Capital numbers and the obscure numericality of configuration

I report here on a deliberately naive attempt to re-count a single number: approximately 29 million code repositories on Github at a particular point in time (late 2015). The number appeared in a research project focused largely on transformations in the social life of code. But the project sought to experiment with data-intensive methods for analysis of re-assemblings and re-associations typical of places such as Github.[^12] On Github, two numbers have capital importance – the number of people using the platform, and the number of different code repositories stored there – but I deal in this paper only with the latter. The counting I undertake will focus on problems of copying, duplicating, and imitation that render large numbers a moving substrate.

What kinds of work can we do with numbers today, amidst perhaps their deep transformation and re-materialisation in technologies? How do we do that amidst the computational count, and many efforts to count everything (species, stars, likes and bitcoins)? While I'm very drawn to attempts to count things, I find it most interesting to try and count whilst paying attention to the joining-together or composition inherent to counting and the very fabric of number. Numbers and counts are site of entanglement between device-specific materialities and imaginaries of order, totality, rank, inclusion, boundary and power. This double sense of numbers as both a calculatively ordered cutting-framing-summing and as materially specific figuring of differences could well be understood in terms of Lucy Suchman's 'configuration' device:

> the device of *configuration* has two broad uses. First, as an aid to delineating the composition and bounds of an object of analysis, in part through the acknowledgment that doing so so is integral not only to the study of technologies, but to their very existence as objects. And second, in drawing our analytic attention to the ways in which technologies materialize cultural imaginaries, just as imaginaries narrate the significance of technological artefacts. Configuration in this sense is a device for studying technologies with particular attention to the imaginaries and materialities that they *join together*, an orientation that resonates as well with the term's common usages to refer to the conjoining of diverse elements in practices of systems design and engineering [@Suchman_2012, 48].

While Suchman's examples of configuration come from software systems, I'm suggesting that a single number could be re-counted as a configuration. The argument is that numbers can be configurations. *Configurative numbers* can be part of the boundary-making of technologies, and a resource for recovering the heterogeneity of technologies.

Re-counting a capital number on a platform

Github, started in in 2007, epitomises and has indeed been central – as the suffix 'hub' suggests – to a mass of configurational events associated with code. Like many social media platforms, it has grown tremendously in the last 7 years to include around 31 million software projects (February 2016). This growth flows from a tremendous variety of processes that are difficult to summarise or classify partly because the actors, topics or domains of coding are so diverse and partly because much of what flows through Github is both technically and socially 'innovative' in the sense that Bruno Latour uses the term: '" innovative" means that we do not know the number of actors involved in advance' [@Latour_1996, 72]. But given we are dealing mainly with code or code-like documents, Github practice heavily concerns configuration.

Like many contemporary assemblages, Github seeks to describe itself through large numbers (what I call 'capital numbers'), and these numbers in their sometimes daily changes and updates attest to an investment in being innovative or open-ended enumerations. In June 2014, the Github homepage showed a photo of a women working in an urban office location, with lights and professional camera focused on her work. It described '6.1 million people collaborating right now across 13.2 million repositories … building amazing things together' [@Github_2014a]. In late November 2015, Github has a slightly more functional description, and the image seems to be a press conference in China:

> GitHub is how people build software. With a community of more than 12 million people, developers can discover, use, and contribute to over 29 million projects using a powerful collaborative development workflow. [@Github_2015](https://web.archive.org/web/20151216055610/https://github.com/about)

The numbers change over time alongside the composition of the actors involved (women coding and working at Github, a sore point for the company; Chinese developers using Github, but also political activists, leading to a denial of service attack on the Github, allegedly by the Chinese government [ˆ3]). But the changing numbers, alongside the sometimes rather sublime scales of community (a 'community' of 12 million people?) certainly form part of Github's description.

The litany of sense-making: enumerating the aggregate through events

In 2013, my hope was that I could simply verify these numbers in some way. I planned to use digital methods to count people and count things, in this case, software projects on Github. In re-counting them, I wanted to en-counter something of their composition in all its heterogeneity and conformity. This would be a both political and culturally significant since those numbers are typical of social media platforms and their capitalisation.

How could I actually count software project or people? Like everyone else, I turned to data published through the Github API (Application Programmer Interface). APIs were not created as a data resource for social scientists, but for software developers working in convergence cultures [@Jenkins_2004] in which connecting different platforms, devices, sites and practices together using code has become standard practice, APIs have great significance [@Bucher_2013].

In Github's case, data published through the API derives from acts of writing and reading code. The data is logged as developers write code and move that code in and out of code repositories using the `git` system [@Git_2014]. Coding today constructs and take place in increasingly complex associative infrastructures. Github seeks to figure coding as a social network practice based on a hub. The format of the data that affords and records its husbandry of the hub collocates people and things. By convention much API data from social media sites has an 'event' structure that links named actors and named entities to specific infrastructural locations (usually coded as an URL) at a particular time (the 'timestamp').

```
{
"id": "2111998059",
"type": "WatchEvent",
"actor": {
"id": 1459103,
"login": "mmemetea",
"gravatar_id": "4532d1e4885f579ca7d9aa8748418817",
"url": "https://api.github.com/users/mmemetea",
"avatar_url": "https://avatars.githubusercontent.com/u/1459103?"
},
"repo": {
"id": 14802742,
"name": "OpenSensorsIO/azondi",
"url": "https://api.github.com/repos/OpenSensorsIO/azondi"
},
"payload": {
"action": "started"
},
"public": true,
}
```

What would it mean to count things amidst this data? The data can be understood as configuration on the move. It conjoins elements together. The relatively simple `WatchEvent` on Github shown in the data extract above documents the act of an `actor` calling themselves `mmemetea` associated with a repository called `azondi`, a software project coordinated by the 'organisation' `OpenSensorsIO`.[^2] Note that the event also has various status designations – it is a `public` event, it has a 'payload' (often much more complicated than simply `started`) – and includes various indexical references or `ids` that link the event to other groups of people, organisations, repositories and images (`gravatar_id`). The intricate syntax of this data – many brackets, inverted commas, colons, commas – attests to a complex social configuration which links actors, actions, places and times in discrete events ordered in time. The discreteness of such events attests to the potential for the precise configuration of elements around a given repository to change or rearranged in some way. New actors might be added; relations might appear between entities; the location of entities might shift, and forms of association ('organizations') might subsume or grow out or around all of this.

Imbuing numbers with importance

Obviously, I'm not the only one interested in counting things amidst the massive stream of event data concerning Github. A datastream purporting to contain the whole public event timeline of Github suddenly appeared in mid-2012. Ilya Grigorik, a 'Web Performance Engineer' at Google, launched a Github repository `igrigorik/githubarchive` linked to a website `GithubArchive.org` dedicated to amalgamating all the Github API public event data – the so-called 'timeline' – in one place [@Grigorik_2012]. Grigorik, or `igrigorik`not only published all the data in a cloud-based data store but published that data in Google's newly launched cloud computing service, GoogleBigQuery.[^1] The Github timeline data was listed, along with all the words in Shakespeare and all the US birth name records, as one of three dataset exemplars that people could use to learn about GoogleBigQuery [@Google_2016]. Like the data on GithubArchive itself, the GoogleBigQuery copy of the Github public timeline data was updated hourly. Reaching back to 2011, it logs around 400 million public events.

The archiving, copying and transformation of Github event data drew the attention many people working to count events in different ways. These forms include *labour-numbers.* For instance, the 'OpenSource Report Card' (http://osrc.dfm.io/) or `dfm/osrc` by Dan Foreman-Mackay [@Foreman-Mackay_2014], is a prize-winning use of the timeline data (see Figure **??**). It ingests all the data from the Githubarchive, counting what developers do, when they do it, and using what programming languages. With this data stored, it then builds a predictive model that allows it to both characterise a given Github user, and to predict who that Github user might have affinities with. Here the mass of events in the Github timeline are brought to bear on finding similarities between people.

In response to the Github Data Challenge in 2012, people looked for feelings or 'sentiments' about coding in the timeline data. Feelings associated with coding were mined by counting emotional words present in comments accompanying the Github events (http://geeksta.net/geeklog/exploring-expressions-emotions-github-commit-messages/). The presence of words in these message can be cross-linked with programming languages in order to profile how different programming languages elicit different emotional reactions.

Configurative numbers are also *nationalised* or regionalised. People began to map coders and repositories by geographic location. The mapping of Github contributions by location performed by David Fischer (http://davidfischer.github.io/gdc2/#languages/All) is typical in that it too counts events, but this time puts the emphasis on the geography of the 'top' repositories, coders and their programming languages. As the David Fischer puts it 'this data set contains contributions to the top 200 GitHub repositories during the first four months of 2013 and plots the location based on what the contributor provided' [@Fischer_2013a].

Logistic numbers can be derived from the data streams. People make live dashboards for Github. `Octoboard` (http://octoboard.com/) animates changes on Github using the timeline data [@Roussell_2015] (see Figure **??**). `Octoboard` ornaments the capital numbers with a range of peripheral live enumerations that point to the liveliness of events on Github. It presents a summary of daily activity in major categories on Github – how many new repositories, how many issues, how repositories have been 'open sourced' today. It offers realtime analytics on emotions. Like many other dashboards associated with social media analytics, `octoboard` suggests that the constant change in associations and projects in software development can no longer be known through leisurely rhythms of analysis, but is increasingly framed as a problem realtime awareness.

Looking slightly more widely, *pedagogical numbers* appear from the data. the

Github timeline data has quickly become a favourite training tool for data mining textbooks books that configure and convey the calculative agencies characteristic of capital numbers. In *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More,* Matthew Russell makes use of the Github timeline to demonstrate ways of uses social network analysis to highlight the important nodes and links between repositories and users [@Russell_2013]. Finally, for academic researchers in computer science and certain parts of organisation studies, Github has been a boon because they study technologically and economically important practices software development in the wild much more easily. Academic researchers in fields such as software engineering do social network analysis in order to gauge productivity, reuse, efficiency and other engineering and management concern [@Thung_2013]. Like the many Github-hosted projects discussed above, they analyse sentiment [@Guzman_2014], collaboration and productivity [@Dabbish_2012], and geography [@Takhteyev_2010].

All of these re-countings enliven, animate, reactivate, localise and qualify the capital numbers by carving off fragments from the 'natural multiplicity' [@Badiou_2008, 211] of the aggregate. They potentialise the numbers in terms of work, geography, liveness and further accumulation by summing them up in different ways (realtime, networks of connections, geographies of work and affect).

Acts of imagined configuration

How could we re-count capital numbers in order to both further highlight their dynamic composition through flows of association and to highlight some of the highly reactive, imaginary boundary work they might do? Could configurative numbers in both their compositional and imaginary-materialising multiplicity be counted? Whilst there is a much configurational work involved even in beginning to count things heterogeneously on an infrastructural scale, I found myself drawn in two directions, one concerned with ephemerality of configurations and the other with their imitative composition.

The vast majority of repositories on Github are highly ephemeral. They attract one or two events. A small number of repositories, counted on the lower right-hand side of the plot capture many thousand events. A great proportion of events in the timeline data were absorbed into ephemeral repositories (if that is not too great a contradiction in terms). Millions of repositories flash into visibility on the timeline for a brief period before falling back into obscurity. They remain in the capital number. The diagram shown in the Figure **??** encapsulates this imitative flux. On the left hand side, millions of repositories receive less than five events during the 18 months. On the right hand side, less than 50 repositories receive more than a thousand events. A similar pattern appears in the other capital number: while some 'people' emit many thousands of events, others only trigger a few.[ˆ5] Already, then, the capital number of repositories on Github takes on a different composition when viewed from the perspective of duration.

But rather than see this ephemerality as waste, noise or something to be discarded, we might trace the working of associative processes through it. I can only give a brief indication of this here. For instance, counting just the unique names of repositories in the timeline data, counting how often and when they appear in events, begins to show how associations are configured on Github. Almost one half of the repositories in Github inject only one event into the timeline data stream. What is this event? The act of copying ('forking') another repository. People 'fork' other repositories frequently. Table 1 shows 10 million ForkEvents, suggesting that around one third of the total 29 million repositories are direct copies. Acts of copying occur on many scales and at various levels of infrastructural and associative complexity. This copying is vital to the 'sharing' practice of Github coding. Of the several hundred million events in the timeline dataset, approximately 95 million result from copying, watching, commenting on or directly participating in other repositories.

It is hard to grasp the texture of imitative processes in event data. We can view localised aspects of propagation of imitation. Both Figure **??** and **??** count imitations in the form of copying code, but in ways that go beyond the formal copying mechanisms offered by Github itself (for instance in the 'Fork' button that appears on the top right hand side of any repository on Github; for example https://github.com/twbs/bootstrap). Imitations appears in two different ways in these figures. The broad bands of color rippling horizontally across the middle of the figures graph the counts of copies being made each day on Github of popular repositories using the Fork action. (In forking, the repository name remains the same.) But the much more dense striations on either side of the central bands, seen for instance in Figure **??**, count repositories whose names incorporate the important repository, but vary it in some way. These repositories have some association with the `android` repository, but diverge from it in a multiplicity of ways. A repository may, for instance, relate to the popular Twitter `bootstrap` repository yet combine it with a range of other platforms and devices such as `android` or `jQuery`.

In striated zones of associative imitation, it is possible to count something different. Imitations augment the average everydayness of coding work with forms of associative investment and affective identification. In this work, important repositories act as high visibility markers around which forms of identity take hold and multiply. Unlike the capital numbers with their total aggregates of people or repositories, the imitative fluxes that bulk out and pad in these diagrams have diverse network-making configurations. The implication here is that by counting elements within the capital numbers, we can begin to glimpse the pathways of generative circulation that enliven and literally join together in contemporary technologies.

De-capitalising configuration

The very act of counting these imitations has a device-specificity, in its reliance on the Github platform API, the GithubArchive timeline dataset and the GoogleBigQuery cloud analytic's platform, and indeed Github itself as the repository of the code and text comprising this article [@Metacommunities_2016]. Unlike the network-tracing approach, it can perhaps take into account the ways in which the platform itself works to shape networked associations. Whenever we count or calculate in any form, device-specific configurations materialise in the numbers. The expansively contagious nature of imitation inevitably encounters device-specific configurations.

One sense of how numbers embody device-specific configurations can be seen by examining high event-count repositories on Github. The top 1000 repositories attract around `r round(total_events/290000000 * 100)`% of all events in the timeline period. The query de-capitalises the repository names for the purposes of aggregation (e.g. a repository called `DotFile` will be counted along with `dotfile`). Given the flattening of the names to lower case produced by this query, we cannot readily see how those events are distributed. But these highly frequented repositories, which only comprise a tiny percentage of the 29 million on Github, absorb many events. What happens in these high event count repositories?

Several major traits appear in these high event-count repositories. Many events come from off-platform. Github, despites its own figuration as a hub or as the capital, functions as an element in others configurations, located elsewhere. For instance, the repository `eclipse.platform.common` accounts for almost 2 million events in the timeline data. A single repository attracting two million events (or almost 1% of the total event count in the timeline data) suggests something highly significant in the geography of coding work. Perhaps the fact that `eclipse` is itself part of the Eclipse Foundation, 'an amazing open source community of Tools, Projects and Collaborative Working Groups' [@EclipseFoundation_2016] with almost a thousand of its own projects, might help explain the large number of events. More significantly, the high-frequency event trade in `eclipse.platform.common` is an example of how Github itself functions as part of a configuration. The `eclipse` repositories are not actively developed on Github. They are mirrored or themselves copied from the hundreds of `git` repositories found at (git://git.eclipse.org)[git://git.eclipse.org]. Like many other significant repositories (`linux`, `android`, `mozilla`, `apache`), high event counts figure Github as a hub.

In addition, even where events do originate on Github, many of them concern Github or technical configurations closely associated with Github. For instance, the list of names of high-event count repositories includes many containing a set of terms centred on a few highly repeated items. On the one hand, the sharing of names – for instance, the many repositories that contain the term `dot` as in `dotfile` or `vimdot` – suggests the imitation or repetition that Tarde describes. On the other hand, terms like `dot`, `test`, `hello`, `config`, `doc`, `build`, `setting` or `demo` also stream through this set of repository names. Many repository names proliferate from these stems. All of these stems relate to device-specific configurations and work done to sustain the configuration of devices in particular places.

Conclusion

Capital numbers, I have suggested, enumerate an aggregate, imbue it with importance linked to infrastructural scale and complexity, and promise to generate further accumulation. As we have seen, the capital numbers attract many further aggregations in the form of data archives (GithubArchive) and oligoptic visualization. Configurative numbers, in resonance with the configuration work that gives rise to them, account for the both the accumulation and the heterogeneity of events, people and things, and for their re-counting.

It takes work to get contemporary digital data and associated large numbers to do something other than add to the count of capital numbers and their platform-centred aggregates. Configurative numbers, I have suggested, as one term for the versions of enumeration that result from such re-countings. We saw how that re-counting points to imitative fluxes, to that which takes places elsewhere but is refracted in the capital numbers, and to device-specific configurations that saturate the data stream. None of this discounts the capital numbers – 29 million, 12 million – as if they were false, reactive or performative numbers [@Espeland_2008].

How do we re-animate the capital numbers that abound in contemporary technologies? To do this, we need to find ways of working on formatted data that goes against its pre-formatted accounting. Device-specific research that attends to the ways in which data is formatted, and how that formatting itself materialises imaginaries of inclusion, relation and wholes that pertain more broadly. At the same time, paying attention the configuration or device-specific composition of numbers, we can construct a view on what remains irreducible to the formatted imaginaries of the data-stream.