7

Operative ethnographies and large numbers

Adrian Mackenzie

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 primarily focused largely on transformations in the social life of coding, programming and software development amidst apps, clouds, virtualisation and the troubled life of code commons. In exploring 'how people build software' (Github 2015), the project explicitly sought to experiment with data-intensive methods, infrastructures and tools – cloud computing services such as Google Compute, data analytic and visualisation tools such as R, Python and IPython notebooks, large data stores such as BigQuery, streaming data from social media platform APIs (Application Programmer Interfaces), predictive models, especially in the form of machine-learning classifiers (support vector machines, Naive Bayes classifiers, random forests) – in making sense of what happens on Github en masse. Somewhat recursively (although I don't want to make too much of this recursion, since I do not think it amounts to a recursive public (Kelty 2008)), Github happens to be the *platform* that hosts the code development of many big data tools and infrastructures (hadoop, tensorflow, ipython, flow, d3.js, spark etc.). It also directly hosts all of the text, code, figures, intermediate results and configuration information for the research project I describe, including this chapter in various operational and executable forms ((Metacommunities 2016a).

The approximate number 29 million will in this chapter serve as a synecdoche for big data in several ways. The number, a count of all the repositories on Github in late 2015, is a small part standing in for the whole. On the one hand, Github as a platform used by millions of people is a typical setting open to, affected by and shaped by big data practices. On the other hand, what happens on Github – coding, software development, configuration, modification and transformation of information infrastructures – shapes big data practices. Crucially, since open or publicly visible code repositories are shadowed by largely invisible private repositories that yield revenue for Github as a San Francisco-based business, it is inherently difficult to count all the repositories. 'All' here inevitably means 'some', but maybe that is always a problem for big data.

A focus limited to one big number (29 million), particularly one that presents itself as the total of all repositories, is inevitably narrow. The story of encounters with this single number might, however, evoke and exemplify the appeal, frustrations, hopes and material configurations associated with contemporary data practices that envision working with 'all the data' (Mayer-Schönberger and Cukier 2013). This number, one of a series of similar numbers, exists and exerts effects in different ways on counting and accounting. I'm interested in its composition, its instability, the 'stories,' imaginings and materialisations it authorises, and its practically problematic relation to big data infrastructures.

What kinds of engagement can we have with such numbers today amidst perhaps their deep transformation and re-materialisation in technologies?[1] Given so much computational counting, so many efforts to count all manner of things (species, clicks, stars, likes, cells and bitcoin transactions), and indeed strong injunctions to tell stories about such numbers, what can social researchers do? One key difficulty in developing an ethnographic sensibility around such numbers resides in the many detours, variations and differences they embody, not least for the social researcher who finds himself or herself trying to re-count them or account for them. These variations, shadings and shifting in numbers might be seen as the everyday frictions of ethnographic writing in its reflexive analytic mode, but they also spawn from the configured, infrastructural and operational complexity of ethnography as an experimental situated practice. As George Marcus suggests, 'experiment today is thus less about writing strategies and more about creating forms that concentrate and make accessible the intermediate, sometimes staged, sometimes serendipitous occasions of distinctively anthropological thinking and concept work' (Marcus 2014: 400). The chapter documents and practically re-enacts a series of operational queries – both in the technical sense of a statement executed by a database and in the sense of a question posed to a given state of affairs – concerned with large numbers and technical ensembles.

While I'm very drawn to attempts to count things and to weave counts into ethnographic writing, during this project I often wondered about how to count things whilst paying attention to their composition. Marcus enjoins ethnographers to 'make forms of processes' (Marcus 2014: 401). The problem is that, seen in a rather literal-minded way, counting is a process that very easily renders the form of a number. An ethnographic sensibility around numbers at a very minimum entails a double counting or re-counting. In the setting I describe (software development, distributed information and knowledge infrastructures), numbers and counts are sites of entanglement between device-specific materialities and collective imaginaries of order, totality, rank, inclusion, boundary and power. This double sense of numbers as both a calculatively ordered cutting-framing-summing-up form and as materially specific figuring of differences could well be understood in terms of what Lucy Suchman describes as a 'configuration':

> 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 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 – 29 million repositories on Github – could be re-counted as a configuration. The argument is that even the listing, enumerating and counting of things in a single number can be seen as a configuration. Practically, a configured number, if such a curly term is permitted, will need to be counted in ways that hold together the two aspects Suchman highlights. A configured number might be integral to the existence of the 'technology', in this case the 'social coding' platform Github. At the same time, that number would, in its counting, call our attention to what is being materialised in terms of an imaginary and all that entails.[2]

Re-counting a capital number on a platform

On Github, two numbers have cardinal and indeed capital importance: the number of people using the platform, and the number of different code repositories stored there. I deal in this chapter only with the things – the repositories – not the people. The counting I undertake will focus on problems of copying, duplicating, and imitation of things – repositories –- that render large numbers a moving substrate.

Software developers (coders, hackers, geeks, software engineers, software architectures, programmers, scientists etc.) turn to Github to find, deposit, discuss, collaborate, publish and tinker with code. Across a gamut of practices, rituals, organisational forms and inventions, they rely on an underlying set of protocols, tools and workflows that focus on the problem of versions and versioning of code called *git* (Straube 2016). Github, started in in 2007, epitomises and has indeed been central to – as the suffix 'hub' suggests – a mass of configurational events associated with the development of big data practices in association with large technical ensembles. Like many social media platforms, Github has grown tremendously in the last ten years to around 55 million software projects (March 2017; the current total of the 29 million that I focus on). Its growth flows from a variety of processes that are difficult to summarise or classify partly because the actors, topics or domains of coding are diverse, and partly because much of what flows through Github is both technically and socially 'innovative' in the sense in which Bruno Latour uses the term: '"innovative" means that we do not know the number of actors involved in advance' (Latour 1996: 72). (Again, a problem of counting appears here, albeit only in the limited form of 'not knowing in advance'.)

Given this massive confluence of coding, configuring, copying, collaborating and much playing around with repositories in sometimes quixotic fashion (as we will see, many repositories are highly transient, and very many have a spam-like quality), Github, like many contemporary assemblages, seeks to describe itself through large numbers such as '55 million projects'. We might call them 'capital numbers' since they serve a vital function in attracting investment (Github, with a market value of $2 billion, has received several hundred million dollars in venture capture (Gage 2015)), imbuing the platform with hub-like importance in contemporary techno-economies (see for instance, Quentin Hardy's writing about Github and 'open everything' (Hardy 2012)). They also, as I will suggest, directly lend weight to the injunction to 'tell stories with the [big] data'. Capital numbers in their sometimes daily changes and updates attest to an investment in being innovative and in open-ended processes of growth.

is paragraphy is q quote. Should be indented and
paced accordingly

As Suchman's concept of a configuration suggests, capital numbers combine with cultural imaginaries of work, value, power and technology in manifold ways. In June 2014, the Github 'about' page 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 2014). In late November 2015, the same page had a slightly more functional description, and the image seems to be of a crowded 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 as a co-founder was involved in sexual harassment complaints; Chinese developers using Github, but also political activists, leading to a denial of service attack on the Github, allegedly by the Chinese government). But the mundane yet sublime scale of these numbers (a 'community' of 12 million people? 29 million software repositories?) certainly forms part of an ongoing cultural, organisational and financial capitalisation.

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

In 2013, a year in which the injunction to do things with data was impacting social science research funders as well as many people in media, commerce, industry and government, my aggressively data-analytic ambition with respect to Github was to re-count the capital numbers – people and repositories – in some way. I planned to use data-analytic methods and infrastructures and code in all their stacked sophistication to count software projects on Github. In re-counting them, I imagined that I would capture something of their composition in all its heterogeneity and conformity, and show how their scale was an effect of specific dynamics of imitation. I felt disposed to contest the easy rhetorical slippages between 'code', 'open' and 'good'. The data-analytic practice would be both political and culturally worthwhile since capital numbers are common in the capitalisation of social media platforms. It would be ethnographically recursive since my writing would be mixed with code in an experimental writing practice derives from the sites – repositories – of the research.

Coding today takes place in increasingly complex associative infrastructures. How could I actually count software projects or people? Like many data scientists and digital sociologists, I turned to data published through the Github API (Application Programmer Interface). APIs were created not as a data resource for social scientists but for software developers working in convergence cultures (Jenkins 2004) where it is standard practice to connect different platforms, devices, and sites using code. APIs have great practical significance (Bucher 2013). In Github's case, data published through the API derives from acts of writing and reading code (although not only code – an extraordinary variety of other documents, texts, images, maps and other forms of data can be found on the platform). Actions are logged as developers write code and move that code in and out of repositories using the *git* version control system (Git 2014). Github seeks to figure coding as a social network practice based on a hub.

By common 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 a URL) at a particular time (the 'timestamp'). While not all big data has this timestamp-actor-location-links texture, it is common enough to stand in as typical of the data that big data infrastructures and practices encounter. As Noortje Marres has persuasively argued, these data formats are not ‘neutral’ or in any way particularly good for social research, since they tend to pre-structure the kinds of engagement that one might have with the data (Marres 2012). Work needs to be done with and against the data format. A slightly recursive instance of this API-effect can be seen in event data for the project:

```
library(curl)
library(jsonlite)
con2 =
curl(‘https://api.github.com/repos/metacommunities/
   metacommunities/languages’)
lang = fromJSON(readLines(con2), flatten=TRUE)
lang
## $‘Jupyter Notebook’
## [1] 5316912
##
## $Python
## [1] 362165
##
## $TeX
## [1] 206395
##
## $R
## [1] 176827
##
## $Scala
## [1] 27835
##
## $Shell
## [1] 23630
##
## $JavaScript
## [1] 16699
##
## $‘GCC Machine Description’
## [1] 15451
##
```

```
con3 =
curl('https://api.github.com/repos/metacommunities/
   metacommunities/contributors')
people = fromJSON(readLines(con3), flatten=TRUE)
people
## login id
## 1 rian39 526523
## 2 Milllss 4489313
## 3 rian32 885027
## 4 MatthewFuller 4973736
## 5 Goffinger 4973562
##      followers_url
## 1 https://api.github.com/users/rian39/followers
## 2 https://api.github.com/users/Milllss/followers
## 3 https://api.github.com/users/rian32/followers
## 4 https://api.github.com/users/MatthewFuller/followers
## 5 https://api.github.com/users/Goffinger/followers
##         following_url
##      repos_url
## 1 https://api.github.com/users/rian39/repos
## 2 https://api.github.com/users/Milllss/repos
## 3 https://api.github.com/users/rian32/repos
## 4 https://api.github.com/users/MatthewFuller/repos
## 5 https://api.github.com/users/Goffinger/repos

con4 = curl('https://api.github.com/repos/metacommunities/
   metacommunities/branches')
branches = fromJSON(readLines(con4), flatten=TRUE)
nrow(branches)
## [1] 30
```

In our/my case – I'm not sure how to report on the work I did with others in this project – I wanted to write code to gather, aggregate, clean, explore, visualise and model what has happening in repositories on Github. Code written for the research project itself exemplifies typical variations in action: languages and versions ('branches') abound.[3] The code vignette shown above summarises some of this (I return to the configurative events behind this summary below). APIs encourage code-dependent and automated engagement. They typically afford live or real-time observation of online processes. They also invite completionist ambitions: gathering all the data, all the events streaming in and out of the Github platform become imaginable and somewhat viable via APIs. The code vignette shown below, for instance, gathers the last few hundred events on the Github timeline, and is a typical starting point.

```
library(curl)
req = curl_fetch_memory(‘https://api.github.com/events’)
print(rawToChar(req$content[1:1100]))
## [1] “[\n {\n \”id\”: \”5774962987\”,\n \”type\”:
   \”IssueCommentEvent\”,\n \”actor\”: {\n \”id\”: 719827,\n
   \”login\”: \”fmueller\”,\n \”display_login\”: \”fmueller\”,\n
   \”gravatar_id\”: \”\”,\n \”url\”: \”https://api.github.com/users/
   fmueller\”,\n \”avatar_url\”: \”https://
   avatars.githubusercontent.com/u/719827?\”\n },\n \”repo\”:
   {\n \”id\”: 76853145,\n \”name\”: \”zalando-incubator/zally\”,\n
   \”url\”: \”https://api.github.com/repos/zalando-incubator/zally\”\n
   },\n \”payload\”: {\n \”action\”: \”created\”,\n \”issue\”: {\n
   \”url\”: \”https://api.github.com/repos/zalando-incubator/zally/
   issues/309\”,\n \”repository_url\”: \”https://api.github.com/repos/
   zalando-incubator/zally\”,\n \”labels_url\”: \”https://api.github.
   com/repos/zalando-incubator/zally/issues/309/labels{/name}\”,\n
   \”comments_url\”: \”https://api.github.com/repos/zalando-
   incubator/zally/issues/309/comments\”,\n \”events_url\”:
   \”https://api.github.com/repos/zalando-incubator/zally/issues/309/
   events\”,\n \”html_url\”: \”https://github.com/zalando-incubator/
   zally/pull/309\”,\n \”id\”: 224909677,\n “
```

The two lines of *R* code in the vignette (or *Gist*, as it would be called on Github) fetch the latest public – that is, visible to anyone – events from the API, and print them. (Private events have exactly the same format, but are available only through the API to authorised requests.) Rather than displaying them, big data practice would feed them all into some datastore awaiting further analysis.

What would it mean to count repositories using formatted, API data? The JSON (JavaScript Object Notation format; a format commonly used for social media data) records conjoin elements. The relatively simple PushEvent on Github shown in the data extract above documents how an *actor* calling themselves *6ijr* adds a file into a repository called *tbd-ace*. Note that the event also has various attributes

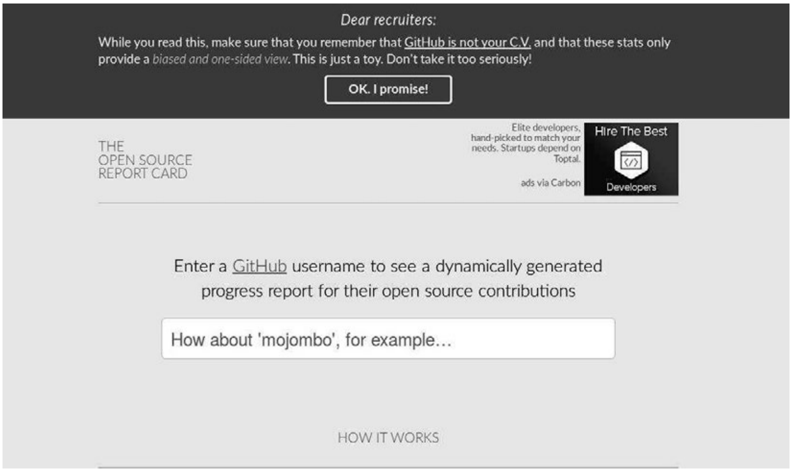
– it is a *public* event, it has a 'payload' (often much more complicated than simply *create README.md*) – and includes various indexical references or *ids* that link the event to other groups of people, organisations, repositories and images (*gravatar_id*). The intricate syntax of this data – many brackets, inverted commas, colons, commas – attests to a social configuration that orders actors, actions, places and times in discrete events in time. The list of components in the event data – actor, organisation, repository, payload gravatar, etc. – attests to the potential for the specific configuration of elements around a given repository to undergo rearrangements. New actors might be added; relations might appear between entities; the location of entities might shift, and forms of association ('organisations') might subsume or grow out of or around all of this. In short, each event in the timeline API suggests another small reconfiguration in the totality of elements comprising Github.

Imbuing numbers with importance

The ethnographic ambition to count or re-count capital numbers in the interests of understanding complex operational environments is not a lone wolf activity. Strikingly often, ethnographic research encounters parallel or similar ambitions associated with different actors. When our project started in 2012, we were clearly not the only people interested in using API data to understand the massive stream of event data converging on Github. A dataset purporting to contain the whole public event timeline of Github 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 archiving 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 as hourly packages in a cloud-based data store but also copied that data to Google's newly launched cloud computing data analysis platform, *GoogleBigQuery*. 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 Google BigQuery (Google 2016). Like the data on GithubArchive itself, the Google BigQuery copy of the Github public timeline data was updated hourly. Reaching back to 2011, it logs around four hundred million public events.[4]

The archiving, copying and transformation of Github event data into big data-ready format drew the attention of many people. The data was publicised through several ‘Github Data Challenges’ (2012–14). Working in different ways using data-analytic techniques and data visualisations, people re-counted events on the Github timeline in order to tell different ‘stories’ about Github’s millions of repositories and people. These stories include many different configurations. ‘Data challenges’, hackathons and the like presented difficulties for my project of recounting the Github repositories. In some ways, the entries and projects that respond to the publication of the data threaten to supplant or render redundant the efforts of social researchers. They ‘socialise’ big data in multiple ways, albeit often retaining if not reinforcing aspects of its capitalisation.

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 7.1). It ingests all the data from the Githubarchive, counting what developers do, when they do it and using what programming languages. With this data



Dear recruiters:

While you read this, make sure that you remember that GitHub is not your C.V. and that these stats only provide a *biased and one-sided view*. This is just a toy. Don't take it too seriously!

Enter a GitHub username to see a dynamically generated progress report for their open source contributions

How about 'mojombo', for example...

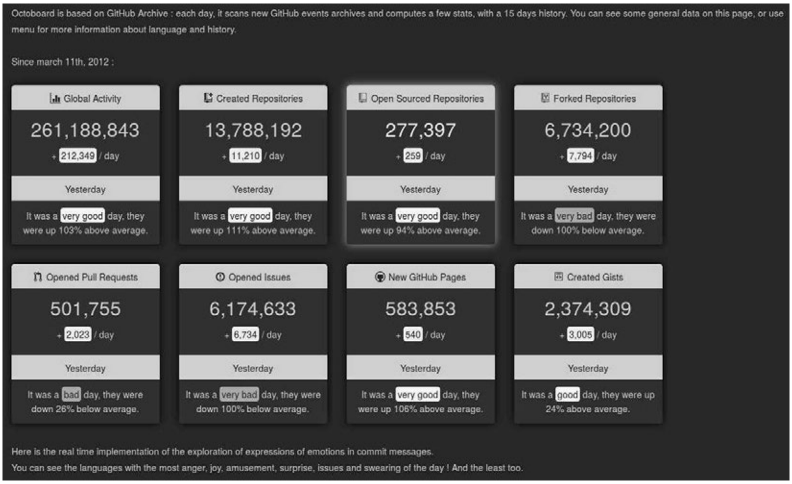
Figure 7.1 The Open Source report card

stored, it then builds a predictive model that allows it both to profile a given Github user in terms of the mixture of programming language they use and to predict whom that Github user might be similar to. Here the mass of events in the Github timeline are brought to bear on finding similarities between people, producing numbers and score to suggest similarities in coding work. Similarly, in response to the Github Data Challenge in 2012, people looked in the timeline data for feelings or 'sentiments' associated with different programming languages. Feelings associated with coding were mined by counting emotional words present in comments accompanying the Github events (Gómez 2012). The presence of words in these messages can be cross-linked with programming languages in order to profile how different programming languages elicit different emotional reactions.

Capital numbers were also *nationalised* or regionalised. People mapped 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 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 2013).

~~**include_graphics**('figure/octoboard.png')~~

Logistic narratives can be derived from data streams. People made live dashboards, a characteristic data-analytic visual form, for Github. Octoboard (http://octoboard.com/) animates changes on Github using the timeline data (Roussell 2015) (see Figure 7.2). Logistic narratives ornament the capital numbers with a range of peripheral live enumerations that point to the productive flow of actions on Github. They do this in the form of summaries of daily activity in major categories on Github – how many new repositories, how many issues, how many repositories have been 'open sourced' today. 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, operational awareness of the flow of social actions.



Octoboard is based on GitHub Archive : each day, it scans new GitHub events archives and computes a few stats, with a 15 days history. You can see some general data on this page, or use menu for more information about language and history.

Since march 11th, 2012 :

Global Activity
261,188,843
+ 212,349 / day
Yesterday
It was a very good day, they were up 103% above average.

Created Repositories
13,788,192
+ 11,210 / day
Yesterday
It was a very good day, they were up 111% above average.

Open Sourced Repositories
277,397
+ 259 / day
Yesterday
It was a very good day, they were up 94% above average.

Forked Repositories
6,734,200
+ 7,794 / day
Yesterday
It was a very bad day, they were down 100% below average.

Opened Pull Requests
501,755
+ 2,023 / day
Yesterday
It was a bad day, they were down 26% below average.

Opened Issues
6,174,633
+ 6,734 / day
Yesterday
It was a very bad day, they were down 100% below average.

New GitHub Pages
583,853
+ 540 / day
Yesterday
It was a very good day, they were up 106% above average.

Created Gists
2,374,309
+ 3,005 / day
Yesterday
It was a good day, they were up 24% above average.

Here is the real time implementation of the exploration of expressions of emotions in commit messages.
You can see the languages with the most anger, joy, amusement, surprise, issues and swearing of the day ! And the least too.

Figure 7.2 Octoboard: a Github dashboard

Looking slightly more widely, the Github timeline data has quickly become a favourite training tool for data mining textbooks 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 using social network analysis to highlight the important nodes and links between repositories and users (Russell 2013). Finally and not least for my own projects, academic researchers in computer science and certain parts of management science, GithubArchive and the BigQuery publication of Github have been a boon because they permit relatively easy study technologically and economically important practices of software development. Academic researchers in fields such as software engineering do social network analysis in order to gauge productivity, reuse, efficiency and other engineering and management concerns (Thung et al. 2013). Like the many Github-hosted projects discussed above, they analyse sentiment (Guzman et al. 2014), collaboration and productivity (Dabbish et al. 2012) and geography (Takhteyev and Hilts 2010).

All of these re-countings enliven, animate, reactivate, localise and qualify the capital numbers. They configure the numbers in terms of work, geography, liveness and further accumulation by summing them up in different ways (realtime status updates and dashboards, networks of associations, geographies of work and affect) commonly found in contemporary data economies and as the outcome of big data practice. Many of the dashboards, maps, sentiment analyses and predictive recommendations are common in big data practice, and the fact that people using Github should so readily analyse Github itself using big data infrastructures such as GoogleBigQuery and other analytic devices is hardly surprising. Coders and software developers are, after all, key workers in the ongoing transformation of systems of controls and configuration associated with big data.

Acts of imagined accumulation

Given this vortex of work around the Github data, which I interpret both as symptomatic of big data practices and as a set of parallel 'storyfications' of the 'datafication' of Github, where would an ethnographic sensibility intervene? George Marcus in his account of prototyping ethnographic experimentation emphasises the ongoing relevance of 'images of moving through natural settings of social action' (Marcus 2014: 3). Is there any place or point at which analysis might ethnographically engage with the code-infrastructure-oriented practices associated with flows of data? Given my – and indeed 'our', since this project was a team effort, including statisticians and social theorists – propensity or somewhat unwitting interpellation as a white, middle-class, middle-age, relatively technically over-skilled male social researcher long drawn into information infrastructural transformations, the viable forms of ethnographic attentiveness to differences, slippages, experiential ambivalences and ambiguities were not obvious. Could writing queries for GoogleBigQuery GithubArchive dataset, developing data analytic and visualising code in languages such as R and python, programming languages commonly used in the big data practices, actively prototype forms that did not simply conform to the implicit and pervasive injunction to 'tell stories with data'?

In this respect, the elementary character of the repository count was usefully grounding. Technically, the number is relatively easy to approximate in the GithubArchive data. A simply GoogleBigQuery produces the number in roughly 2.0 secs having processed around 40Gb of repository URLs:

```
library(bigrquery)
query1 = “SELECT count(distinct(repo.url))
FROM
[githubarchive:year.2011],
[githubarchive:year.2012],
[githubarchive:year.2013],
[githubarchive:year.2014],
[githubarchive:year.2015]”
repo_count = query_exec(‘metacommunities’, query = query1)
## Auto-refreshing stale OAuth token.
```

The result: 31490949 repositories.

Could we re-count such capital numbers in order both to further highlight their dynamic composition through flows of association and to highlight some of the highly reactive, imaginary boundary work they might do as they materialise imaginaries of totality, globality or infrastructural control? Could configurative numbers in both their compositional and imaginary-materialising multiplicity be counted? Whilst there is much configurational work involved even in beginning to count things heterogeneously on an infrastructural scale, I found myself drawn by the transience of configurations and their imitative composition. Both concerns – transience and imitation – are not highlighted in any of the other re-counts of the GithubArchive data. They only became salient to me amidst many other failures to find any interesting story, stable signal or statistical regularity in the Github timeline data.[5]

When we count people or things (such as software projects), the working assumption is that they have some duration and substantial presence. This assumption, however, does not hold very true at the scale of large data streams, where, like subatomic particles in a collision, people and things rapidly and unpredictably flash in and out of existence. Transience and instability are artefacts of the data stream. Frustrated by our seeming inability to find anything in the GithubArchive data apart from the obvious importance of well-known software projects such as linux or android, we were forced to accept transience and ephemeral visibility as a common mode of existence

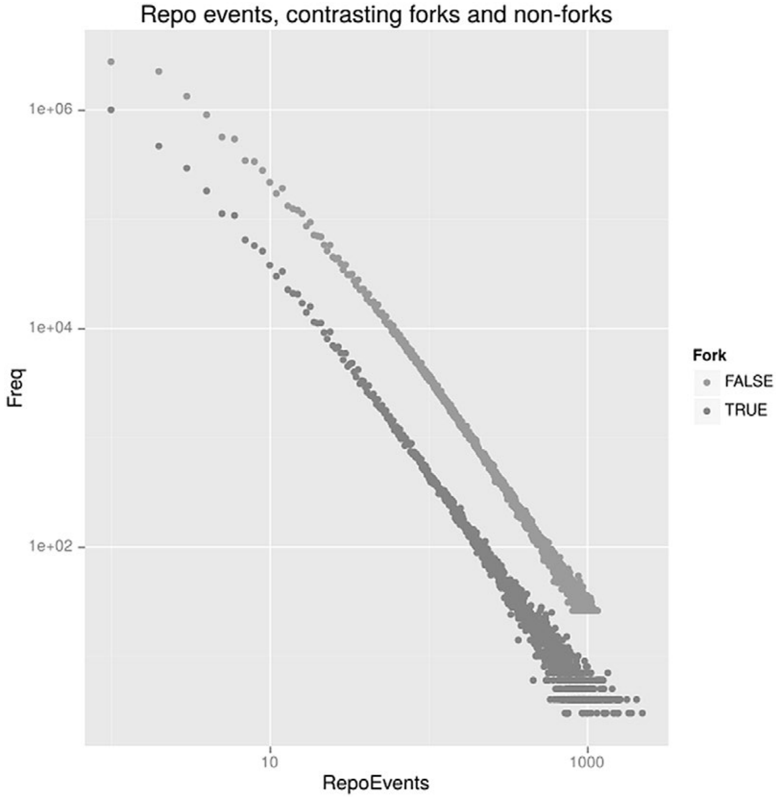


Figure 7.3 Events associated with repositories

on Github. The transient visibility of people and things in the data takes many forms. The vast majority of repositories on Github are very short-lived. They attract one or two 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 blink into visibility on the timeline for a brief period before falling back into obscurity. Yet they survive in the capital number. Figure 7.3 encapsulates this imitative flux. On the left side, millions of repositories receive fewer than five events during the 18 months. On the right, fewer than fifty 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 trigger only a few. Already, then, the capital number of repositories on Github takes on a different composition when viewed from the perspective of duration.

The massive asymmetry between the relatively few long-duration repositories and the vast mass of transient, abandoned repositories is definitely not another corroboration of 'the long-tail' distributions that social media exponents such as Clay Shirky and Chris Anderson began discussing more than a decade ago and that may have had a large part to play in many of the intensely individualising tendencies of big data analytics, targeted advertising, predictive feeds and recommendations. (For instance, one reason that Amazon.com stocks so many obscure products is that the long tail of sales of these items is potentially more important than the sales of a smaller number of best-sellers (Brynjolfsson et al. 2006; Anderson 2009).) Rather than the long tail of coding, a scale-free network of code-hubs on Github (itself a hub in networks of code-related infrastructure), or even simply waste, noise or something to be discarded, we might trace the working of associative processes through highly skewed distribution of events.

I can give only a brief indication of some prototyping experiments. For instance, counting just the unique names of repositories in the timeline data, counting how often and when they appear as event in the data stream begins to suggest something about the composition of the capital number of repositories. Almost two-thirds of the repositories in Github inject only one or two events into the timeline data stream ever. This means that twenty million of the thirty million repositories are just flashes in the timeline data appearing as one or two events.

What are these events? The event of creating a named repository is primary, followed by the event of pushing (putting something into a repository – a 'commit'), and then act of copying ('forking') another repository. People 'fork' other repositories frequently. If we just count the event of creating a repository by copying or forking it, more than half all repositories are copies of existing repositories (1.665343710^{7}). If every ForkEvent on Github creates a new repository, more than half of the 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 151891374 or 31 per cent of all events in the Github

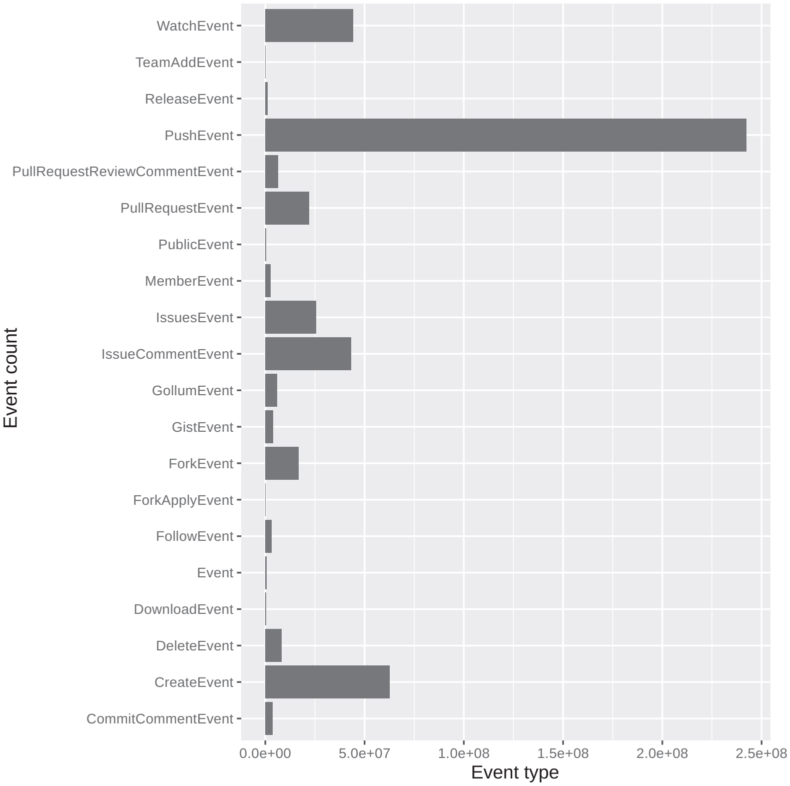


Figure 7.4 Event counts on Github 2011–2015

timeline data arise from copying, watching, commenting on or otherwise observing other repositories. (See the WatchEvent, ForkEvent, PullRequestEvent,and IssueEvent counts in Figure 7.4.)

The growth of associative imitation

```
##
## Attaching package: 'readr'

Retrieving data: 2.1s
Retrieving data: 4.0s
Retrieving data: 5.9s
Retrieving data: 7.8s
Retrieving data: 9.7s
Retrieving data: 12.2s
Retrieving data: 14.1s
Retrieving data: 15.7s
# 160 lines removed
Retrieving data: 161.9s
Retrieving data: 165.4s
Retrieving data: 170.3s
Retrieving data: 174.2s
Retrieving data: 178.2s
```

While it is hard to grasp the texture of imitative processes in event data, they figure deeply in the composition of the capital numbers. We can observe some localised aspects of propagation of imitation by going beyond the formatted data of the API or GithubArchive data. Much social media platform data relies heavily on unique names (hence, for instance, Facebook's insistence that every user has a single unique identity). Yet the associative play of names (of people and things) attests to processes of associative imitations that continually overflow any singular naming. In the years 2012–15, both mobile devices and user interfaces were being intensively transformed in complex ways (e.g. the growth of apps, and the transformation of webpages from static HTML-formatted text to thoroughly dynamic script-driven surfaces composed for many elements). While these transformations were not directly the work of big data practices, they are intricately connected to it by virtue of the proliferation of devices, the deeper integration of networks of association they implement and, not least, the changes in the texture of contemporary experiences of interfaces. Both Figures 7.5 and 7.6 attempt to figure something of this proliferating-integrating process of change. Both figures 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 of any repository on Github).

```
## Parsed with column specification:
## cols(
## repo_name = col_character(),
## created_at = col_datetime(format = ““),
## cnt = col_integer()
## )
```

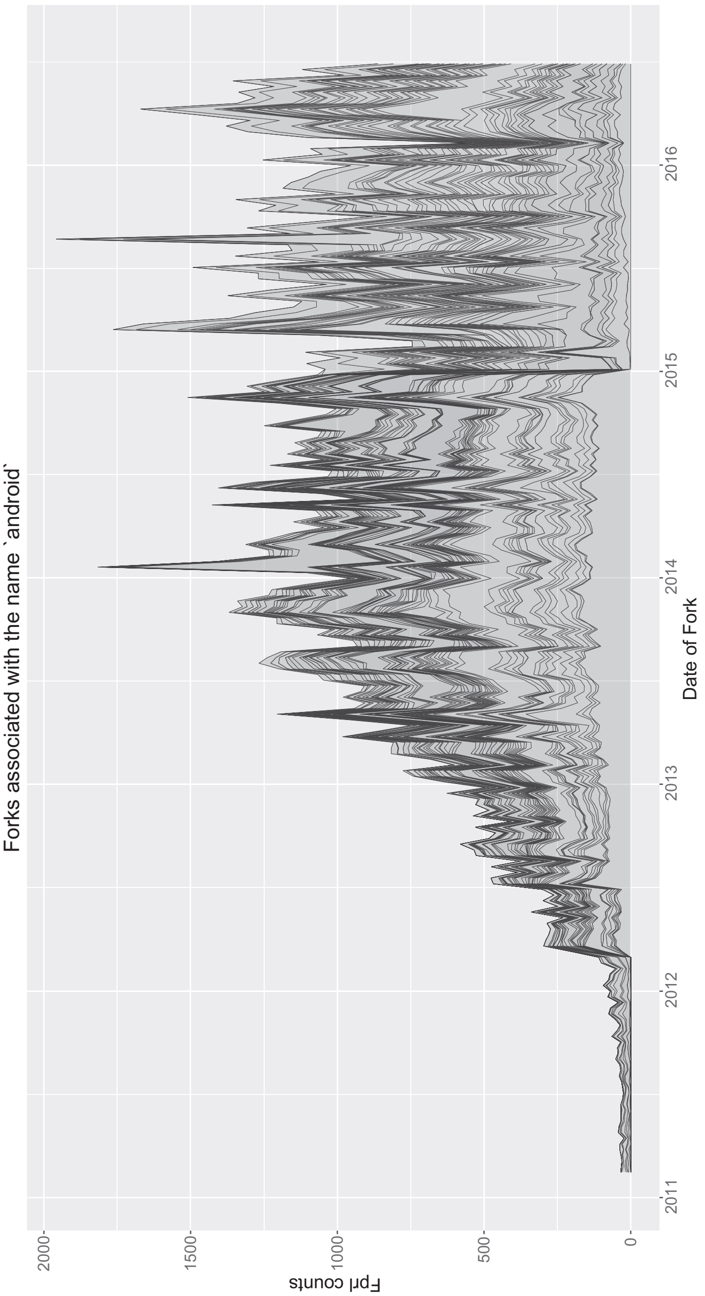


Figure 7.5 Repository forks associated with the name android

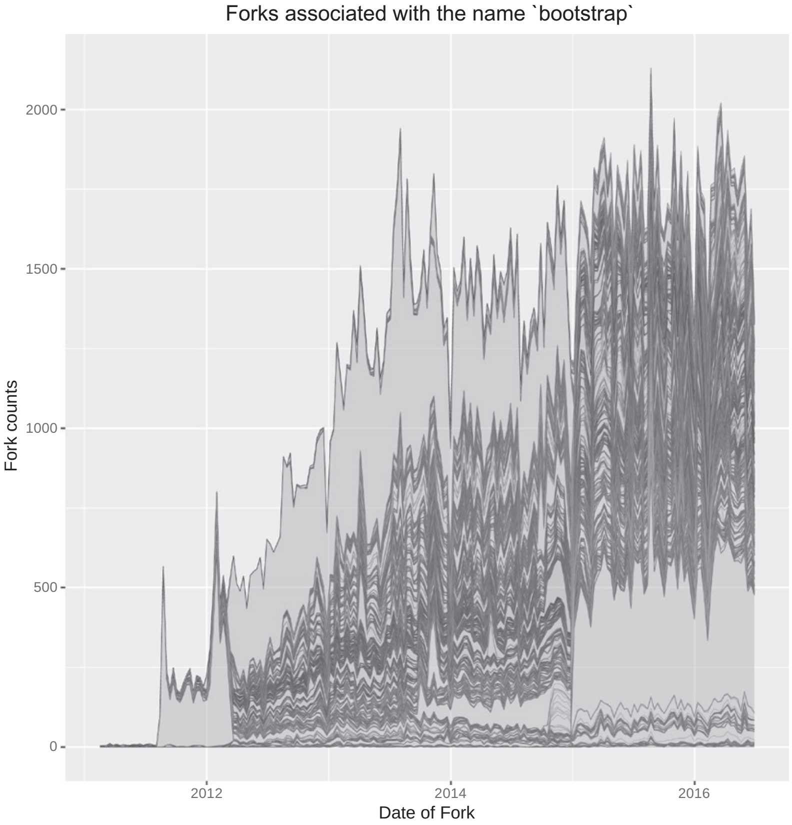


Figure 7.6 Repository forks associated with the name bootstrap

We might call this form of composition, as it configures capital numbers, *stacked associative imitation*. Associative imitation appears in two different ways in these figures. The broad bands of colour rippling horizontally across the base of the figures graph the counts of copies being made each day on Github of popular repositories such as android or bootstrap using the Fork action. (In forking, the repository name remains the same.) But the much more dense striations running above the base ribbons, seen for instance in Figure 7.5, count repositories whose names combine the base repository (e.g. android) but vary it in some way. These repositories associate with the base repository, but diverge from it in a multiplicity of ways. A repository may, for instance, relate to the popular bootstrap repository yet combine it with a range of other platforms and devices such as android or jQuery.

In the striated zones of associo-imitation plotted in the figures, it is possible to count something that is neither the long tail nor a link-based network of affiliations, or any figure that is part of a business solution or an implementation. The cross-ply of repository imitations identifies the average everydayness of coding work with forms of associative investment and affective identification that range across many different repositories. In this work, certain repositories act as high-visibility markers around which waves of hybridisation stack up. Unlike the capital numbers with their total aggregates of people or repositories, the imitative fluxes that bulk out these diagrams have diverse networks of association and affiliation. The implication here is that, by counting elements within the capital numbers, we can begin to glimpse the pathways of generative circulation that join together contemporary technologies.

Decapitalising configuration

Densely populated with ephemeral repositories, stacked with associative imitations, the capital number 29 million seemed in the light of these re-countings to be a sieve-like entity, overflowing with differently configured processes. Yet still the count was not exhaustively accounted for in these different counts. The very act of counting these imitations has a device-specificity, in its reliance on the Github platform API, the GithubArchive timeline dataset, the GoogleBigQuery cloud analytic's platform, and indeed Github itself as the repository of the code and text comprising this article ((Metacommunities 2016a). 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 0.1 per cent of repositories attract around 26 per cent of all events during 2011–15. (The query decapitalises 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 eventful repositories, which comprise only 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, despite its own figuration as a hub where coding is done socially, functions as an element in wider configurations. For instance, the repository eclipse.platform.common accounts for almost two million events in the timeline data. A single repository attracting two million events (or almost 1 per cent 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' (Eclipse Foundation 2016) with almost a thousand of its own projects, might help explain the large number of events. More significantly, the high-frequency event traffic 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 – frequently copied – from the hundreds of git repositories found at git://git.eclipse.org. Mirroring practices attest to the capital importance of Github in coding cultures, but they also suggest that the value of Github is not paramount. Organisational life carries on elsewhere, outside the workflows and social coding mechanisms facilitated by Github. Like many other significant repositories (linux, android, mozilla, apache), high event counts often configure Github itself as part of a wider network of relations.

italics

italics

Conversely, even where repository events do originate on Github, many of them concern Github or technical configurations closely associated with Github itself. For instance, the list of names of high-event-count repositories centres on a few highly repeated names such as *test*, *dot*, *sample* and *try_git*. The many repositories that contain the term *dot* as in *dotfile* or *vimdot* suggest that Github repositories act as stores for the settings and configurations specific to coding work (e.g. *vimdot* repositories hold customised settings for the popular *vim* text editor). Terms like *test*, *hello*, and *try* or *demo* also stream through this set of repository names. These repositories often attest to non-code uses of Github and sometimes (for instance, in the case of *KenanSulayman/heartbeat* with its several million commit events) to surprising or experimental repurposings of the Github platform.

Do configuration-oriented practices around Github affect the capital numbers and the value and importance they are meant to convey? Configuration is, as science and technology studies scholars have emphasised, vital to the very existence of technologies. If many of the 29 million repositories on Github function as elements in the configuration of coding work rather than as bodies of code in their own right, then any account of what happens cannot assume that analysis of Github data primarily concerns 'how people build software' (Github 2015). Or, put more constructively, it suggests that much of what counts as software on Github – code repositories – in actuality concerns the arrangements that people make with each other and themselves in order to work with and inhabit information infrastructures. In this respect, the capital numbers are always also configurative numbers. There is no work on software without the 'how' of building software, the configuration work.

Even if we only added one repository to Github, our – the five researchers directly involved in the research – work on the Github API, GithubArchive datasets and the GoogleBigQuery platform was full of configurative events. The repository for this project contains thousands of lines of code written in Python, in R and in specialised languages such as SQL (Structured Query Language) ((Metacommunities 2016a) as well as tens of thousands of lines of text distributed across thirty branches of the repository. According to the Github API, user rian39 added around ten million lines of code and deleted eight million, leaving a net contribution of around two million lines to the metacommunities repository during the years 2012–16 ((Metacommunities 2016b). In terms of ethnographic notebook writing, even if these lines are never read, this is an impressive level of inscription. While those numbers might be read as attesting to extraordinary levels of code productivity, they actually include many lines of data stored in the Github repository, alongside code, drafts of documents and configuration-related information. The deletion of eight million lines suggests that some of the main activities in the repository were changes to do with cleaning up, tidying and rearranging files and documents in the repository.

```
## opening file input connection.
## ..
Found 22499 records...
Imported 22499 records. Simplifying...
```

I think all the highlighted text should go. Won't mean much to readers ...

```
## Warning: closing unused connection 7 (https://api.github.com/
   repos/
## metacommunities/metacommunities/branches)
## Warning: closing unused connection 6 (https://api.github.com/
   repos/
## metacommunities/metacommunities/contributors)
## Warning: closing unused connection 5 (https://api.github.com/
   repos/
## metacommunities/metacommunities/languages)
## closing file input connection.
## Saving 5 x 4 in image
```

Perhaps this level of contribution activity to a repository suggests something important about the prototyping of configurative numbers in big data practice. Most of the ten million lines added and eight million lines deleted were configuration-related work as we – I in particular – sought to make sense of capital numbers such as 29 million, and tried to differentiate that sense-making from the flood of data challenges, hackathons and other usages of the GithubArchive data that were also offering geographical, sentiment, logistic and temporal narratives about 'how people build software'. In the process of trying to re-count the capital number, we 'built software' that generated four hundred or images, one hunndred or so data files, and around one thousand other files. More than 22,000 queries on GoogleBigQuery took place. Figure 7.7 shows something of the flow of data involved in this effort. By mid-2016, 80 terabytes of Github data had been searched, queried, tallied, counted, sorted and arranged according to the job logging supplied by the Google Compute Platform. This includes one day in August 2013 where 12,000 jobs were executed. It appears as a steep ascent on the left side of the graph. Traversing 40 terabytes, and generating a bill of around $US2000, the events of that day triggered a 'reaching out' from Google Compute marketing department who were interested in our 'use case'. Why write so much code, process so much data, create so many figures and images, and indeed organise all of this work in the sometimes maddeningly precise version control system of a git repository on Github?

Conclusion

Where does this leave 29 million repositories? Halved by all the repositories with no more than a single event, halved again by all the

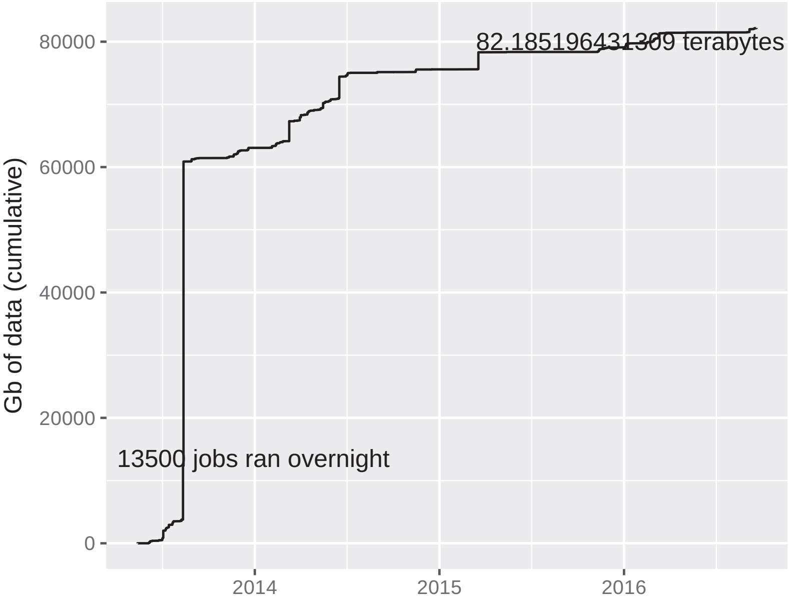


Figure 7.7 BigQuery processes terabytes in counting a capital number

repositories that are a copy of other repositories, hollowed out by all the repositories where people do not build software but use the repository for some other purpose (as a mirror site for instance), millions of code repositories remain. Github placed the full source code of two million repositories on GoogleBigQuery in mid-2016, perhaps reflecting the reality that the thirty million or so other repositories were not going to add new stories to the platform.

Suchman suggests that configuration always entails both composition of elements and materialising imaginaries. It takes work to get contemporary digital data and associated large numbers to do something other than augment the count of capital numbers and their platform-centred aggregates. *Configurative numbers*, I have suggested, is one term for prototypical enumerations and re-countings that seek to map the composition of capital numbers, and to follow some of the imaginaries aggregating in them. We saw how that re-counting runs along the same lines as the many attempts to ‘storify’ data flows, yet whenever we turn to associative-imitative fluxes, or to very extensive configuration work that saturates the data stream with links to what happens elsewhere, the stories told with data start to become unstable, unfamiliar and at times a little more interesting.

Notes

[1] A recent special issue of the *Journal of Scandinavian Social Theory* on numbers offers a useful variety of engagements with the status of number. See for instance Guyer (2014) on percentages or Gerlitz and Lury (2014) on 'kudos' ratings.

[2] What is an imaginary? I will not address that question directly here, but all concepts of imaginary (ranging from Lacan's to the more recent proliferation of imaginaries in humanities and social science (McNeil et al. 2016)) concern a sense of wholeness or completion that allows formations of identity, inclusion, belonging and otherness to take shape. For an example of the concept of imaginary in an account of software and coding, see Kelty 2005.

[3] Code for the research project can be found at https://github.com/metacommunities/metacommunities.

[4] Slightly complicating matters, the single Github timeline dataset has now been retired on GoogleBigQuery and the data now takes the form of separate tables for each day, month and year since February 2011. More recently, the full contents and histories of over two millions Github repositories have been published as a GoogleBigQuery dataset (GoogleInc_2016a).

[5] Nearly all of our attempts to make sense of the data can be found in the Github repository https://github.com/metacommunities/metacommunities. The scripts, plots and pieces of writing found there, however, are not highly ordered. The key results lie scattered across different branches of that repository. The somewhat tangled, messy aggregate of materials there attests to our strenuous but often incoherent efforts to find valid statistical regularities in a dataset characterised by tremendous heterogeneity and messiness. More interestingly perhaps, that repository can be understood as an ethnographic notebook.

References

Anderson, Chris. 2009. *The Longer Long Tail: How Endless Choice Is Creating Unlimited Demand*. London: Random House Business.

Brynjolfsson, Erik, Hu, 'Jeffrey' Yu and Smith, Michael D. 2006. 'From Niches to Riches: Anatomy of the Long Tail'. SSRN Scholarly Paper ID 918142. Rochester, NY: Social Science Research Network. http://papers.ssrn.com/abstract=918142 [accessed 4 March 2018].

Bucher, Taina. 2013. ‘Objects of Intense Feeling: The Case of the Twitter API: Computational Culture’. *Computational Culture* 3: 1–9. http://computationalculture.net/article/objects-of-intense-feeling-the-case-of-the-twitter-api [accessed 4 March 2018].

Dabbish, Laura, Stuart, Colleen, Tsay, Jason and Herbsleb, Jim. 2012. ‘Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository’. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Wor*: 1277–86. ACM. http://dl.acm.org/citation.cfm?id=2145396 [accessed 4 March 2018].

Eclipse Foundation. 2016. ‘Eclipse – The Eclipse Foundation Open Source Community Website’. www.eclipse.org [accessed 4 March 2018].

Fischer, David. 2013. ‘GitHub Data Challenge II’. David Fischer (blog). 4 May 2013. www.davidfischer.name/2013/05/github-data-challenge-ii/ [accessed 4 March 2018].

Foreman-Mackay, Dan. 2014. ‘The Open Source Report Card’. 14 February. https://web.archive.org/web/20140214105201/http://osrc.dfm.io [accessed 4 March 2018].

Gage, Deborah. 2015. ‘GitHub Raises $250 Million at $2 Billion Valuation’. *Wall Street Journal*, 29 July. www.wsj.com/articles/github-raises-250-million-at-2-billion-valuation-1438206722 [accessed 4 March 2018].

Gerlitz, Carolin and Lury, Celia. 2014. ‘Social Media and Self-Evaluating Assemblages: On Numbers, Orderings and Values’. *Distinktion: Journal of Social Theory* 15(2): 174–88.

Git. 2014. ‘Git’. http://git-scm.com [accessed 4 March 2018].

Github. 2014. ‘Results of the GitHub Investigation’. *GitHub*, 21 April. https://github.com/blog/1823-results-of-the-github-investigation [accessed 4 March 2018].

Github. 2015. ‘About · GitHub’, 16 December. https://web.archive.org/web/20151216055610/https://github.com/about [accessed 4 March 2018].

Gómez, Ramiro. 2012. ‘Exploring Expressions of Emotions in GitHub Commit Messages’. *Gekksta*. http://geeksta.net/geeklog/exploring-expressions-emotions-github-commit-messages [accessed 4 March 2018].

Google. 2016. ‘Github Timeline Data on Google BigQuery’. https://bigquery.cloud.google.com/table/githubarchive:github.timeline [accessed 4 March 2018].

Grigorik, Ilya. 2012. ‘GitHub Archive’. http://www.githubarchive.org [accessed 4 March 2018].

Guyer, Jane I. 2014. ‘Percentages and Perchance: Archaic Forms in the Twenty-First Century’. *Distinktion: Journal of Social Theory* 15(2): 155–73. doi:10.1080/1600910X.2014.920268.

Guzman, Emitza, Azócar, David and Li, Yang. 2014. ‘Sentiment Analysis of Commit Comments in GitHub: An Empirical Study’. In *Proceedings of* *the 11th Working Conference on Mining Software Repositories*; 352–55. MSR 2014. New York, NY: ACM.

Hardy, Quentin. 2012. 'Dreams of "Open" Everything'. *Bits Blog*, 28 December. http://bits.blogs.nytimes.com/2012/12/28/github-has-big-dreams-for-open-source-software-and-more [accessed 4 March 2018].

Jenkins, Henry. 2004. 'The Cultural Logic of Media Convergence'. *International Journal of Cultural Studies* 7(1): 33–43. http://ics.sagepub.com/content/7/1/33.short [accessed 4 March 2018].

Kelty, C. 2005. 'Geeks, Social Imaginaries, and Recursive Publics'. *Cultural Anthropology* 20(2): 185–214.

Kelty, Christopher. 2008. *Two Bits: The Cultural Significance of Free Software*. Durham, NC: Duke University Press.

Latour, Bruno. 1996. *Aramis, or the Love of Technology*. Translated by Catherine Porter. Cambridge, MA, and London: Harvard University Press.

Marcus, George. 2014. 'Prototyping and Contemporary Anthropological Experiments with Ethnographic Method'. *Journal of Cultural Economy* 7(4): 399–410.

Marres, Noortje. 2012. 'The Redistribution of Methods: On Intervention in Digital Social Research, Broadly Conceived'. *The Sociological Review* 60(S1): 139–65. http://onlinelibrary.wiley.com/doi/10.1111/j.1467–954X.2012.02121.x/full [accessed 4 March 2018].

Mayer-Schönberger, Viktor and Kenneth Cukier. 2013. *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Boston, MA: Eamon Dolan/Houghton Mifflin Harcourt.

McNeil, Maureen, Haran, Joan, Mackenzie, Adrian and Tutton, Richard. 2016. 'The Concept of Imaginaries in Science and Technology Studies'. In *Handbook of Science and Technology Studies*. Edited by Ulrike Felt, 3rd ed. London & Thousand Oaks, CA: Sage Publications Ltd, 435–64.

Metacommunities. 2016. 'Metacommunities/Metacommunities'. *GitHub*. https://github.com/metacommunities/metacommunities.

Roussell, Dennis. 2015. 'Octoboard'. *Github Activity Dashboard*, 1 August. https://web.archive.org/web/20150801193208/http://octoboard.com [accessed 4 March 2018].

Russell, Matthew A. 2013. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. O'Reilly Media, Inc. http://books.google.co.uk/books?hl=en&lr=&id=_VkrAQAAQBAJ&oi=fnd&pg=PR4&dq=github&ots=JqiqtzTxmK&sig=sfea4ce1ue2XYt_dERD41VpSTS4 [accessed 4 March 2018].

Straube, Theodore. 2016. 'Stacked Spaces: Mapping Digital Infrastructures'. *Big Data & Society* 3(2): 1–12.

Suchman, Lucy. 2012. ‘Configuration’. In *Devices and the Happening of the Social*. Edited by Celia Lury and Nina Wakeford. London: Routledge, 48–60.

Takhteyev, Yuri and Hilts, Andrew. 2010. ‘Investigating the Geography of Open Source Software Through GitHub. http://takhteyev.org/papers/Takhteyev-Hilts-2010.pdf [accessed 4 March 2018].

Thung, Ferdian, Bissyandé, Tegawendé F., Lo David and Jiang Lingxiao. 2013. ‘Network Structure of Social Coding in GitHub’. In *17th European Conference on Software Maintenance and Reengineering (CSMR), 2013*, 323–26. IEEE.