WEBSITE CHANGES AND USER BEHAVIOR
USING PANJIVA DATA TO EXAMINE CODE CHANGES

JOHN WANG
14.27 FINAL PAPER

ABSTRACT.

CONTENTS

1. Dataset Explanation

The proprietary Panjiva Dataset comes from the back-end databases collected by Panjiva, Inc. Panjiva's website `http://www.panjiva.com` acts as a medium for buyers and suppliers of manufactured goods. The site provides a communication platform so that bulk buyers of a particular good can search and obtain unbiased information on factories and suppliers of that good. These two parties can then communicate and send messages over Panjiva's interface, attempting to strike a deal.

Panjiva's competitive advantage rests in its ability to parse government import and export data in order to obtain unbiased information about suppliers. Panjiva determines a supplier reliability score and also provides recent history of a supplier's shipments, and allows buyers to search and aggregate this information easily. Most firms that use Panjiva are large to medium size buyers of components. For example, a department store would use Panjiva to search for suppliers of shirts or clothing, or a home improvement store would search for suppliers of socket wrenches. In addition, Panjiva provides data on trends in global manufacturing and shipping by leveraging the government data it already mines for individual supplier information.

The dataset used in this paper comes from the event and activity logs of Panjiva's website. Each time a user performs some significant event or activity on the Panjiva website, an entry will be created in either the event or activity log. If the user has a registered account with Panjiva, the action will be recorded in the activity logs. All activities, regardless of whether a user has an account, are recorded in the event logs. All nontrivial features of the website, including supplier search, U.S. import and export search, and profiles views, are accounted for and stored in a SQL database.

The data in the event and activity logs are organized so that one can trace the exact user or subscribed account for which the entry. In particular, the logs contain information on the ip address of the user, the time the activity was performed, the webpage the activity occurred on, and extra data depending on the type of activity performed.

The enormous quantity and granularity of this data enable the analysis of user-level interactions. The event logs contain about 124 million entries while the activity logs contain about 13 million entries. Moreover, the data in each of these logs extends for multiple years, allowing one to analyze the growth of Panjiva as a company and the effect of different changes to the website.

1.1. **Summary Statistics.** Summary statistics providing an overview of Panjiva's business and website are provided in table 1. Panjiva was incorporated in 2008 and has since developed a customer base composed mostly of buyers of manufactured goods. The website provides free services (a limited number of searches) for free users, and provides many more services to subscribers (there are multiple levels of subscription).

Table 1. Panjiva Overview

Total Users	121,653
Subscribing Users	2,985
Monthly Site Visits	903,426
Monthly Unique Visitors	762,723
Average Pages per Visit	1.99
Average Visit Duration	1 min 18 sec

Commits can be thought of as changes to the code. In each commit, a software developer will introduce new code or delete old code which will then be launched to the production website. Typically a single developer at Panjiva will commit a couple of times throughout the workday. Figure **??** shows a histogram of the number of commits throughout a 52 week period ending on 11/24/2012. Commit activity varies depending on the season and the week. In particular, commits during the summer spike upwards and there will be one or two weeks each quarter when commits fall to low levels. The second phenomenon is due to the fact that Panjiva holds a quarterly retreat where engineers reflect upon the work done over the quarter. Typically the amount of code written decreases during these weeks.
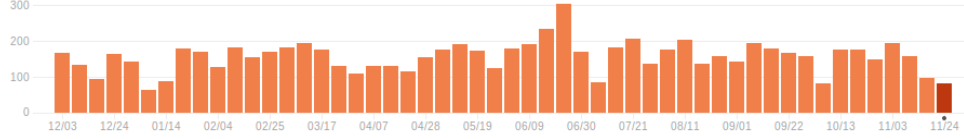
FIGURE 1. 52-Week Commit Activity

Table **??** shows statistics on a snapshot of the commit repository made on 11/25/2012. The table shows an enormous number of changes throughout the history of the website. In addition, the 1.3 million lines of code show that one can use the variation in the lines of code that a commit changes in order to better understand how each commit affects user behavior.

TABLE 2. Overall Commit Statistics - 11/25/2012

Active Days (at least 1 commit)	1,983
Total Current Files	20,901
Total Lines of Code	1,313,235
Total Lines of Code Added	3,989,295
Total Lines of Code Removed	2,676,060
Total Commits	29,924
Total Authors/Developers	33

Examining commits in more detail, one can see that the majority of commits occur during working hours (from 11am to 6pm EST). Moreover, each day is fairly regular in the number of commits that occur. Commits are at a high level throughout the work day, and fall off to a low, relatively constant level throughout the night. Figure **??** shows the times of day during which commits happen the most frequently.

FIGURE 2. Commit Frequency by Time of Day
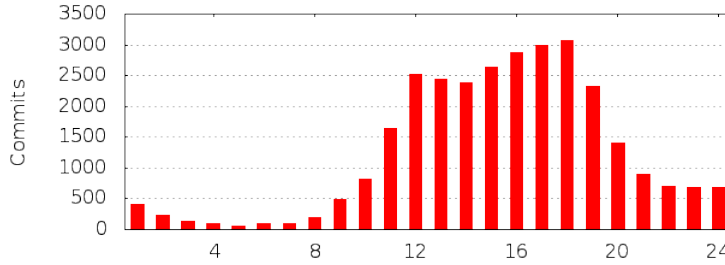
Finally, note that almost no commits occur on weekends. Only 5% of the commits happen on either Saturday or Sunday.

2. MACRO-LEVEL RESULTS

This section examines how code changes affect the total amount of user activity across the entire website (macro-level effects). In particular, this section analyzes the change in a number of metrics of user activity controlling for the amount of code changed in a particular day. The regression specification used will the as follows:

$$(1) \qquad y_t^i = c_0 + \bar{\beta}_0 \bar{M}_t + \bar{\beta} \bar{\chi}_t + \epsilon_t$$

Where t indexes day, y_t^i corresponds to the ith metric of user activity on day t, $\bar{M}_t$ corresponds to a vector of covariates that represent changes in the code, $\bar{\chi}_t$ is a vector of controls, and ϵ_t is an error term. We will use a number of different metrics for user activity and the specification will be run with each of them.

The first two metrics for user actions are the total number of entries in the activity or event logs for a particular time period. Recall that event logs are database records for actions performed by anyone, while activity logs are records for only those users who have a registered account with Panjiva.

Another set of metrics are the total number of distinct users performing either activities or events in a given time period. Finally, one might also use the average number of actions per user, whether action is defined using records from activity or event logs. One would expect the each of these metrics to measure a different aspect of user behavior.

2.1. **Daily Effects of Code Changes.** Table **??** displays the results from using the total action count as a metric for user activity. The regressions in table **??** attempt to find the effect of different changes in the code base on different measures of user activity. A commit can be measured by the number of files it changes, or the number of lines of code it changes. Changes in lines of code can be either insertions or deletions. Each of these measures is percentilized (sorted rank is divided by the total number of observations). The regressions below are performed on aggregated statistics per day, with a dummy variable equal to 1 on the weekends and 0 otherwise.

The table shows the effect of commits on same day user activity. In general, changes to the code have minor and insignificant effects on metrics derived from the event logs. There are only significant effects for the eventlogcount. However, code changes have a stronger effect on metrics derived from the activity logs. Notice that for activitylogcount, the insertionspercentile variable is significant and the fileschangedpercentile and deletionspercentile are within the 10% significance level. Moreover, fileschangedpercentile has a significant effect on the average number of activities per user. The number of files changed and number of insertions are the only significant regressors, and they have opposing signs.

Increasing the number of files changed tends to decrease user activity, while increasing the number of code insertions tends to increase user activity. Although this result seems perplexing, there are a number of possible explanations. First, it is possible that the commits which change fewer files tend to be higher quality commits. Thus, if a commit touches a large number of files, there is a higher likelihood that this commit would introduce a bug or contain low quality changes. However, it seems that introducing new code, instead of deleting code, has a positive effect on user activity, controlling for the number of files changed.

Another, though possibly less satisfying explanation, is that code changes hurt average user activity in general. Since average user activity provides a measure of the organic growth of the website, it may be a better measure for the effects of code changes. Note that increasing total user count could result from a large number of new distinct user performing one or two actions each on the website. Although this could be construed as an increase in activity, each user could leave quickly, which would negatively affect long term traffic. Thus, one can think of the negative coefficient on the fileschangedpercentile variable as decreasing the long-term user activity with more code changes.

However, the second explanation seems unlikely because of the fact that code changes are constantly occuring. The regression is measuring the marginal effect of commits. Each additional file changed could provide a negative effect due to diminishing returns. Thus, one could have a negative coefficient on fileschangedpercentile even when increasing the number of files changed at very low levels (i.e. from 0 to 1) would increase user activity.

Also, notice that commits have no significant effect on the total number of distinct users. This is unsurprising since it is likely that it takes a significant period of time to increase the number of distinct users. Changes to the website will probably not spill over to other users during short time spans (like a day). Users do not have the time yet to share their new experiences on the site, so each user can be thought of as independent of others during these short time periods. Therefore, it is unsurprising that code changes fail to increase the total number of distinct users on the website.

2.2. **Lagged Effects of Code Changes.** Note that the analysis of the previous section uses the implicit assumption that for a given day t, only the code changes on day t influence user actions on day t. The previous regression assumes first that there is no lag between a code change and the change in user activity resulting from the code change, and second that

TABLE 3. Effect of Commits on User Activity

	(1) activitylogcount	(2) eventlogcount	(3) activitylog distinctusercount	(4) eventlog distinctusercount	(5) avguseractivity	(6) avguserevents
fileschangedpercentile	-5530.7 (-1.91)	-60589.4* (-2.26)	460.5 (0.65)	492.7 (0.71)	-10.40* (-2.01)	-148.6 (-1.52)
insertionspercentile	4868.8* (2.12)	47053.7* (2.22)	284.2 (0.51)	118.6 (0.22)	2.817 (0.69)	79.92 (1.03)
deletionspercentile	2778.9 (1.29)	29970.6 (1.50)	-192.2 (-0.36)	-184.4 (-0.36)	4.670 (1.21)	49.47 (0.68)
weekend	-14708.0*** (-16.48)	-79769.8*** (-9.65)	-1060.6*** (-4.86)	-816.2*** (-3.82)	-0.297 (-0.19)	257.0*** (8.49)
_cons	22396.6*** (25.05)	224482.8*** (27.12)	1635.7*** (7.49)	1395.9*** (6.52)	25.00*** (15.58)	340.6*** (11.24)
N	474	475	474	475	474	475

t statistics in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

The table provides regression estimates of how different measures of user actions vary with changes in the codebase. The table was created by aggregated data from the Panjiva activity and event logs over the period from 7/14/2011 to 11/24/2012. Variables are broken apart by day, so that counts are per day. Activity logs record information about users with registered accounts, while event logs record information about all users regardless of whether they have registered with the Panjiva.

Activitylogcount and eventlogcount aggregate the total number of records seen per day in the activity and event logs respectively. Activitylogdistinctusercount and eventlogdistinctusercount aggregate the number of distinct users that have performed an action in either the activity or event logs respectively. Finally, avguseractivity and avguserevents divides the total number of records by the total number of distinct users per day to arrive at an average number of actions per day.

Weekend is a dummy variable equal to 1 if the day is a Saturday or Sunday and 0 otherwise. The other covariates are percentilized by sorting each day and taking the rank and dividing by the total number of days. Files changed, insertions and deletions are obtained from commit history and are aggregated over each day. Insertions and deletions are the total number of lines of code inserted or deleted, respectively.

only a single day's code changes affects the user activity on day t. Both of these assumptions are simplifications, and this section will relax the first one.