%)	**Control apps**	680

Details of significant releases (percentages reported over 8,700 significant releases)			
Metric	**Total (of significant)**	**+ve**	**-ve**
R	4,781 (55.0%)	2,563 (29.5%)	2,218 (25.5%)
N	4,747 (54.6%)	4,747 (54.6%)	0
NW	2,226 (25.6%)	862 (9.9%)	1,364 (15.7%)
R ∩ NW	701 (8.1%)	220 (2.5%)	199 (2.3%)

Table 3: RQ3.2: Results reveal minimal effects arise from different control set size choices using the full set of 26,339 target releases. Control sets were sampled randomly from the maximal control set of 680 apps. The '680' result calibrates, by assessing (maximal) agreement between repeated runs on maximal sets, capturing variance due to the inherent underlying stochastic nature of causal analysis.

		Control Set								
	Metric	3	5	10	21	42	85	170	340	680
Agreement	R	0.93	0.93	0.93	0.91	0.91	0.91	0.93	0.93	0.94
	N	0.91	0.91	0.91	0.91	0.91	0.90	0.91	0.91	0.92
	NW	0.97	0.97	0.97	0.97	0.97	0.95	0.97	0.97	0.98
	All	0.94	0.94	0.94	0.93	0.93	0.92	0.93	0.94	0.94
Cohen's Kappa	R	0.78	0.76	0.76	0.72	0.72	0.73	0.76	0.77	0.79
	N	0.69	0.70	0.70	0.70	0.69	0.66	0.69	0.69	0.73
	NW	0.78	0.78	0.80	0.83	0.81	0.71	0.81	0.83	0.85
	All	0.75	0.74	0.75	0.74	0.73	0.70	0.74	0.76	0.78

Those releases that occur near the beginning or end of the time period will therefore not have sufficient information available, and so causal impact analysis cannot be applied. Hence we select a subset of releases that must also belong in the range of weeks [4, 49], out of a possible [1, 52]. Of these target releases, some are significant and some are not according to causal impact analysis. As Table 2 reveals, we found that 33.0% of releases were significant.

The remainder of Table 2 reports the observed change in success metrics for significant releases, thereby identifying candidate causes of these effects. For each success metric change, we report the total number of releases that exhibited a significant change in the associated metric and the percentage (of all app releases) that exhibited the change. We further subdivide this total into those that are considered 'positive' and 'negative' (from a developer's perspective).

From the 33.0% of significant app releases in Google Play, approximately a third (30.0%) affected more than one success metric. The releases of most potential interest to developers are likely to be those that affect both rating and rating frequency (due to their potential for increasing user base and revenue). Of these, there were 701 significant releases.

These results support the hypothesis that there is a subset of releases that cause significant changes to their app's success in the store.

RQ3.2: How does the causal control set size affect results? Table 3 reports the effect of choosing different control set sizes, from among those apps that did not undergo any release during the time period studied.

The results in Table 3 reveal strong agreement for each control set size, indicating that the model is stable in our case (using non-releasing apps). We can see from our two runs (using the full 680 apps in the control set), that there is between 0.92 and 0.98 agreement due to the stochastic element of Causal Impact Analysis. Our results show that restricting our choice of control set does not have advantages, and so we opt to use the full 680 apps for our control set for subsequent experiments.

Answer to RQ3: Do releases impact app success? There is strong evidence ($p \leq 0.01$) that 33% of the target releases in the Google Play store significantly affected a success metric, and approximately 11% significantly affected more than one success metric.

RQ4: What characterises impactful releases? The finding from RQ3, tells us that there are significant releases, but it cannot identify the causes, merely that there has been a significant change in post-release success. We now turn to identify any candidate causes, which may have played a significant role in the changes we have observed, and analyse their effects.

RQ4.1: What are the most prevalent terms in releases? Table 4 reports the results of information retrieval, using TF.IDF and the topic modelling on the release text of significant releases. In this table, we consider only those apps for which release text is available. The results show only the top TF.IDF terms, and terms from the top topic, respectively; thereby indicating the most prevalent terms and topic.

Of the 26,339 target releases (those with sufficient evidence for causal impact analysis), 20,014 have release text available. The remainder of the table consists of four overall columns, giving the metric for which a release is found to be significant (leftmost column), followed by the most prevalent terms (for TF.IDF) and topics (for topic modelling), followed by a subdivision of these prevalent terms into those whose effects are positive and negative (from a developer's perspective).

Table 4 reveals that terms and topics themed around bug fixes and features occur frequently overall in the text of significant releases. The text of releases that positively or negatively affected rating is slightly more specific to certain sets of apps, i.e. "card map feature", "wallpaper live christmas", yet still appear to refer to features. These observations motivate our subsequent analysis in RQ4.2 for the terms 'bug' and 'fix' and 'new' and 'feature'.

RQ4.2: How often do top terms and topics occur in each set of releases? Table 5 shows the number of occurrences, within significant and non-significant releases, of the terms 'bug' and 'fix', and 'new' and 'feature' which emerged as the top terms from our information retrieval in RQ4.1.

We can see from the results in Table 5 that the terms 'bug' and 'fix' are far more common than 'new' and 'feature' in both significant and non-significant releases. This is unsurprising because bug fixing occupies a large proportion of development effort [43]. However, it is noteworthy that, in both cases, the terms are more common in the releases that positively affected metrics, as opposed to those that negatively affected metrics.

Table 4: RQ4.1: Top release text terms: TF.IDF terms on the left and Topic Modelling topics on the right. Terms which occur in all groups are removed for comparison: new, app, game, play, free

Type	**Overall**		**Apps**	14,592		
Non-significant	feature fix time	mode challenge friend	**Target releases**	26,339		
Significant	feature fix device	hero monster battle	**Release text**	20,014		
Metric	**All**		**+ve**		**-ve**	
R	feature word fix	hero monster battle	feature word time	wallpaper live christmas	bible fix support	account mobile card
N	feature word time	wallpaper live christmas	fix feature word	wallpaper live christmas		
NW	map feature word	tip local city	card map feature	hero monster battle	map feature time	tip local city
R ∩ NW	feature video map	wallpaper live christmas	account card feature	account mobile card	video time feature	hero monster battle

Table 5: RQ4.2: Occurrences of the terms 'bug' and 'fix' and 'new' and 'feature' in release text.

Type	**Overall**	**Apps**	14,592
Non-significant	4,690 / 13,200 (35.5%)	**Target releases**	26,339
Significant	2,432 / 6,809 (35.7%)	**Release text**	20,014

Metric	**Total**	**+ve**	**-ve**
R	1,336 / 3,794 (35.2%)	745 / 2013 (37.0%)	591 / 1781 (33.2%)
N	1,300 / 3,693 (35.2%)	1,300 / 3,693 (35.2%)	0 / 0
NW	626 / 1,783 (35.1%)	260 / 685 (38.0%)	366 / 1098 (33.3%)
R ∩ NW	190 / 578 (32.9%)	63 / 179 (35.2%)	48 / 170 (28.2%)

Occurrences of the terms 'bug' and 'fix' in release text.

Type	**Overall**	**Apps**	14,592
Non-significant	2,473 / 13,200 (18.7%)	**Target releases**	26,339
Significant	1,355 / 6,809 (19.9%)	**Release text**	20,014

Metric	**Total**	**+ve**	**-ve**
R	795 / 3,794 (21.0%)	437 / 2013 (21.7%)	358 / 1781 (20.1%)
N	687 / 3,693 (18.6%)	687 / 3,693 (18.6%)	0 / 0
NW	393 / 1,783 (22.0%)	151 / 685 (22.0%)	242 / 1098 (22.0%)
R ∩ NW	149 / 578 (25.8%)	52 / 179 (29.1%)	37 / 170 (21.8%)

Occurrences of the terms 'new' and 'feature' in release text.

Table 6: RQ4.3: Probabilistic analysis of candidate contributions (P)rice, (RT_{size}) release text size and (RT_{change}) release text changes.

	Significant / Non-significant		**+ve R / -ve R**	
Metric	**Wilcoxon**	$\hat{A}_{12}$	**Wilcoxon**	$\hat{A}_{12}$
P	0.000	0.539	0.000	0.419
RT_{size}	0.000	0.532	0.006	0.479
RT_{change}	0.110	0.505	0.434	0.499

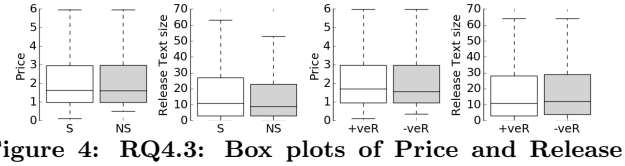

Figure 4: RQ4.3: Box plots of Price and Release Text size, comparing (S)ignificant and (NS) non-significant releases, and releases that increased rating (+veR) and decreased rating (-veR).

RQ4.3: What are the effects of each of the candidate causes? Based on the low p-values reported in Table 6 we further analyse price and the size of the release text as candidate causes. Fig. 4 presents box plots showing the distribution of (paid app) price and release text size, comparing significant releases against non-significant releases, and releases that positively affect rating against those that negatively affect rating. Since 61% of the apps are free, we plot paid apps in the price boxplot, and compute the proportion of free and paid apps in each set, as well as the mean and median prices.

The results for price are interesting, somewhat surprising, and nuanced. We found that significant releases (positive and negative) have higher prices than non-significant releases. The mean prices of all significant releases (including free) and all non-significant releases are £0.79 and £0.59, respectively. Paid releases were more likely to be significant: 40.2% of paid app releases were significant, compared with 30.7% of free app releases. A somewhat surprising finding is that higher priced (paid) app releases are more likely to have a positive effect: a mean of £3.25 and median of £1.72 compared with £2.45 and £1.58 for those with negative effects, respectively. However, a greater proportion of significant paid app releases *negatively* affected rating: 62.6% compared with 41.0% for free apps.

Overall, a larger proportion of significant releases are paid than non-significant releases (29.6% compared with 21.7%, respectively). As a result, the mean price of significant releases is higher by £0.20. Of course, a price difference of £0.20 in Google Play appears relatively trivial. However, the difference in revenue that accrues can be substantial. We conservatively calculate that for the app, OfficeSuite Pro + PDF (which had a release on 23rd December 2015, for which we observed a significant effect on the subsequent rating), over its (minimum) 50,000 installs [13], the price difference of £0.20 extrapolates to (minimum) £10,000 in accrued revenue. Our revenue calculation is particularly conservative, since at the time of this release, OfficeSuite Pro was priced at £11.66. This finding suggests that developers need not fear a 'race to the bottom' with competitors over pricing. Unsurprisingly, these results confirm the intuition that users can be expected to be price-sensitive.

We can also see from Table 6 that there are significant differences between the distributions of price and release text size, comparing significant with non-significant releases and releases that increase rating to those that decreased rating. The median number of changed, stopword-filtered words in significant release text is 11, compared to only 9 for non-significant release text. This provides evidence that users may be influenced by release text, but the effect size is relatively small. Nevertheless, developers might wish to spend time carefully choosing their release text to maximise positive influences on the users.

Answer to RQ4: What characterises impactful releases? There is evidence that releases that significantly affect subsequent app success have higher prices and more descriptive release text. Releases that positively affect rating are more common in free apps, and in paid apps with high prices. We also note that the release text of releases that positively affect success make more prevalent mentions of bug fixes and new features.

Table 7: RQ5: Developer responses to questions after contact was established and a CIRA report was shared with them. Not all developers responded to all questions, and so the bottom 'total' row indicates the number of developers who expressed an opinion on a given question.

	Receiving further reports	Agree with detected significance	External cause of changes	Learning contributing factors	Would change strategy
Yes	29	39	23	39	19
No	16	11	23	6	25
Total	45	50	46	45	44

RQ5: Is causal impact analysis useful to developers? We sent 4,302 emails to the email addresses available in the Google Play app store pages of companies with significant releases, as detected by our tool, CIRA. Of course, this is something of a 'cold call', and we suspect that many emails never even reached a human. We can report that 90 immediately failed due to invalid email addresses used on app store pages, and 127 were immediately assigned to a support ticket. Those that received a response are the only cases in which we can verify that contact was established with developers, of which there were 138.

Of these 138 developers, 56 (distributed across 25 of 41 app store categories) replied to express their opinion in response to follow up questions. All respondents' apps were established in the store; the smallest had 31 reviews and the largest had 78,948 at the time of our experiments. We summarise developers' opinions in Table 7, in each case indicating the instances where developers expressed an opinion: the most answered question concerned agreement with CIRA's assessment, of which there were 50 respondents.

We can observe that 39 out of 50 development teams who expressed an opinion, agreed with CIRA's assessment that their app release was causally significant. For example, the developers of a dictionary application said: "in our case it was obvious for ourselves because it was a totally new release and with lots of new features", resonating with our earlier finding that new features increase the chance for significant releases (see RQ4.3). Only 11 developers disagreed with CIRA's assessment. However, some of them were still able to identify a cause for the significant change detected by CIRA (but did not think that the release itself could be the cause). For example, the developers of a security caller app said: "We did not release anything. We just upload builds for our beta version which uses few users :-) So your tools are WRONG." In this case, although the developers did not consider a beta version as a full release, the app's version identifier changed on its app store page resulting in a 'release', by our definition in Section 2.2. This detected release, combined with increased user activity, resulted in the causal significance detected by CIRA.

About one half of those who expressed an opinion (23 out of 46) indicated that they knew of an external reason for the changes, and several teams elaborated further. We might expect the developers who know of an external cause to be a subset of those who believe there to be a significant causal effect. However, 5 of 23 developers who claimed to know of external causes also disagreed with CIRA that the corresponding release was significant. One set of developers described the release as: "a minor 'bugfix-only' release. Therefore I doubt that this release was the main reason for this change.". However, as we know from RQ4 results, mentions of bug fixes in release text are more prevalent in significant releases that positively affect rating.
Indeed, this particular release did mention bug fixes in its release text, and significantly and positively affected rating.

Over half of the developers who expressed an opinion (29 of 45) were interested in receiving further reports. The majority of developers (39 of 45) indicated that they would like to learn more about the characteristics of significant releases, and 19 of 44 indicated they would consider changing their release strategy based on our findings. Only 10 of the 19 developers who would change their strategy knew of an external cause of the changes, thus suggesting that this consideration will be based on our general findings. These results provide initial evidence that causal impact analysis can be useful to app developers.

Answer to RQ5: Is causal impact analysis useful to developers? Three quarters of developers who expressed an opinion (39 of 50) agreed with CIRA. Most developers (39 of 45) were keen to learn more about the characteristics of significant releases, and 19 of 44 said that they would consider changing their release strategy based on CIRA's findings. This provides initial evidence that causal impact analysis is useful to developers.

7. THREATS TO VALIDITY

In this section we discuss threats to the construct, conclusion and external validity of our findings.

Construct Validity: The gap between data analysis and causality is large, forcing any user to make very strong assumptions if they hope to effectively imply causality. Causal analysis can hope to reduce this gap, but no such analysis could hope to fully close it; there will always be unknown factors which may nevertheless have affected the data. In the case of app stores, there will always be potential external influences for which no data is available to capture them.

We apply causal impact analysis in our experiments, but there are other forms of causal analysis such as differences-in-differences. Nonetheless, we believe this method is the most suitable due to its independent consideration of each app release, and the ability to use all non-releasing apps as the control set in every experiment, thus reducing the risk of control set choice influencing results. We have shown how causal impact analysis is a useful way to identify releases that cause significant changes in subsequent success. In order to assess the actionability of the technique we ask RQ5.

Conclusion Validity: Our conclusion validity could be affected by the qualitative human assessment of 'top terms' and topics for sets of releases in RQ4.1. We mitigate against this threat by asking a quantitative question of the number of times 'bug' and 'fix' and 'new' and 'feature' occur in each set of releases in RQ4.2.

External Validity: Naturally, care is required when extending our findings to other app samples and app stores. Nevertheless, the methods we used to analyse causal effects can be applied to other app stores. We believe that conclusions about the characteristics of significant releases over this sample will yield interesting and actionable findings for developers, and we contact app developers for their opinions on the usefulness of our technique in RQ5.

We can only report the views of those developers with whom contact could be established (see Section 6), and care is required when interpreting their responses. Since we were able to effectively reach 56 developers, we cannot claim that our sample is representative of the entire population. However, this is still a fairly large sample with respect to those used in other studies involving app developers (e.g., [20, 30]).

8. RELATED WORK

For an overview of app store analysis literature, we point the reader to our survey on app store analysis for software engineering [26]. In this section, we discuss previous work on software releases and causal analysis in software engineering.

There has been a large amount of recent work linking software quality with user perceived quality. Ruiz et al. [37] studied how ad library usage affected user ratings. Bavota et al. [2] investigated how the changes and faults present in APIs used affected apps' ratings. Panichella et al. [33] classified user reviews for software maintenance. Palomba et al. [32] studied how developers responding to user feedback can increase the rating. Moran et al. [28] presented an Android bug reporting tool that can increase the engagement between users and software quality.

It therefore stands to reason that software releases affect quality and consequently may affect user rating behaviour. In 2011 Henze and Boll [16] analysed release times and user activity in the Apple App Store, and found that Sunday evening is the best time for deploying games. In 2013 Datta and Kajanan [12] found that apps receive more reviews after deploying updates on Thursday or late in the week. In 2015 Gui et al. found that 23% of releases from frequently-releasing apps contained ad-related changes [15]. Comino et al. [9] studied the top apps in Apple and Google stores, finding that releases can boost user downloads.

McIlroy et al. [27] studied update frequencies in the Google Play store, finding that only 1% of studied apps received more than one update per week. These findings support our weekly data collection schedule, as very few releases can be 'missed' by collecting data weekly; additionally the target releases we use (defined in Section 2.3), mandate that very frequently updated apps are excluded due to lack of sufficient prior and posterior time series data. McIlroy et al. [27] also found that rating was not affected by update frequency, however the findings by Guerrouj et al. [14] indicate that high code churn in releases correlates with lower ratings. Nayebi et al. [29] surveyed developers and users, finding that half of developers had clear releasing strategies, and many experienced developers thought that releasing strategy affects user feedback. Users were not more likely to install apps based on release date or frequency, but preferred to install apps which have been infrequently, but recently, updated.

All of these previous findings on app releases tantalisingly point to the possibility that certain releases may have higher causal significance than others. However, no previous study (excepting our technical report [25] and two-page student research abstract [23]) has specifically addressed the question of how we identify the set of releases that are significant. Furthermore, no previous work attempted to identify the characteristics of highly significant app releases: the primary technical and scientific contributions of the present paper. In our preliminary work [23, 25], we studied apps which were in 'most popular' lists every week in Google Play and Windows Phone store, between July 2014 and July 2015. This very strict interpretation of 'popular' resulted in a much lower number of apps than the ones considered in the present study: 307 and 726 apps in the Google and Windows sets, respectively.

In the present study, using a larger dataset that does not suffer from the app sampling problem [24], we confirm and extend the findings of our preliminary study [23, 25]. Indeed, some of the results suggest that certain findings hold between different stores and datasets, suggesting that they may be more widely applicable: releases mentioning the terms new and feature are more likely to positively affect success; higher priced (paid) app releases are more likely to have a positive effect. Other results, such as the mentions of the terms bug and fix, perhaps highlight the difference between dataset maturity: releases are more likely to significantly affect success if they mention bug fixing in this large dataset, but slightly less likely in the dataset used in our technical report [25]. We can speculate that this is expected, as the apps used in this study are not consistently the most popular apps in the store, so it is more acceptable for them to fix bugs in releases; while in the most popular apps (used in our technical report [25]), it is perhaps less acceptable to acknowledge bugs, or to spend a large proportion of development effort on bugs. In future work we will investigate this possibility by comparing sets of apps which have different popularity.

To the best of our knowledge, we are the first authors to apply causal impact analysis to app store analysis. However, causal inference has been discussed in empirical science papers [19] and it has been previously used in software engineering. For example the work using the Granger causality test by Couto et al. [10, 11] and Zheng et al. [44] as a means of software defect prediction, and the work by Ceccarelli et al. [7] on identifying software artefacts affected by a change. Since `CIRA` allows us to apply causal impact analysis on any time series vector, future work will use it to analyse other metrics from app store data, and other time series datasets. Future work may also seek to identify the causal effect of app feature migration [38] on susequent success.

9. CONCLUSIONS

In this work we propose the use of Casual Impact Analysis to identify causally significant releases in app stores. In particular, we introduce our tool `CIRA` to perform causal impact analysis on 26,339 Google Play app releases between February 2015 and February 2016, for all apps that appeared at least once in the top rated apps in the year prior to February 2015. For these apps, we found that overall release frequency is not correlated with subsequent app success, but that there is evidence that price and release text size and content all play a role in whether a release is significant and the type of effect it has on success. Higher priced releases are more likely to be significant and, perhaps surprisingly, to *positively* affect rating; there were more prevalent mentions of new features and bug fixes in releases that positively affected rating, and successful releases also had longer (more descriptive) release text. We have shown that causal analysis can be a useful tool for app developers by eliciting their opinions: most of those who responded were interested in our findings and agreed with `CIRA`'s assessment, and some said they would consider changing their releasing strategy based on our findings.

10. REFERENCES

[1] B. Adams, S. Bellomo, C. Bird, T. Marshall-Keim, F. Khomh, and K. Moir. The practice and future of release engineering: A roundtable with three release engineers. *IEEE Software*, 32(2):42–49, 2015.

[2] G. Bavota, M. Linares-Vasquez, C. E. Bernal-Cardenas, M. D. Penta, R. Oliveto, and D. Poshyvanyk. The impact of API change-and fault-proneness on the user ratings of Android apps. *IEEE Transactions on Software Engineering*, 41(4):384–407, 2015.

[3] Y. Bejamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal statistical Society (Series B)*, 57(1):289–300, 1995.

[4] M. Bertrand, E. Duflo, and S. Mullainathan. How much should we trust differences-in-differences estimates? Technical report, National Bureau of Economic Research, 2002.

[5] D. M. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[6] 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.

[7] M. Ceccarelli, L. Cerulo, G. Canfora, and M. Di Penta. An eclectic approach for change impact analysis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 163–166. ACM, 2010.

[8] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[9] 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.

[10] 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.

[11] 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.

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

[13] Google and MobiSystems. OfficeSuite Pro + PDF Android Apps on Google Play. https://play.google.com/store/apps/details?id=com.mobisystems.editor.office_registered. Retrieved 8th March 2016.

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

[15] J. Gui, S. McIlroy, M. Nagappan, and W. G. J. Halfond. Truth in advertising: The hidden cost of mobile ads for software developers. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 100–110. IEEE Press, 2015.

[16] 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.

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

[18] 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.

[19] B. A. Kitchenham, T. Dyba, and M. Jorgensen. Evidence-based software engineering. In *Proceedings of the 26th International Conference on Software Engineering*, ICSE '04, pages 273–281, Washington, DC, USA, 2004. IEEE Computer Society.

[20] M. Linares-Vásquez, G. Bavota, C. E. B. Cárdenas, R. Oliveto, M. Di Penta, and D. Poshyvanyk. Optimizing energy consumption of guis in android apps: A multi-objective approach. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 143–154, 2015.

[21] M. H. Maathuis and P. Nandy. A review of some recent advances in causal inference. *Handbook of Big Data*, page 387, 2016.

[22] C. D. Manning, P. Raghavan, and H. Schütze. Scoring, term weighting, and the vector space model. In *Introduction to Information Retrieval*, pages 100–123. Cambridge University Press, 2008.

[23] W. Martin. Causal impact for app store analysis. In *Companion Proceedings of the 38th International Conference on Software Engineering*, ICSE Companion 2016. ACM, 2016. 2 page Student Research Competition (SRC) entry.

[24] 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, 2015.

[25] 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.

[26] 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.

[27] 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.

[28] K. Moran, M. Linares-Vásquez, C. Bernal-Cárdenas, and D. Poshyvanyk. FUSION: A tool for facilitating and augmenting Android bug reporting. In *Proceedings*

of the 38th International Conference on Software Engineering, ICSE '16, pages 609–612. ACM, 2016.

[29] M. Nayebi, B. Adams, and G. Ruhe. Mobile app releases – a survey research on developers and users perception. In *IEEE 23rd International Conference on Software Analysis, Evolution and Reengineering (SANER'16)*. IEEE, 2016.

[30] M. Nayebi, B. Adams, and G. Ruhe. Release practices in mobile apps – users and developers perception. In *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 552–562. IEEE, 2016.

[31] G. Neumann, M. Harman, and S. M. Poulding. Transformed vargha-delaney effect size. In *Search-Based Software Engineering - 7th International Symposium, SSBSE*, pages 318–324, 2015.

[32] F. Palomba, M. Linares-Vásquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia. User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In *31st International Conference on Software Maintenance and Evolution (ICSME) '15*, 2015.

[33] S. Panichella, A. D. Sorbo, E. Guzman, A. Visaggio, G. Canfora, and H. Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, 2015.

[34] K. Pearson. Notes on regression and inheritance in the case of two parents. *Proc. of the Royal Society of London*, 58:240–242, June 1895.

[35] X.-H. Phan and C.-T. Nguyen. Gibbslda++. http://gibbslda.sourceforge.net. Retrieved 3rd March 2016.

[36] X.-H. Phan and C.-T. Nguyen. Jgibblda. http://jgibblda.sourceforge.net. Retrieved 3rd March 2016.

[37] I. J. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan. Impact of ad libraries on ratings of Android mobile apps. *IEEE Software*, 31(6):86–92, 2014.

[38] 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.

[39] S. L. Scott and H. R. Varian. Predicting the present with bayesian structural time series. *International Journal of Mathematical Modelling and Numerical Optimisation*, 5(1-2):4–23, 2014.

[40] C. E. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

[41] A. Vargha and H. D. Delaney. A critique and improvement of the "CL" common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2):pp. 101–132, 2000.

[42] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[43] H. Zhang, L. Gong, and S. Versteeg. Predicting bug-fixing time: An empirical study of commercial software projects. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1042–1051, Piscataway, NJ, USA, 2013. IEEE Press.

[44] 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.