48 million configurations and counting: platform numbers and their capitalization

## Abstract

Recent work on processes of capitalization has identified platforms as central actors in cultural economic processes. Technical platforms include social network sites, online content management systems, streaming media platforms, mobile communication infrastructures, supply chain logistics management solutions, or cryptocurrency systems. This paper argues that the analysis of platform capitalization should take into account a constitutive opacity of platforms as ensemble realities. It focuses on the role played by the counting of people and things in enacting platforms as assets. Via a case study of the code repository platform [Github.com](https://github.com/), it analyzes how such enumerations or 'platform numbers' participate in capitalization. It describes various attempts to enumerate the elements of the platform by counting, mapping or listing them in the form of large numbers. The paper shows how attempts to enumerate people and things constantly encounter unstable aggregates, and forms of association, duplication, combination, imitation and configuration that are both crucial to the ensemble but remain somewhat refractory to asset-structuring capitalization and its asset enactments.  
It proposes configurative enumeration of the platform numbers as a way of approaching these un-enacted excesses. In a configurative enumeration, the diverse composition, the rhythms of imitation, variation and commutation, and the constant relating, tuning, repairing and adjusting of technical elements crucial to the ongoing existence of ensemble realities come into view. Configurative enumerations, the paper suggests, might critically articulate some of the inventive realities of platformization, realities which precede and sometimes overflow their capitalization.

## Introduction

This paper describes and theorises encounters with a single large number, currently standing at 48 million. '48 million' is a numbering of the code repositories on the social media platform Github. In complex ways, the number conveys the value and significance of (i) what happens on Github: coding or software development: 'Github is how people build software' as Github rather tersely puts it (Github [2015](#ref-Github_2015)); (ii) what happens on platforms more generally. Platforms, as many authors (Gillespie [2010](#ref-Gillespie_2010); Lovink, Tkacz, and de Vries [2015](#ref-Lovink_2015); Plantin et al. [2016](#ref-Plantin_2016); Van Dijck [2013](#ref-VanDijck_2013a); Langley and Leyshon [2016](#ref-Langley_2016)) have suggested, stand at the centre of the processes of capitalization and are themselves interwoven with code drawn from Github and the like.

Numbers such as Github's 48 million present a cultural-analytical artefact rooted in the dynamism of technical platforms. It attests to 'the scope and range of an operation that typically connects distant situations and configures large social realities' (Muniesa et al. [2017](#ref-Muniesa_2017), 17). The associative, imitative and often highly layered fabric of technical platforms gives rises to series of such numbers, numbers that have important sociotechnological functions. Large technical or 'tech' platforms often figure their growth using such numbers. We need only think of the prominence of Facebook with its reported 1.9 billion users (Statista [2017](#ref-Statista_2017)). These numbers enter into calculative devices such as business models and their financialization. They influence how work, sociality, political or cultural life are felt, debated and analyzed. The numbers are highly enacted in the sense that acts of stating them, presenting, monitoring, and comparing their changes matter to many actors.

Large technical platforms, however, depend on processes of concretisation and abstraction, and distributed operational realities, that cannot be easily enumerated or conceptualised.[[1]](#footnote-2) These dependencies are both central to platformization -- the process of rendering an ensemble as a platform -- and capitalization -- the process of re-configuring a given setting as a site of accumulation, growth and revenue (Muniesa et al. [2017](#ref-Muniesa_2017)). Platform analysis has highlighted the entanglement of capitalization and platformization. On the capitalization front, platforms emerge and endure through the conversion of hitherto intangible or uncontrolled social processes into potential asset streams. As Liliana Doganova and Fabian Muniesa argue, following Andrew Leyshon and Nigel Thrift's earlier work (Leyshon and Thrift [2007](#ref-Leyshon_2007)), capitalization transforms a 'technology into an *asset* that has the power to generate a steady stream of future cash flows, that is, into capital' (Doganovam and Muniesa [2015](#ref-Doganova_2015), 110). In many platforms, the constructed asset will largely comprise social practices or behaviours of large numbers of people, or what Nathan Tcaz and Geert Lovink term 'the platformed masses' [Lovink\_2015, 14]. On the platformization front, matters are slightly different. While capitalization as a process includes any 'any system for aggregating ground rents into a mass', platformization typically engineers specific forms of connection and accumulation, whose capitalization remains contingent or even promissory. As Langley and Leyshon conclude, 'the platform is not merely a manifestation of wider transformations in the relations and structures of contemporary capitalism' (Langley and Leyshon [2016](#ref-Langley_2016), 15). 'Analytical attention,' they suggest, 'should be given to the contingent configuration and consequences of the platform as a discrete mode of socio-technical intermediary and capitalist business arrangement' (15).

The difficulty of engaging with platform capitalization begins upstream of any business arrangements. If the philosopher Gilbert Simondon is right, it has some roots in the way in which technological cultures have increasingly gathered collections of technical elements such as tools and machines in ensembles that are neither instruments/tools nor simply taken-for-granted infrastructures (Simondon [1989](#ref-Simondon_1989), 81). Platforms certainly have many machine elements. (Much of the recent work on algorithms and algorithmic cultures could be seen as responding to the effects of machine-populated platforms; see (Hallinan and Striphas [2014](#ref-Hallinan_2014))). Their construction of configured future-oriented asset geographies, however, cannot be explained in terms of machines or technologies, because the grouping of these often quite generic elements requires other relations and configurations to cohere as a platform. As I will suggest, despite their seeming remoteness from configured operational realities, platform numbers such as '48 million' offer scope for sociological engagement with the configured capitalizing ensemble realities of platforms.

## Platformizing practice

If technical platforms do have a configured reality that overflows tool, machine, capital or labour, how would we engage with it sociologically? Given that technical platforms as such cannot be radically novel (since they must aggregate and concretise, group and connect existing technical elements), how are they made? How are machines, technologies, tools and networks platformized?

Github, the case study of this paper, lies at the heart of contemporary platformizing practice. On Github, a social network layer of programmers is immersed in making platforms. Github, itself a capitalizing platform, is a platformizing platform, or a hub of platformizing practice. Its mundane operational reality concerns how platforms (amongst other things: laws, recipes, plans, policies, scientific articles, books, etc. can also be found there) are made, maintained and deployed. Github instances how coding practices set up, concretise and manage platforms.

Platformization depends on programming. Coding practices have changed over the last decade or so in two ways profoundly entangled with technical platforms. First, coding has shifted away from relatively closed proprietary machine-centred software development (the kind epitomised by Microsoft Windows or IBM mainframe software during the latter part of last century (Campbell-Kelly [2004](#ref-Campbell-Kelly_2004); Ensmenger [2010](#ref-Ensmenger_2010))) to a much more legally and technically open, de-centralized practice of varying scope threaded across communication and network infrastructures.[[2]](#footnote-3) An explosion of digital devices and infrastructures, ranging from proliferating mobile devices and sensors to mushrooming data centres, have widened the reach and variety of coding. The lines and borders separating the production, distribution, and operation of code in technical platforms have thoroughly blurred. Code flows in much more diverse ways through infrastructures and devices. For instance, 'devops' (development-operations) moves constantly revised code directly into operational platforms through continuous deployments, often directly from code repositories on sites such as Github (Mackenzie [2016a](#ref-Mackenzie_2016b)). Similarly, the geography of coding work still centres on well-established urban centres such as San Francisco and London (Mackenzie [2016b](#ref-Mackenzie_2016)), but by virtue of the networked distribution of coding tools and skills, occurs in many other places as well.

Second, across all of these changes, coding has become a much more *public* economic and cultural activity (see (Kelty [2008](#ref-Kelty_2008))), and participation in coding has become a major concern for business, art, government and science, albeit for different reasons (for instance, in scientific work, open data has been accompanied by a rapid proliferation of techniques, resources and tools for publishing code written for scientific purposes; for instance, in the many attempts to train school students and others to code as a putative 'basic life skill'). These tendencies in coding work, all of which traverse and structure Github itself, merit much closer attention. Here they signal some of the complexities involved in making sense of contemporary technical platforms. Github, we might say, is a typical technical platform because it intensifies all of these tendencies and tensions around the ensemble, distribution, and publicness/participation in coding practice.

## Platform numbers and aggregating practices

The social media platform Github, started in in late 2007, epitomises and has indeed become a hub of coding. Like some other social media platforms, it has grown tremendously in the last ten years. Github's growth flows from a variety of processes that are difficult to summarise or classify partly because the actors, topics or domains of coding are so diverse. At first glance, it seems to a kind of online file store where, initially for free, coders can publish, store and find code. But much of what flows through Github is meant to be 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](#ref-Latour_1996), 72). As we will see, the growth of Github has many facets. Such growth is not incidental or extrinsic to platform capitalization. As (Doganovam and Muniesa [2015](#ref-Doganova_2015)) argues, the capitalizing devices at work in many platforms predicate growth as the asset or resource that supports revenue streams. Capitalization by definition turns things into assets (Muniesa et al. [2017](#ref-Muniesa_2017), 12). While online code repositories have existed for several decades (most famously [SourceForge](http://sourceforge.net/) or [GoogleCode](http://code.google.com/)), which has closed and migrated to Github), more recent code repository platforms such as Github, [BitBucket](https://bitbucket.org/), or [GitLab](https://about.gitlab.com/), attempt to encompass a much wider domain of distributed, participatory and public coding practice. They tend to invite and promote collaborative coding as a social networked practice, with all the appurtenances of reputational metrics, publication tools, user profiling and organizational management we have come to associate with social network platforms.

Like many contemporary technical platforms, Github has describes and performs its own growth. It presents itself through large numbers such as 48 million projects, and these numbers in their sometimes daily changes and updates attest to an investment in future-revenue asset construction. These platform numbers are often accompanied by images that concretise the social identity and geographic location of developers. In June 2014, the Github homepage showed a photo of a woman working in an urban office-loft (presumably San Francisco, where Github is based), with lights and professional camera focused on her work. It described '6.1 million people collaborating right now across 13.2 million repositories ... to build amazing things together' (Github [2014](#ref-Github_2014)). In late November 2015, Github had a slightly more functional description, and the image seemed to be of Github founders in 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](#ref-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 Github, allegedly by the Chinese government[[3]](#footnote-4)). But the changing numbers, alongside the sometimes rather sublime scales of 'community' (a 'community' of 18 million people?) in several respects only highlight and even exacerbate the difficulty of grasping the composition of Github as an ensemble.

Numbers such as 18 million 'people' and 48 million 'projects' are part and parcel of platform capitalization. Github has attracted large investments of capital in the last few years ($US100 million from the venture capital investment fund Andressen-Horowitz in 2012) on the basis of these numbers. They may be somewhat 'false numbers' as Martha Lampland suggests, designed to enable certain forms of rationalization (Lampland [2010](#ref-Lampland_2010)) or investment (Plassnig [2016](#ref-Plassnig_2016)). They have the feel of what Carolin Gerlitz and Celia Lury describe in their study of the social reputation website Klout as 'reactive numbers' meant to evoke further work, action and engagement (Gerlitz and Lury [2014](#ref-Gerlitz_2014)). They may support claims of global importance. And they change: the series of numbers 29, 31, and now 48 million announce and perhaps enact Github as an 'enumerated entity' in time (Verran [2012](#ref-Verran_2012), 61). The platform numbers that feature on Github's webpages re-play in other settings (for instance, in the many articles appearing in newspapers, magazines and business press about the popularity and growth of Github; see (Hardy [2012](#ref-Hardy_2012); Meyer [2013](#ref-Meyer_2013); Gage [2015](#ref-Gage_2015); Newcomer [2016](#ref-Newcomer_2016); Plassnig [2016](#ref-Plassnig_2016))).

Platform numbers resemble other numbers in their cardinality (the count of elements): population statistics for small nations, amounts of money spent on a new building, a research programme or a CEO's remuneration. But in platform capitalization they become visible and circulate in ways that diverge from other enumerations. Their salience and pluri-potency will be described further below, but play a particular role in relation to platforms and their capitalization. The simultaneously enumerate the gathering or aggregation of things and people around the platforms, and in their changes, evoke its potential to grow and perhaps to accommodate new actors.

## Aggregating events

At the end of 2016, the Github platform counted 18 million people and 48 million projects. The people are software developers or coders (although, as ever with online platforms, a 'person' may be automated software or a bot (Niederer and van Dijck [2010](#ref-Niederer_2010)). The Github projects counted in 48 million are git repositories, named collections of files mainly containing code, but also a great variety of operational documents (settings, manuals, installation instructions, etc.), stored in many versions and varieties (Mackenzie [2016a](#ref-Mackenzie_2016b)). Any enumeration of Github contends with not only the sheer number of people visiting and using the platform, but the sprawling diversity of ways in which they associate with each other through repositories using one or more of several hundred different programming or scripting languages. It is difficult to say which platform number matters more -- the head count or the repository count. On Github, they often appear in tandem: '2,641,337 people hosting over 4,442,708 repositories' announces the Github home page on 27 November 2012 (Github [2012](#ref-Github_2012)).

How is the number 48 million actually produced? The count of repositories, or 'projects' as Github more recently terms them, is, however, an artefact of the way that the flow of code into and out of Github is structured by the platform. In order to allow many versions to exist, and many people to work on the same code without introducing conflicting or incompatible changes, every repository or project on Github closely tracks the state of the files it contains. Versions of the same project may exist in parallel and/or in time. Control of the merging and branching of versions of code allows platforms to be updated, reconfigured, integrated, expanded, deployed and localised. In practice, many different ways of working with code exist side by side on Github. A remarkable variety of workflows occur on Github, and these workflows, sometimes comprising hundreds of thousands of distinct actions carried out on group projects, attest to the highly relational and commutative reality of technical platforms.

The relational intricacy associated with code version control is premised on uniquely naming or otherwise identifying every single event or action on Github, even those that appear to be quite elementary. Take the act of copying a repository, which is called 'forking' by Github or 'cloning' by developers. Such acts are published by Github on its API or Application Programmer Interface. The act of publishing events on a API, or a programmatically accessed internet address, attests to the fact that Github is part of an ensemble that connects different platforms, devices, sites and practices together through code (Bucher [2013](#ref-Bucher_2013)).

An extract from a single ForkEvent presents some of the unique naming practice that renders platform numbers countable:

{  
 "actor": {  
 "avatar\_url": "https://avatars.githubusercontent.com/u/7136540?",  
 "gravatar\_id": "",  
 "id": "7136540",  
 "login": "stas-g",  
 "url": "https://api.github.com/users/stas-g"  
 },  
 "created\_at": "2016-12-07 00:09:21 UTC",  
 "id": "4979965325",  
 "org": null,  
 "other": "{\"actor\":{\"display\_login\":\"stas-g\"}}",  
 "payload": " ...,  
 "public": "true",  
 "repo": {  
 "id": "22321806",  
 "name": "ntd/tccv",  
 "url": "https://api.github.com/repos/ntd/tccv"  
 },  
 "type": "ForkEvent"  
}

The ForkEvent on Github shown in the data extract documents the act of an actor calling themselves stas-g copying or forking the repository named tccv, a software project coordinated by the 'actor' calling themself ntd.[[4]](#footnote-5) The data has various status designations -- it is a public event, it has a 'payload' (not included in the extract) -- and contains various indexical references or ids (repo\_id, actor\_id, gravatar\_id) that both connect 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 specific technical elements in the Github ensemble -- database architectures in particular -- that coordinate and align actors, actions, and code in time. More generally, the formatting of all actions as events with attached 'payloads' (the contents of which are not shown in the data extract) permits precise enumeration of human and non-human elements, and opens the possibility of their changing configuration on the platform. New actors might be added; relations might appear between repositories; the location of entities might shift, and forms of association ('organizations') might subsume or grow out or around all of this.

## Differentiating the flux of events into asset geographies

Formatting action in named events not only provides momentary apprehensions of the platform numbers. It creates space for what Leyshon and Thrift term 'asset geographies' (Leyshon and Thrift [2007](#ref-Leyshon_2007), 109). As well as affording lump sum totals such as 48 million repositories, the flow of action formatted as named events yields a quite fine-grained sequence of numbers relating to different actors and things. The platform formatting of fluxes of action is both crucial to the operation of the platform and to its capitalization as a future-oriented asset geography.

Table 1 Github event counts 2012-2015

|  |  |
| --- | --- |
| type | events |
| PushEvent | 140,739,368 |
| CreateEvent | 35,483,741 |
| WatchEvent | 26,101,411 |
| IssueCommentEvent | 25,056,189 |
| IssuesEvent | 16,166,541 |
| PullRequestEvent | 11,208,920 |
| ForkEvent | 10,037,523 |
| GistEvent | 4,816,399 |
| GollumEvent | 4,253,087 |
| DeleteEvent | 3,680,053 |
| FollowEvent | 3,435,804 |
| PullRequestReviewCommentEvent | 3,066,117 |
| CommitCommentEvent | 2,493,741 |
| MemberEvent | 1,492,529 |
| ReleaseEvent | 418,180 |
| DownloadEvent | 302,247 |
| PublicEvent | 261,821 |
| TeamAddEvent | 175,909 |
| ForkApplyEvent | 5,628 |
| Total | 289,195,208 |

Table 1 shows a more differentiated view of events on Github.[[5]](#footnote-6) Github itself publicly presents all the events on the platform in various ways: through the platform-defining Application Programmer Interfaces (APIs; (Github [2016](#ref-Github_2016))), through online archives such as [GithubArchive](http://www.githubarchive.org/) or [GHTorrent](http://ghtorrent.org/) (GHTorrent [2017](#ref-GHTorrent_2017)) and through cloud-based data analytics platforms such as GoogleBigQuery (Inc [2016](#ref-GoogleInc_2016a)). These data-streams embody the changes, aggregations, and commutations occurring as technical platforms take shape, concretise, localize or indeed disintegrate. With around one million events each day (in December 2016), making sense of these processes might seem difficult. The 300 million or so events summarised in Table 1 range across a variety of different code repository practices ('push', 'create,' 'watch,' 'pullrequest', 'fork') that I will not explore in depth here.

Github, like many contemporary technical platforms, tracks and measures everything that happens on its platform. Metrics for its own operation are part of the ongoing re-invention of the ensemble. (See for instance, the [Github Engineering blog](http://githubengineering.com/) for many examples of this metric aspect of the ensemble). Knowing what is happening at various points in the distributed operation of Github is a key practical concern, just as it is in financial markets (Knorr-Cetina and Bruegger [2002](#ref-Knorr-Cetina_2002)) or in control rooms more generally. While some operational metrics -- response times of servers in the data centres, number of read/write operations/minute, etc. -- are constantly monitored and sometimes acted upon by engineers, the variety, heterogeneity, significance and possible economic value of what flows through the platform in the form of actions and practices of coding is much harder to gauge for Github.

Uncertainty about or obscurity in what accumulated on the platforms does not thwart or damage platform capitalization. It can become a provocative or potentialising uncertainty in its own right if techniques or forms of realisation can be applied to it. During 2012-2014 Github organised three 'Data Challenges' (Github [2013](#ref-Github_2013)) focused on the data that Table 1 draws on. The challenges (or competitions) invited entrants to use the GithubArchive or GoogleBigQuery data to make sense of what was happening on Github by producing visualizations, websites or interactive systems. Given the technical hurdles of working with large amounts of Github data, most entrants to the competitions were themselves developers or data scientists using Github. In this sense, the Data Challenges staged a recursive Github public (Kelty [2005](#ref-Kelty_2005)) concerned with making sense of platform numbers or engaged in practical re-countings of the data.

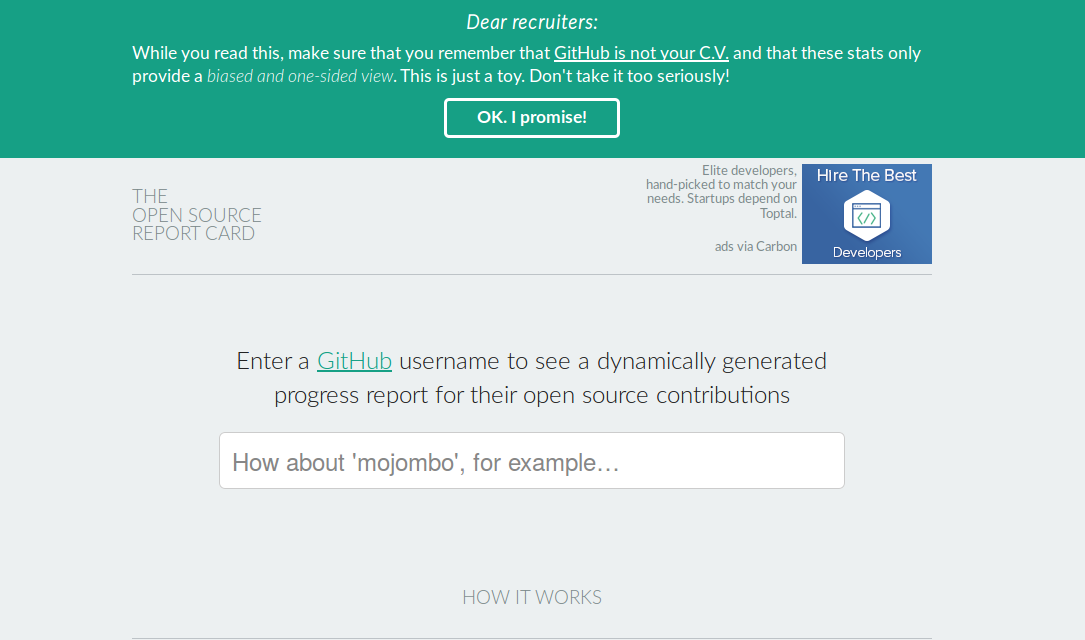


Figure 1 The Open Source Report Card

Entrants to the Data Challenges elaborated the platform numbers in various experiential, geographic and infrastructural registers. For instance, the 'OpenSource Report Card' (<http://osrc.dfm.io/>) or dfm/osrc by Dan Foreman-Mackay (Foreman-Mackay [2014](#ref-Foreman-Mackay_2014)), is a prize-winning use of the timeline event data (see Figure 1). It ingests all the data from the Githubarchive, counting what developers do, when they do it, and with what programming languages. With this data stored, it then builds a predictive model to both profile a given Github user and predict similarities between that user and others. The osrc filters and arranges the stream of events in the Github timeline in order to find similarities between people. An admonition from Foreman-Mackay – '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! ' – suggests that even explicitly playful applications of the data quickly encounter processes of capitalization. For coders, programmers and software developers, the profile of their activity on Github repositories can be part of getting work.

In response to the Github Data Challenge in 2012, contestants looked for feelings or 'sentiments' in the event 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. Certain languages attract exasperation and others pleasure (see (Coleman [2012](#ref-Coleman_2012)) for a broader account of these feelings). The enumeration of affects animates the platform numbers with an underpinning affectivity associated with coding in both its vicissitudes and its unevenly shared dividends of participation in and control of technical platforms (Chun [2011](#ref-Chun_2011)).

The platform numbers associated with Github also begin to acquire a geography through the Data Challenges. Many entrants mapped coders and repositories by geographic location. The mapping of Github contributions by location performed by David Fischer is typical in that it too counts events, but emphasises the geography of the 'top' repositories, coders and their programming languages (Fischer [2013](#ref-Fischer_2013a)–2013-05-05T02:47:58+00:00).

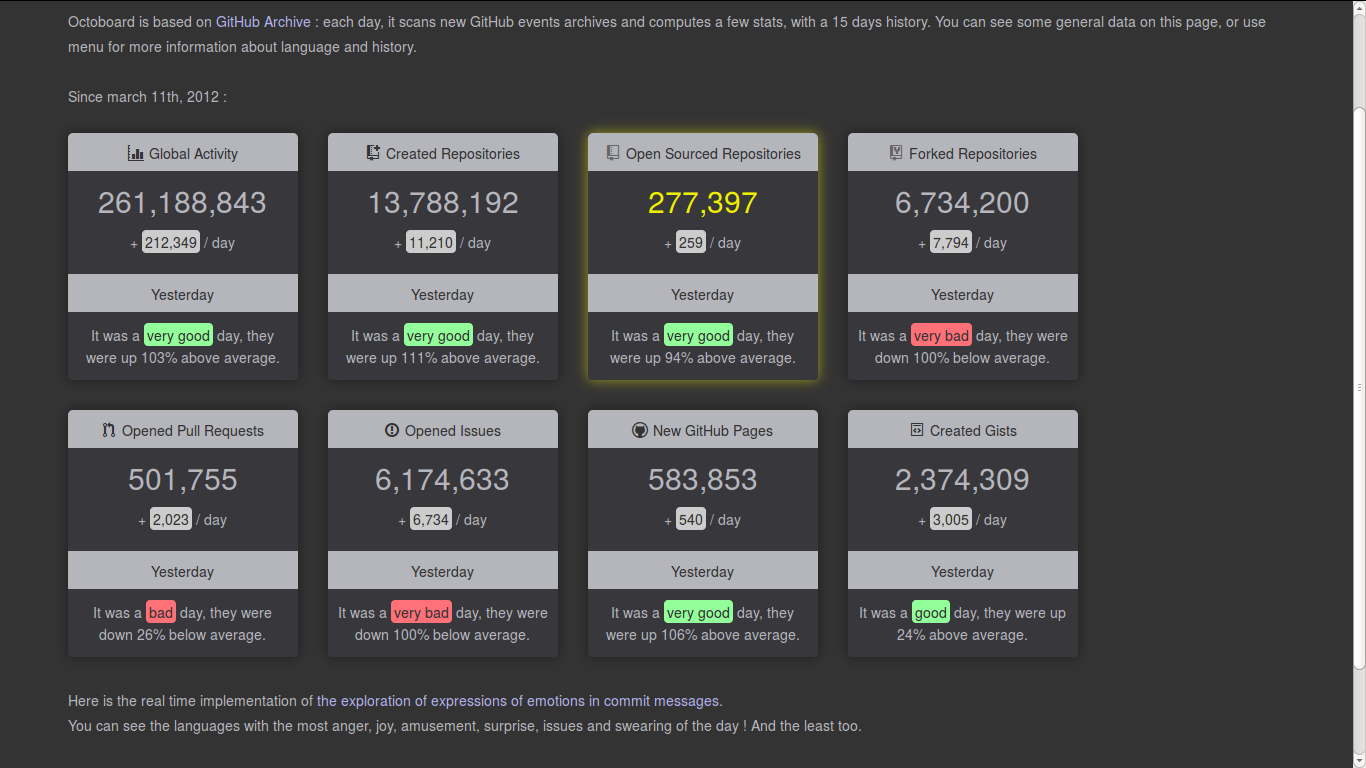


Figure 2 Octoboard: Github activity dashboard

Echoing the metrics and gauges that Github constructs to monitor and regulate its own operations, other entrants to the Data Challenges made live dashboards for Github. Octoboard (<http://octoboard.com/>) animates changes on Github using the timeline data (Roussell [2015](#ref-Roussell_2015)) (see Figure 2). Octoboard elaborates and differentiates the platform numbers in a range of 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 almost real-time analytics on emotions. Like many other dashboards associated with social media platforms, octoboard suggests that the constant change in associations between people and things can no longer be known through leisurely rhythms of analysis, but be treated as a problem real-time awareness. Significant shifts, trends, hotspots of activity, improbably important marginal developments or breakdowns need to be brought into immediate view through 'stream analytics.' That is, not also does the data stream, but the analysis is meant to stream as well in order to be timely, lively and responsive to change.[[6]](#footnote-7)

The Data Challenges also simply highlight the publication of the flow of platform events. As the event data-stream is taken up in various quarters, its public availability and transformation into maps, reports, dashboards, network visualizations and sentiment analyses augments or supports the platform numbers that Github counts on. The figurations of labour, geography, affect, and logistics enliven, animate, reactivate, localise and qualify platform numbers such as 48 million. They potentialise the numbers in terms of expansiveness, liveness and further accumulation by summing them up in different ways (temporalities, networks of connections, geographies of work and affect) that might yield further ways of constructing assets with revenue streams.

## From platform number to configurative enumeration

I have been arguing that platform capitalization pivots on enumerations of a future-oriented asset geographies. These enumerations of people and things construct an uncertain magnitude whose capitalizing potential derives from the multiple ways in which what is enumerated might be re-counted or re-grouped in asset forms. These enumerations are synthetic, platform-specific counts. They commonly drawn on reputational measures such as 'likes,' 'watchers', and in the case of Github, the act of forking or copying code. It is not, however, the case that platform capitalization necessarily produces stable forms of organisation or circulation. Like Karl Marx's decomposition of the commodity form into shifting ratios of forms of capital (fixed, relative and absolute surplus value, etc. (Marx [1986](#ref-Marx_1986)), we might critically decompose platform numbers in the interests as (a) seeing how the numbers capture or fail to capture the forms of life they purport to enumerate and stabilise as assets; and (b) gauging whether platformization, or the process of rendering social life through platforms overflows the parameters of the associated asset geographies. The diversity seen in the Github Data Challenges also suggest the possibility of viewing platform numbers from a different angle: as an incitement to explore differences and changes in what happens on Github (and similar technical platforms).

I am suggesting, then, that platforms numbers have configurative, compositional and figural modalities that do not simply act performatively (as most current understandings of capitalization and related cultural-economic processes presume; see (Mohamed et al. [2011](#ref-Mohamed_2011); **???**; Cardwell [2015](#ref-Cardwell_2015); Paudyn [2015](#ref-Paudyn_2015); Jerne [2016](#ref-Jerne_2016); Yarrow and Kranke [2016](#ref-Yarrow_2016))), but derive from pre-capitalizing processes specific to the platform in question.[[7]](#footnote-8) More than performative, platform numbers are configurative.

The configurative modality of platform numbers appears already in a seemingly trivially abundant yet elementary entity in code and on Github more generally, the *name.* As Ludwig Wittengstein suggests, 'only in the nexus of a proposition does a name have a meaning' (Wittgenstein [2003](#ref-Wittgenstein_2003), 3.3). 'Naming' lies at the heart of the massive systems of addressability associated with software (see (Thrift and French [2002](#ref-Thrift_2002))). The sample data shown in Data Extract 1 suggests that events in the Github timeline data and on the platform more generally can be understood as nexus of named entities. The event data format is a nexus of names referring to people, organisations, repositories, platforms, code libraries or packages, as well as a very wide range of devices and places. This seemingly trivial feature of code-driven technical platforms -- the reliance on names as a form of address amidst huge numbers of technical elements -- powerfully informs both the platform number 48 million, and the configurative dynamism associated with it. Github's platform numbers count people and repositories precisely because entities (people, devices, locations, etc.) have been named.

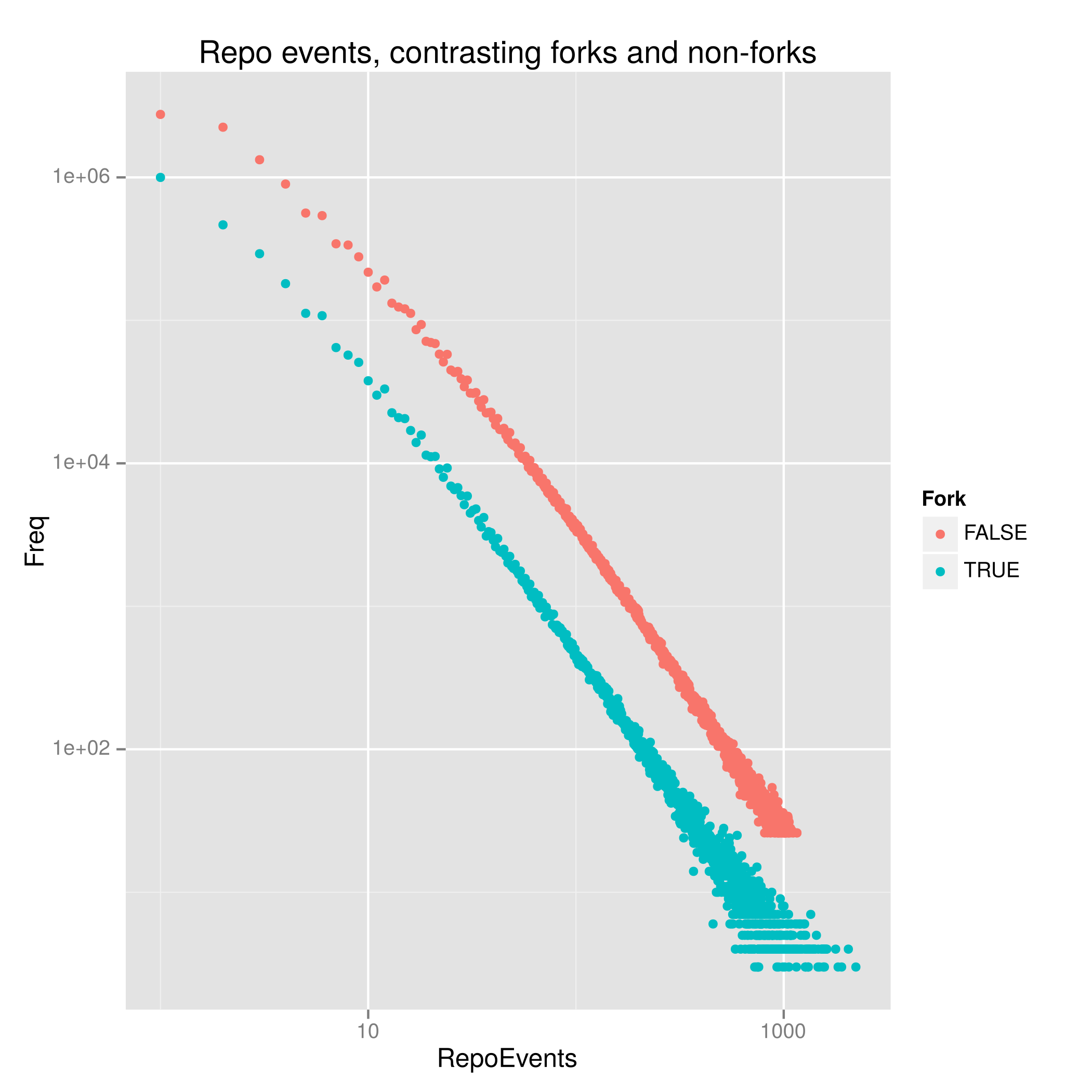


Figure 3 The power of events over time

What would happen if we approach the platform number 48 million with an eye on the diverse temporal, associative and above all configurative composition of named entities in the timeline event data (as published on GoogleBigQuery)?[[8]](#footnote-9) The beginning of such an encounter appears in Figure 3. This not-so simple plot of the number of events associated with different Github repositories suggests, first of all, that the vast majority of repositories on Github are highly ephemeral. On the top left hand side of the figure, millions of repositories receive less than five events. On the right hand side, less than 50 repositories receive more than a thousand events.

The simple act of counting the number of events received by each repository re-scales the platform number of repositories in one key respect. Ephemeral repositories (if that is not too great a contradiction in terms) vastly outnumber repositories that endure. Ephemera constitute the main body of the 48 million repositories (that is, repositories comprising one event constitute more than 50% of the 48 million). Millions of repositories flash into existence on the timeline for a brief period before falling back into uneventful obscurity. (A similar pattern appears in the other platform number: while some 'people' emit many thousands of events, others only trigger a few.[[9]](#footnote-10)) Already, then, this re-configuration splits the platform number of repositories on Github: we might differentiate eventful and uneventful repositories. Given that repositories are a primary asset form on Github, it might seem strange that the vast bulk of them are so uneventful. Rather than seeing this ephemerality as waste, noise or something to be discarded, we might regard some of these uneventful repositories as pointing to differential processes at work in platform numbers.

A second configurative differentiation appears in Figure 3. Most repositories come into existence through an act of copying --'forking'-- another repository. In Figure 3, two separate sets of points distinguish repositories that are forks from those are not. People 'fork' other repositories frequently and we can this glimpse this forking in the event counts. Each time a repository is forked, an ForkEvent appears in the timeline event data. Table 1 shows 10 million ForkEvents, suggesting that around one third of the total 29 million repositories (in existence at the end of 2015) began life as copies. Surprisingly, the fork repositories attract more events than those that are not forked. The lower right hand end of Figure 3 suggests that repositoriesi created through forking or copying have some of the highest event counts.

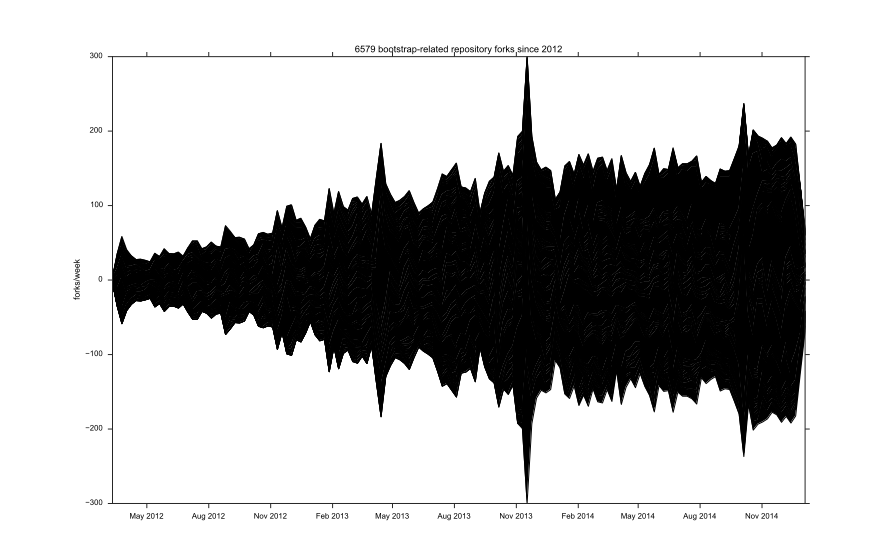


Figure 4 bootstrap repository forks

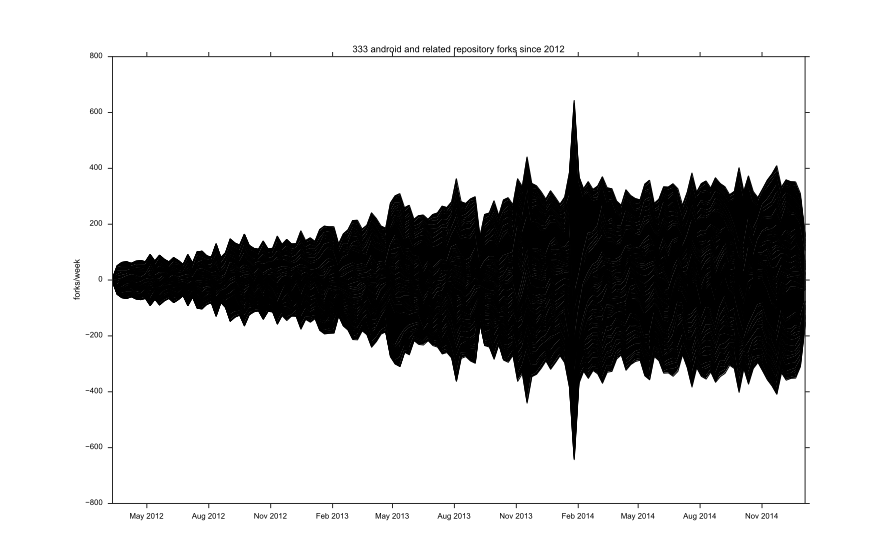


Figure 5 android repository forks

A final example of the convoluted growth forms occurring on platforms can been seen in how repositories combine with each other. If copying powerfully drives the growth of platform numbers, but repositories also derive from commutations of existing repositories. Both Figure 4 and 5 count the number of repositories that commute or recombine existing repositories. Rather than counting overall or fork events, these plots how some repositories (bootstrap, android, etc.) attract a comet's tail of variations that augment and mutate its operations. Important repositories act as high visibility technical landmarks around which ensembles concretise in widely varying configurations.

Two processes interweave here. People or other actors fork repositories such as bootstrap (a very widely used code library developed by Twitter that supplies visual elements for web-based user interfaces) or android (the operating system developed by Google and used on most smart phones, tablets and mobile computing devices).[[10]](#footnote-11) Each day, new combinations of existing repositories appear as new repositories on the platform -- e.g. bootstrap-android, android-tensorflow. The striations seen in the figures count forks made of repositories whose name incorporates the base repository. These repositories associate with the bootstrap or android repository, but diverge from it in different directions. A repository may, for instance, relate to bootstrap yet combine it with some other platform, device or infrastructure such as android or jQuery. The repositories that populate the striations in the diagram generate large numbers of events on Github, and hence strongly animate platform numbers. But recombining repositories populating these figures will never have prominence comparable to the base repositories they relate to. These striations mark processes -- an incessant commutation of technical elements in the ensembles -- that can barely appear as such.

## The generalization of configuration

Platform numbers, and their associated future-asset geographies cover over these ephemeral, associative and combinatory processes. The abundance of short-lived, copied or re-combined repositories is not inimical to growth and future revenue streams, but it suggests that any such growth is secondary to the underlying dynamism of this background activity, activity that lies very close to the practices and lives of people using the platform. Platformizing and platform capitalization depends on the fabric woven through ephemeral, associative and combinatory processes. In enumerating the repository count, one final very small fragment of the 48 million, effectively 1000 repositories or 0.002 % of the total number of repositories, looms large. The top 1000 repositories attract around 16% of all events in the timeline period.[[11]](#footnote-12) Two different tendencies stand out in this subset of the 48 million. The small number of very high-event rate repositories is led by a repository called eclipse.platform.common. The repository eclipse.platform.common accounts for almost 2 million events on Github to date. A single repository receiving two million events (or almost 1% of the total event count for the platform) suggests an important site for coding cultures. 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](#ref-EclipseFoundation_2016)) with almost a thousand of its own projects all focused on developing tools, systems, and environments for software development, helps explain the large number of events. The flow of events around eclipse attests to the ongoing genesis of technical tools essential to platformization. The coding, maintenance and even deployment of platforms takes place in software development environments such as eclipse and even the decades old vim code editor. Their configuration, maintenance, and openness to change are constant objects of attention.

The high number of events received by eclipse.platform.common suggests, perhaps more importantly for our purposes, that the dynamism of platforms, their capacity to grow, to change or stabilise, to hybridize or even infrastructuralize (Plantin et al. [2016](#ref-Plantin_2016)) is a highly distributed process. For instance, the eclipse repositories are not actively developed on Github. The code found in that repository is a mirror or frequently updated copy of the hundreds of git repositories found at [git.eclipse.org](https://git.eclipse.org/). The development and much of the associative fabric of eclipse does not reside on Github, but is copied there.[[12]](#footnote-13) In other words, the many events flowing into the eclipse.platform.com repository on Github are copies of work done elsewhere. For many other significant repositories (linux, android, mozilla, apache), high event counts attest here to Github's function as a mirror for technical action taking place elsewhere. Many events animating the platform numbers of repositories and 'people' imitate things happening elsewhere.

A final tendency in high event-count repositories is seemingly trivial yet vital to the ongoing genesis of technical platforms as configured operational realities. The list of names of high-event count repositories cluster around a few highly repeated items: test, hello, config, doc, build, setting or demo recur frequently as repository names. The repetition of such names -- for instance, the many repositories that contain the term dot as in dotfile or vimdot -- exemplifies the copying and commutation discussed above, but it points to something specific to configuration as the composition of a technical platform. These repositories store configurations and record work done to sustain configuration of technical elements and machines in particular places. In the platform number 48 million, the proportion of events absorbed by repositories explicitly concerned with configuration is substantial. Of the events received by the top 1000 repositories, 18% belong to configuratively-named repositories. These repositories explicitly focus on setting up and maintaining the small but crucial technical details of configuration that connect code libraries, editors, build tools, databases, network connections, online access and authorisations needed by developers in order to position themselves within the operational ensemble of technical platforms.[[13]](#footnote-14)

## Conclusion

Platform capitalization can be understood as a organized asset structuring focused on networking forms of everyday life as future revenue streams. The case of Github presents typical features of platform capitalization. It solicits sticky reputational behaviours associated with the habitus of coding and programming. It configures coding as a 'social' practice, replete with organisational and collaborative platform affordances. While code itself is the tangible asset here, code is indissolubly mixed with coding, in all its practical varieties and ntangible dimensions. Processes of collaboration, and programmers' identification with bodies of code are structured as assets alongside the code itself.

Platform capitalization enacts high-level summaries or totals -- platform numbers -- that anchor capitalization and connect it to other capitalization devices such as business models. The highly circulated user-counts and, in the case of Github, project counts enumerate an aggregate, imbue it with quantitative importance linked to platform scale and complexity, and promise to generate further accumulations of work and value. As we have seen, these platform numbers attract further enactments in the form of data archives (GithubArchive), financial investment and diverse data-enabled sense-making.

Even as platform numbers enact platformization, they offer little insight into the associative, commutative, imitative and configured dynamism that animates technical platforms. They evoke dynamism, a form of change vital to capitalization in its promissory modality, but they cannot subsume or fully digest it. In key respects, Github might be a singular case of platform capitalization because it makes this indigestibility more obvious. It is platform where platforms are assembled, and where the ongoing shaping of platforms in their programmability is practically accomplished. Because it is a platformizing platform, a place where the coding masses technically and socially negotiation platform configurations, Github presents the possibility of observing some of the internal dynamics of platformization. Github presents, almost despite itself, and even as it seeks to centre forms of coding life on the platform, much evidence of margins of indeterminacy and instability in practices of association, copying, imitations and localized configuration.

Compared to Facebook and many other platforms, Github is relatively little known, despite its importance as a site of ongoing platform configuration, invention, maintenance and assembly. This relative obscurity is perhaps, however, actually typical of platform capitalization more generally if, as I have been suggesting, all platforms have a constitutive, configurative opacity rooted in their multiple or ensemble reality.

In their recent work, Fabian Muniesa and co-authors advocate engagement with capitalization: 'Studying capitalization is also redefining it, engaging with it, against it, alongside it' (Muniesa et al. [2017](#ref-Muniesa_2017), 127). In this paper, I have approached platform numbers as they derive from naming practices native to code. This approach is platform-specific and pays close attention to 'contingent configuration' (Langley and Leyshon [2016](#ref-Langley_2016), 15). Configurative enumeration views platform numbers in ways that pay more attention to the their heterogeneous ensemble realities. Departing from platform numbers, configurative enumeration seeks to render visible the margin of indetermination central to contemporary technical platforms. Treating platform numbers as configurative enumerations might give us a more plural sense of platformizing processes. We have seen the platform number of repositories on Github as a temporally, socially, materially variable configuration.

The configurative re-enumerations presented in the preceding discussion do not exhaust the proliferating combinations of machines, infrastructures, and connections taking and platforms. The successive re-countings -- almost half of the repositories are copies or forks; the vast majority of repositories only ever receive 1-2 events; a substantial proportion of code repositories have no code in them; oriented recombination of projects heavily contribute to the repository count; many repositories mirror work done elsewhere, configurative repositories hold things together -- are not exhaustive enumerations. They are limited in their grasp of what happens to that aspect of the platform ensemble held together by names, counted in their copies and variations. But such configurative re-counting points to imitative fluxes, to patterns of variation, to the organizational life of platforms as ensemble realities, to the ongoing processes of concretisation that technical platforms undergo as part of their genesis, and to the configurative processes that in many ways precede and sometimes de-stabilise the capillary extraction of value in capitalization.

## Acknowledgments

[REMOVED FOR PEER REVIEW]

## References

Bratton, Benjamin H. 2016. *The Stack: On Software and Sovereignty*. 1 edition. Cambridge, Massachusetts: MIT Press.

Bucher, Taina. 2013. “Objects of Intense Feeling: The Case of the Twitter API : Computational Culture.” *Computational Culture*, no. 3. <http://computationalculture.net/article/objects-of-intense-feeling-the-case-of-the-twitter-api>.

Bynens, Mathias. 2016. “Mathiasbynens/Dotfiles.” *GitHub*. <https://github.com/mathiasbynens/dotfiles>.

Campbell-Kelly, Martin. 2003. *The History of Mathematical Tables*. Oxford University Press.

———. 2004. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. New Ed edition. Cambridge, Mass.: MIT Press.

Cardwell, Emma. 2015. “Power and Performativity in the Creation of the UK Fishing-Rights Market.” *Journal of Cultural Economy* 8 (6): 705–20. <http://www.tandfonline.com/doi/abs/10.1080/17530350.2015.1050441>.

Chun, Wendy. 2011. *Programmed Visions: Software and Memory*. The MIT Press.

Coleman, Gabriella. 2012. *Coding Freedom: The Ethics and Aesthetics of Hacking*. Princeton N.J.: Princeton University Press.

Cooper, Melinda, and Martijn Konings. 2016. “Pragmatics of Money and Finance: Beyond Performativity and Fundamental Value.” *Journal of Cultural Economy* 9 (1): 1–4. <http://www.tandfonline.com/doi/abs/10.1080/17530350.2015.1117516>.

Dabbish, Laura, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. “Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository.” In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, 1277–86. ACM. <http://dl.acm.org/citation.cfm?id=2145396>.

Doganovam, Liliana, and Fabian Muniesa. 2015. “Capitalization Devices: Business Models and the Renewal of Markets.” In *Making Things Valuable*, edited by Martin Kornberger, Lise Justesen, Anders Koed Madsen, and Jan Mouritsen, 109–25. Oxford University Press.

Eclipse Foundation. 2016. “Eclipse - The Eclipse Foundation Open Source Community Website.” <https://www.eclipse.org/>.

Ensmenger, Nathan. 2010. *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. MIT Press. <http://www.jstor.org.ezproxy.lancs.ac.uk/stable/j.ctt5hhjdh>.

Fischer, David. 2013–2013-05-05T02:47:58+00:00. “GitHub Data Challenge II.” *David Fischer Dot Name*. <https://www.davidfischer.name/2013/05/github-data-challenge-ii/>.

Foreman-Mackay, Dan. 2014. “The Open Source Report Card.” February 14. <https://web.archive.org/web/20140214105201/http://osrc.dfm.io/>.

Gage, Deborah. 2015. “GitHub Raises $250 Million at $2 Billion Valuation.” *Wall Street Journal*. July 29. <http://www.wsj.com/articles/github-raises-250-million-at-2-billion-valuation-1438206722>.

Gerlitz, Carolin, and Celia Lury. 2014. “Social Media and Self-Evaluating Assemblages: On Numbers, Orderings and Values” 15 (2): 174–88. doi:[10.1080/1600910X.2014.920267](https://doi.org/10.1080/1600910X.2014.920267).

GHTorrent. 2017. “The GHTorrent Project.” <http://ghtorrent.org/>.

Gillespie, Tarleton. 2010. “The Politics of ’Platforms’.” *New Media & Society* 12 (3): 347–64. doi:[10.1177/1461444809342738](https://doi.org/10.1177/1461444809342738).

Github. 2012. “GitHub · Social Coding.” <http://wayback.archive.org/web/20121127113950/https://github.com/>.

———. 2013. “Data Challenge II Results.” *GitHub*. June 26. <https://github.com/blog/1544-data-challenge-ii-results>.

———. 2014. “Results of the GitHub Investigation.” *GitHub*. April 21. <https://github.com/blog/1823-results-of-the-github-investigation>.

———. 2015. “About · GitHub.” *Internet Archive*. December 16. <https://web.archive.org/web/20151216055610/https://github.com/about>.

———. 2016. “Event Types & Payloads | GitHub Developer Guide.” <https://developer.github.com/v3/activity/events/types/>.

Grigorik, Ilya. 2012. “GitHub Archive.” <http://www.githubarchive.org/>.

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](https://doi.org/10.1080/1600910X.2014.920268).

Guzman, Emitza, David Azócar, and Yang Li. 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, USA: ACM. doi:[10.1145/2597073.2597118](https://doi.org/10.1145/2597073.2597118).

Hallinan, Blake, and Ted Striphas. 2014. “Recommended for You: The Netflix Prize and the Production of Algorithmic Culture.” *New Media & Society*, June, 1–21. doi:[10.1177/1461444814538646](https://doi.org/10.1177/1461444814538646).

Hardy, Quentin. 2012. “Dreams of ’Open’ Everything.” *Bits Blog*. December 28. <http://bits.blogs.nytimes.com/2012/12/28/github-has-big-dreams-for-open-source-software-and-more/>.

Inc, Google. 2016. “GithubArchive Timeline.” *Google BigQuery*. <https://bigquery.cloud.google.com/table/githubarchive:github.timeline>.

Jerne, Christina. 2016. “Performativity and Grassroots Politics: On the Practice of Reshuffling Mafia Power.” *Journal of Cultural Economy* 9 (6): 541–54. <http://www.tandfonline.com/doi/abs/10.1080/17530350.2016.1214849>.

Kelty, C. 2005. “Geeks, Social Imaginaries, and Recursive Publics” 20 (2): 185–214. doi:[10.1525/can.2005.20.2.185](https://doi.org/10.1525/can.2005.20.2.185).

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

Knorr-Cetina, Karin, and Urs Bruegger. 2002. “Traders’ Engagement with Markets. A Postsocial Relationship.” *Theory, Culture and Society* 19 (5/6): 161–85.

Lampland, Martha. 2010. “False Numbers as Formalizing Practices.” *Social Studies of Science* 40 (3): 377–404. doi:[10.1177/0306312709359963](https://doi.org/10.1177/0306312709359963).

Langley, Paul, and Andrew Leyshon. 2016. “Platform Capitalism: The Intermediation and Capitalisation of Digital Economic Circulation.” *Finance and Society.* <http://dro.dur.ac.uk/19828/>.

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

Lazzarato, Maurizio. 2014. *Signs and Machines: Capitalism and the Production of Subjectivity*. Cambridge, MA: Semiotext (e). <http://mitpress.mit.edu/books/signs-and-machines>.

Leyshon, Andrew, and Nigel Thrift. 2007. “The Capitalization of Almost Everything: The Future of Finance and Capitalism.” *Theory, Culture & Society* 24 (7-8): 97–115. doi:[10.1177/0263276407084699](https://doi.org/10.1177/0263276407084699).

Lovink, Geert, Nathaniel Tkacz, and Patricia de Vries, eds. 2015. *MoneyLab Reader: An Intervention in Digital Economy*. Institute of Network Cultures.

Mackenzie, Adrian. 2006. “Java: The Virtuality of Internet Programming.” *New Media & Society* 8 (2): 441–66.

———. 2016a. “Infrastructures in Name Only? Identifying Effects of Depth and Scale.” In *Infrastructures and Social Complexity*, edited by Penny Harvey, Caspar Bruin Jensen, and Morita Atsuro. London & New York: Routledge.

———. 2016b. “Mindful Infrastructures.” *Cultural Anthropology*.

Marczak, Bill, and Nicholas Weaver. 2015. “China’s Great Cannon.” *The Citizen Lab*. April 10. <https://citizenlab.org/2015/04/chinas-great-cannon/>.

Marres, Noortje, Carolin Gerlitz, and others. 2015. “Interface Methods: Renegotiating Relations Between Digital Social Research, STS and Sociology.” *Sociological Review*. <http://research.gold.ac.uk/11343/>.

Marx, Karl. 1986. *Capital A Critique of Political Economy. The Process of Production of Capital*. Moscow: Progress.

Meyer, Robinson. 2013. “Github, Object of Nerd Love, Makes Play for Non-Programmers.” *The Atlantic*. August 44. <http://www.theatlantic.com/technology/archive/2013/08/github-object-of-nerd-love-makes-play-for-non-programmers/278971/>.

Mohamed, Abdel-rahman, Tara N. Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E. Hinton, and Michael A. Picheny. 2011. “Deep Belief Networks Using Discriminative Features for Phone Recognition.” In, 5060–3. IEEE. doi:[10.1109/ICASSP.2011.5947494](https://doi.org/10.1109/ICASSP.2011.5947494).

Muniesa, Fabian, Liliana Doganova, Horacio Ortiz, Álvaro Pina-Stranger, Florence Paterson, Alaric Bourgoin, Véra Ehrenstein, Pierre-André Juven, David Pontille, and Basak Saraç-Lesavre. 2017. *Capitalization: A Cultural Guide*. Paris: Presses de l’Ecole des Mines. <https://ideas.repec.org/p/hal/journl/halshs-01426044.html>.

Newcomer, Eric. 2016. “GitHub Is Building a Coder’s Paradise. It’s Not Coming Cheap.” *Bloomberg.com*, December 45. <https://www.bloomberg.com/news/articles/2016-12-15/github-is-building-a-coder-s-paradise-it-s-not-coming-cheap>.

Niederer, Sabine, and José van Dijck. 2010. “Wisdom of the Crowd or Technicity of Content? Wikipedia as a Socio-Technical System.” *New Media & Society* 12. <http://dare.uva.nl/cgi/arno/show.cgi?fid=337567>.

Ong, Aihwa, and Stephen J Collier. 2005. *Global Assemblages : Technology, Politics, and Ethics as Anthropological Problems*. Malden, MA: Blackwell Publishing.

Paudyn, Bartholomew. 2015. “The Struggle to Perform the Political Economy of Creditworthiness: European Union Governance of Credit Ratings Through Risk.” *Journal of Cultural Economy* 8 (6): 655–72. <http://www.tandfonline.com/doi/abs/10.1080/17530350.2015.1062792>.

Pellandini-Simányi, Léna. 2016. “Non-Marketizing Agents in the Study of Markets: Competing Legacies of Performativity and Actor-Network-Theory in the Marketization Research Program.” *Journal of Cultural Economy* 9 (6): 570–86. <http://www.tandfonline.com/doi/abs/10.1080/17530350.2016.1214614>.

Plantin, Jean-Christophe, Carl Lagoze, Paul N. Edwards, and Christian Sandvig. 2016. “Infrastructure Studies Meet Platform Studies in the Age of Google and Facebook,” 1461444816661553. <http://nms.sagepub.com/content/early/2016/08/02/1461444816661553.abstract>.

Plassnig, Moritz. 2016. “GitHub Is Doing Much Better Than Bloomberg Thinks - Here Is Why.” *Medium*. December 39. <https://medium.com/@moritzplassnig/github-is-doing-much-better-than-bloomberg-thinks-here-is-why-a4580b249044>.

Roussell, Dennis. 2015. “Octoboard.” *Github Activity Dashboard*. August 1. <https://web.archive.org/web/20150801193208/http://octoboard.com/>.

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>.

Simondon, Gilbert. 1989. *Du Mode d’existence Des Objets Techniques*. Paris: Aubier. <https://philpapers.org/rec/SIMDMD-2>.

Star, Susan Leigh, and Karen Ruhleder. 1996. “Steps Toward an Ecology of Infrastructure: Design and Access for Large Information Spaces” 7 (1): 111–34.

Statista. 2017. “Number of Facebook Users Worldwide 2008-2017 | Statistic.” *Statista*. June 12. <https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/>.

Suchman, Lucy. 2006. *Human and Machine Reconfigurations: Plans and Situated Actions*. 2nd ed. Cambridge University Press.

Takhteyev, Yuri, and Andrew Hilts. 2010. *Investigating the Geography of Open Source Software Through GitHub*. <http://takhteyev.org/papers/Takhteyev-Hilts-2010.pdf>.

Thrift, Nigel, and Shaun French. 2002. “The Automatic Production of Space.” *Transactions of the Institute of British Geographers* 27 (3): 309–35. <http://onlinelibrary.wiley.com/doi/10.1111/1475-5661.00057/abstract>.

Thung, Ferdian, Tegawende F. Bissyande, David Lo, and Lingxiao Jiang. 2013. “Network Structure of Social Coding in Github.” In *Software Maintenance and Reengineering (Csmr), 2013 17th European Conference on*, 323–26. IEEE. <http://ieeexplore.ieee.org/abstract/document/6498480/>.

Van Dijck, José. 2013. “Facebook and the Engineering of Connectivity: A Multi-Layered Approach to Social Media Platforms.” *Convergence: The International Journal of Research into New Media Technologies* 19 (2): 141–55. <http://con.sagepub.com/content/early/2012/09/17/1354856512457548.abstract>.

Verran, Helen. 2012. “The Changing Lives of Measures and Values: From Centre Stage in the Fading ‘Disciplinary’ Society to Pervasive Background Instrument in the Emergent ‘Control’ Society.” *The Sociological Review* 59: 60–72. doi:[10.1111/j.1467-954X.2012.02059.x](https://doi.org/10.1111/j.1467-954X.2012.02059.x).

Wittgenstein, Ludwig. 2003. *Tractatus Logico-Philosophicus: Volume 123*. 2 edition. London & New York: Routledge.

Yarrow, David, and Matthias Kranke. 2016. “The Performativity of Sports Statistics: Towards a Research Agenda.” *Journal of Cultural Economy* 9 (5): 445–57. <http://www.tandfonline.com/doi/abs/10.1080/17530350.2016.1202856>.

1. Concepts of infrastructure (Star and Ruhleder [1996](#ref-Star_1996)), sociotechnological project (Latour [1996](#ref-Latour_1996)), global assemblage (Ong and Collier [2005](#ref-Ong_2005)), configuration (Suchman [2006](#ref-Suchman_2006)), interface (Marres, Gerlitz, and others [2015](#ref-Marres_2015)), machinic assemblage (Lazzarato [2014](#ref-Lazzarato_2014)), and more recently, 'stack' (Bratton [2016](#ref-Bratton_2016)) probe specific aspects of the highly mutable combinations of machines, signs, people, platforms, tools, processes of invention, practices of work, technical, geographical, social and energetic materialities comprising platforms. The ensemble reality of platforms remains, it seems to me, difficult to analytically conceptualise. [↑](#footnote-ref-2)
2. I don't address in any depth here the discourses of free and open software that were so vital and prominent in the constitution of social media platforms as a relational, connective sociality. See (Mackenzie [2006](#ref-Mackenzie_2006); Coleman [2012](#ref-Coleman_2012)) for discussion of these discourses. [↑](#footnote-ref-3)
3. Neither image -- the woman at work in a loft-conversion office or the press conference in China -- is arbitrary, again suggesting that enumeration is always also configurative in the sense of materialising imaginaries of wholeness, difference, inclusion, etc. Github was rocked in 2012 by allegations of sexism and mismanagement, particularly associated with one of the co-founders, who subsequently resigned and left Github (Github [2014](#ref-Github_2014)). In early 2015, Github itself was subject to a massive Denial of Service attack, emanating from China. Reports alleged that the DoS attack targeted pro-democracy repositories on Github. Unlike its regulation of other social media platforms, China does not block Chinese citizens' use of Github because Github is seen as economically and commercially important for the software industry (Marczak and Weaver [2015](#ref-Marczak_2015)). [↑](#footnote-ref-4)
4. In what follows, I use monospace font like Github when referring to entities named in the Github data. The typography highlights entities subject to enumeration by platform numbers. [↑](#footnote-ref-5)
5. The table results from the query select type, count(type) as events from [githubarchive:github.timeline] group by type order by events descending run on the GithubArchive public dataset hosted on the cloud computing platform, GoogleBigQuery. An archive purporting to contain all of the Github data 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](#ref-Grigorik_2012)). Grigorik, or igrigoriknot only published all the data in a cloud-based data store but transformed that data, whose formatting we have glimpsed above, into the flat tabular forms familiar in much statistical work, (Campbell-Kelly [2003](#ref-Campbell-Kelly_2003)) and made it available through Google's newly launched cloud computing service, GoogleBigQuery.The GoogleBigQuery copy of the Github public timeline data is updated hourly. In late 2015, the main GithubArchive timeline dataset on GoogleBigQuery was frozen. Data after that date flows into new datasets named by the month. e.g. <https://bigquery.cloud.google.com/table/githubarchive:month.201107> points to the data for July 2011. This partitioning of the data on a 'big data' platform such as GoogleBigQuery attests to the voluminous flows of events through Github. For the purposes of my argument, and to simplify code slightly, I only make use of the main Github timeline dataset covering 2012-2015. [↑](#footnote-ref-6)
6. Looking slightly more widely, 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 platform 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](#ref-Russell_2013)). Again, the propensity to apply network analysis approaches is widespread and endemic to the data itself, given the way that the event format is already implicitly framed by a network or 'social media' understanding. 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 et al. [2013](#ref-Thung_2013)). Like the many Github-hosted projects discussed above, they analyse sentiment (Guzman, Azócar, and Li [2014](#ref-Guzman_2014)), collaboration and productivity (Dabbish et al. [2012](#ref-Dabbish_2012)), and geography (Takhteyev and Hilts [2010](#ref-Takhteyev_2010)). [↑](#footnote-ref-7)
7. For the beginnings of discussion moving beyond the performative, see (Pellandini-Simányi [2016](#ref-Pellandini-Simanyi_2016); Cooper and Konings [2016](#ref-Cooper_2016)). In a post-performative vein, we might consider numbers such as prices, scores, rankings, and indexes that sociological and anthropological accounts have explored precisely in terms of such overflow modalities (see (Verran [2012](#ref-Verran_2012); Guyer [2014](#ref-Guyer_2014)) for instance). Numbers, or 'figures' in the numerical sense of the term, *configure* through enumeration, through seriation and through evocations of scale, limit and change. [↑](#footnote-ref-8)
8. Examining just the names of repositories in the Github API data, counting how often and when they appear in events begins to show how associations are configured. A host of methodological questions cluster around the question of counting names. For the purposes of this discussion, I leave aside the many sidetracks, dead-ends and plain failures associated with working on a several hundred gigabyte dataset. While we rely on repository names as the principal identifier in this paper, it would also be possible to count and re-count other identifiers or indexes, such as event IDs, timestamp, user names, etc. [↑](#footnote-ref-9)
9. This re-counting of platform numbers only includes the public repositories and actors on Github. An unknown but substantial number of repositories are private. In the light of Github's financial valuation at $US 2 billion in 2015, this number may be quite large (Gage [2015](#ref-Gage_2015)). [↑](#footnote-ref-10)
10. The ForkEvents are not included in the Figures 5 and 4 [↑](#footnote-ref-11)
11. The GoogleBigQuery query de-capitalises the repository names for the purposes of enumeration (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 individual events are distributed. But these highly frequented repositories, which only comprise a tiny percentage of the 29 million on Github, absorb many events. The query is SELECT lower(repository\_name) as repository\_name, count(lower(repository\_name)) as count FROM [githubarchive:github.timeline] group by repository\_name order by count desc LIMIT 1000 [↑](#footnote-ref-12)
12. The presence of eclipse on Github suggests another important analytical problem to consider when enumerating: how do know for a given setting or place that what we count belongs there? Sometime repositories on Github mention they are 'mirrors' of something else. Other times they may not. [↑](#footnote-ref-13)
13. For instance, a dotfile repository will typically contain the settings for a software developer's terminal windows, code editor, and 'shell' or command line interface (see the heavily forked [mathiasbynens/dotfiles](https://github.com/mathiasbynens/dotfiles) repository for typical contents (Bynens [2016](#ref-Bynens_2016))). The name dotfile refers to the fact that some operating systems hide from normal view file names beginning with the character . and configuration information is often, by convention, stored in such hidden files dotfile. The fact that many users will never see such dotfiles only highlights their relevance for developers organizing, interpreting and aligning machines, objects, devices and sub-ensembles with each other. We could look further for implicit configurative work in the timeline events. A query on the timeline dataset yields around ten million push events containing either a comment or message relating to 'config,' 'test' or 'build.' Similarly, a query for repositories that use no programming language results in around eleven million repositories. While null-code repositories vary greatly in their contents, many of them act as stores for configuration-related documentation, settings and data files. [↑](#footnote-ref-14)