Sampling GitHub

Research Theme

Why is CI valuable for community-based projects?

https://github.com/BonnyCI/ci-plunder

What data do we need?

Publicly available repositories with additional metadata describing key project parameters including language and contributors.

Github Data

GitHub is the best source of this data

- Huge user base
- Huge project representation
- Public Activity
- Public API

The Problem

Github is huge - Too much data

Sampling is hard - a disproportionate representation of one group could skew results or mask population subsets

Dealing with Huge Data

Cluster - divide into random groups, sample from each group (often used for geography)

Stratify - divide into groups by characteristic, sample from each group

Splitting Github into Strata

Of the datasets available, which one is the best one for us to work with?

What parameters are available in this dataset? How much effort to determine these parameters?

Which of these parameters will help to answer our questions?

Our needs

- Interviews to understand the role of CI - need to be able to contact contributors
- Focus on current CI technology rather than historical - eg, services like TravisCI
- Historical analysis - what changed when CI was introduced?
- Future predictions - what do we expect to change when a project uses CI?
- Project outreach for BonnyCI - What CI value does BonnyCI provide? Who is most likely to benefit?

Repository Characteristics

Dependent on CI value discovery, however we think we are interested in repositories that are:

- Currently active
- Have (or want) a user base
- Do releases (or want to)
- Have (or want) more than one active developer
- A mix of mature (have history) and new

Translating Needs to Requirements

- Timestamped data and ability to search and filter based on this
- Usernames of end users (eg, username, downloads, bug authors, who cares about the repo)
- Release information (eg, version number, tag)
- Contribution information (eg, username, commits, documentation, bug comments)
- Repository metadata (eg, age, summary of activity per time period)

Available Datasets

- Github Archive
  - GOOD: available in Google Big Query, very complete, updated daily, goes back to 2011
  - BAD: Only event data; Payloads are text dumps; No additional metadata
- GHTorrent
  - GOOD: Full profile dump archive from the Github API; available in GBQ
  - BAD: Highly normalized, incomplete/out of date, inconsistent “calculated” fields
- Github API
  - GOOD: Best source for up-to-date data, JSON
  - BAD: Rate Limiting, 300 results (no archive), JSON (has to be parsed)

Available Datasets Demo

Github Event Data

Plus:

- Github event data best suits our needs.
- Current, easy to filter on date
- Usernames at top level

Minus:

- Repository metadata, release metadata, contribution info is available but has to be extracted
- Additional Repository data must come from the API

What’s the best way to stratify this data?

Discovery phase 1

How do the available parameters correlate with the repository characteristics we care about?

Time is the obvious one so, let’s decide on a time frame and take a sample

June through December 2016

Why 6 months?

Seems long enough to show activity trends but at this point we don’t really know anything or what to expect.

For instance, how active are mature repositories? Will they have more or less events than less active, less mature repositories?

Does event distribution from month to month indicate anything about the maturity or overall number of contributors?

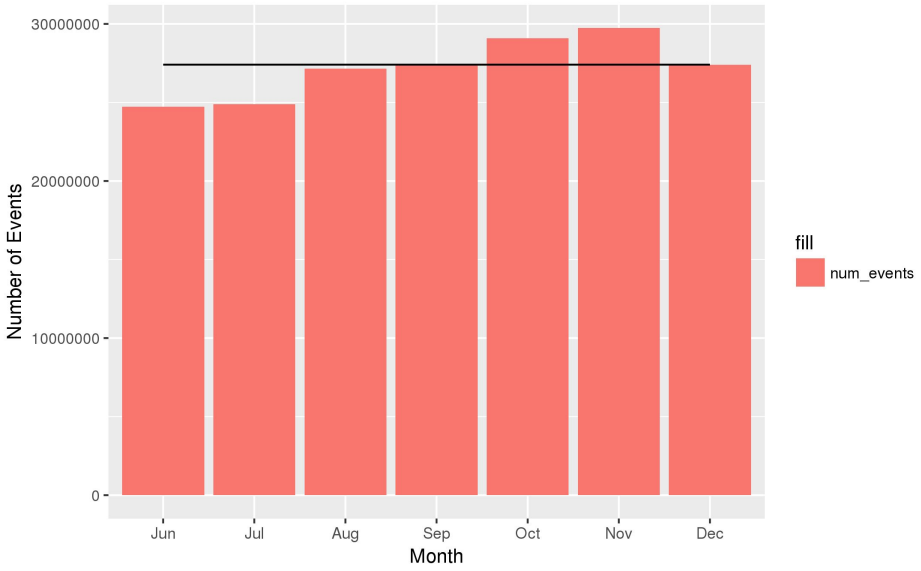
Github Events June - December 2016

Can we reliably sample Github events?

How much variation is there from month to month? Is there a pattern to the variation or does it appear to be totally random?

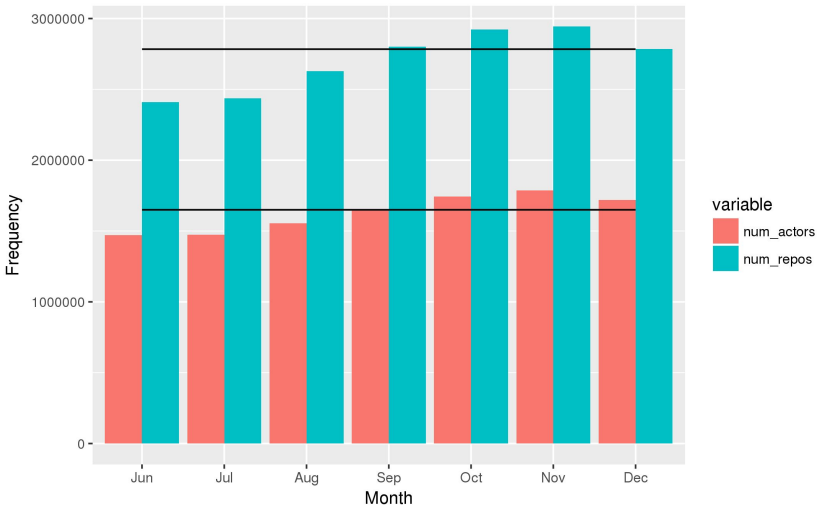
What factors vary the most and can we reason as to why?

Month to Month Events

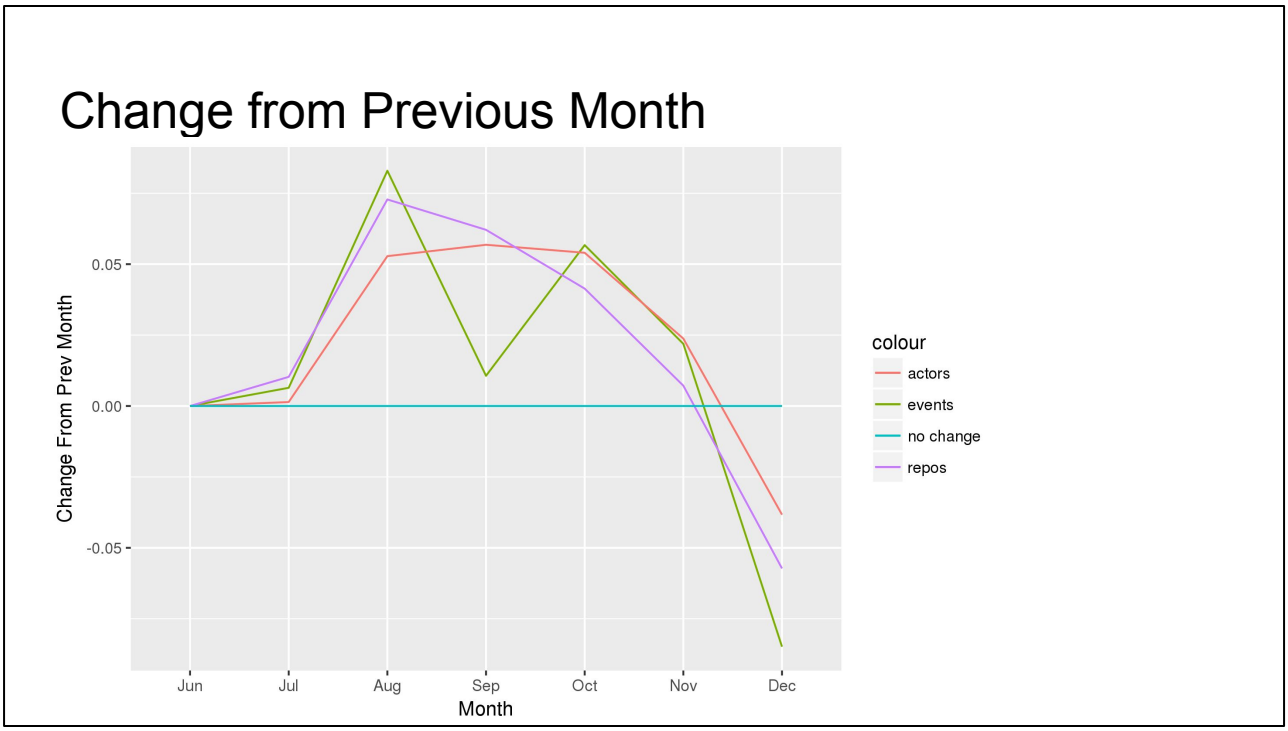


When we examine the total number of events, unique actors, and unique repositories over the 6 month period, there appears to be a lot of variation.

Month to Month Unique Actors and Repositories



Number of actors and repositories seem to follow the same shape as the number of events



However we see less variation when we look at the change from month to month and the deviation from the median. The month-to-month change and the deviation from median have been adjusted to relative proportions to allow for a comparison between the different totals, and to show some sense of the overall scale. Further analysis needs to be done over a broader span of time for any solid conclusions about overall variability. However, for the purposes of this study, the variability indicated below does not appear to be significant enough to skew further analysis results.

Event Types

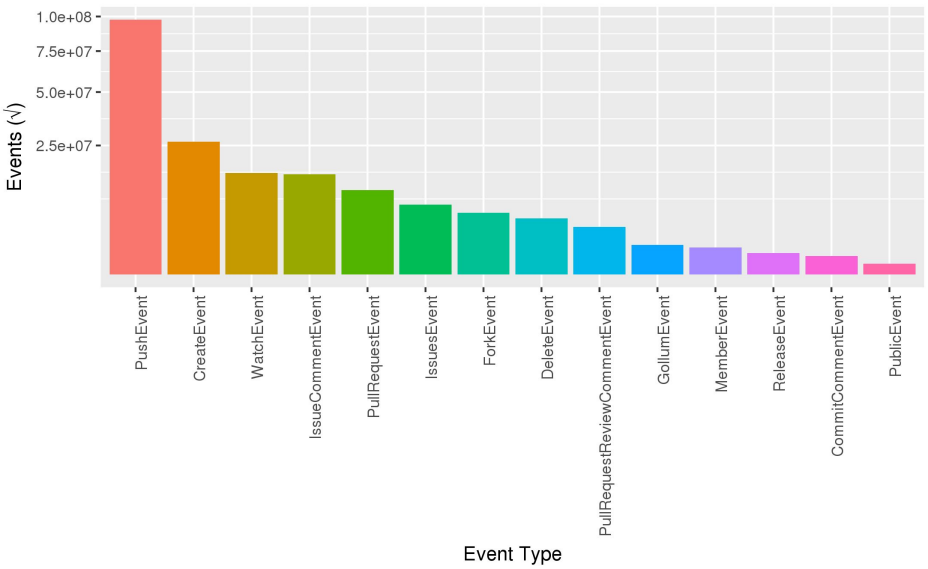
Different types of activity are associated with Github event types.

Do some types of events occur more often than others?

How much do event types vary from month to month?

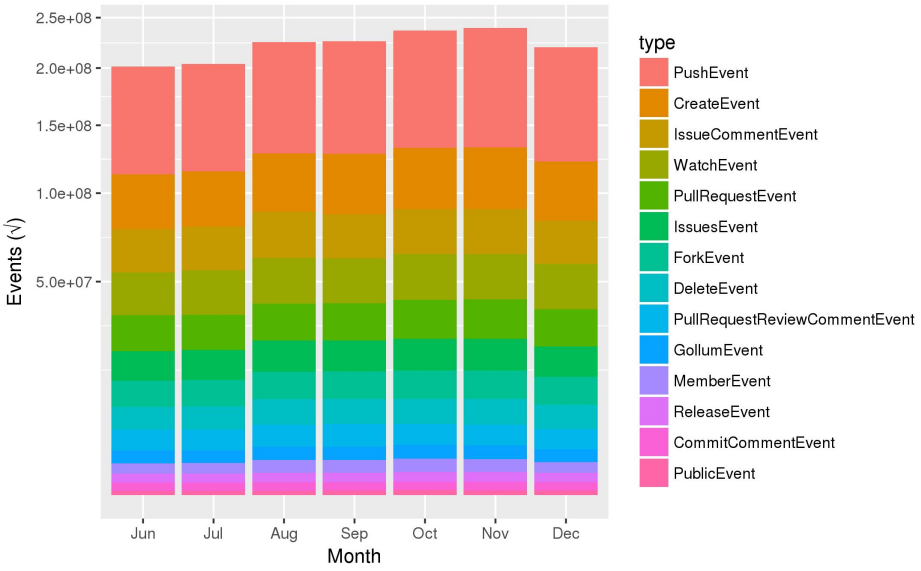
Are there any patterns to event type frequencies? How do these compare to overall events?

Overall Event Type Frequency



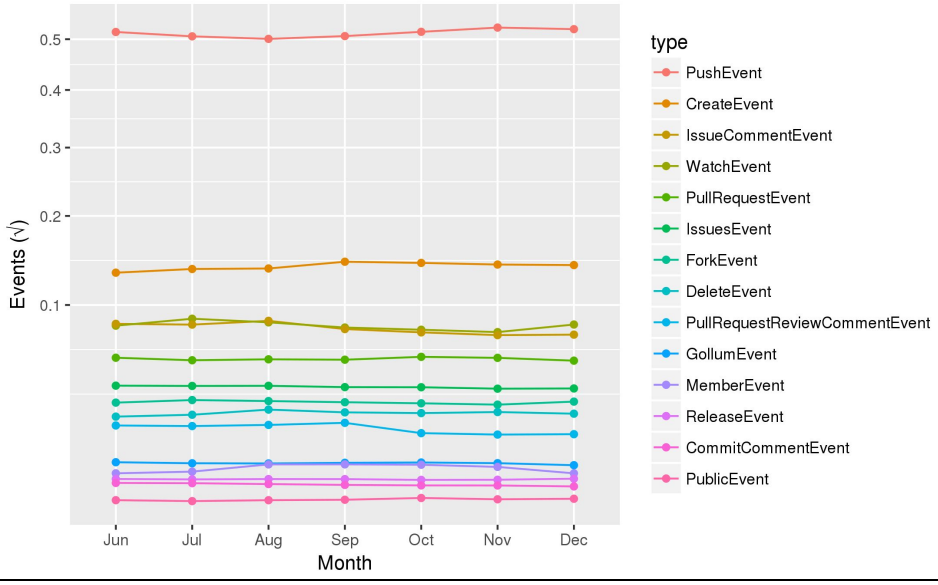
Event types represent different types of activity that occur in each project. Event types that occur more frequently might be distributed across more repositories and therefore may show less correlation to special repository parameters. Samples drawn from more frequent event types might show more variability between repository types than samples drawn from less frequent types. Further analysis needs to be done on each type to see if it is correlated with repository parameters easily available in the events data.

Event Types Per Month (Proportion)

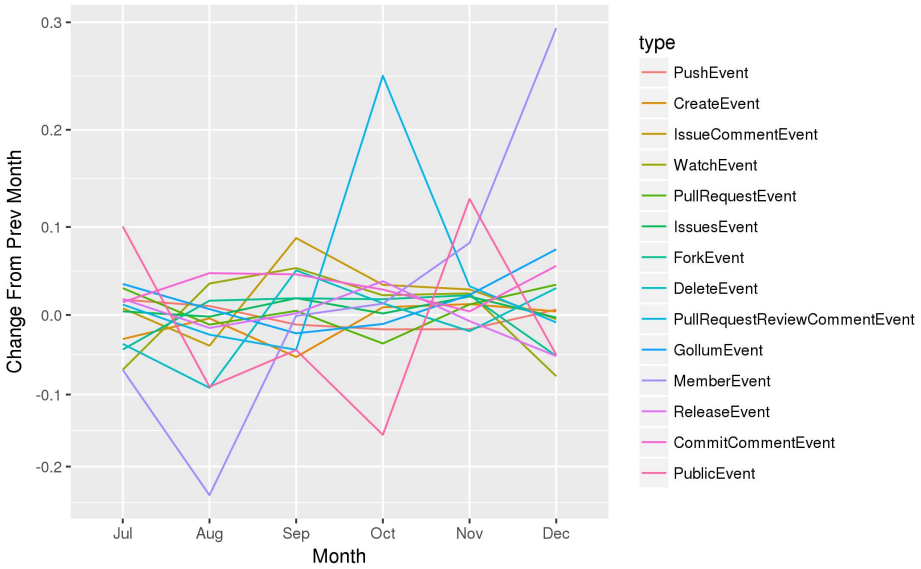


The proportions of events over the 6 month period did not change in terms of rank, except for Issue Comment and Watch Events. For the most part, the events that occurred most frequently in one month occurred most frequently in the subsequent months.

Event Types Per Month (Change Over Time)



Event Types Change from Prev Month



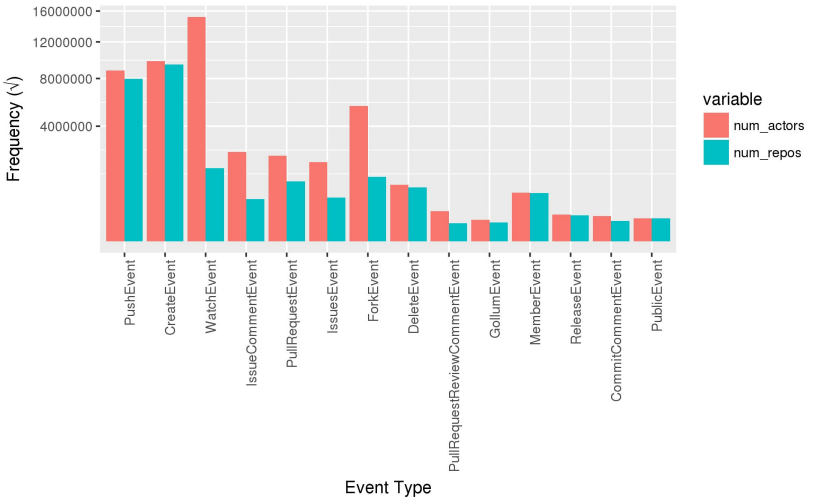
The overall change from month to month does vary. Some events show a lot of change from month to month while others show less. The most frequent events (Push, Create, Issue Comment, Watch, and Issue) appear to show less variability than the least frequent events.

Event Types and Repositories

How many unique actors and repositories contributed to each event type?

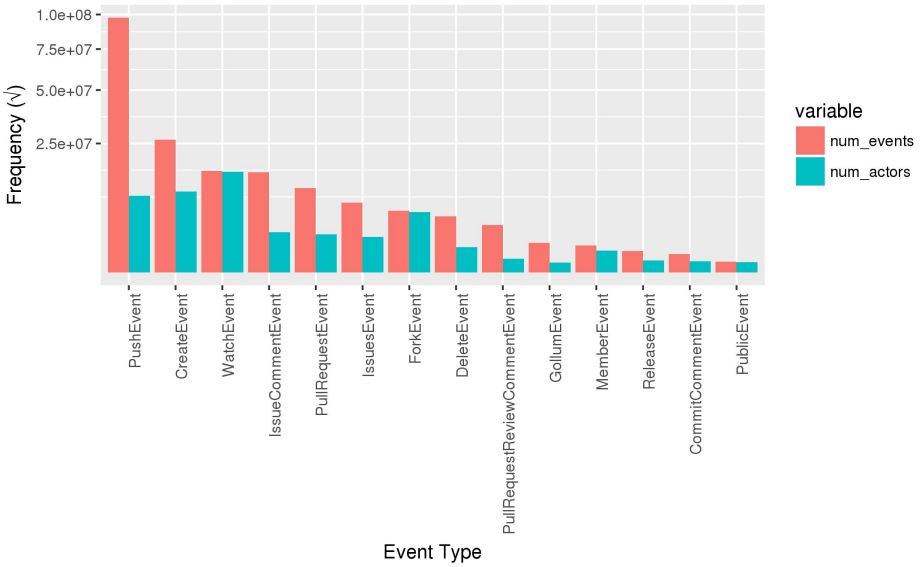
Is there any relationship between these values?

Actors + Repositories per Event Type



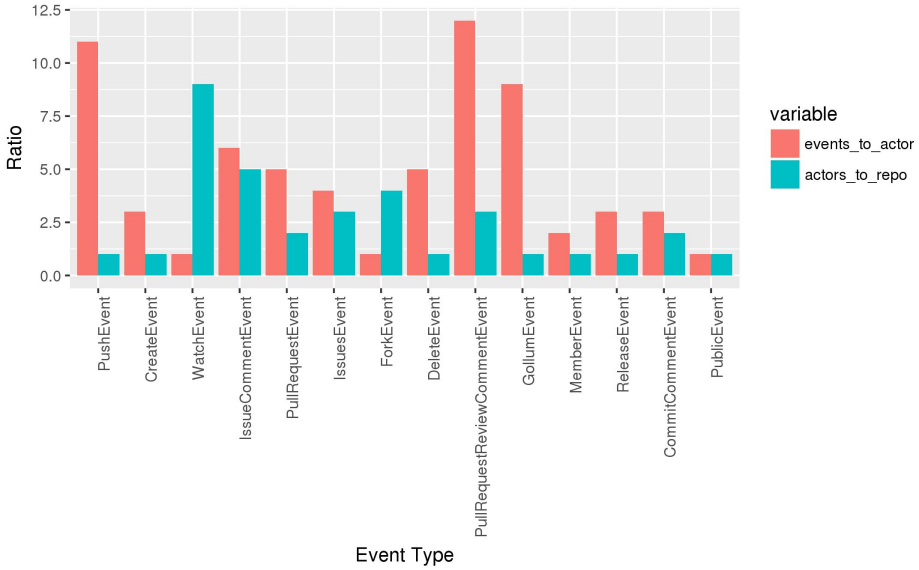
Event types that have a high number of actors and a low number of repositories could indicate many actors interacting with a small number of repositories. Frequencies closer in value indicate a smaller number of actors per repository. This distribution suggests a possible correlation between the type of event and the number of unique repository actors.

Actors + Events per Event Type



A high number of events and a small number of actors suggests a small number of actors generating a large number of events. The Watch Event distribution is particularly interesting. It suggests one watch event per unique actor in the sample. This is pretty unexpected, one would expect a high number of watch events per actor, not a 1:1 relationship. Github uses Watch Events as a popularity metric, but this suggests that it may only be an activity performed minimally by github users.

Actors to Repo / Events to Actor



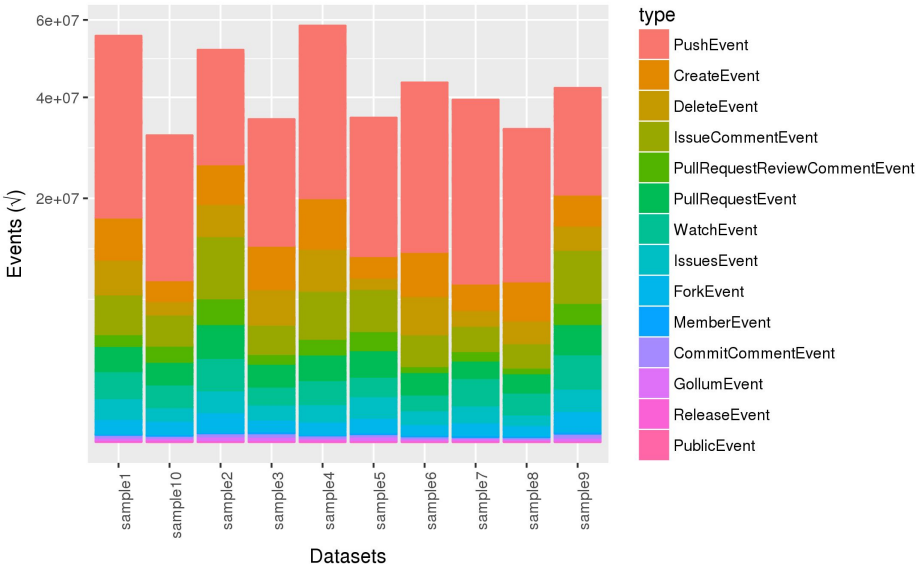
This takes the previous plots and turns them into ratios which are then plotted together to look for a pattern.

Sampling Events

What does a sample of repositories taken from events clustered only by time look like?

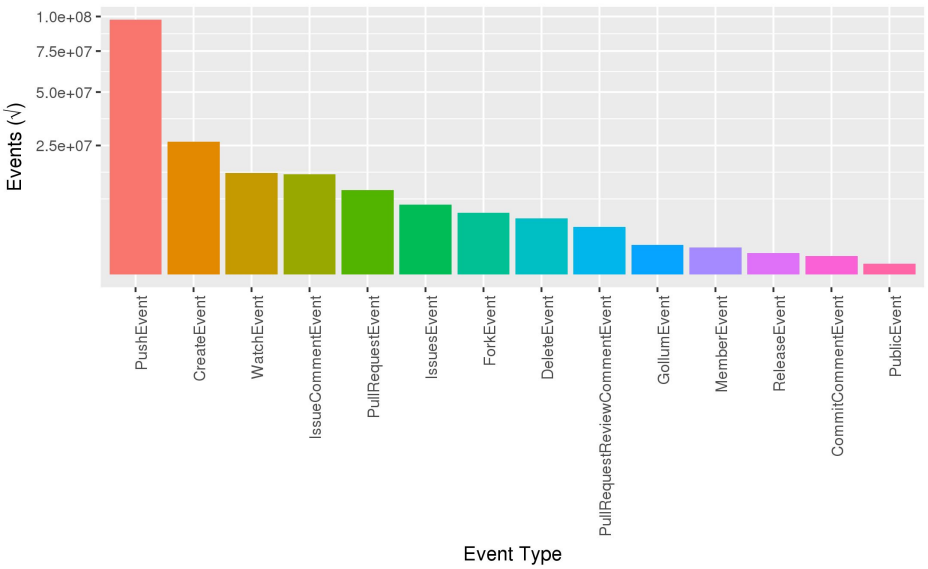
Random samples were drawn from the 6 month population described above. Let’s see how they compare to the population.

Distribution of event types in sample



Plotting the total number of events represented by each sample and the event types, we see a lot of variation between the samples. The top represented events appear to be Push, Create, Delete, IssueComment, PullRequestReviewComment, PullRequest, and Watch.

Overall Event Type Frequency (Reprise)

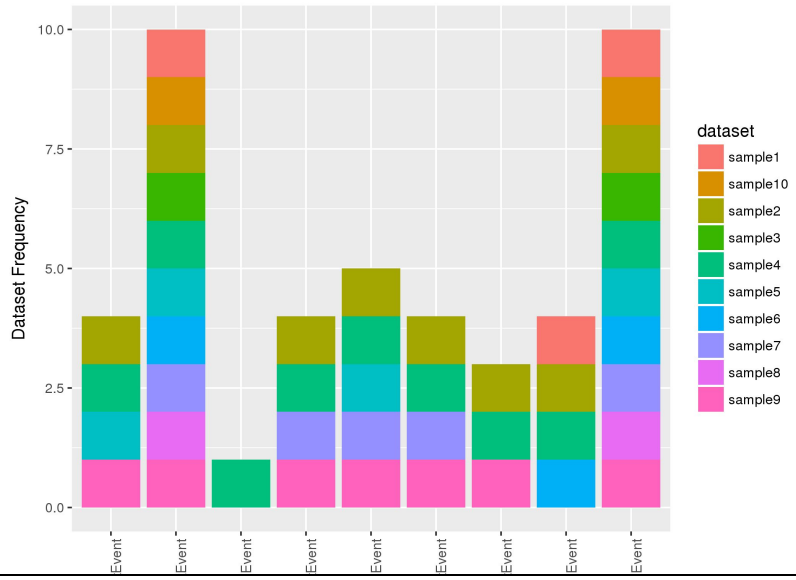


Event types represent different types of activity that occur in each project. Event types that occur more frequently might be distributed across more repositories and therefore may show less correlation to special repository parameters. Samples drawn from more frequent event types might show more variability between repository types than samples drawn from less frequent types. Further analysis needs to be done on each type to see if it is correlated with repository parameters easily available in the events data.

Event Types Correlation to Activity

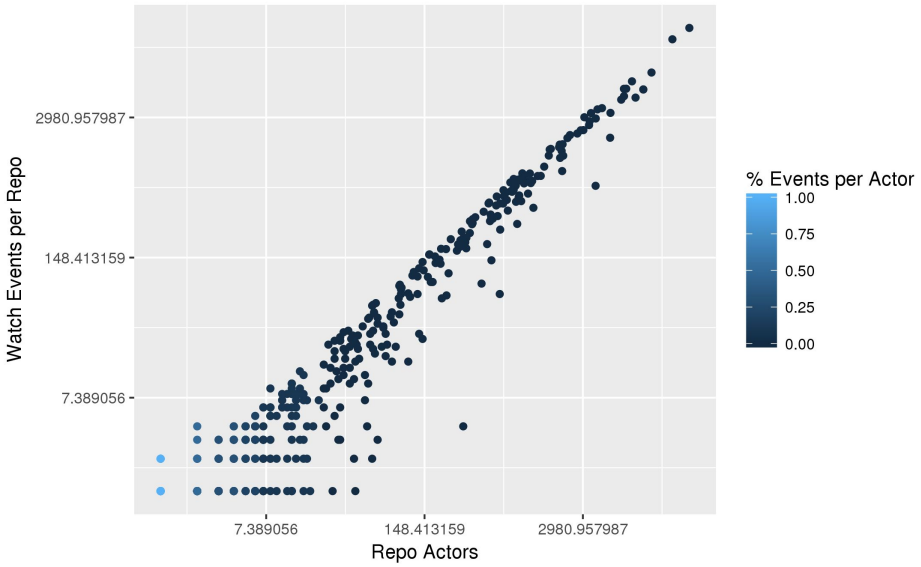
Is there any relationship between certain event types and the number of actors per repo or events per actor?

Event Types vs Unique Actors per Repo



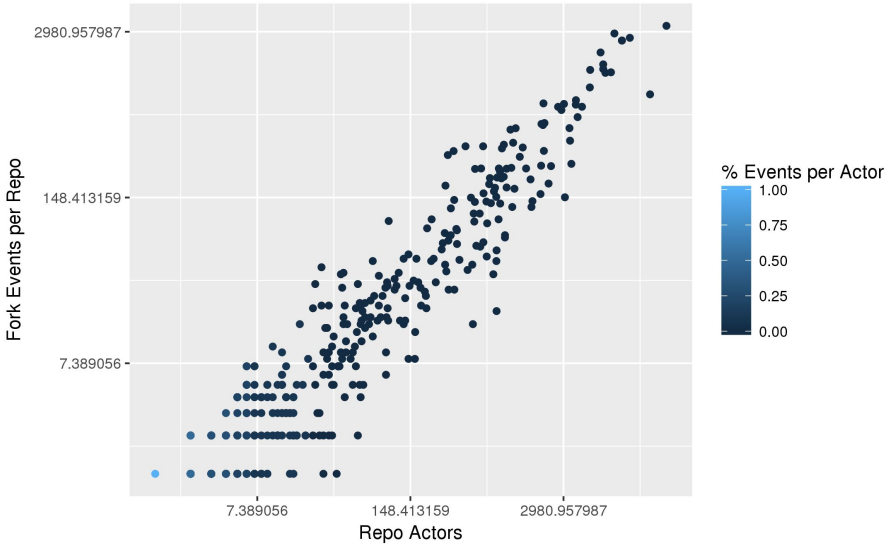
When evaluating the correlation (R cor() function) between actors and event type frequency per dataset, Fork and Watch events correlated strongly in all 10 datasets. Issue events only correlated 50% of the time.

Watch Events vs Actors Per Repo



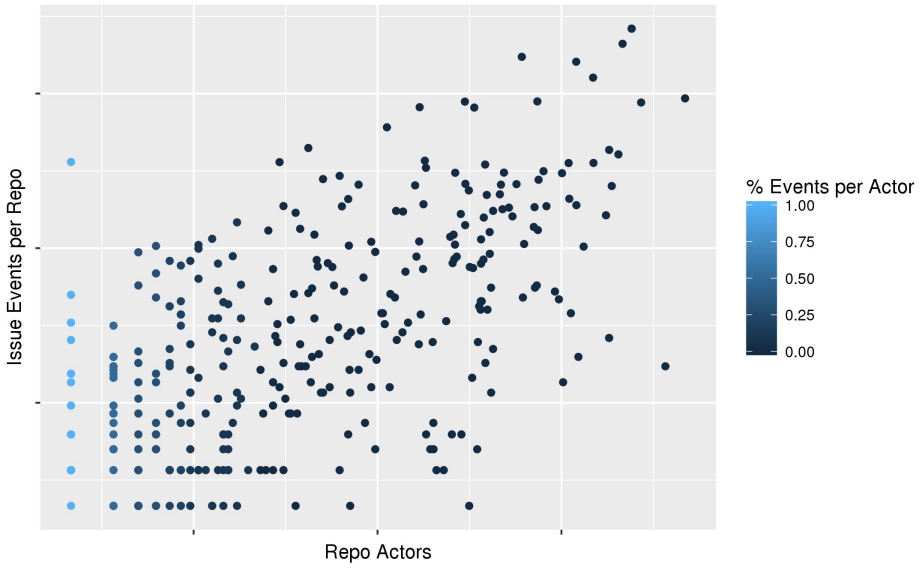
Really strong correlation as expected. As the number of actors per repo increases, so does the number of watch events. Given that watch events are typically 1 per actor, repos with a high number of watch events will show a high number of events. Because these are dependent factors, we need to examine actual repository data to see if it holds up. However because we can logically deduct this correlation in plain english, it's probably a good one.

Fork Events vs Actors per Repo



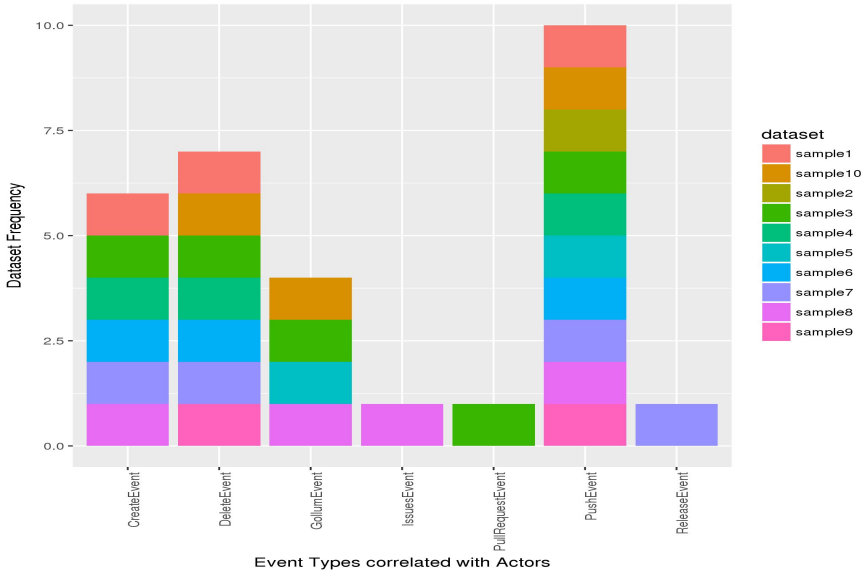
Fork events also tended towards one per actor so we see a fairly strong correlation here, although not quite as strong as Watch Events.

Issues Events vs Actors per Repo



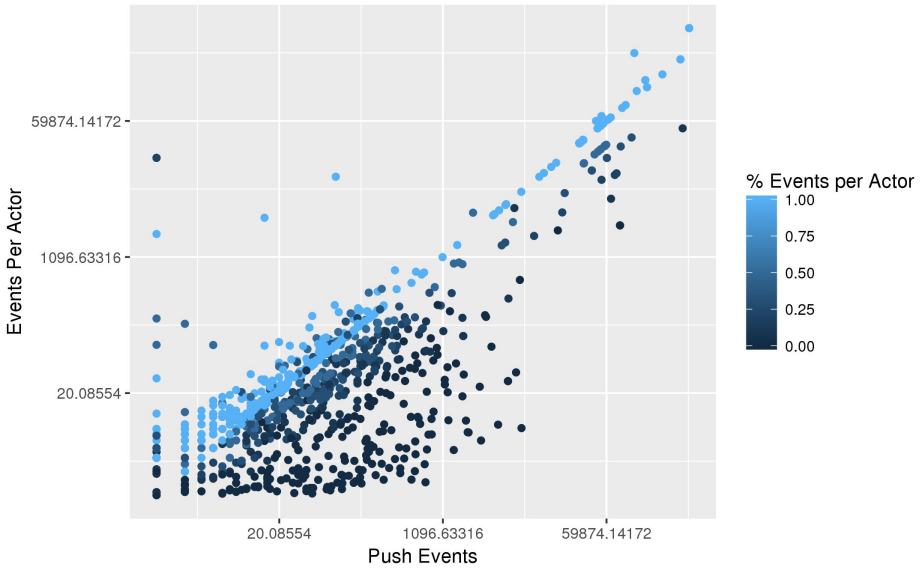
What we'd expect for 50% of datasets showing a correlation

Event Types vs Events Per Actor



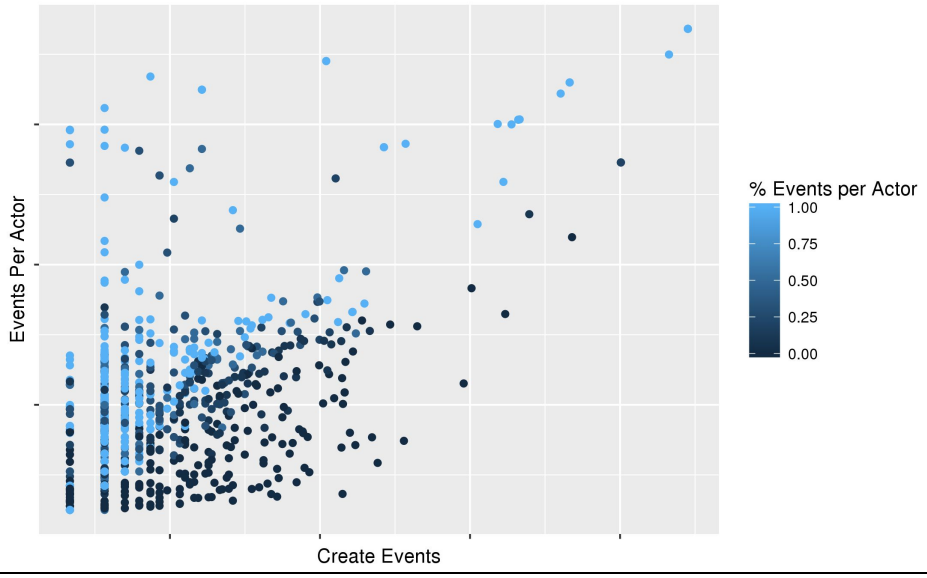
The next series of plots explores the possible correlation between the frequency of an event type and the ratio of events per actor per repository. Ideally a higher events to actor ratio indicates less unique actors per repository. This does not tell us anything about the proportion of those events, so a percentage value has been applied when examining individual event types. The percentage value indicates what percentage of repository events the events per actor ratio represents. A higher percentage indicates the events are spread across a smaller number of actors. Push, Create, and Delete events show the highest correlation to events per actor.

Push Events vs Events Per Actor

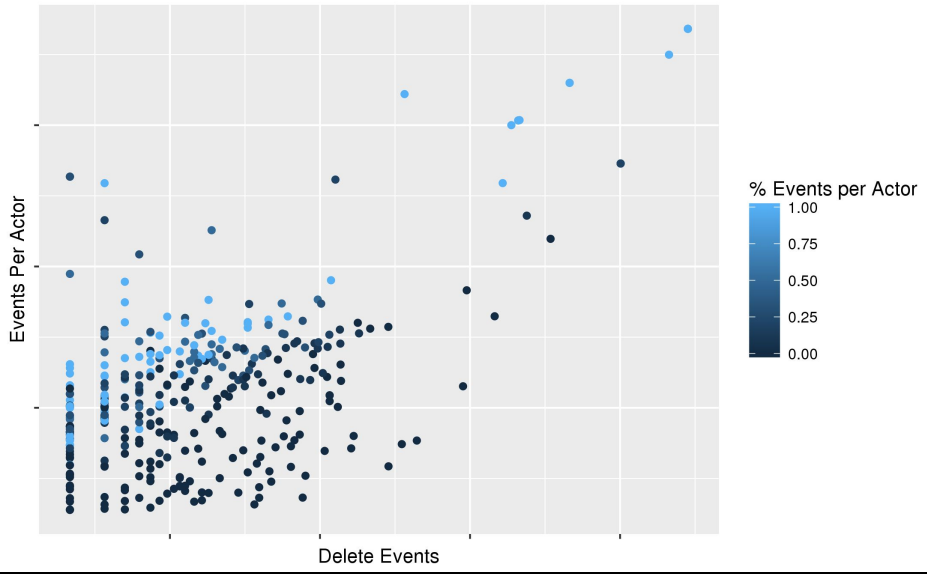


Push events appear to be associated the highest events to actor ratios meaning they are more likely to represent repositories with a smaller number of unique actors. Other event types showed interesting patterns, like Watch and Fork events, that indicated a smaller events to actor ratio overall. For events other than Push events, however there does not appear to be a strong correlation between the frequency an event type occurs for a repository and the distribution of events overall per actor.

Create Events vs Events Per Actor



Delete Events vs Events Per Actor



Hypothesis

Repository participation is strongly correlated with certain types of GitHub events.

Therefore, repository samples drawn from these event types will show a higher proportion of repositories that share a level of participation correlated with that event type.

Measuring Participation

percent of events per actor = events per actor / total number of events
where events per actor = total number of events/total number of unique actors

For example, for a total of 100 events:
1 unique actor = (100/1)/100 = 1 or 100% of events per actor
2 unique actors = (100/2)/100 = .5 or 50% of events per actor
5 unique actors = (100/5)/100 = .2 or 20% of events per actor
100 unique actors = (100/100)/100 = .01 1% of events per actor
Lower value = higher participation level

Participation Levels

High 0%

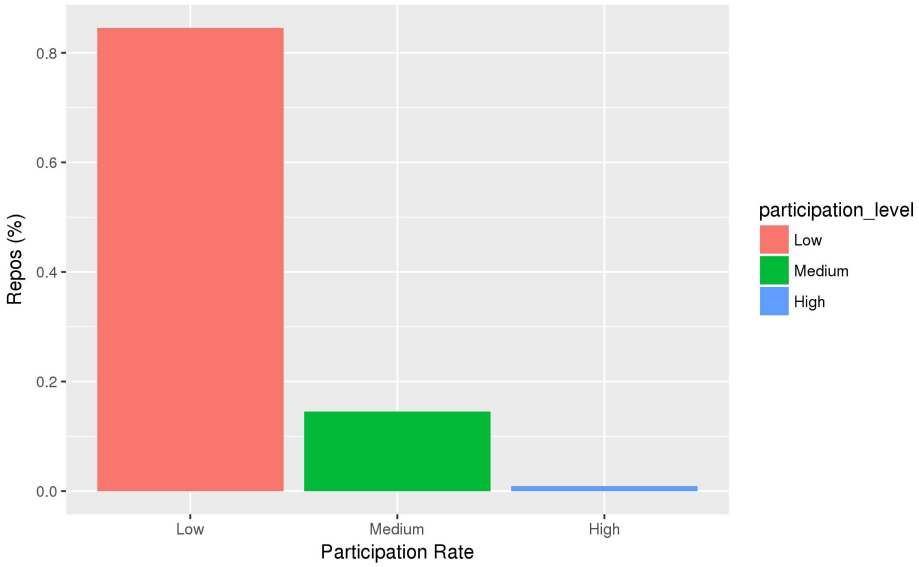
Medium 10-50%

and Low 100%

Repositories Proportion

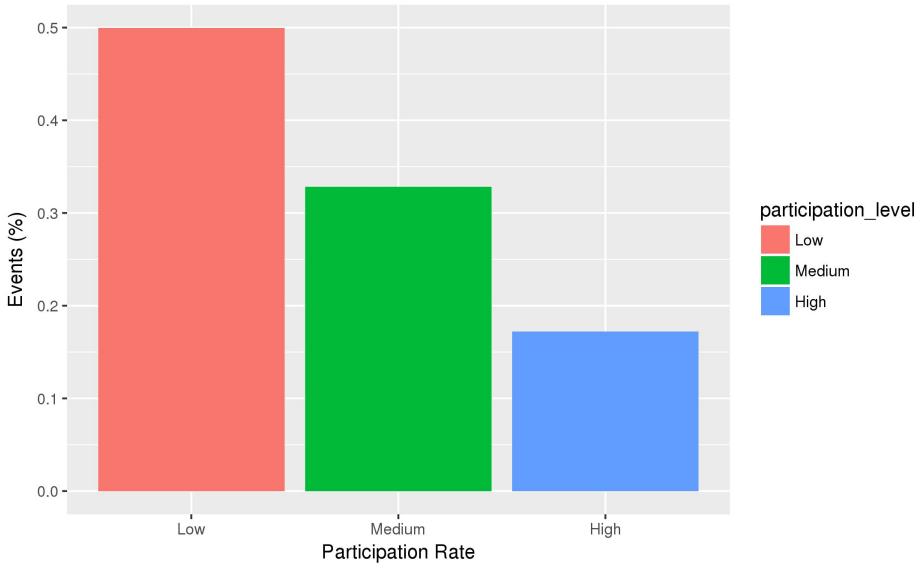
What is the overall distribution of participation?

Repositories Proportion



The majority of repositories in the population are Low participation, meaning they only have 1 unique actor generating events.

Events Proportion



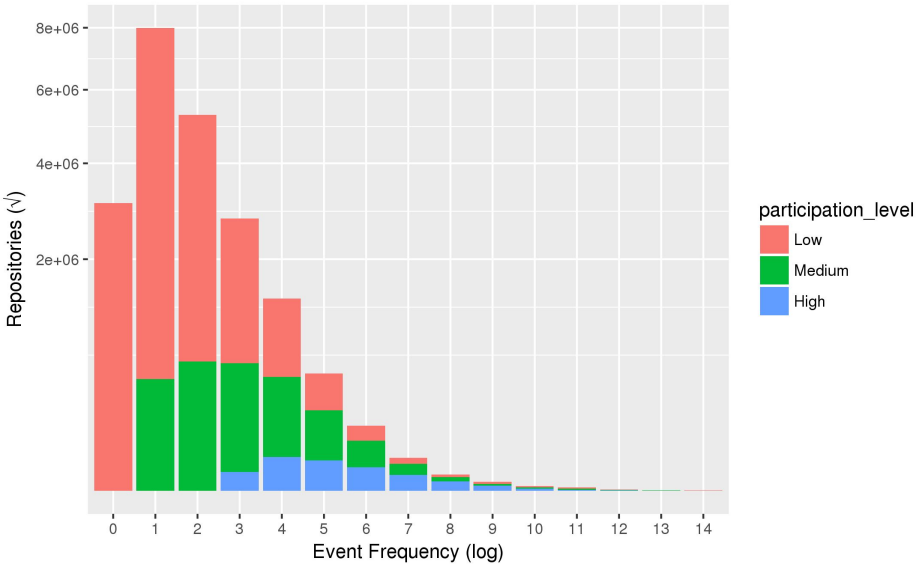
Low participation repositories were responsible for almost half of the events. Medium participation repositories accounted for nearly 30% while High participation repositories accounted for under 20% of all event activity.

Events per Repository

How many events did repositories have?

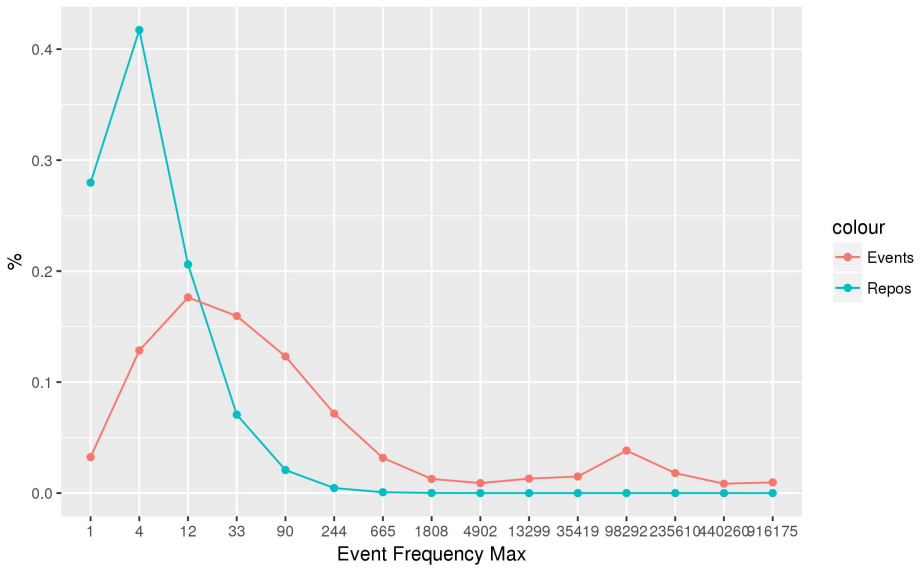
How much does this vary between participation rates?

Frequency of Events per Repo



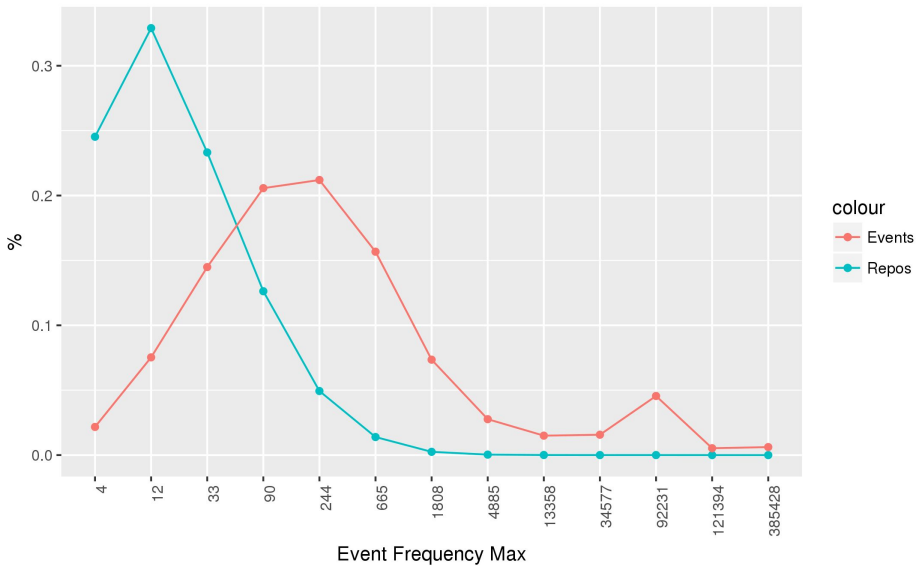
Repositories were grouped into buckets based on a rounded log of their total events and then counted. The distribution is heavily right skewed, showing that the majority of repositories contributed to activity in the lower end of the events range. However, the minimum number of events represented by Medium and High activity repositories is much higher than the minimum number of events represented by the low activity repositories. Low participation repositories show the greatest spread overall with data points in both the lowest and highest ends.

Low Participation: % of repos -> % of events



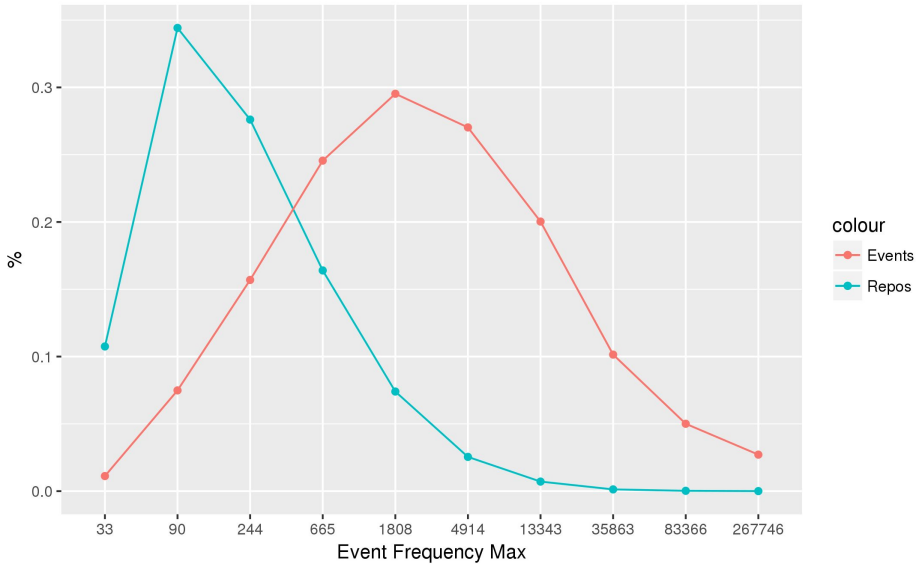
This plot shows the proportion of all events the repositories in each bucket were responsible for overlayed with the proportion of low participation repositories represented in each group. We would read it as, repositories that had between 2 and 4 events represent almost 50% of the low participation repositories. These repositories accounted for just under 15% of low participation events.

Medium Participation: % of repos -> % of events



The plots are shifting to the right, Medium has no repositories in the single event bucket because 2 contributors cannot make 1 event.

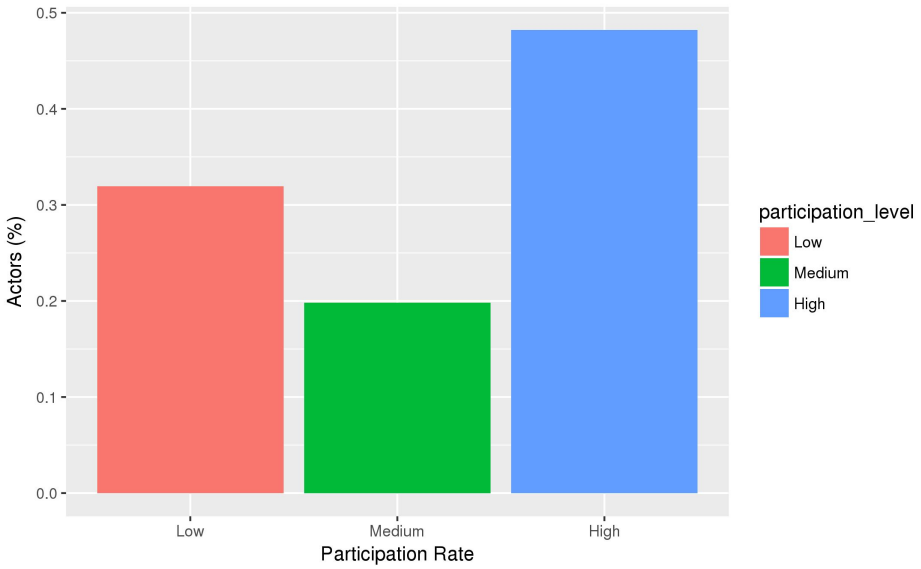
High Participation: % of repos -> % of events



Actors per Repository

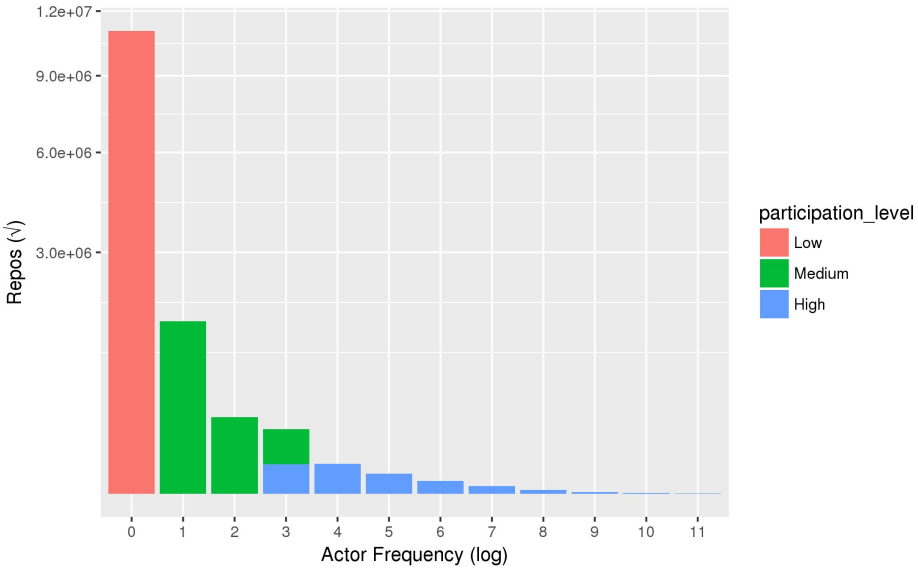
How many unique actors were responsible for events in each participation rate? Per repository?

Proportion of Actors per Repo

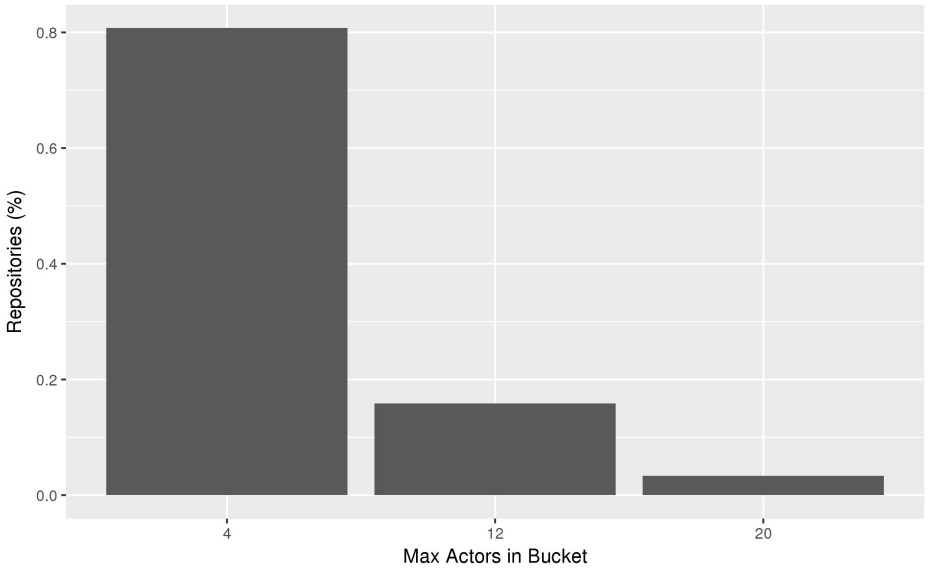


Low and High participation repositories had the most unique actors.

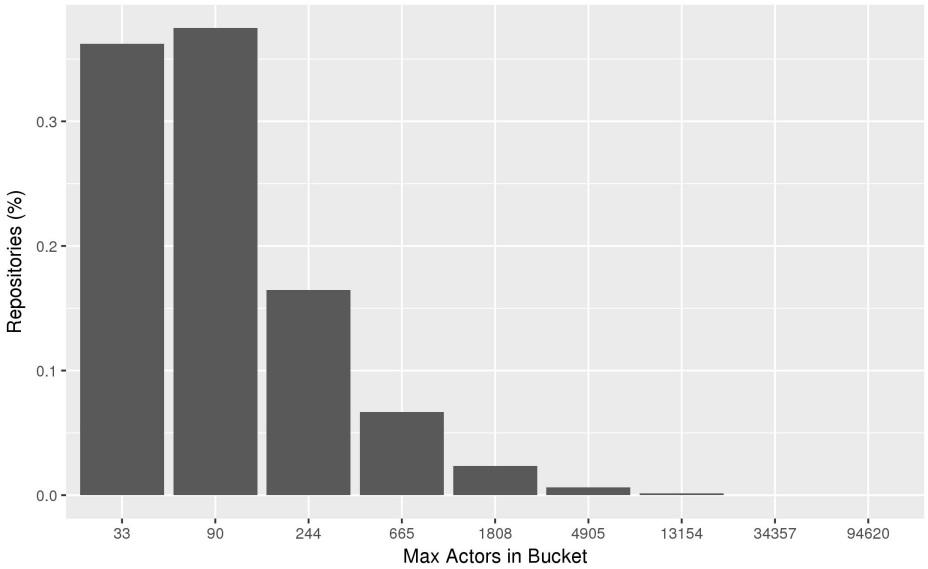
Frequency of Actors per Repo



Medium Participation Actors



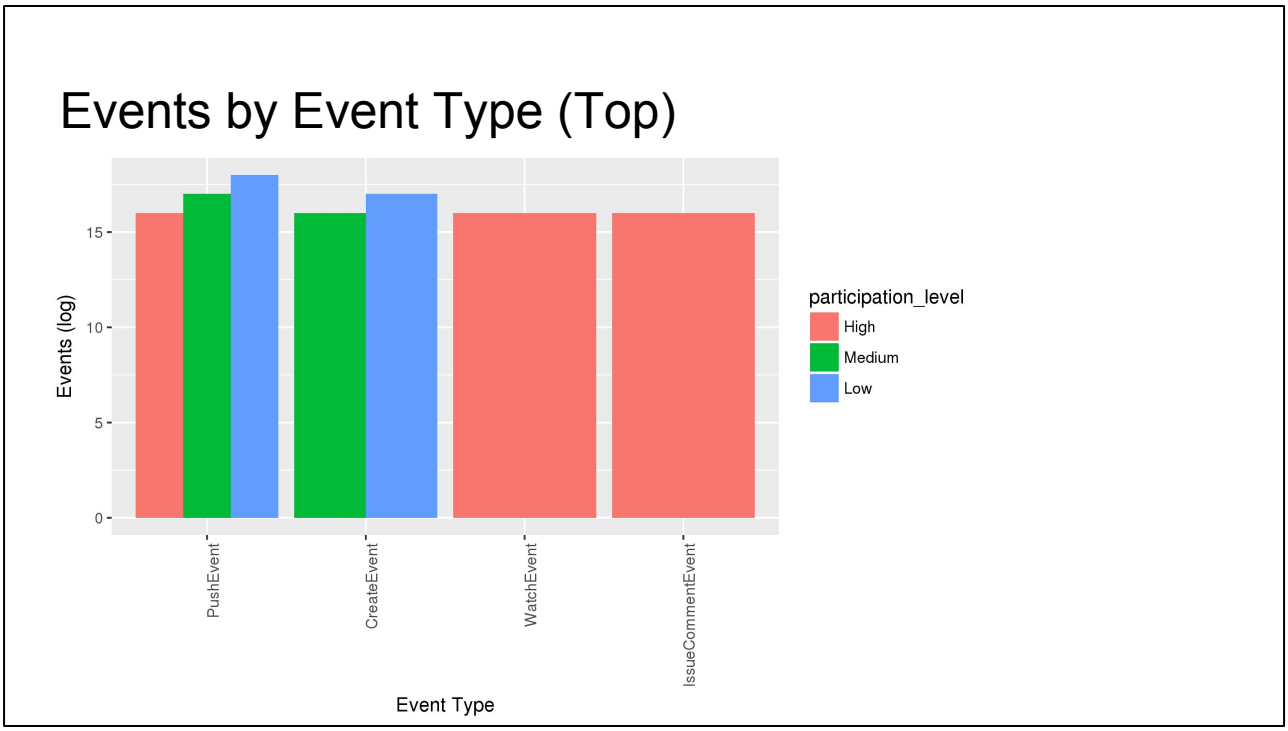
High Participation Actors



Event Types per Repository

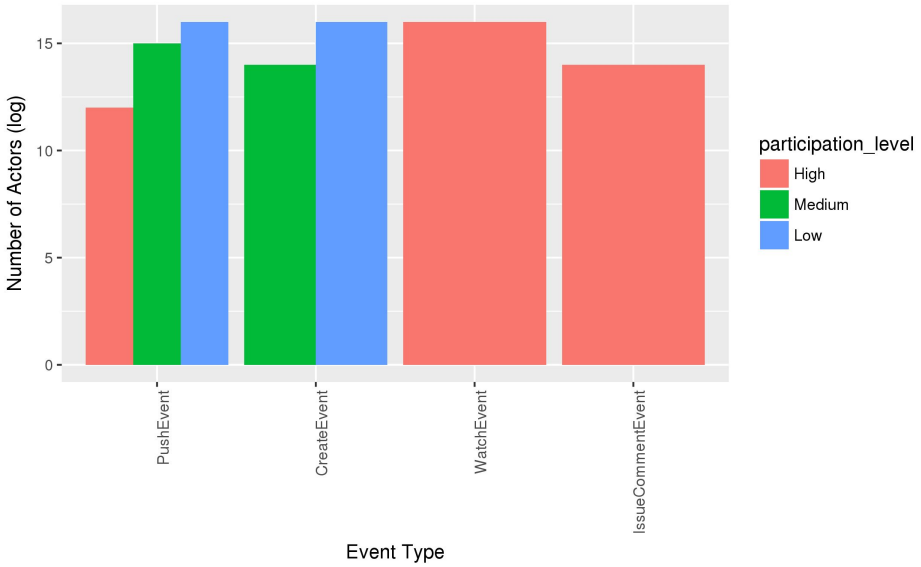
What is the distribution of event types per repository? How does this vary between participation rates?

Now we get to the MEAT!



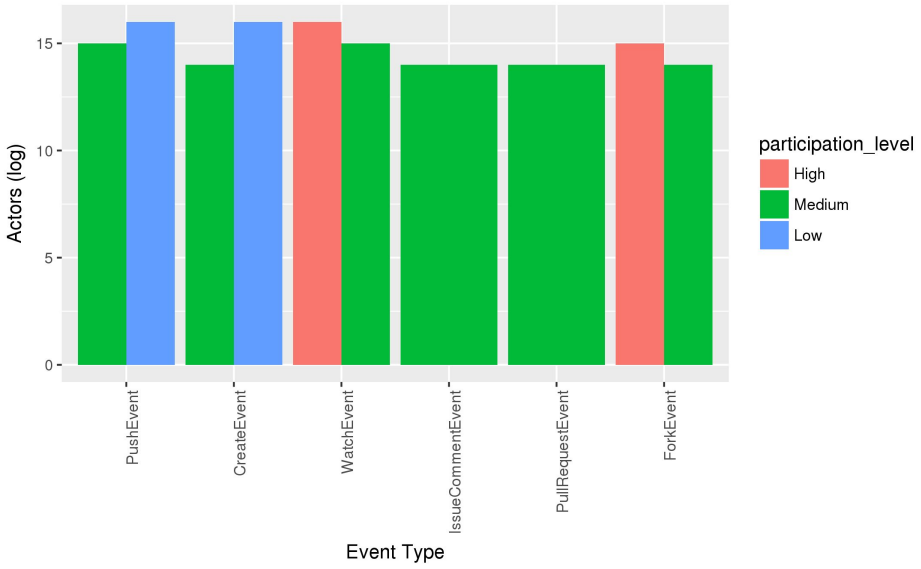
All repositories had a high frequency of Push events. High participation repositories had a high number of Watch and Issue Comment. The top event types for Medium and Low participation repositories were Create events.

Actors by Event Types (Top Events)



This shows the frequency of repository actors for the top events shown in the previous slide.

Actors by Event Type (top actors)



This shows the event types that had the highest number of repositories with the same number of actors per participation level.

Sampling Experiments

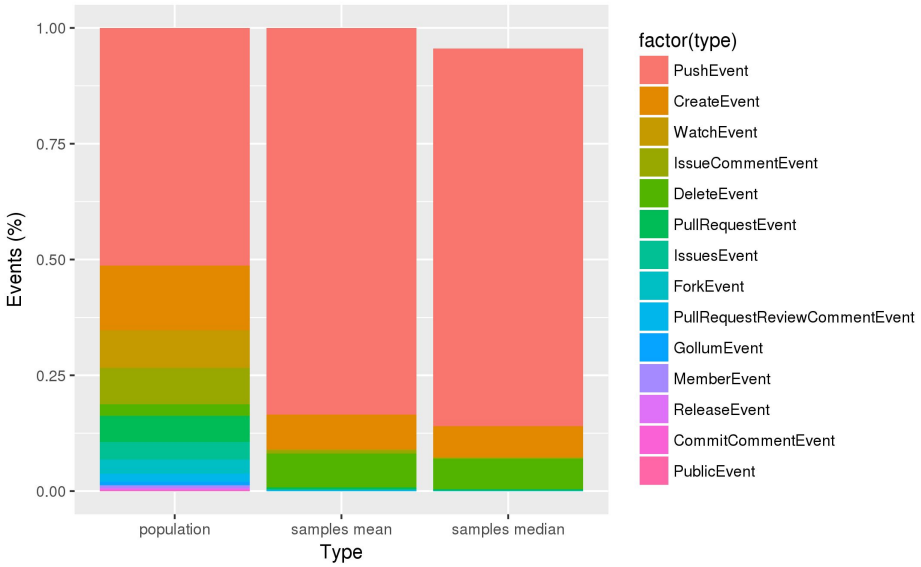
1. Samples taken from Push events would show the most variation and diversity.
2. Create events show the highest association with Medium and Low repositories.
3. Watch, Issue Comment, and Fork events occur most frequently with High participation repositories.

Samples are pulled from event types that had the highest association with one participation level. The participation level distribution will be evaluated to see if by sampling events of a certain type we can guarantee a majority of repositories at the desired participation level. If this turns out to be the case, it means we do not have to calculate the participation levels of repositories and could potentially just pull samples directly from events in the archive. 10 samples of 100 repos were pulled for each event type.

Push Events

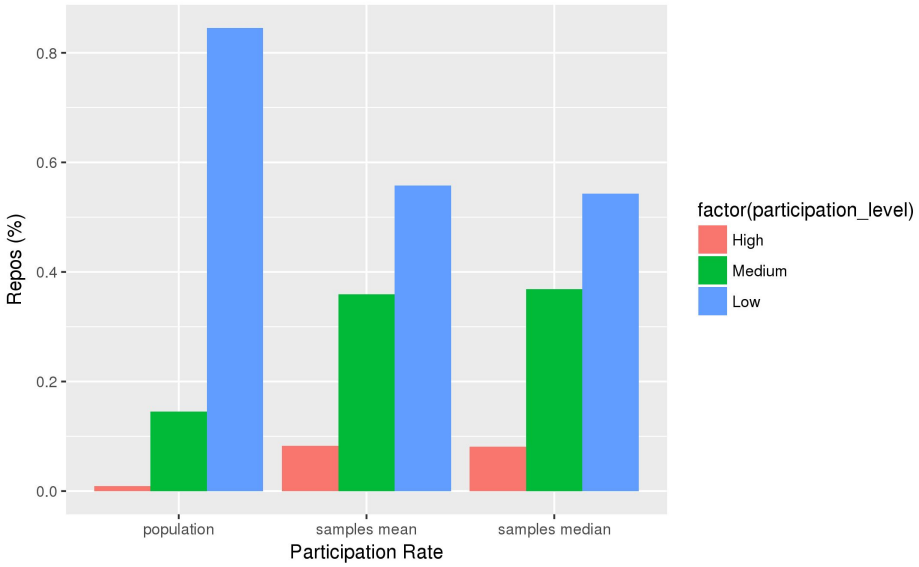
How do repositories sampled from Push Events compare to the overall population?

Push Event Types vs Population



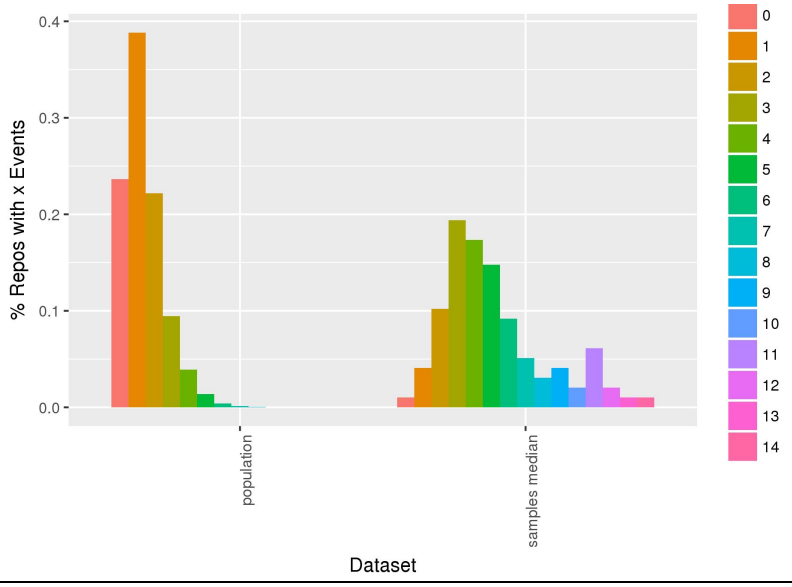
Samples from Push Events have less diversity in event types than the population

Push Participation vs Population



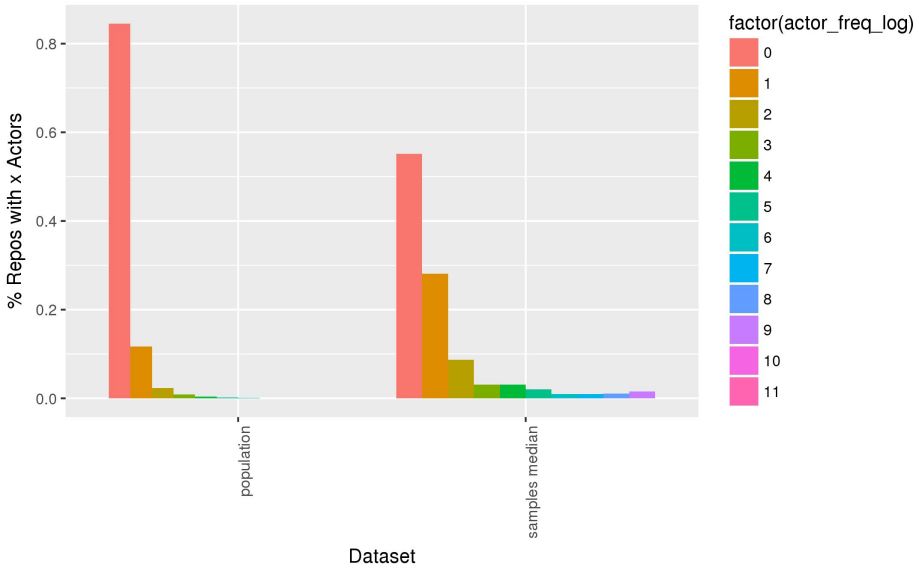
Push samples have a greater diversity of participation rates

Push Events per Repo vs Population



Push samples have a greater diversity of repositories with different event activity levels

Push Actors per Repo vs Population



There were some outliers in the population but overall Push appears to show a similar distribution considering the smaller scale

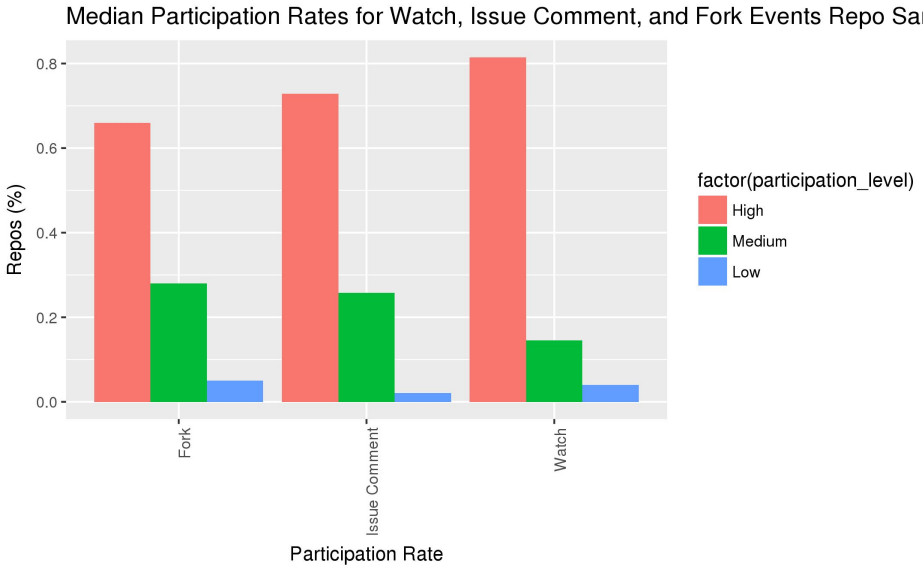
Push Events Conclusion

A sample of repositories taken from Push Events results in a more normally distributed sample than what was found in the population. Therefore, samples drawn from Push Events would allow us to study many different types of GitHub repositories without the heavier skew seen in the overall population.

Watch, Fork, and Issue Comment Events

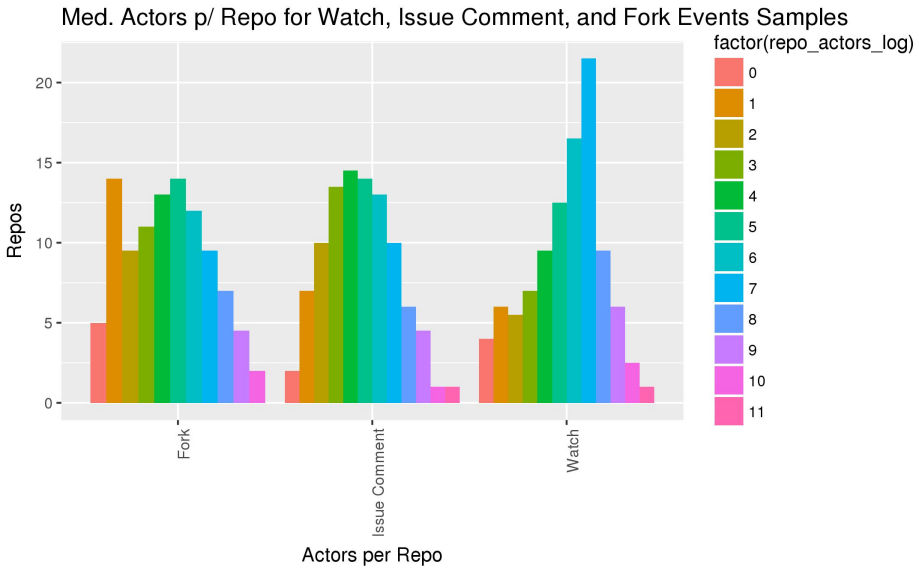
How do repositories sampled from Watch, Fork, and Issue Comment Events compare to the overall population and to the High Participation repositories studied earlier?

High Participation: Watch, Fork, Issue Comment



All three event types show a high frequency of High participation repositories

High Participation: Watch, Fork, Issue Comment



All 3 show a high frequency of repositories with a high number of unique actors

Watch, Fork, and Issue Comment Events Conclusion

A sample of repositories taken from Watch Events results in repositories that have a larger number of events distributed amongst its actors. It also results in a repositories having a high number of unique actors.

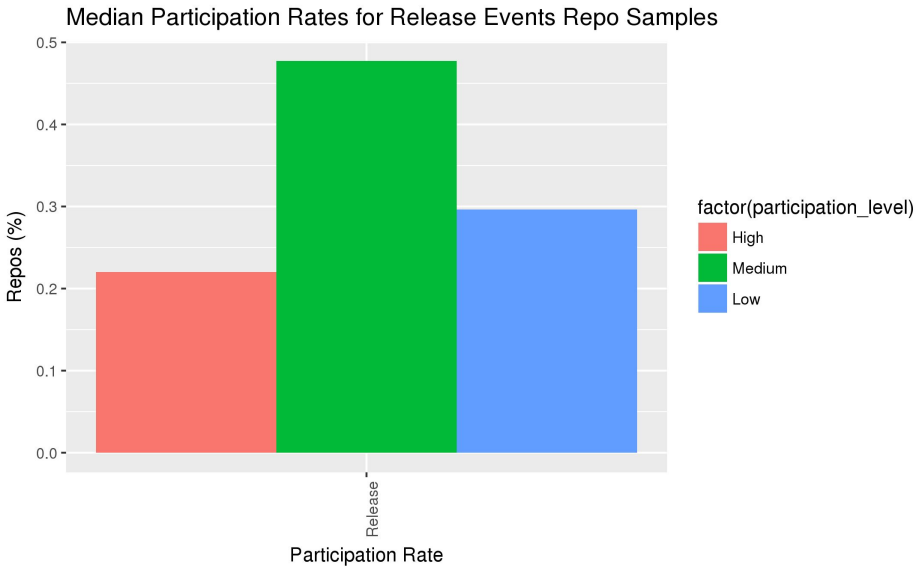
Both Fork and Issue Comment events also leaned towards a higher number of events distributed among its actors. Fork Event repositoires leaned towards having a low to medium repo actor frequency, while Issue Comment Event repositories showed a more normal distribution of repo actor frequency.

Therefore, samples drawn from Watch Events would produce a sample repositories with a higher number of actors. Fork events previously showed a strong correlation to number of actors, so it should also be studied.

Release Events

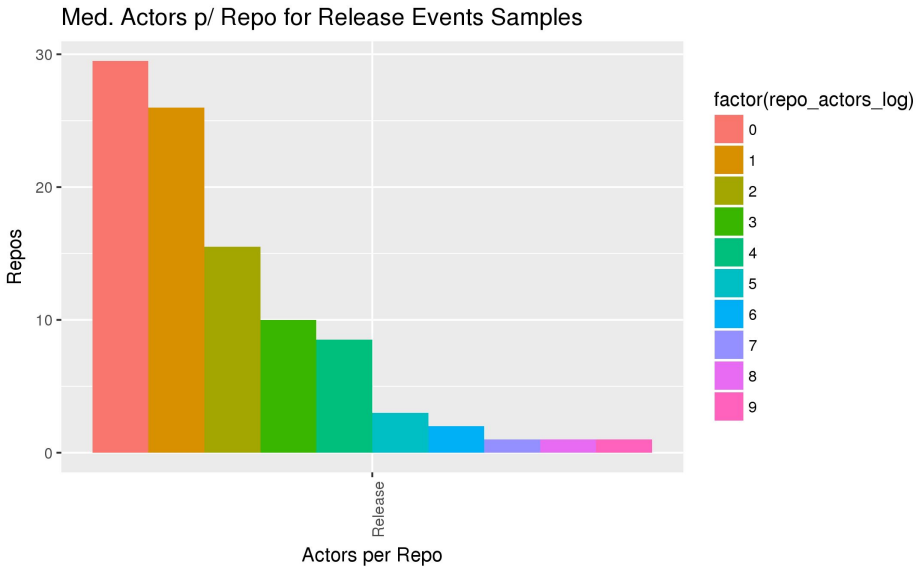
How do repositories sampled from Release Events compare to the overall population and to the Medium Participation repositories studied earlier?

Medium Participation: Release



Release events skew towards a lower participation rate

Medium Participation: Release



Release events tend to have more repositories with a lower number of actors per repository.

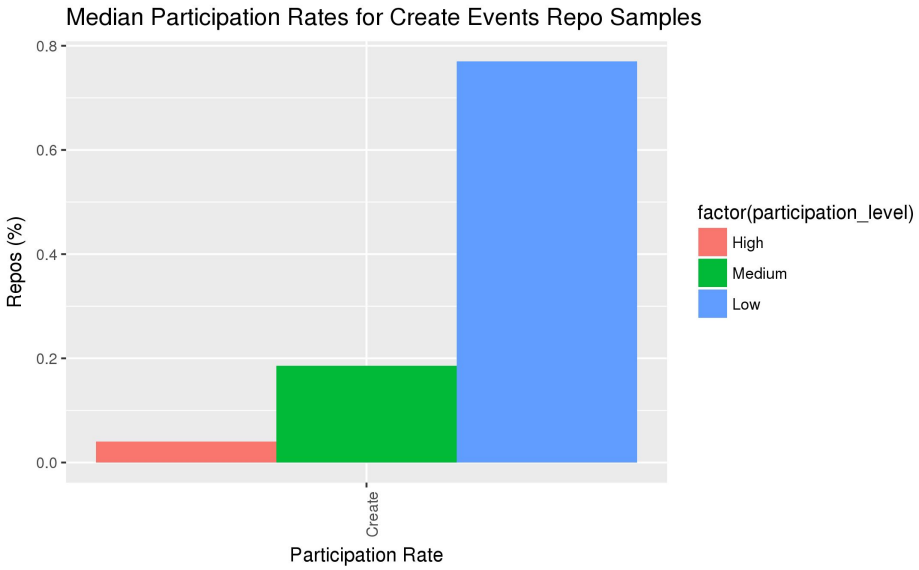
Release Events Conclusion

A sample of repositories taken from Release Events results in a sample that skews towards lower participation and a lower number of actors per repository.

Create Events

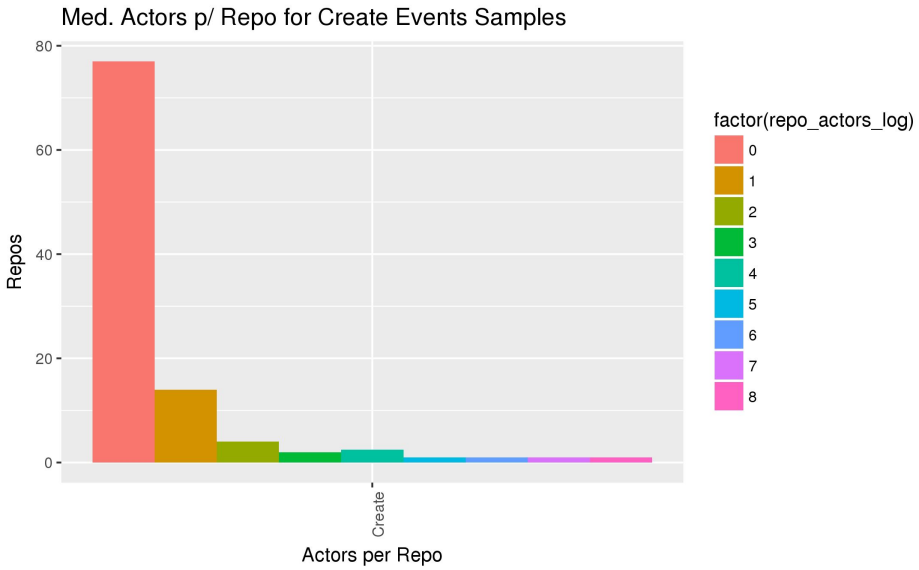
How do repositories sampled from Create Events compare to the overall population and to the Low Participation repositories studied earlier?

Low Participation: Create



Create events show a large proportion of low participation repositories.

Low Participation: Create



Create events tend to have more repositories with only 1 actor per repository.

Create Events Conclusion

A sample of repositories taken from Create Events results in a sample that leans towards lower participation and high frequency of repositories with only a single actor. Therefore, samples drawn from Create Events would produce a high proportion of repositories with only one actor.

Because we are less interested in these types of repositories, further analysis will not be done on Create Events at this time.

Hypothesis (Reprise)

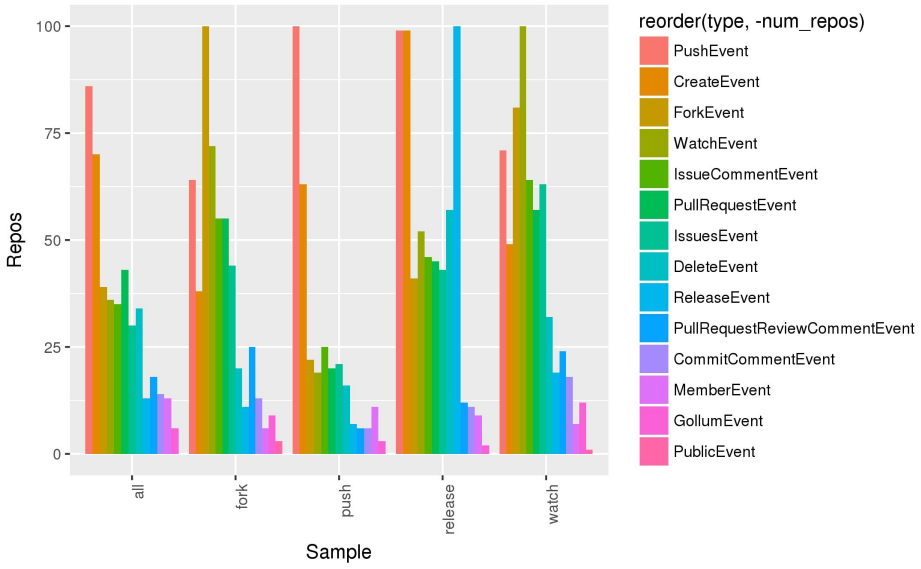
- ❏ Repository participation is strongly correlated with certain types of GitHub events.
- ❏ Therefore, repository samples drawn from these event types will show a higher proportion of repositories that share a level of participation correlated with that event type.
- ➔ Repositories sampled from these event types will also share other parameters

Github Repo Samples (aka testing our methods)

Do the repository samples line up with our expectations?

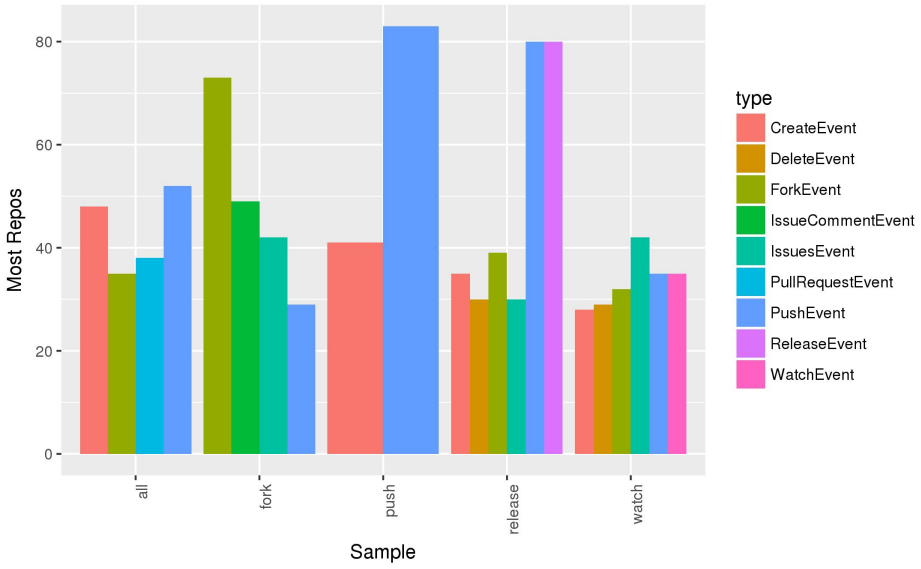
Do we see things in common within each cluster that we don’t see in common outside of that cluster?

Event Types per Repository



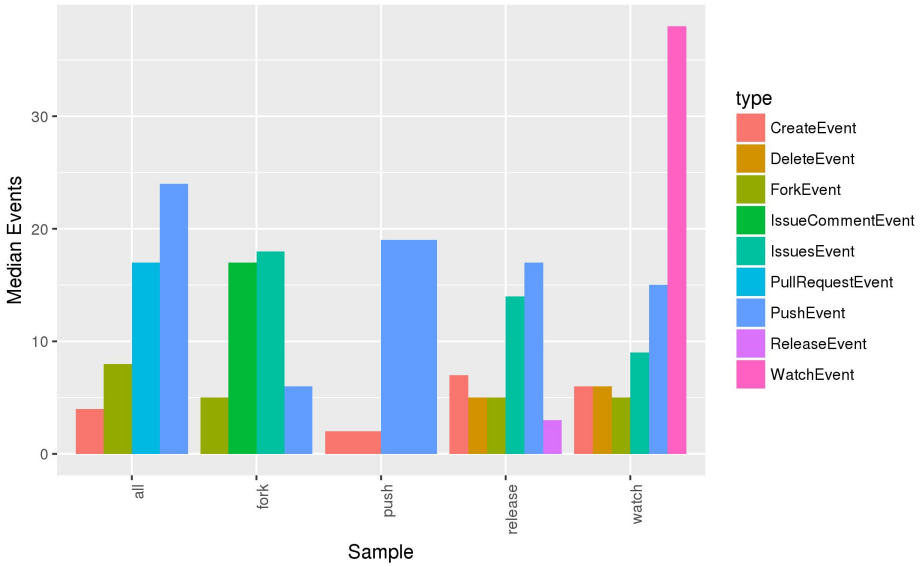
Total count of repos that had an event of the given type of the sample.

Top Repos sharing Event Frequency



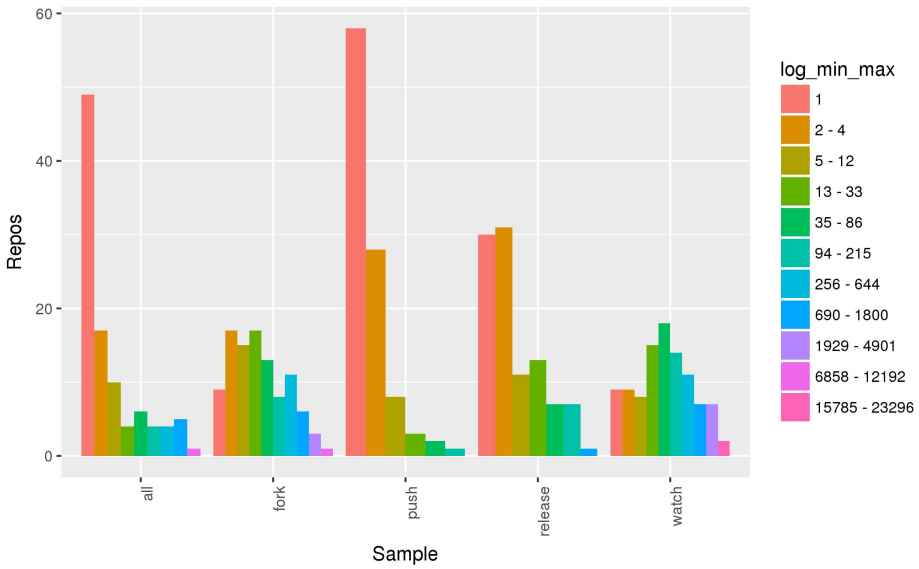
Buckets were created by taking the natural log for number of events per event type for each repo in each sample. I then counted the number of repos that fell into each bucket and ranked them. This plot shows the top number of repos per sample that fell within the same event type frequency bucket where the total number of repos is at least 25 or ¼ of the sample. So, for example, we can read it as, the majority of repositories in the Push sample had a similar number of Push events per repository where only a minority of repositories shared the same number of commit comment events. Remember our hypothesis is that we believe some parameters will be less variable between repositories in our event type samples than in our control sample.

Median Events for Highest Repo concentration

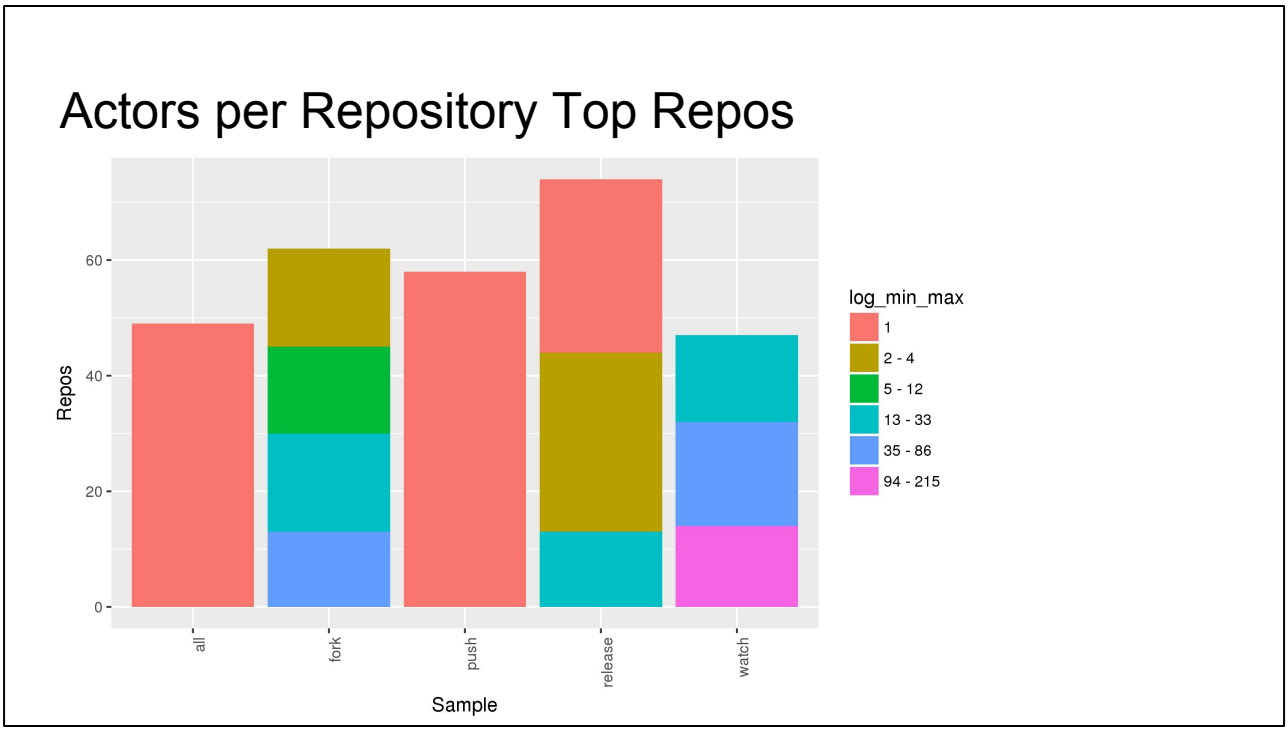


This is the median number of events for the highest concentration of repo activity (see previous slide). This plot shows the median number of events for the buckets with the highest numbers of repositories. Or in other words, here we're looking at the median number of events per type for the highest concentration of repositories with common activity. One thing to keep in mind when looking at this chart is that a high number of events may only be shared by a small proportion of repositories. We're looking for a pattern of a high concentration of repositories and a high number of events.

Actors per Repository

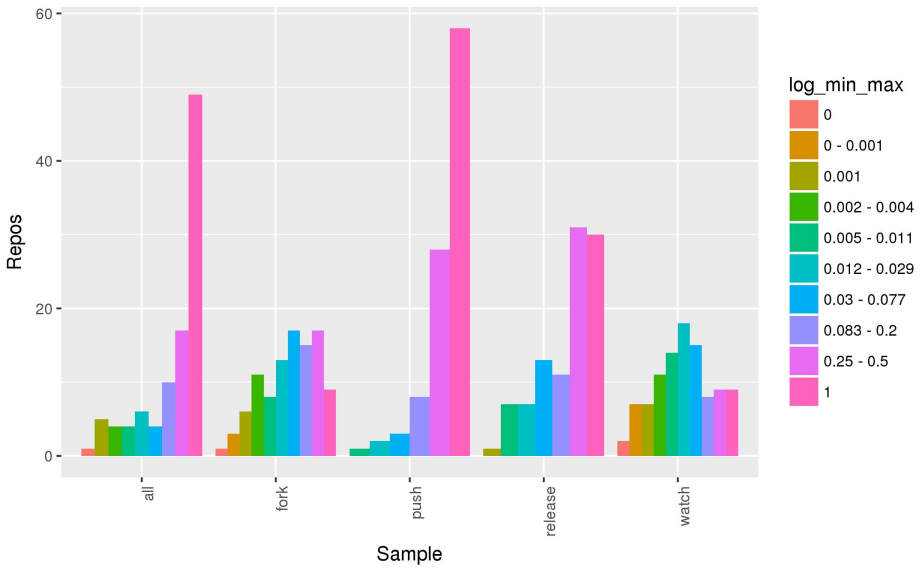


Fork and Watch have significantly more buckets than the all repo sample which means they will have less repositories per bucket. This means Fork and Watch have a greater variability between total actors per repository than the other samples. The control sample is left skewed, with the majority of repositories having only 1 actor per repository.

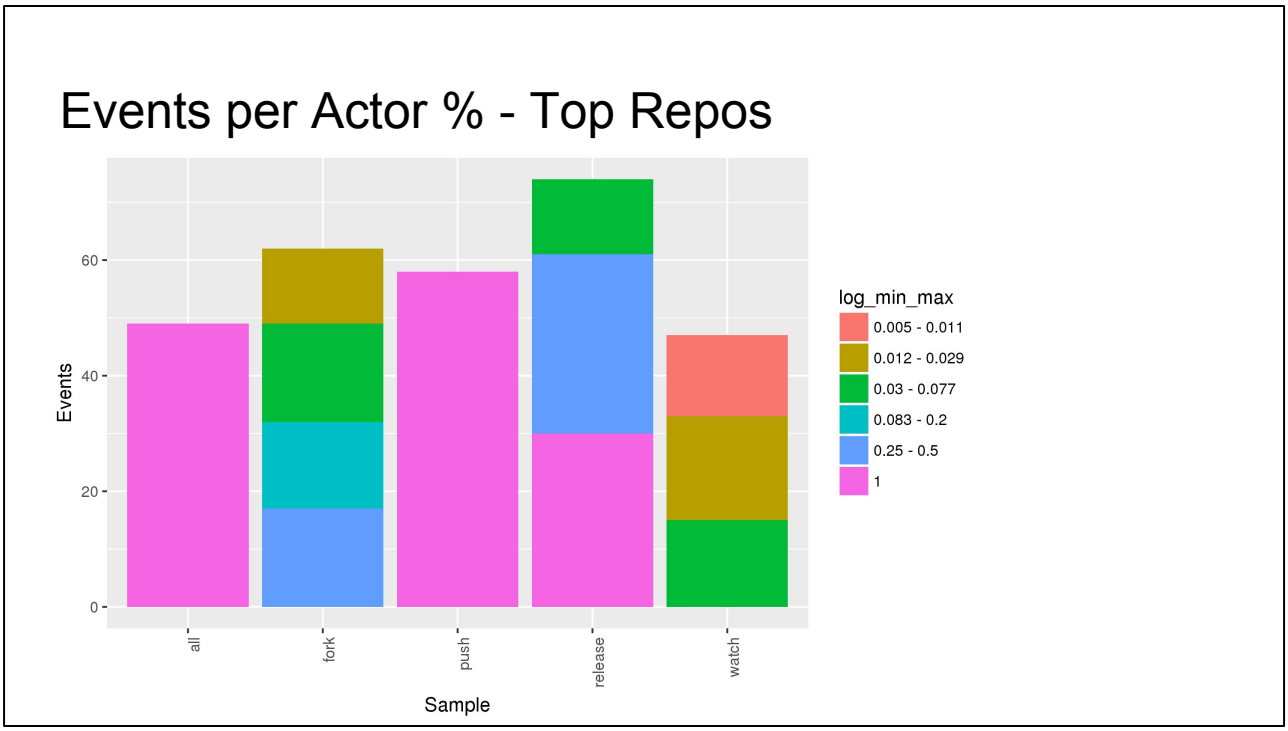


This plot groups the total actors in the previous plot by repo frequency. That is, the highest concentrations of repositories sharing a similar actor frequency were combined to better assess how variable the samples are. All and Push are the least variable. It appears that the more actors per repo in a sample, the more likely repositories are to vary.

Events per Actor %



Watch has a nearly normal distribution for events per actor percent. Fork is slightly left skewed while the other samples are more heavily left skewed.



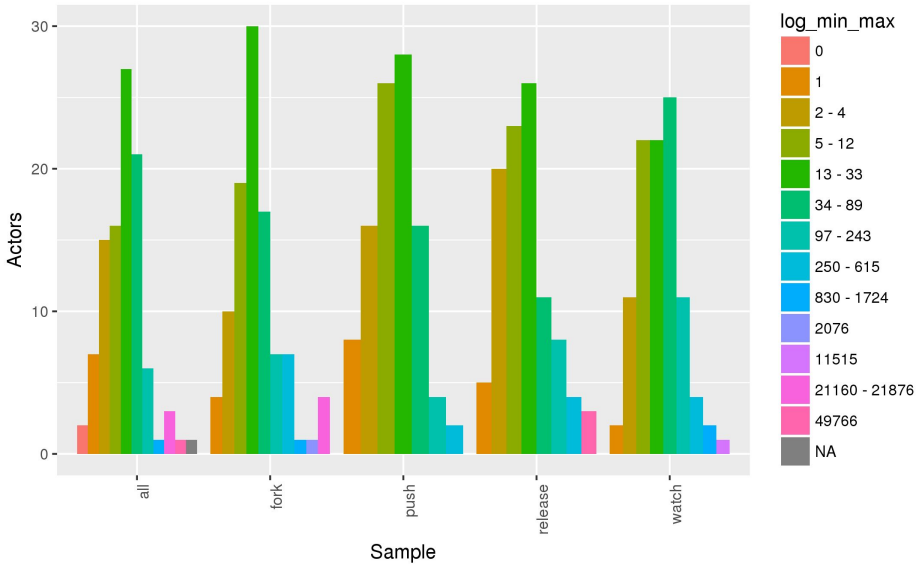
Here we combine repo frequencies that are in a similar range and compare the totals. The possible e2a percent values are 0 to 1, or 0 to 100% of events attributed to one actor. 50% of Watch repositories fell between .5 and 7.7% of events per actor. Release had the highest number of repositories sharing a common e2a percent. The control sample and Push showed the least diversity with the majority of samples sharing an e2a value of 1, which means 100% of events can be attributed to one actor.

Github API Parameters

What patterns do we see for repos sampled by event type for parameters outside of the Event data?

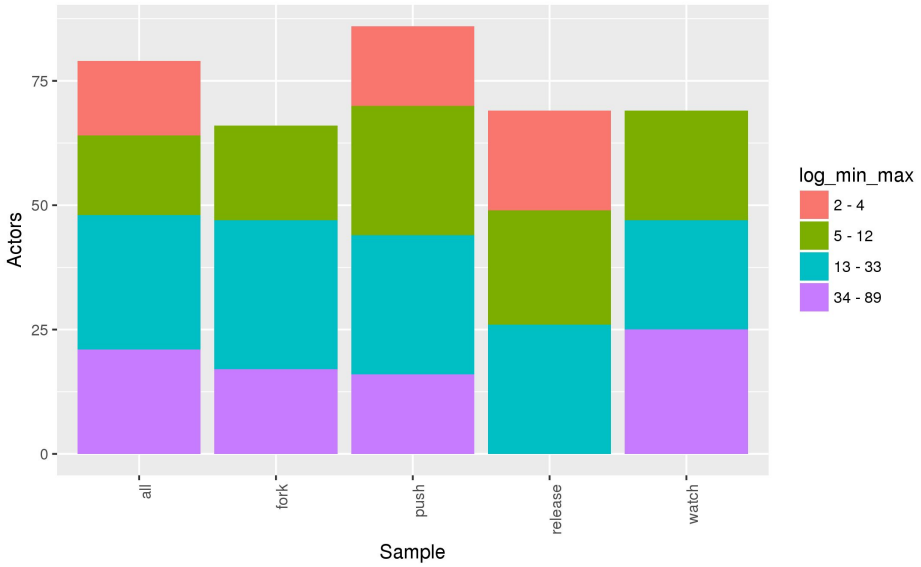
Do we see common patterns amongst repos in each sample that are different from the other samples?

Repositories per Owner



Number of repositories owned by each Repository Owner (according to Github API) are shown by color. The bars represent the number of repo owners that shared a common repository ownership bucket. For example, in the Watch sample, the highest number of actors sharing a similar number of repositories was 25 and they owned between 34 and 89 repositories. All of the samples are right skewed but Watch looks to be the closest to a normal distribution with the majority of actors sharing a closer range of repositories owned than the other types.

Repositories per Owner - Top Owners



This plot shows the top concentrations of actors who shared the same number of common repositories. Push and All had the most actors who shared a common repository ownership frequency spread across all ranges. Watch and Fork had more actors who shared a high repository frequency than in the control sample and therefore less variability than the other samples. Watch has a nearly normal distribution with the majority representing the center of the distribution as per the previous slide while the other types show a left skew.

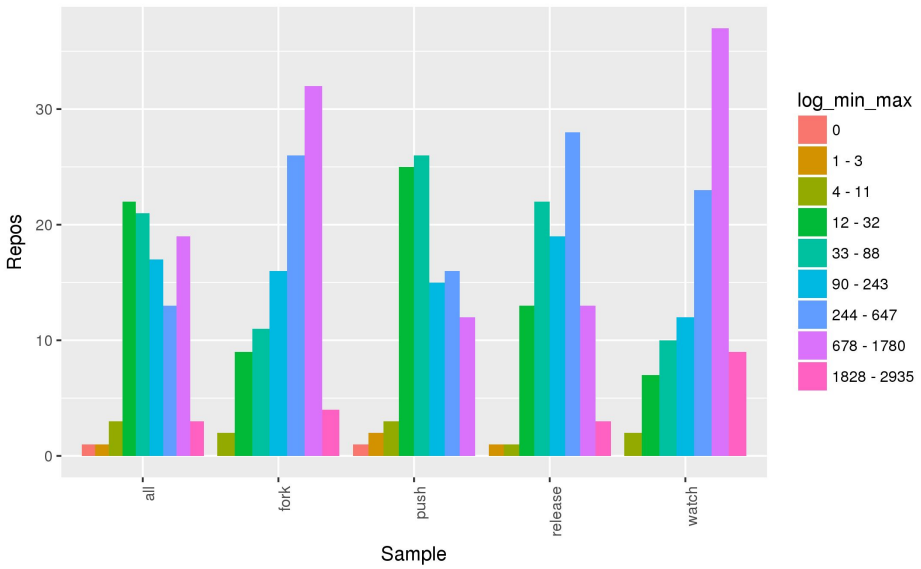
Repository Age and Updated

Which event types are more likely to have “stale” repos?

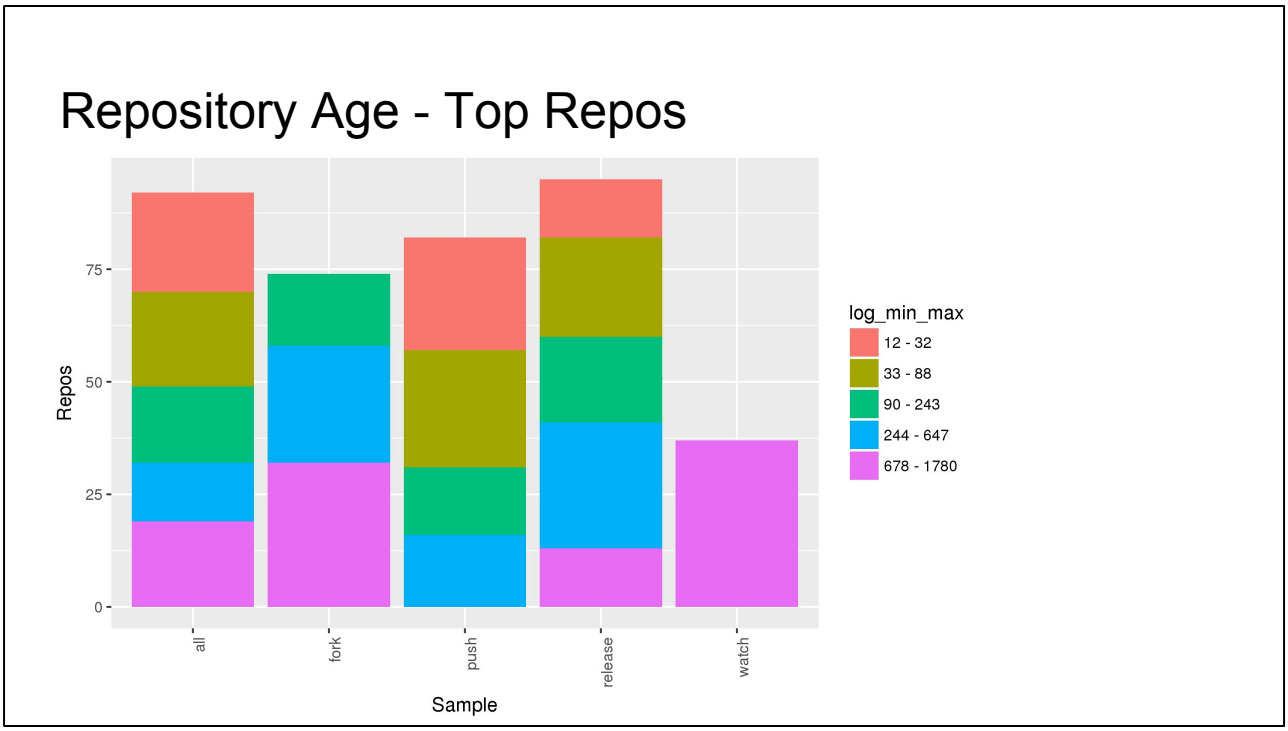
Which event types are more likely to have “fresh” repos?

Probability of each Updated-Age score for each event type?

Repository Age

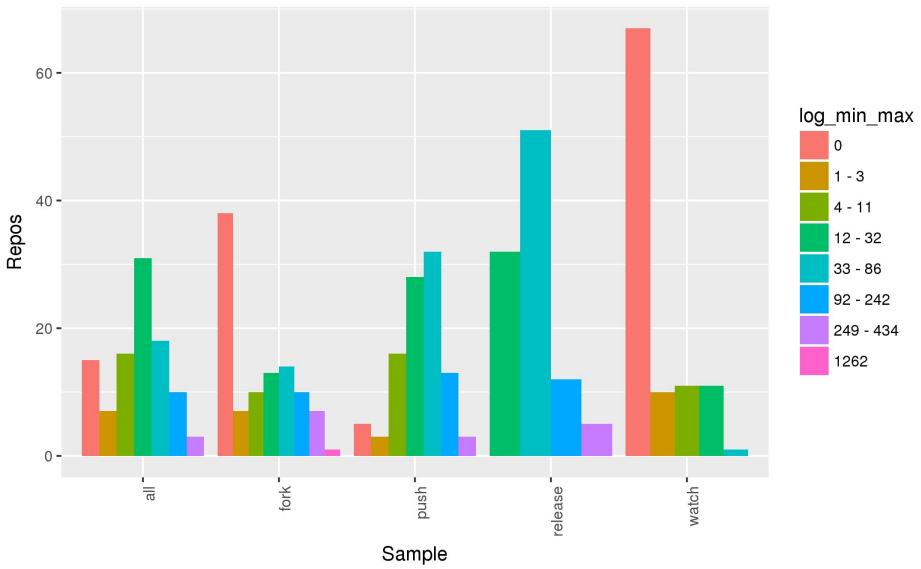


Repository age is computed as the difference between when the sample was drawn and the “created at” field provided by the Github API. For deleted repositories, the created at field was determined by finding the first event for the repo in the event history, either a create event of type repo as indicated by the payload or a fork event from another repo. In some cases the first first event was a push event. Each sample has a fair amount of variability for this value, however Fork and Watch show a strong skew towards older repositories than the other types. Based on this plot, one can hypothesize that if one is interested in studying older repositories, Fork and Watch events would provide a better representation of that.



Looking at the top repository frequencies, we see the least variability in Watch and the most variability in Release and the Control. The top was taken by taking the max of the rounded natural log of the total number of repos in each bucket per sample to see how many different buckets fell into each one. Watch had only one group falling into its maximum while the other samples had several.

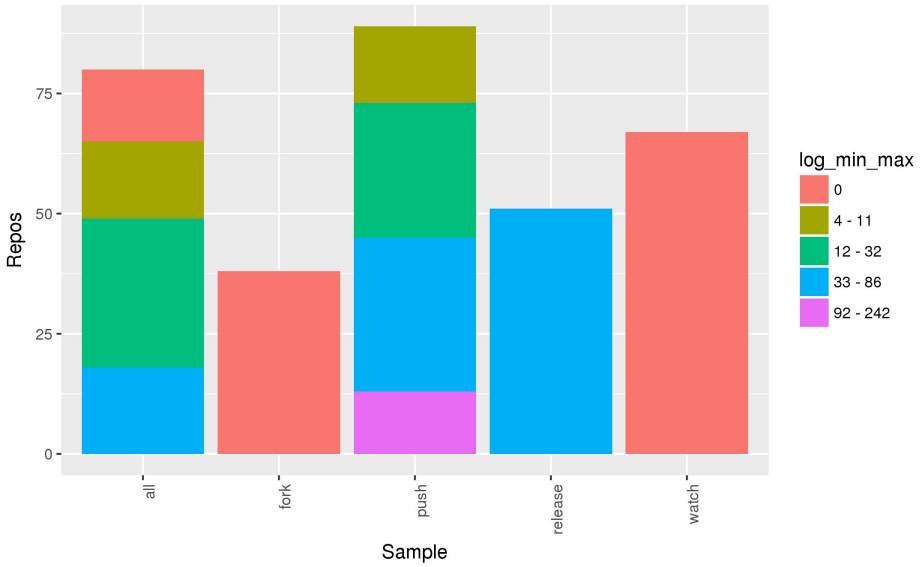
Updated Since



The number of days since a repository was updated from the time the sample was taken according to the Github API. For repos that did not have available data through the github api, this was determined by the last event recorded in the sample set taken from the Github Events archive. Watch shows the strongest majority with most of the repos sharing an updated since value of 0 days. Release shows the least variability with the majority of repos having updated within 12 to 86 days.

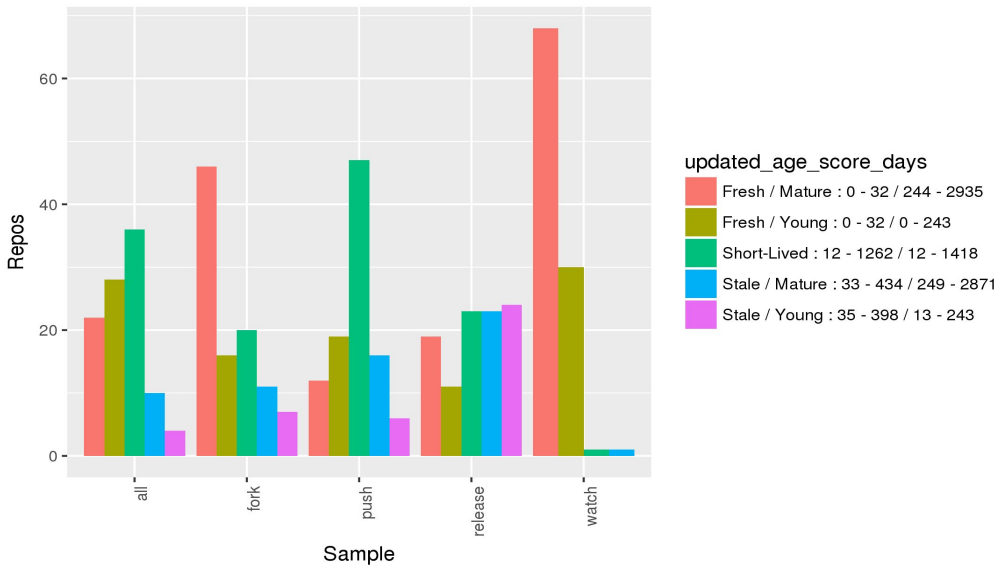
Note that updated at includes all changes to the repository; might be a commit, changing the description of the repo, creating wiki pages, changes to branches, etc.

Repository Updated Since - Top Repos



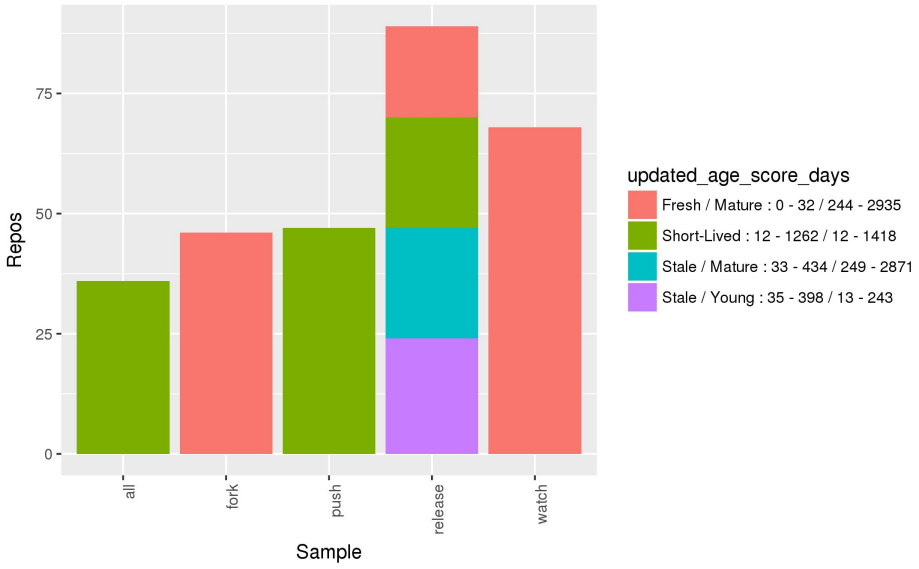
The range of updated since days for the maximum of the rounded natural log for the number of repos in each bucket is shown here. Or rather, this shows the majority updated since values for the highest concentration of repos in each sample. Fork, Watch, and Release show a clear majority that only has one updated since bucket.

Age vs Updated Since



This compares the age with the updated since value by creating a score, described in the legend. Watch has the highest proportion of repositories that are recently updated and nearly none that are “short lived” (meaning their age and updated since are the same value and over 12 days).

Age vs Updated Since - Top Repos

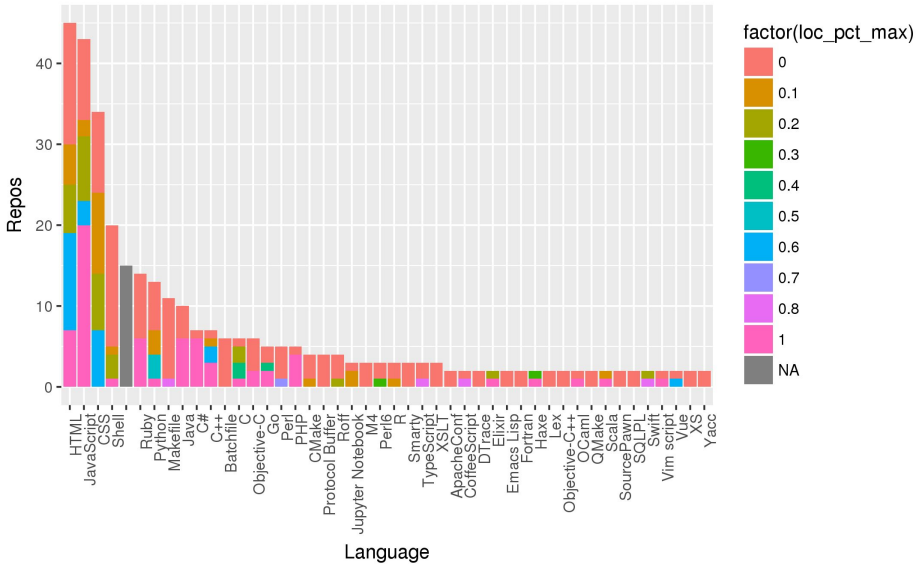


Combining the Updated Since and Repository Age values is a little tricky because there is a lot of variation between the two. A single ratio or other computation doesn't differentiate between a high updated since and a low repo age or vice versa. To categorize these properly, we need a combined score. Repository age is rated as "Mature" or "Young". Updated Since is rated as "Fresh" or "Stale". Mature means the log of the age in days is greater or equal to 6. Fresh means the log of the updated since in days is less than 3. The days ranges are indicated in the legend with the minimum and maximum updated since coming first. Short-Lived are repositories that had the same value for updated since and repository age indicating they were only worked on when they were created. "Young" repositories were excluded from this and put in the "Fresh/Young" category.

Repository Languages

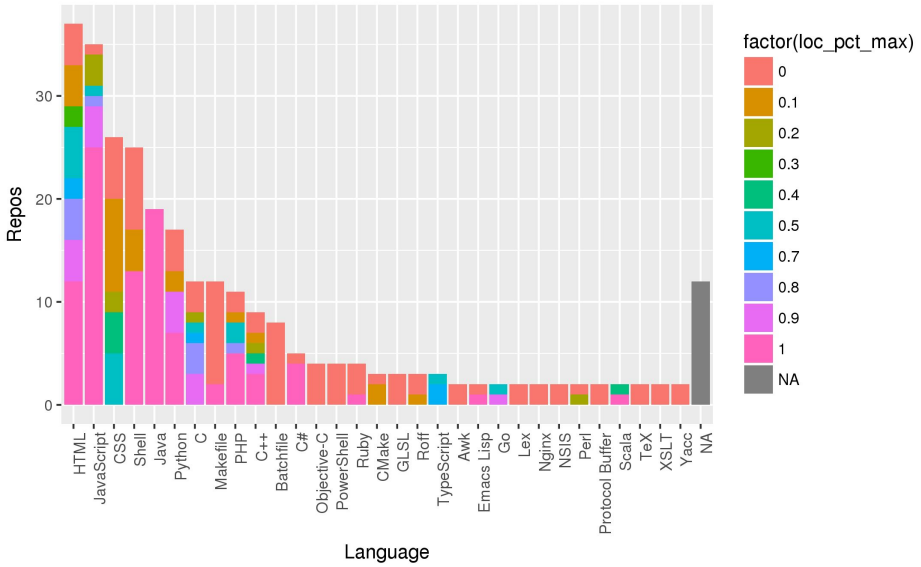
1) Number of most frequent languages per repo - which event types had a smaller number of top languages?

Control: Top Repository Languages (>1 Repo)

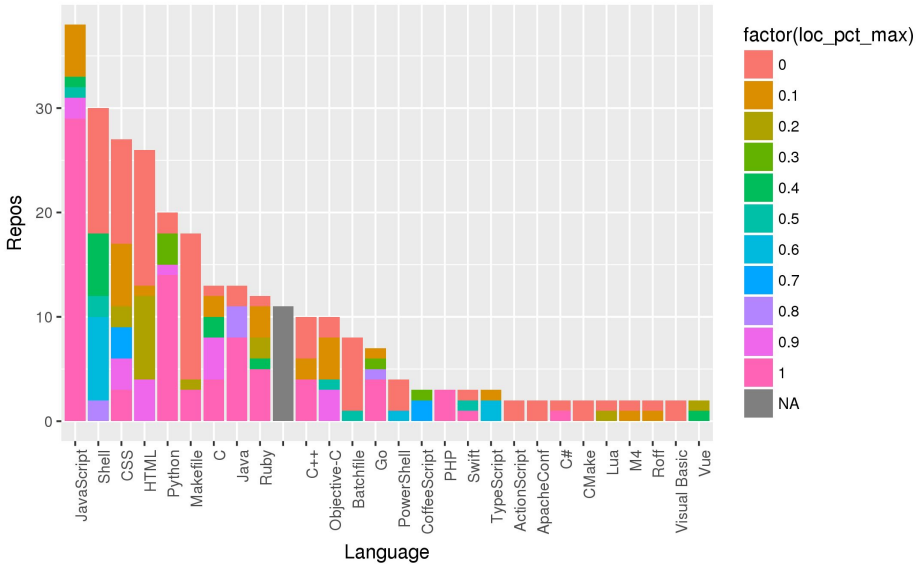


Total repos per language (as identified by Github), additionally what the maximum lines of code percent was for each repo in that language.

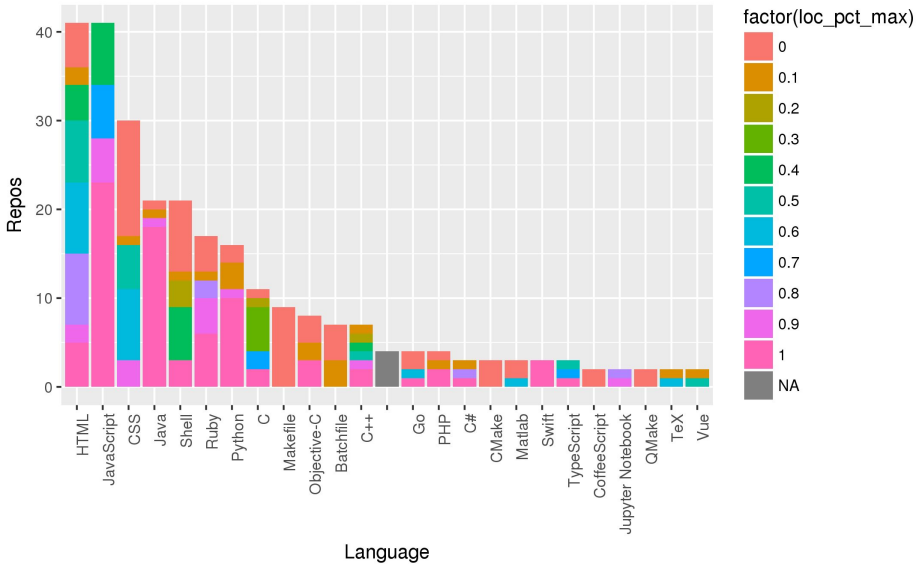
Push: Top Repository Languages (>1 Repo)



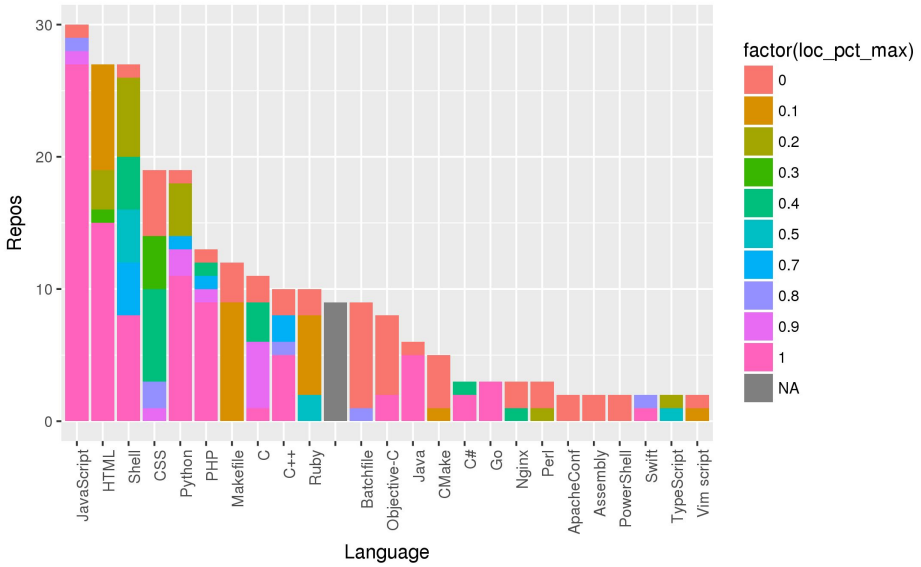
Watch: Top Repository Languages (>1 Repo)



Fork: Top Repository Languages (>1 Repo)



Release: Top Repository Languages (>1 Repo)



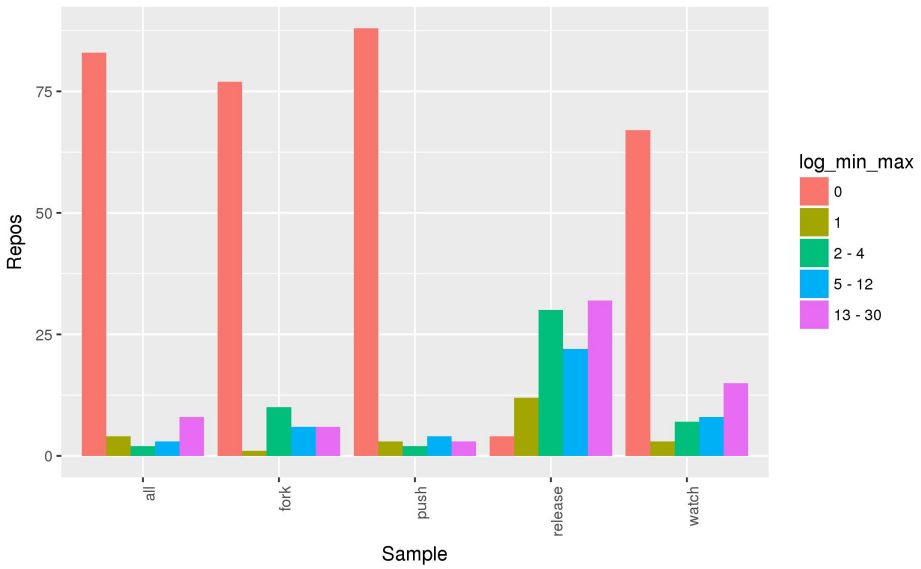
Releases

How many repos had releases?

How many repos appeared to do releases on a regular release schedule?

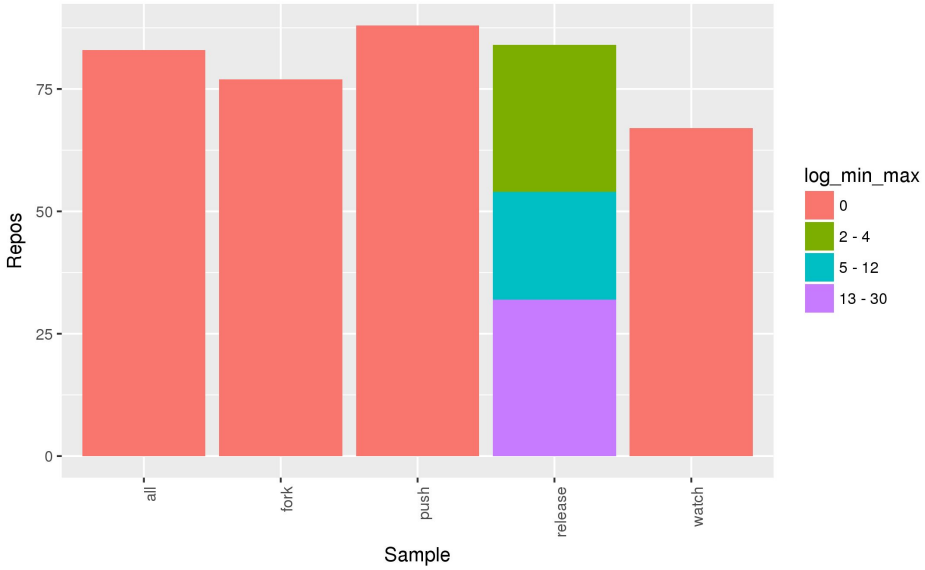
From looking at the samples it's not clear that the Github Release API tells the whole story about releases. Only a minority of repositories in most cases had release activity. To better determine this, one would need to use tools like Library.io to match these repos with release activity to common code distribution tools to see if repos are actually doing releases but not in a way Github recognizes.

Releases

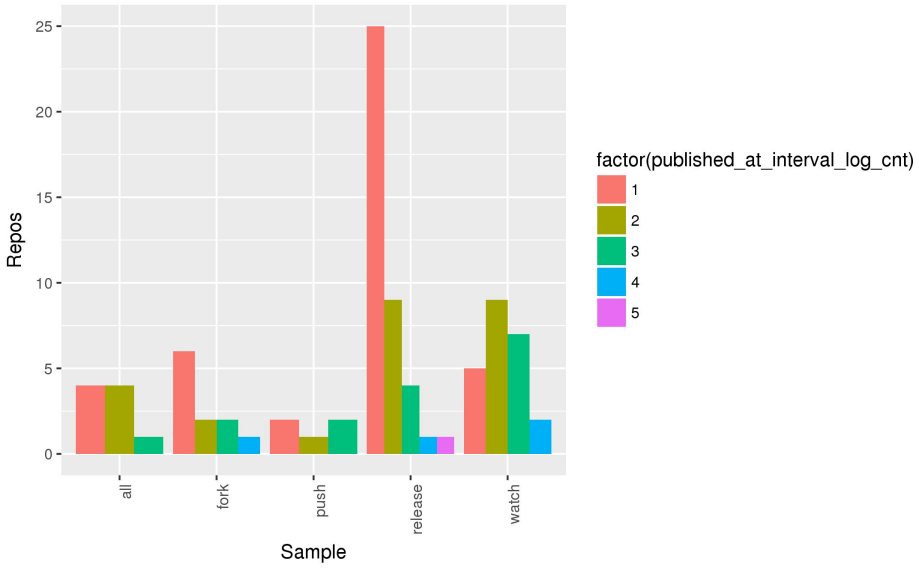


The majority of repositories in all samples except for Release had no releases. Not surprisingly, the Release sample had the smallest number of repos without releases. How is this possible if we are sampling release events? Release data came from the Github API rather than from the event data. In the case of those repos, their data were unavailable due to repository deletion or a change to private. Would capturing release data for deleted repositories make a difference? Not likely since this only affected 1-3 repos per sample.

Releases - Top Repos

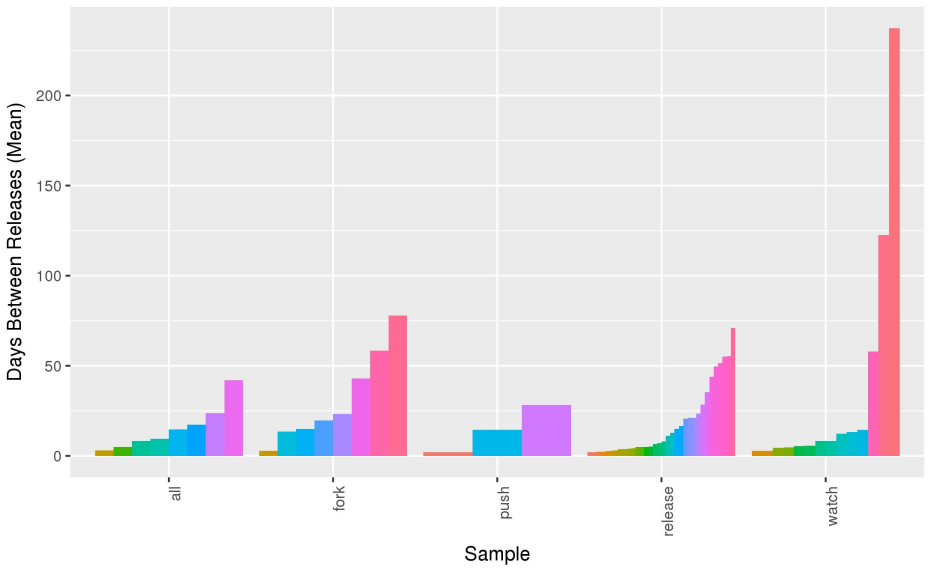


Time Between Releases - Interval Frequency



Is there a regular interval between releases? Which event type sample has the most regular intervals?
Intervals between releases were calculated in days and then a natural rounded log was used to create categories. Total releases per repo were then tallied for each interval and then a natural rounded log of number of releases was calculated to see where the maximum number of releases fell per repo. In most cases, more than one release interval had the maximum number of releases. Repos with a maximum release count of 1 (or log 0) were excluded. This shows, per sample, the number of intervals per maximum releases per repo. So Release samples had the highest proportion of repos with only 1 interval bucket which means these are most likely to release on a regular interval. It also had the highest number of interval buckets (5) suggesting these repos did not release on a regular interval.

Time Between Releases - Most Consistent Intervals



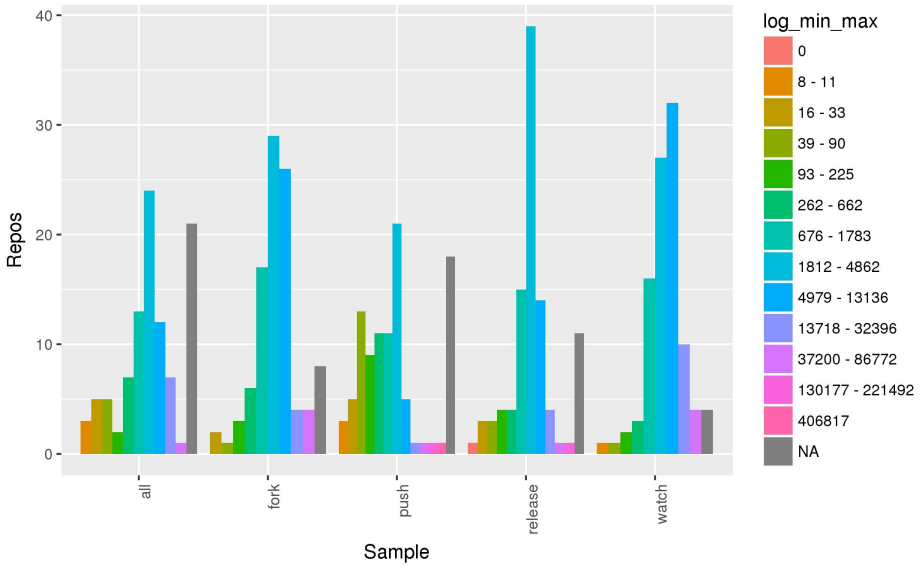
Repos with a release interval bucket count of 2 or less are plotted here, meaning that there is likely more consistency between release times amongst these. This shows the average time between releases for repositories in each sample where each color represents a repository in the sample. This is just to get a feel of what the release interval distribution is among these repositories. It would be interesting to see if there is any correlation between the age of a repo and its release frequency. Of course further analysis is needed to determine if there actually is a regular release interval.

Readme

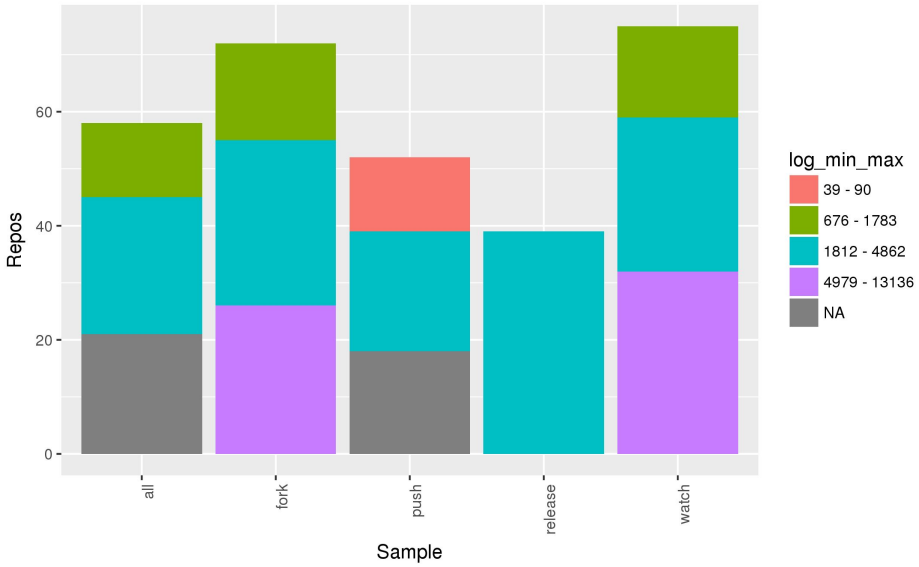
How do event type samples compare in terms of readme size?

This is an easy parameter to calculate but it’s not clear if it actually has any meaning. The only thing we are considering here is how Readme size is distributed among the samples and if there is any correlation between Readme size and event types.

Readme


Repos
40
30
20
10
0
all
fork
push
release
watch
Sample
log_min_max
0
8 - 11
16 - 33
39 - 90
93 - 225
262 - 662
676 - 1783
1812 - 4862
4979 - 13136
13718 - 32396
37200 - 86772
130177 - 221492
406817
NA

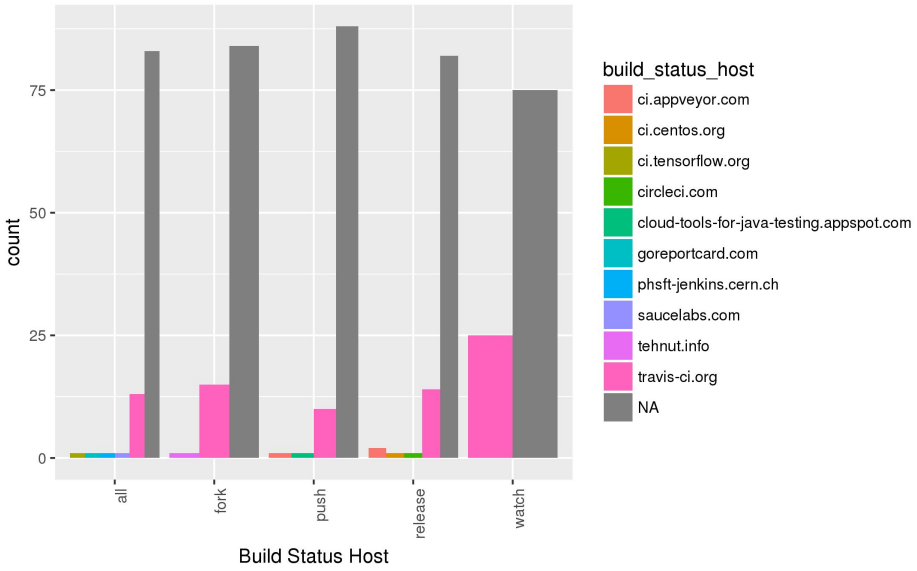
Readme - Top Repos



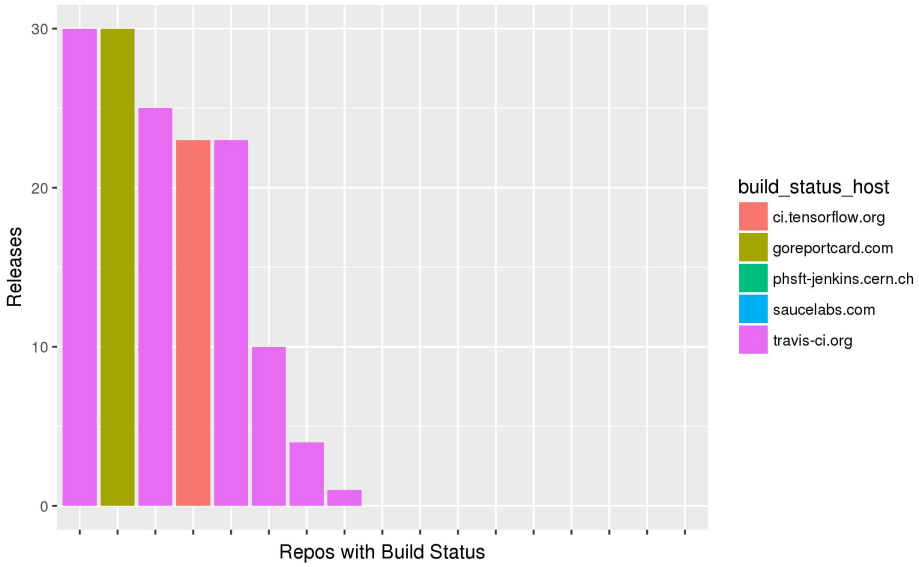
Build Status in Readme

Which event type samples are more likely to have a “build status” tag in the Readme?

Build Status in Readme

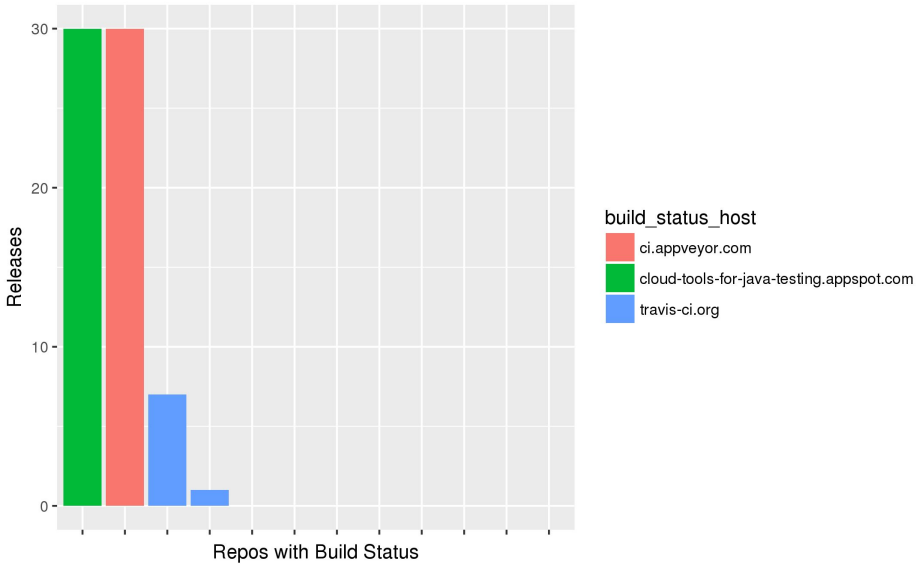


Control: “build status” in Readme vs Releases



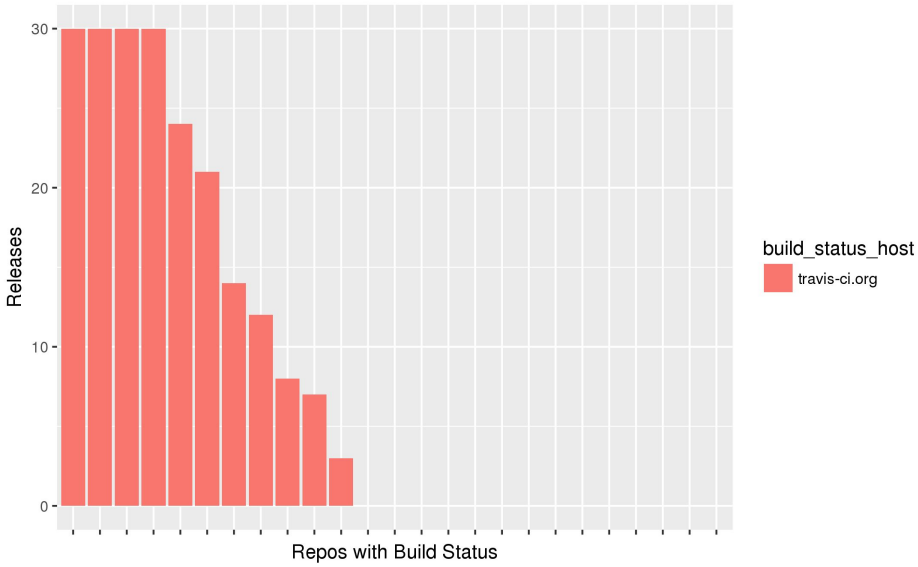
Individual repos with a readme and a build status tag are represented on the y-axis. Of the repositories that had release events and also had a README with a “build status” tag, 100% used TravisCI. This does not mean that other repositories did not do releases, it just means that they did not register release events through the Github API. Further analysis is needed possibly with a tool like Libraries.io to track releases against package managers to discover more about releases that could be happening outside of Github.

Push: “build status” in Readme vs Releases



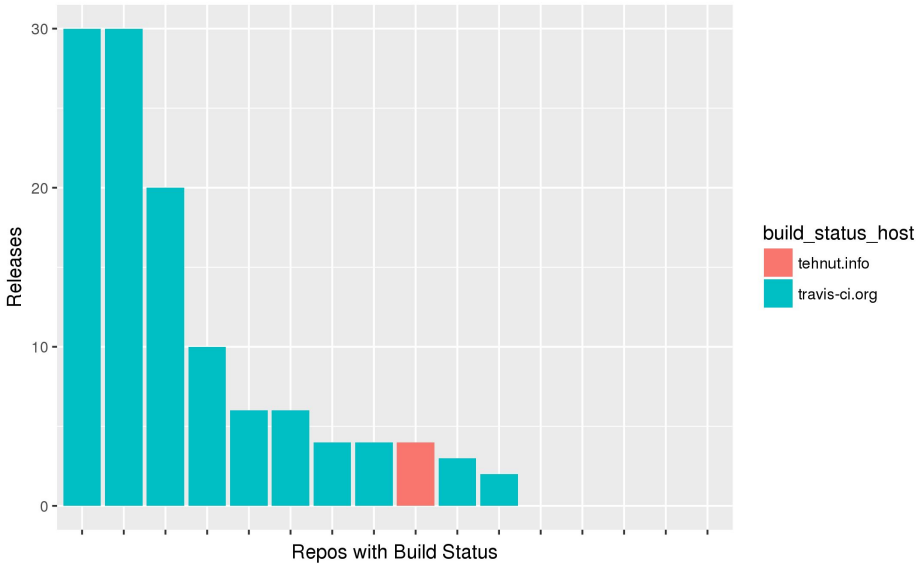
Individual repos with a readme and a build status tag are represented on the y-axis

Watch: “build status” in Readme vs Releases



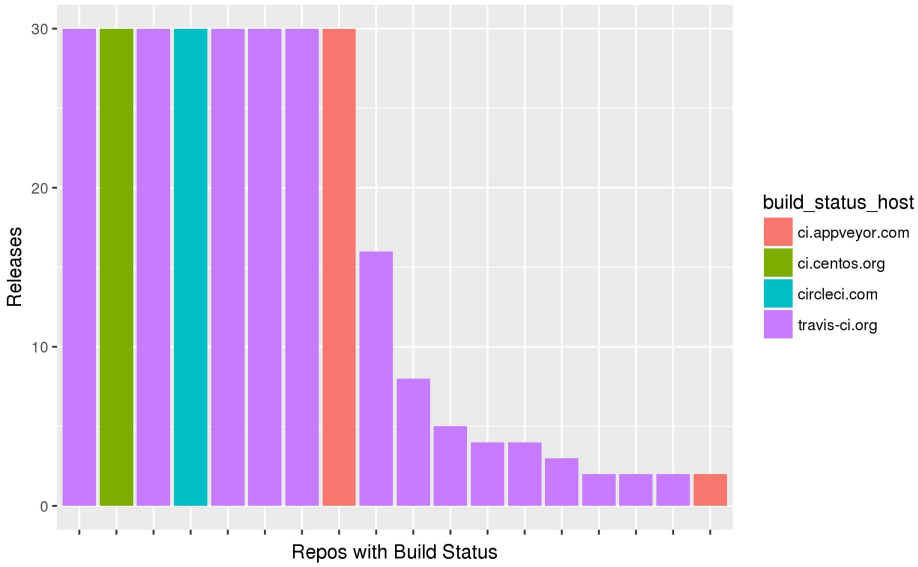
Individual repos with a readme and a build status tag are represented on the y-axis

Fork: “build status” in Readme vs Releases



Individual repos with a readme and a build status tag are represented on the y-axis

Release: “build status” in Readme vs Releases



Individual repos with a readme and a build status tag are represented on the y-axis

Github Repo Samples Summary

Which parameters had the strongest correlation to event type and which ones didn't?

Age, Readme size, Releases, Build status in Readme (somewhat)

Which event types showed the strongest skew?

The Watch sample showed a consistent skew throughout all parameters.

Conclusions

Can we stratify based on event type?

A sample of repos using Watch events will bias towards the type of repos we are interested in studying while still representing the overall Github population in a way that can be accounted for.

TODO: figure out the best license to use for this stuff so it's free/open but we get credit for our work