1 – Select base repos from pull requests table – those which obviously have some repeat head repos. Going for DIstinctPullRequests/DistinctHeadRepos >= 5 – 13269 repos

2 – BQ query for each base repo, get a list of heads with some variables (e.g. number of pull requests, number merged/closed, first and last dates)

3- Use this to pick out interesting pairs, then maybe look at the full history of interaction between them – getting the full history, need to be CAREFUL how to do this on bigquery….

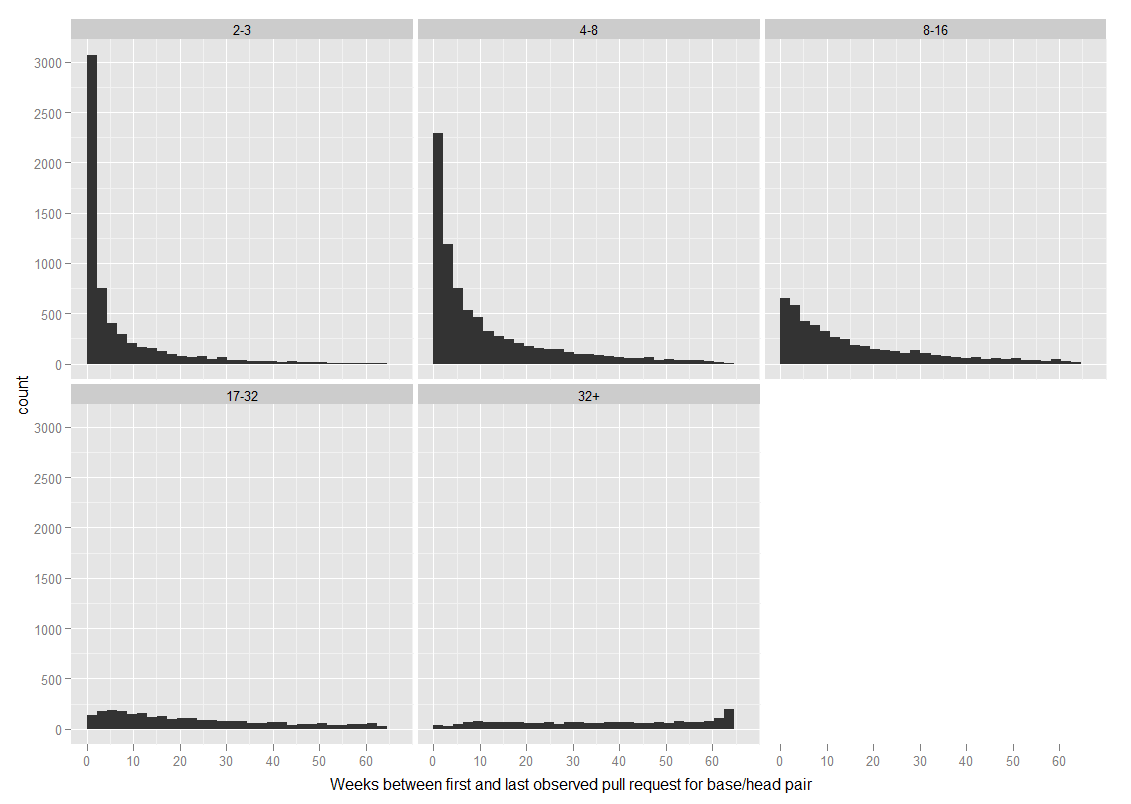
When those 13k repos were run through bigquery this generated a data-set having 38390 rows each concerning a head repo. There are 12,751 different base repos represented here, so some of the queries must have failed but I think I have all recorded heads for the 12,751 base repos that are in here.

5786 of the head repos here are the same as the base repo – i.e. they represent (ongoing) intra-repo pull request activity. Setting these aside.

9610 (30%) of the remaining head repos were single-use, setting these aside.

So the sdata set I’m working with has 22994 rows, for most of these the head repo is a fork (presumably of the base repo) – only 796 are not forks.

The duration of relationships follows the kind of pattern one would expect: where the relationship consists of a smaller number of pull requests it tends to be short in duration, when it consists of a large number of pull requests it tends to be long (for some of the repos in this data-set the relationship pre-dates timeline data and persists for all of the timeline).



bin by distinct pull requests, panel graphs showing: histogram duration, histogram gap between created and first PR

**Looking for users who pull-requested their way into a project**

df$pr1.bp1 is positive – first pr is before first base push  
df$pr1.add is positive – first pr is before added to base

df$hp1.bp1 is positive – first head push is before first base push

15,029 users meet these criteria

Should I map out all the possible sequences of: Forked base repo, First push to head repo, First Pull Request, Added to base repo, First push to base repo

Prototypical sequence is: first push to head, first pull request, (added to base), first push to base

length(df$hp1.bp1[df$hp1.pr1 > 0 & df$pr1.bp1 > 0])

14,348 users who match this sequence

length(df$hp1.bp1[df$hp1.pr1 > 0 & df$pr1.add > 0])

If we extend this to include people who have an add memberevent after their first pull request the number drops to 3,416.

Look at whether there are any SameUser merges, are there heads which are 100%?

**Summary of work on base/head relationships by user**

One of the primary goals here was to look for users who pull-requested themselves unto a project – users who first interact with a base repo through pull requests and then go on to become members of the base repo and make pushes.

Started with 200k pairs of base/head repos that had at least 4 pull request events between them. (bq\_base\_head\_pairs\_nointra) – some base repos appear more than once because they had “relationships” with a number of heads.

Got a list of all users who pushed to each base and head repo, and a list of all pull requests by user and base/head repo. Augmented this second table with variables about pushes to base and head repos.

The 'pull request users' table has 196,437 rows – representing 99,274 distinct users (some were involved in pull requests for multiple base/head pairs).

There are 25,678 cases where the user has at least one push on the base and head repo – but its not clear that all of these users became base contributors AFTER they had made a pull request.

By the strictest criteria I can think of (sequence where first head push precedes first pull request precedes being added as a contributor to base) there are 3,416 users who appear to have received contributor rights as a result of making pull requests.

However, the limiting factor here is being added as a member of the base repo... these events are sparse in the data-set (only 5,538 users have a recorded add memberevent, whereas 25,678 users have at least one push to head and base repo – and therefore must have had contributor rights to the base repo). I'm either missing some events where users were added as contributors, or these events occurred before the timeline data began.

So, if we relax the criteria a bit (a sequence where first head push precedes first pull request precedes first base push) there are 14,348 users who meet it. These users made a median of 18 pushes to the head repo and 8 to the base repo, but the variance on this is huge (its roughly power law).

Once these users had the ability to make pushes on the base repo they no longer needed to make pull requests, technically at least. 5,244 stopped making pull requests once they had made their first push to the base repo, for the remainder there is an overlap.

The major takeaway message from all this is that the situation is messier than expected regarding pull requests. It seems fairly common for people who have contributor rights on a repo (and have exercised these to make pushes directly) to also make pull requests (when they could presumably have achieved the same thing by pushing their changes directly).

Some possibilities: 1) they use pushes for a certain type of change and pull requests for other types 2) there are other contributors to the head repo, and so the pull requests represent a kind of management (external workers change the head repo, the head repo's owner is also a base repo contributor and manages the merging of these changes through pull requests).

**Notes on base/head relationships between repos themselves**

Based on the table of 200k base/head pairs having at least 4 PR events.

There are around 26k base repos which have a 'relationship' with more than one head repo – some have relationships with many heads (e.g. mxcl/homebrew has 1345 different heads, rails/rails has 691).

There are also around 4k head repos which have a 'relationship' with more than one base repo, a fair number of these are related to 'liferay-portal', not sure how many exactly.

There are at least a few 'networks' of repos, where a group of smaller repos share pull requests and are also related to a large repo through pull requests.

It also seems that pull requests can 'flow' in more than one direction (reciprocal pull requests between base/head repos where each acts as both base and head) – have yet to put a number on this.

Liferay-portal repos seem to be exactly this kind of network. There are 877 'pairs' of repos involving 195 distinct repos, with 98 different repos serving as a base and 190 unique head repos.

I've done some inspection of the 'liferay-portal' repos and they're actually fairly interesting. There are 28 repos which served as the base in at least 10 pull request pairings (receiving a total of 49k pull requests between them!). The repo which was presumably the 'original' one (liferay/liferay-portal) isn't actually the 'hub' for pull requests – brianchandotcom/liferay-portal was the base in 68 pairs and received 24k pull requests, it was also the head in 2 pairs and made 10 pull requests in this fashion... I was expecting that one of these pairs would involve liferay/liferay-portal (and that brianchandotcom would turn out to be a working repository with liferay being reserved for stable release versions or somesuch) but there's actually no direct link between liferay and brianchandotcom through pull requests. The liferay/liferay-portal repository itself was the base in 46 pairings, receiving a total of 456 pull requests.

I'm kind of suspicious of the very high levels of pull requesting going on between all these liferay repos... but the project itself seems genuine, and if all the activity is also genuine then this would make for a very interesting (and somewhat unique) example. At this stage I could fairly easily find out anything we'd want to know about the liferay-portal 'network', between the pull request data-sets and the census stuff I've already got the data to answer any number of questions about it. I also feel like I've got a good enough command of bigquery and the github API to get and use just about any kind of data that's available to us for this kind of analysis, so that's some kind of progress I guess!