**Repo Census**

Everything here is event-based as that’s what the timeline is based on – at the moment this is mostly about which repos the events (and therefore data) relate to.

There are 4,089,837 distinct repo urls in the timeline data.

**Single-Event Repos**

I’m dealing with the single-event repos separately for now. There are 838,443 in total.

For 556,645 of these the single event is a CreationEvent. These would be repos which were created within the timespan covered by timeline and then never saw any other activity. There’s some strange stuff going on with some of the other single-event repos – most notably repos where the url has no ‘name’ component (12,490 of the single-event repos are such), I don’t know what causes this but I’m going to exclude them for now.

There are some single-event repos which were active in the past (before timeline begins) – for instance 36,955 of them had more than 1 watcher. There are probably far more formerly active repos which don’t show up in this data because they had 0 events since timeline began.

**All the other repos**

Suspicious/Spam repos

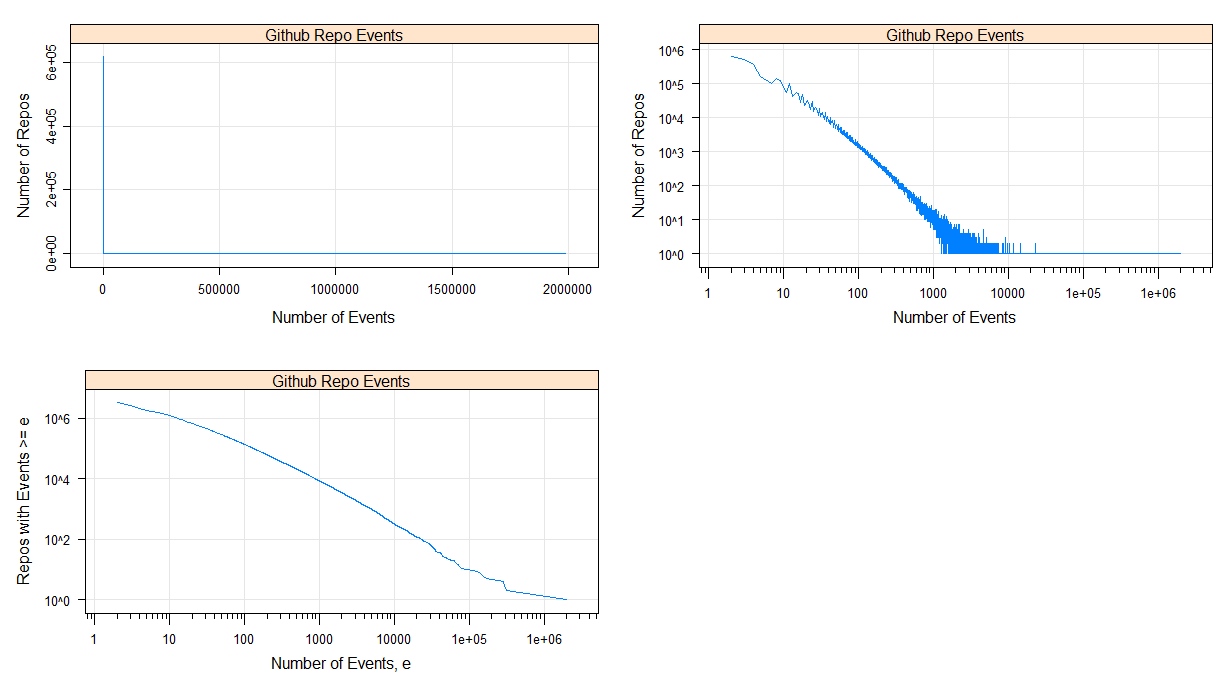
The first thing I noticed when looking at the repos with the most events is that some of these have very unusual profiles. By very unusual I mean that they have a LOT of one or two types of event and virtually none of any other type, they were often active for a short period of time and only had a handful of different user accounts associated with them in the whole timeline.

For instance, the techradical/euro2012 repo which I think has been mentioned before. It has 154232 events total, and 154224 of these are Pushevents – there are only 2 ‘Actors’ who have activity related to this repo and it was active for just over 1 month – it has no more than a handful of Watchers or Forks at any point. For another example see lukaseder/jOOQ-trac-ticket-import-test – where one Actor racked up 40k IssueEvents in just over a week and the repo’s Size never rose above zero. To me these are fairly obviously cases where someone has created a repo and let some sort of bot loose on it, spamming the github timeline with meaningless events (I’ve just noticed one repo is called ‘reach-github-limit’).

For now I think its prudent to devise basic measures which isolate repos like these, flag them and set them aside – otherwise the profiles which follow will be distorted by their presence. Firstly I’m removing the 315,066 repos with a maximal size of 0.

Secondly, I made some variables representing Events per Actor (Actors are any users who interacted with the repo, includes Watchers) and Events per Week. I flagged Repos with more than 5k Events per week AND more than 5k Events per Actor – it isolates around 40 of the worst offenders. I could spend a lot more time looking at these ‘event spam’ repos and could probably identify many more which should be excluded from analysis but I’m leaving it at this for now.

Distribution of Events between Repos



Above are the plots I use to investigate whether a distribution follows the Power Law. Top left is a standard line graph, top right has logarithmic axes, bottom left is the inverse cumulative on logarithmic axes (if the distribution is Power Law the last two graphs should have lines which are straight diagonals – so we’re not far off a power law here by the look of it). These plots include all the repos with at least 2 events.

99006943 Events total

79205554 = 80%

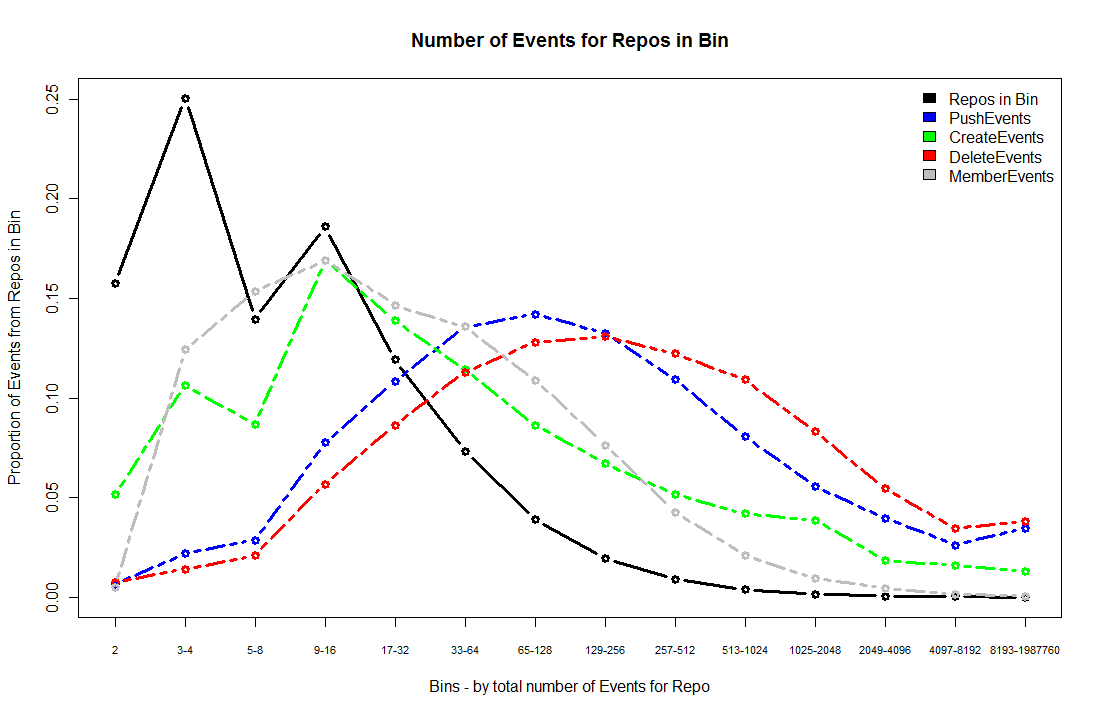
Repos witj 1k Events account for 30% of all data-

Repos with 500 Events (0.5%) account for 37.5% of all Events

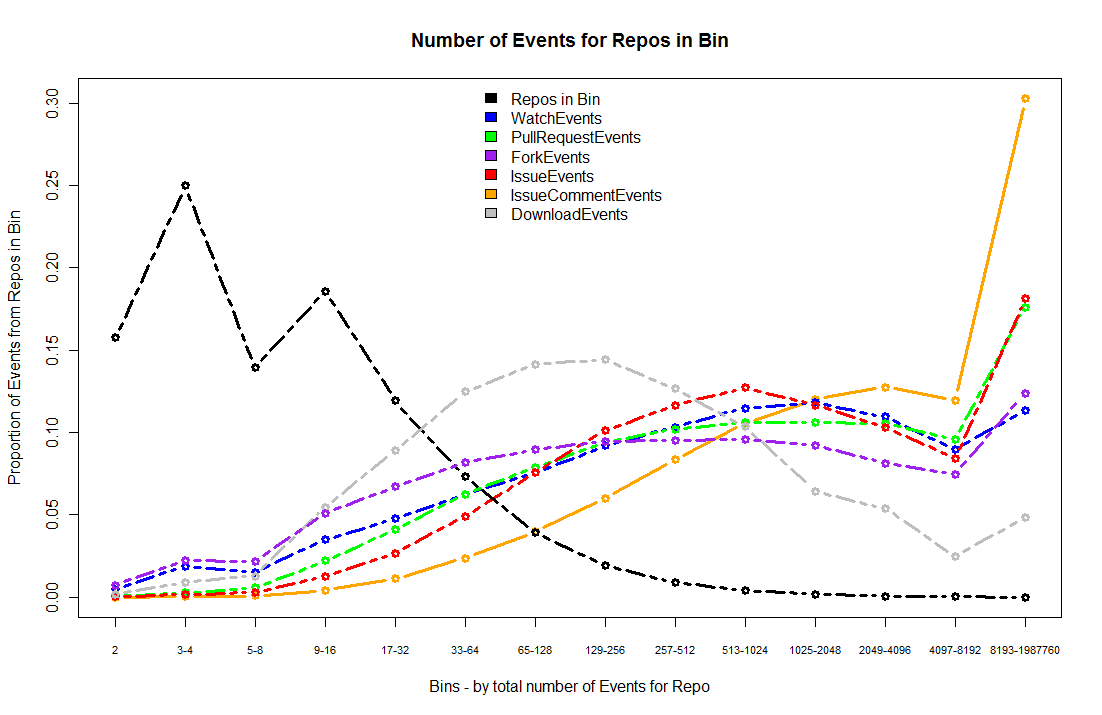
Where do Events originate?

I split the Repos into 14 ‘logarithmic’ bins based on their number of events (and expanded the last bin to include the largest repos). For repos in each bin I calculated the total number of each type of event (for graphing purposes I have expressed these as a percentage of the total number in the data-set).

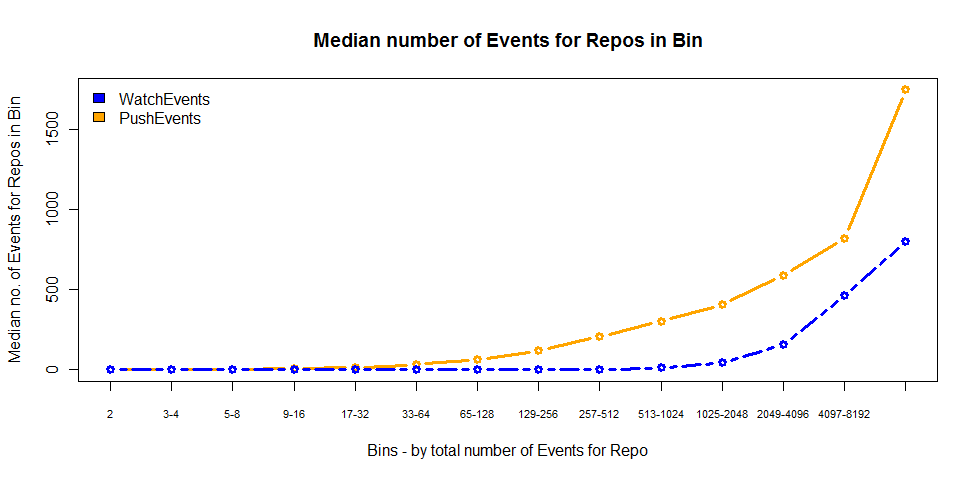
I’ve split the events into two graphs – the first shows ‘owner events’ (events which are instigated by a contributor to the repo), the second shows ‘social events’ (events which can be instigated by a user external to the repo).



So, the low-activity repos account for a lot of CreateEvents (which can be the repo itself, a branch or a tag) and MemberEvents. Most of the PushEvents (and DeleteEvents) relate to moderately active repos – all makes sense to me.

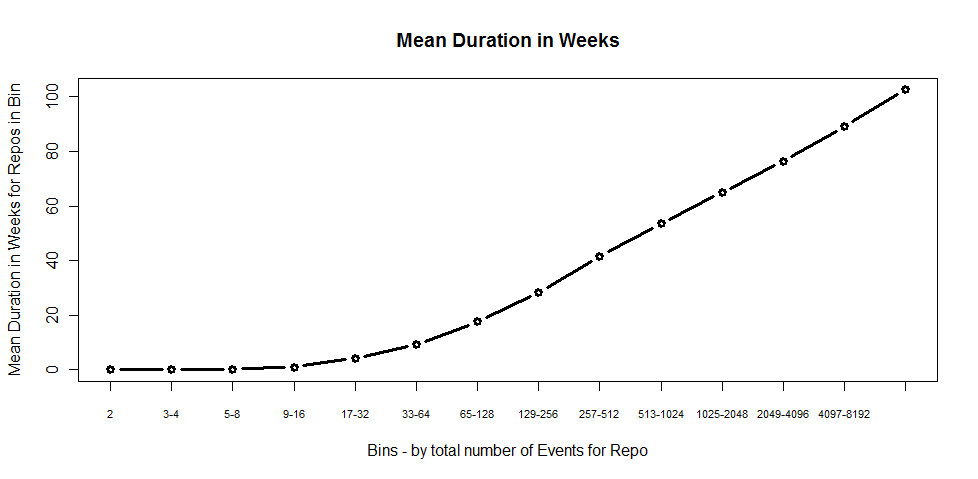


For ‘Social Events’ on the other hand, many of these relate to the highest-activity repos. The 420 repos in the largest bin account for a considerable proportion of all Issue, Fork and Pull Request events. To me this suggests that when we order or group repos by the total number of events (of ALL types) – the repos with the most events are the most well-known or popular repos, because large numbers of ‘social events’ have served to increase the total number of events for these repos.

To demonstrate this I’ve graphed some bin medians (median number of push and watch events for repos in each of the bins). It is common for repos in almost all bins to have at least some push events, and the median number of push events seems to increase at a stable rate with the total number of events. On the other hand most of the bins have a median number of Watch Events which is 0, and it’s only in the higher-activity bins that having a considerable number of WatchEvents becomes common.

Duration

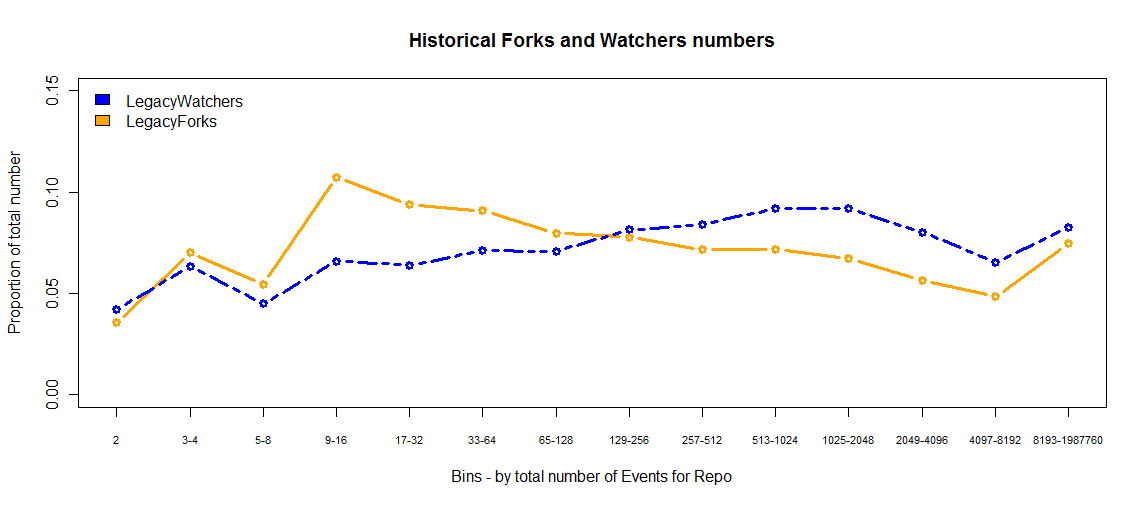
Here’s a graph showing the time in weeks between a repo’s creation date and last recorded push



Historical activity

In addition to the number of timeline events (which everything so far has been based on) there are a couple of variables which give an insight into the ‘pre-timeline era’. On every row of data there are forks and watchers values – giving the number of forks and watchers the repo had at the moment when that event occurred. As I was compiling the event-based data-set I also recorded the maximal (and minimal) number of forks and watchers for each of the repos. These numbers are in some senses better than counting the number of Watch and Fork Events, as they track the number of existing forks and watchers at a given time (whereas when someone watches then un-watches a repo this just adds 2 to the total number of WatchEvents).

I summed the maximal number of forks and watchers for repos in each of the bins and then expressed this as a proportion of the total sum. So, if we consider the question of which repos have watchers and forks throughout github history – they are much more evenly distributed between the event-activity bins (but bear in mind that the top bins have relatively few repos). I guess the message here is about turnover in github repos… the repos which were highly active/popular a few years ago are probably not the same repos which are highly active this year.



Finally, to test the data’s integrity as much as anything else, I looked at the differences between the minimum/maximum observed number of Forks/Watchers for a repo – can be thought of as growth in these values throughout the timeline period. This growth is a bit more evenly distributed between the bins than the number of Fork and WatchEvents would have suggested, the discrepancy is not alarming but it does give another indication that we need to be careful in how we use some of the timeline data.

