What is an important event? 175 million events and counting

Now 195 million events!

Abstract

Even a moderate sized media platforms like Github.com, a 'social coding' site widely used by software developers, record millions of events. Such sites, with their millions of projects, participants, groups and organisations, can be understood as expressions of contemporary social life. But given such vast and complex forms of expression, how can social researchers get a feeling for what counts as an important event? Even if they have open access to the data, infrastructural, configurational, analytical, philosophical and financial problems make it hard to decide what counts as an important event amidst massive streaming data. Against the drive for immediacy and the allure of totality that characterises many engagements with data streams, the paper will describe our attempts to realise in numbers, graphics, code and language a feeling for important events on Github.

Overview of paper on Github

Github is a social media platform for software development. It claims to be the largest code repository 'on the planet.' Given that software is generally acknowledged as a pervasive component of our lives, could we say something about the world by viewing it from the perspective of Github?

This rather grand question invites some obvious and less obvious answers about what is happening:

1. **Participation through sharing:** Platform that is typical of digital media today – massive, complex and dynamic. It is 'social' in the sense that it puts a lot of weight on 'sharing,' in this case, of code and work on code. (For overview, see (Fish et al., 2011)).
2. **Recursive:** Platform that is typical of software cultures today – that is, it

operates on a model of publicness or openness – strangers do things together there, alongside groups or communities. This publicness is often recursive. That is, people work on software that changes the conditions of their own participation in publics and in culture more generally.

3. **Device-specific:** Platform that is a typical object of digital social research, digital sociology or digital STS in that it opens possibly new avenues of research in which we might see what is happening at different scales as it happens. Given the sheer diversity of projects, their sprawling geography, the sheer numbers of developers, platforms and software libraries, as well as the increasingly tangled digital economies and cultures, could this dataset provide a way of doing something like what Bruno Latour has recently asked: ‘let the agents produce a dynamics and collect the traces that their actions leave as they unfold so as to produce a rich data set’ (Latour et al., 2012) He asks: ‘is it possible to do justice to … common experience by shifting from prediction and simulation to description and data mining?' The common experience he refers to here is precisely that which is shared not in the sense of divided but held in common. This is because what happens on Github is recorded and published as data. But this data is specific to the Github platform. It is formatted in platform-specific ways.

As I will suggest, these different characteristics are not separate: the sharing practices of social media, the recursive reconstruction of one's own position of presence, and the challenges and potentials of device-specific data are all entwined. But the last one is the most important for my purposes. The way in which we respond to the device-specific data we encounter will shape what we think happens, how it happens, and the forms of publicness, the practices of participation, sharing and sociality associated with it. In other words, to put it very bluntly, I'm suggesting that what happens in the world are device-specific events, and that engaging with device-specificity is a way to grapple with the problems of scale, massiveness and complexity that vex much digital research.

I'm hoping you will be able to identify this last point in relation to debates about the intersection of big data and empirical social research. I don't have time to explore those debates in any detail, but they form the backdrop of what I'm talking about here. That is, on the one hand, we all have some degree of access superabundant, almost realtime data

on what people are doing. On the other hand, very well-funded and well-equipped teams of people try to make sense of what people are doing online, often using large data infrastructures and sophisticated modelling techniques. Machine learning, data mining and data science promise to make sense of what is happening, and to filter amidst the vast noise of contemporary cultures, what matters and what doesn't. But is digital social research the same as data science?

What we might do differently is specificity and relationality. 'Device-specific social research' is a term for this. I'm drawing quite heavily on Noortje Marres' work here, as well as critically responding to some parts of the STS – here I include Latour – that treat databases as smooth spaces. The point is that we can't make sense of any of the obvious trends or topics or issues around social processes unless we take into account their device-specific characteristics. This is an old point from STS work. But the new twist that Marres' supplies, and I'm attempting to follow here, is this: we can't make sense of device-specific patterns of events unless we grapple with the changing, formatted shape of the data that is both the fabric of events and a resource to help describe them. Marres and Weltevrede call this research 'live social research':

> Live social research, to summarise, is social research that seeks to render analytically productive the formatted, dynamic character of digital networked data. As such, this research practice endorses the dynamism or 'shape-shifting' of online data, turning these into a resource and an object of digital social research. 327

Rather than asking what is trending, or how is most social right now, live social addresses the liveliness or dynamism of the data itself as platforms, people and data sources change in relation to each other. This liveliness has a limited duration, but it is less a matter of prediction or modelling using the data, and more a matter of seeing how the data itself, in the forms and shapes it takes, helps us understand what is happening in and around the platform.

The rest of my talk walks through these points in more detail with examples. I've already stated the conclusion. The question is how to get there by navigating various forms of data, sources of information, using infrastructures and software tools. And along the way, what happens to the relation between **sharing, recursive** and **device-specific?**

The meta-question here is: does device-specific social research help us understand

the participatory practices and the recursive processes of software cultures?

A Software expresses the social: the world according to Github

1 The company itself – no managers, 66% remote, 12 million repositories

GitHub.com is one relatively small social media platform. It is basically a code repository for software developers and programmers. The company, like so many other Web 2.0 startups, is based in San Francisco. It started in 2008, ostensibly has no managers, and has around 300 employees, many of whom work remotely (> 65%). In 2012, it attracted $US100 million in venture capital (Hardy, 2012), a significant investment for a relatively small company. In many ways, Github exemplifies contemporanry open source ethos and business model. It is a social media platform, it is massively open source and the Github ethods of no-managers only teams exemplifies the agile forms of startup busineess found in their thousands in New York, London, Berlin, Shanghai, Melbourne, and probably Copenhagen (see TechCrunch for a listing).

2 The showcases – what they think matters to us – SLIDE

What Github presents as important are events, trends and topics. Like many other social media platforms, it promotes trending repositories and showcases repositories that represent different aspects of Github. Both are done in the interests of conveying what is important and what is happening now.

This is recent – previously no way to explore Github apart from searching for things; this set of 'showcases' is meant to make things easier.

3 Openssl – the heart bleed – example of how software matters – rough graphics

An example of what might be an important event – 2 SLIDES on this – Guardian and then openssl/openssl

B Sharing on Github

SLIDE OF GITHUB home

Obviously programmers have nearly always swapped and shared source code. They also

almost always borrow, cite, invoke or otherwise draw on existing bodies of code, even if it is only in the form of copying patterns or architectures. Writing code is nearly always closely tied up with reading code. Despite its aura of monolithic algorithmic force, code is highly *intertextual*, and this intertextuality or intersecting of inscriptions is both powerful and problematic. Many of the problems around code concern unruly forms of change and sharing. The ease of changing code creates problems: it breaks software systems or creates vulnerabilities.

In Github.com's own terms, the sprawling flow of texts on the platform attests to the power of 'social coding', and the *sharing* of code: 'GitHub is the best place to share code with friends, co-workers, classmates, and complete strangers. Over four million people use GitHub to build amazing things together' (https://github.com/about). Github showcases (https://github.com/showcases) this sharing and sociality. It encourages people to 'be social' (https://help.github.com/articles/be-social) by copying, watching, following and contributing code.

But sharing is a complicated process, and not automatic. It has different meanings, and practices. At a time when sharing was the modus operandi of media platforms in the way it is today, the anthropologist Ulf Hannerz more than twenty years ago wrote that 'there is nothing automatic about cultural sharing,' (Hannerz, 1992: 44) . Hannerz suggests that there are real complexities in the way that things are shared. His argument remains important, even as sharing is somewhat automated through devices such as git and platforms like Github. If Github.com is widely regarded as the exemplar of code sharing today, the ways in which that sharing is accomplished might also need to be understood through the processes of *non-sharing* on which it relies.

1 SCM - cvs, subversion, git

SLIDE/working through of using git

Developers take great efforts to order writing and reading practices around code. Revision control systems or version control systems, programming methodologies such as object oriented languages or extreme programming, or disciplines such as software engineering, are just some of the ways in which the seething textuality of code has been channelled and ordered. Code repositories are also one way in which this intertextuality or dependencies between pieces of code have been managed.

Github pivots on a single piece of software *git* and a series of seemingly trivial

operations performed most basically by typing commands of form `git *do_something*'. Although they are somewhat more complicated than tweeting, the basic operations on git and github are simple. (There are many tutorials on git and Github.) A code repository is established using the command:

git init

Files are created or added to the repository:

git add some_files

Some changes are made to the files – adding new lines of text for instance to a text file and then added to the repository like so:

git commit files

The files are usually 'committed' to the repository along with a message. Prior to Github all of this might be happening a single laptop or in a small team of developers. Github adds a bit to git in order to make Git-Hub:

git add remote origin https://github.com/user/repository_name

The repository can be mirrored at github as long as you have joined Github. Any new commits can be sent to Github like this:

git push

And now anyone can copy the entire repository or just some branch of it to their machine using this line:

git clone https://github.com/user/repository_name

I am not going to say too much more about this seemingly small piece of software *git,* written by perhaps the very famous programmer Linus Torvalds, who is still the key developer on the Linux kernel project. Git embodies a set of values and propositions about coding and other practices that valorises distribution, de-centralisation, and a certain vision of the coherence of code as text.

Unlike some other version control systems, or revision control systems, git does not discourage different versions of code. In some ways it encourages them, and it allows different versions to coexist without much problem.

While there are more sophisticated ways of using git and graphical user interfaces for using git, this simple command-driven approach matters a lot in git and github. It

connotes a directness and transparency that matters a lot to many software developers.

This directness or immediacy is not necessarily about sharing code, but about distributing work so that many people can work together in predictable ways. There is a de-centralisation and distribution of work, but this does not necessarily mean anything is shared, apart from the commitment to a style of work.

2 The github mechanisms – fork-pull requests; issues, watching –

SLIDE OF OPENSSL

We can see this fundamental tension in Github by looking at the same repository in different ways. What happens when ordering practices of integration, merging, accumulating and distributing associated with code encounter the forms of interaction and exchange associated with social media?

The Github view of a repository looks very different to the git view. There is a mapping between them, but Github has 'socialised' it with all the social media buttons of watching, starring, following, as well as some specific ones that rely on git, such as **fork**.

In a public repository on Github, anyone can clone or 'pull' the repository. Not everyone can make changes to it. A stranger to repository can propose changes to a repository by cloning/forking it, and then creating a PullRequest that invites the repository owner to accept the suggested changes from the cloned repository. This process of forking-pullrequest-pull is often regarded as the epitome of the social life of code sharing on Github.

3 git API – make everything available so people use it more – LIVE?

A distributed revision control system such as *git* generates many records as people make changes to code and *commit* those changes to repositories. Repositories are read and written in overlapping patterns on Github at a high speed.

These events can be read from the Github platform api https://api.github.com/events, along with many other types of attributes associated with users, organizations, repositories, the code itself, etc. Some of this data is limited. For instance, the publically timeline data only really goes back a few minutes.

These records are themselves shared by the Github platform. Every hour, thousands of events occur. Between 10-11am, 10 January 2013, the following events occurred:

PushEvent	4248
IssueCommentEvent	1104
CreateEvent	935
IssuesEvent	770
WatchEvent	758
PullRequestEvent	382
ForkEvent	328
FollowEvent	193
GollumEvent	153
PullRequestReviewCommentEvent	119
CommitCommentEvent	86
DeleteEvent	83
MemberEvent	70
GistEvent	30
PublicEvent	2
DownloadEvent	1

Or one day – 1 October 2012, there were around 155k events, of which around

type	events
PushEvent	67376
CreateEvent	29847
IssueCommentEvent	12185
WatchEvent	12148
IssuesEvent	7466
PullRequestEvent	4914
ForkEvent	4626
GistEvent	4487
FollowEvent	3125
GollumEvent	2681
CommitCommentEvent	1403
PullRequestReviewCommentEvent	1213
MemberEvent	962
DeleteEvent	920
DownloadEvent	587
PublicEvent	175

SELECT type, count(type) as events from githubarchive:github.timeline where PARSE_UTC_USEC(created_at) < PARSE_UTC_USEC('2012-10-02 00:00:00') and PARSE_UTC_USEC(created_at) > PARSE_UTC_USEC('2012-10-01 00:00:00') group by type order by events desc;

These events are public, in the sense that anyone can see them. But we can see how the forms of sharing and participation in sharing **reformat the git events and operations** I have just been discussing. For instance, we can see that some of them are Pushes – content contributions – this is always common, but **many are different kinds of watching, following, copying or commenting** type events.

The events are mainly PushEvents where people put code into repositories. But many of them are meta-events, such as making comments on code, creating new repositories, making copies of existing repositories or starting to watch repositories. So we have these events occurring in continuously varying quantities.

4 But also keep total numbers invisible – QUOTE FROM QUORA

5 githubarchive – show that everything is available – SLIDE

6 Google BigQuery githubarchive – show what people do with data when available – how many events are there? – live query on this? 'SELECT count(type) as event_count FROM [githubarchive:github.timeline] LIMIT 1000'

C Recursive – R

A second point of departure comes from existing work on free and open software. A huge amount of work has been done on free and open software. I'm drawing on Chris Kelty's notion of a recursive public to a certain extent:

> A recursive public is a public that is vitally concerned with the material and practical maintenance and modification of the technical, legal, practical, and conceptual means of its own existence as a public; it is a collective independent of other forms of constituted power and is capable of speaking to existing forms of power through the production of actually existing alternatives. (Kelty, 2008: 3)

Kelty developed this notion as a way of engaging with certain groups such as Free Software, open access science, and today open data initiatives that concern themselves with the 'radical technological modifiability of their own terms of existence' (3). All of the software we can see on Github is free and open. Much of it is concerned with modifying its own terms of existence, but **they are recursive in somewhat different ways.**

The core contents of Github are *repositories* or 'repos.' At the moment (March 2014), Github itself says there are over 12 million repositories on Github (https://github.com/about) . This number needs to be treated carefully. On the one hand, this number of software projects would suggest an enormous amount of software is being made. But these repositories have very different lives, which we cannot see even by counting events using the social categories.

SHOW SLIDE OF POWER LAW

1 Repos on Github – the count overall - 'SELECT count(*) as repo_count FROM [metacommunities:github_proper.repo_list]'

2 Many repos about git itself – GitRec, the founder repos, rails, etc

Direct exemplification of recursive idea are the founder repos

repository_url	repository_name	repository_owner	event_count	repository_created_at
https://github.com/mojombo/grit	grit	mojombo	2314	2007-10-29 14:37:16
https://github.com/rubinius/rubinius	rubinius	rubinius	16514	2008-01-12 16:46:52
https://github.com/mojombo/god	god	mojombo	1817	2008-01-13 05:16:23
https://github.com/technoweenie/restful-authentication	restful-authentication	technoweenie	243	2008-01-14 14:44:23
https://github.com/technoweenie/attachment_fu	attachment_fu	technoweenie	205	2008-01-14 14:51:56
https://github.com/macournoyer/thin	thin	macournoyer	2184	2008-01-19 05:39:04
https://github.com/bmizerany/sinatra	sinatra	bmizerany	289	2008-01-24 04:49:54
https://github.com/schacon/ruby-git	ruby-git	schacon	1058	2008-01-27 17:23:23

https://github.com/abhay/calais	calais	abhay	152	2008-01-29 04:10:12
https://github.com/mojombo/chronic	chronic	mojombo	2655	2008-01-29 06:48:49
https://github.com/sr/git-wiki	git-wiki	sr	294	2008-01-29 20:39:42
https://github.com/drnic/ruby-on-rails-tmbundle	ruby-on-rails-tmbundle	drnic	199	2008-01-30 23:20:38
https://github.com/grempe/amazon-ec2	amazon-ec2	grempe	230	2008-02-01 04:27:00
https://github.com/evilchelu/braid	braid	evilchelu	115	2008-02-03 21:14:46
https://github.com/engineyard/eycap	eycap	engineyard	267	2008-02-04 11:37:14
https://github.com/sevenwire/forgery	forgery	sevenwire	473	2008-02-06 22:33:31
https://github.com/collectiveidea/acts_as_audited	acts_as_audited	collectiveidea	369	2008-02-09 16:14:05
https://github.com/collectiveidea	audited	collectiveidea	1087	2008-02-09 16:14:05

/audited				
https://github.com/collectiveidea/graticule	graticule	collectiveidea	121	2008-02-09 17:23:49
https://github.com/collectiveidea/tinder	tinder	collectiveidea	422	2008-02-09 18:18:23
https://github.com/adamwiggins/rush	rush	adamwiggins	332	2008-02-10 20:21:42
https://github.com/defunkt/facebox	facebox	defunkt	1056	2008-02-11 22:49:27
https://github.com/nex3/haml	haml	nex3	741	2008-02-11 22:55:26
https://github.com/haml/haml	haml	haml	4835	2008-02-11 22:55:26
https://github.com/jnunemaker/twitter	twitter	jnunemaker	2004	2008-02-14 02:20:50
https://github.com/sferik/twitter	twitter	sferik	4976	2008-02-14 02:20:50
https://github.com/delynn/userstamp	userstamp	delynn	190	2008-02-14 07:15:27
https://github.com/toretore/barb	barby	toretore	446	2008-02-17

y				13:40:24
https://github.com/tobi/delayed_job	delayed_job	tobi	801	2008-02-17 21:00:03
https://github.com/rsl/stringex	stringex	rsl	1683	2008-02-19 15:31:07
https://github.com/collectiveidea/awesome_nested_set	awesome_nested_set	collectiveidea	2008	2008-02-21 16:03:25
https://github.com/stevedekorte/io	io	stevedekorte	1409	2008-02-22 08:41:30
https://github.com/churchio/onebody	onebody	churchio	116	2008-02-23 15:53:26
https://github.com/seven1m/onebody	onebody	seven1m	119	2008-02-23 15:53:26
https://github.com/mislav/will_paginate	will_paginate	mislav	2947	2008-02-25 20:21:40
https://github.com/lsegal/yard	yard	lsegal	3514	2008-02-26 00:01:52
https://github.com/drnic/ruby-tmbundle	ruby-tmbundle	drnic	112	2008-02-26 00:49:22

https://github.com/crafterm/sprinkle	sprinkle	crafterm	808	2008-02-26 12:58:26
https://github.com/sprinkle-tool/sprinkle	sprinkle	sprinkle-tool	1232	2008-02-26 12:58:26
https://github.com/ctran/annotate_models	annotate_models	ctran	1787	2008-02-27 18:04:38

Another way of looking at this would be to look at all the repos that have github in their name – approximately **125k repos have github in their**

3 Many repos are tests or config attempts – 'test' Individual or tools: .dot files, editors, programming languages or tools:

'SELECT top(repository_url), count(*) as events from githubarchive:github.timeline' has eclipse

4 Many repos are forks of others: 'SELECT count(*) as fork_count FROM [metacommunities:github_proper.repo_list] where fork=1;'

5 Mirroring – many uses of Github as mirror

Many major repositories are clones or mirrors of code held elsewhere. As it became more popular, Github also became a place to put things made elsewhere. We need to decide in looking at a given repository whether we are seeing something made elsewhere and put on Github to advertise and publicise it, or seeing something that exists primarily on Github. That is, is Github being used as a download site, a cleaned-up version of Pirate Bay or MegaUpload, a place that is used to *mirror elsewhere*? Second, even if we decide that the repository is indigeneous to Github, that is, it grew there, further questions arise. Many repositories are derivatives of other repositories. Many repositories are *forks:* repositories that copy or mirror other repositories and then vary them slightly, reconfiguring certain aspects. This process of forking repositories is a key event in the social life of Github platform. It is something

like 'following' on Twitter or 'liking' on Facebook, but with potentially much more open-ended consequences. A fork can become more important than the original repository it copied. Forks are sometimes re-incorporated into the original or parent repositories that came from. Forks and mirroring complicate the flow of code on github. Do forks increase the importance of a repository or do they dilute it? That question remains open.

6 Colonising: organisations colonise it – BBC; social media – Facebook, twitter, google

In addition, repositories on Github do not stand in isolation. They are grouped around other actors on various scales. For instance, there are around 80,000 organizations on Github, alongside the roughly hundreds of thousands of individual 'actors' . Like the repositories themselves, the organisations can be indigenous to or growing on Github, or can be seen as colonising or immigrating. While existing organisations sometimes have repositories, these repositories may not be very significant organisationally or not even a coherent expression. For instance, at first glance the organisation BBC has only has 6 repos (https://github.com/BBC) and they are not updated very often. But BBC has many different organisations on the platform: BBCNews, BBCFrameworks (https://github.com/bbc-frameworks), BBC R&D (https://github.com/bbcrd), BBC Data (https://github.com/BBC-Data) BBC Future Media or BBC World Wide, etc. Repos are distributed more widely on Github. Large organisations like the BBC are present in complicated ways on Github. Sometimes their repos contain work specific to their activities, but very often their repos are clones or forks of repositories outside the organisation. This plurality of relations suggests that organisations inhabit the platform in order to leverage the work of the github metacommunity. (Something similar holds for *The New York Times*).

repository_organization	repository_count
bbcrd	66
BBC-News	51
bbcarchdev	21
bbcf	19
BBC-Knowlearn	19
fablabbcn	15

BBCVisualJournalism	13
MapBBCode	12
BBC	12
bbcrew	9
bbc-fm-nk-core-eng	6
BBCLab	6
bbcsnippets	6
bbc-sport	3
SnabbCo	2
BBCConnectedStudio	2
labbcb	1
BBC-Childrens	1
bbbcddd	1
bbc-test	1
githubbcn	1
bbccoin	1
BBCGlobalNews	1
bbcwwtechnology	1
bbch	1
BBC-Data	1
BBCWW	1
bbc-degrees	1
BBC-Location-Services	1
BBCMS	1
BBCRMDEVCommunity	1

Or to take another example, the social media platform Facebook has hundreds of

repository on Github.

repository_organization	repository_count
facebook	140
Google	185
Twitter	135

In many ways this is strange because Facebook as a social media platform would normally not put part of itself on another platform. But the way that is developers coordinate their work Github, and the profile of their many repositories there can tell us something about the organisation of Facebook itself. A part of Facebook and perhaps an important part is on Github. Almost all social media platforms and media companies show something similar here.

7 Key repos – linux, android, apache, mozilla,

Some of these repositories are highly significant in the world of software for cultural and commercial reasons (e.g. the Linux kernel - https://github.com/torvalds/linux or the open source firmware for Android devices, CyanogenMode, https://github.com/CyanogenMod). Others have broader significance politically (e.g. the Obama administrations 'We the People' petitioning system is at https://github.com/WhiteHouse;) or in terms of contemporary media politics (e.g. Django and its connections to the growth of data journalism at *The Washington Post* https://github.com/django/django). Many have importance to particular web platforms or media (for instance, core components of Mozilla Foundation's Firefox web browser code https://github.com/mozilla/gecko-projects). Undoubtedly, many commercially important software projects rely on github and its repositories. For instance, heavily used pieces of information infrastructure are made and worked in Github (e.g. the noSQL database MongoDb https://github.com/mongodb/mongo; or the Apache Foundation's CouchDb https://github.com/apache/couchdb). Others are major repositories for scientific or engineering projects. Nearly all of the 3D printing software is hosted on Github (https://github.com/josefprusa/PrusaMendel). Important scientific and data wrangling packages such as NumPy (https://github.com/numpy/numpy) or visualization packages such as ggplot2 (https://github.com/hadley/ggplot2) can be found there. We also see the tools for configuring and managing large collections of servers or

information infrastructures (e.g. https://github.com/puppetlabs/puppet). And of course, an enormous number of entertainment related projects such as Minecrafts mods (https://github.com/Bukkit/Bukkit).

8 Less well-known but important – django, jquery, couchdb, mongodb, bootstrap, nodejs

9 Expansive: websites/blogs done using Git; Git as a platform for writing/publishing docs, laws, policies, plans, lists

Many Github repositories aren't actually software repositories. People use the repository mechanisms, and the 'source code management' systems of git to write and maintina blogs or websites rather than code repositories. That is, people use the git mechanism as a way of coordinating writing on websites or blogs. So many repositories are not about software. They are part of the blogosphere or Web. In addition, the popularity of Github as a way of storing and coordinating work by many people on shared texts has attracted other users who are not developing software: how-to guides, metadata on the Tate's art collection (https://github.com/tategallery/collection), the White House's open data policy (http://project-open-data.github.io/policy-memo), the US Pirate Party's efforts to transparently archive their own processes (http://uspirates.github.io/), the complete genome of Kenneth Reitz (https://github.com/kennethreitz/genome), collections of maps (https://github.com/djaiss/mapsicon), legal documents, recipes, books, and blogs are just some of the diversifying use-cases now found in repositories on Github.

D Device-specific social research

The data I have been using mainly comes not from Github itself. Its APIs are too limited. These events can be read from the Github api https://api.github.com/events very readily, along with many other types of attributes associated with users, organizations, repositories, the code itself, etc. Some of this data is limited. For instance, the publically timeline data only really goes back a few minutes. But developers have so much interest in this data that is it is archived on its own site http://www.githubarchive.org/ and also loaded every hour onto Google's cloud computing service called BigQuery https://bigquery.cloud.google.com/table/githubarchive:github.timeline.

Ironically, other non-social data on the github.com API, for instance, data on git 'commits' to repositories https://api.github.com/repos/torvalds/linux/commits is completely unlimited, and has to be, otherwise the git mechanism would begin to break

down. And it is this non-social data that allows Github to actually work as a code repository.

This tension between two different modes of data is symptomatic of the tensions in sharing and non-sharing often encountered today. The timeline data is almost impossible to follow or keep hold of without turning to large scale infrastructures such as BigQuery. That brings new complications in practically dealing with the data. In terms of the radical empiricist account of experience I'm pursuing here, it suggests that we should expect events to hang together by themselves, but only by virtue of intermediaries.

This tension goes to the core of the question of how we research anything about what happens on these platforms. In contrast to the massive turn to data analytics as a direct line to the happening of the social, I'm following Noortje Marres and Esther Weltevrede suggestion about the need to focus on the devices through which data on platforms is produced. They frame their point in terms of web scraping here:

> Scraping, we propose, makes it possible to render traffic between the object and process of social research analytically productive. It enables a form of 'real-time' social research, in which the formats and life cycles of online data may lend structure to the analytic objects and findings of social research. (Marres and Weltevrede, 2013: 3)

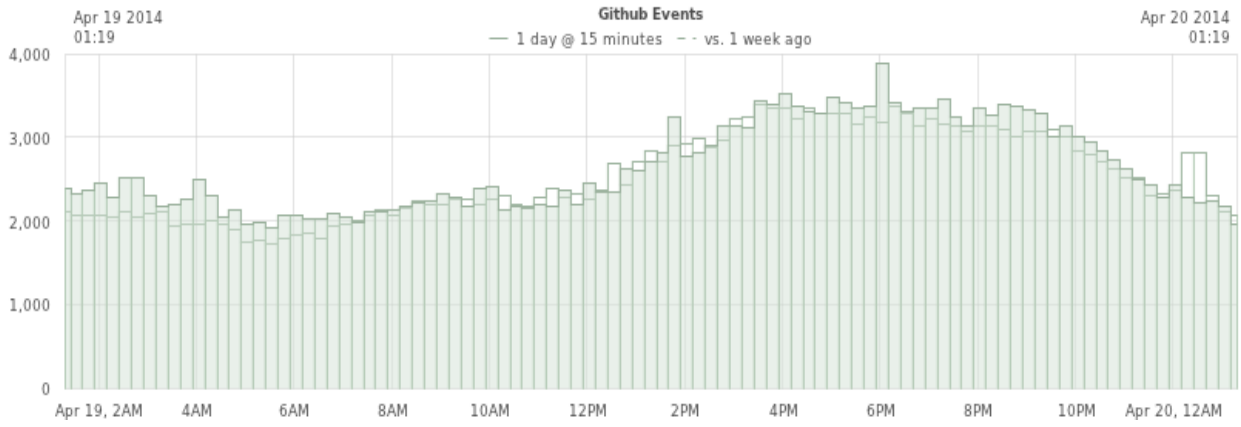
Marres and Weltevrede describe how to engage with online data as a more live or real-time social than the data itself might offer. I think this a really useful analytic addition to the much more flat data analytic approaches that we often see around social media platforms.

1 What is shared: the platform or events?

So if something happens in relation to this shift to sharing, what is it? While Github does not share its non-sharing numbers publically, Github does distribute huge amount of data in the form of 'timeline' data and in the form of Application Programmer Interface (API) data. Like many social media platforms, these data data feeds are not meant for social scientists as such. They are meant to be used by software developers to create new services, applications or apps that will bring more people and traffic to the

social platform. This data is often close to realtime, although limited in various ways (see (Uprichard, 2012) for an analysis of this). But something more is going on with the Github data. Since 2012, a massive archive of events on github has been updated hourly at both GithubArchive.org and on Google's BigQuery cloud analytics platform (https://bigquery.cloud.google.com/table/githubarchive:github.timeline).

2 Github visualises itself – SLIDE OF SCREENSHOT



3 Time limits in the API

4 Closing of the global search –

5 The time limits in the githubarchive:

> Open-source developers all over the world are working on millions of projects: writing code & documentation, fixing & submitting bugs, and so forth. GitHub Archive is a project to **record** the public GitHub timeline, **archive it**, and **make it easily accessible** for further analysis (Grigorik, 2012)

writes Ilya Grigorik, a 'Developer Advocate @ Google, working on everything web performance related: protocols, standards, browser performance' according to his Twitter feed. Much more detail on how to access and use this data can be found on Grigorik's in one of Grigorik's own Github repositories (https://github.com/igrigorik/githubarchive.org/tree/master/bigquery). For instance, the approximately 200 columns in the dataset are described in the 'schema.js' document there (https://github.com/igrigorik/githubarchive.org/blob/master/bigquery/schema.js). Each row of the dataset is one timestamped event. So, to take an almost random example, the query

'SELECT created_at, type, actor, repository_name, url FROM [githubarchive:github.timeline] LIMIT 50'

yields:

Query Results 11:19am, 31 Mar 2014

Row	created_at	type	actor	repository_name	url
1	2014-03-15 09:04:43	PushEvent	vvakame	DefinitelyTyped	https://github.com/borisyankov/DefinitelyTyped/compare/3d3832de3e...93a2313f62
2	2014-03-15 09:04:42	PushEvent	gizmomogwai	plist	https://github.com/gizmomogwai/plist/compare/12eb82d283...83ad8b5f26
3	2014-03-15 09:04:48	IssueCommentEvent	ben-lin	node.inflection	https://github.com/dreamerslab/node.inflection/issues/16#issuecomment-37721104
4	2014-03-15 09:04:48	WatchEvent	cbmd	mongofill	https://github.com/koubas/mongofill
5	2014-03-15 09:04:48	WatchEvent	svett	molokai	https://github.com/tomasr/molokai
6	2014-03-15 09:04:47	GollumEvent	swordray	wiki	https://github.com/ruby-china/wiki/wiki/RubyGems
7	2014-03-15 09:04:46	PushEvent	elfet	purephp	https://github.com/elfet/purephp/compare/dc24b70f00...c94c524e76
8	2014-03-15 09:04:46	PullRequestEvent	nikkypx	rets_data	https://github.com/arcticleo/rets_data/pull/2
9	2014-03-15 09:04:46	WatchEvent	mjaneczek	google-authenticator	https://github.com/jaredonline/google-authenticator
10	2014-03-15 09:04:53	PushEvent	micmath	Rye	https://github.com/micmath/Rye/compare/69c02b1aea...7885ea9e36
11	2014-03-15 09:04:53	PushEvent	fitret	daigon	https://github.com/fitret/daigon/compare/8897fa9404...77e5f48276
12	2014-03-15 09:04:53	WatchEvent	joeyates	spork	https://github.com/sporkrb/spork
13	2014-03-15 09:04:53	CreateEvent	grcnva	mcs-chatboard	https://github.com/grcnva/mcs-chatboard
14	2014-03-15 09:04:51	PushEvent	albatrossen	bungeebouncer	https://github.com/albatrossen/bungeebouncer/compare/57a3771590...7794ffb531
15	2014-03-15 09:04:51	PushEvent	catalinstanciu	ChessEngine-Xboard-cpp	https://github.com/catalinstanciu/ChessEngine-Xboard-cpp/compare/9bedce1744...8b98d2ff45

Query Table

First < Prev Rows 1-15 of 50 Next > Last

4	https://github.com/jonathanstowell/My-Application-Framework	true	2011-11-03 16:56:16	true	Generic components that are useful across many scenarios (Mainly Web Foc
5	https://github.com/yehster/openemr	false	2012-01-13 19:55:08	false	Mirror of official OpenEMR Sourceforge repository

I don't think this is a meaningful query, but we can see events occurring second by

second. The events of various types – PushEvent, IssueCommentEvent, WatchEvent, PullRequestEvent, CreateEvent, GollumEvent – performed by various actors with mostly unpronounceable names feed into repositories with often quite generic names – 'wiki', 'google-authenticator', 'mcs-chatboard', 'plist', but also odd sounding repositories such as 'bungeebouncer' (prevents impersonators in chats sessions connected to Minecraft servers connected together using Bungeecord software) or 'spork' ('Tim Harper‘s implementation of test server (similar to the script/spec_server provided by rspec-rails), except rather than using the Rails constant unloading to reload your files, it forks a copy of the server each time you run your tests' http://spork.rubyforge.org/).

6 Duplicates in the BigQuery data – SLIDE of this

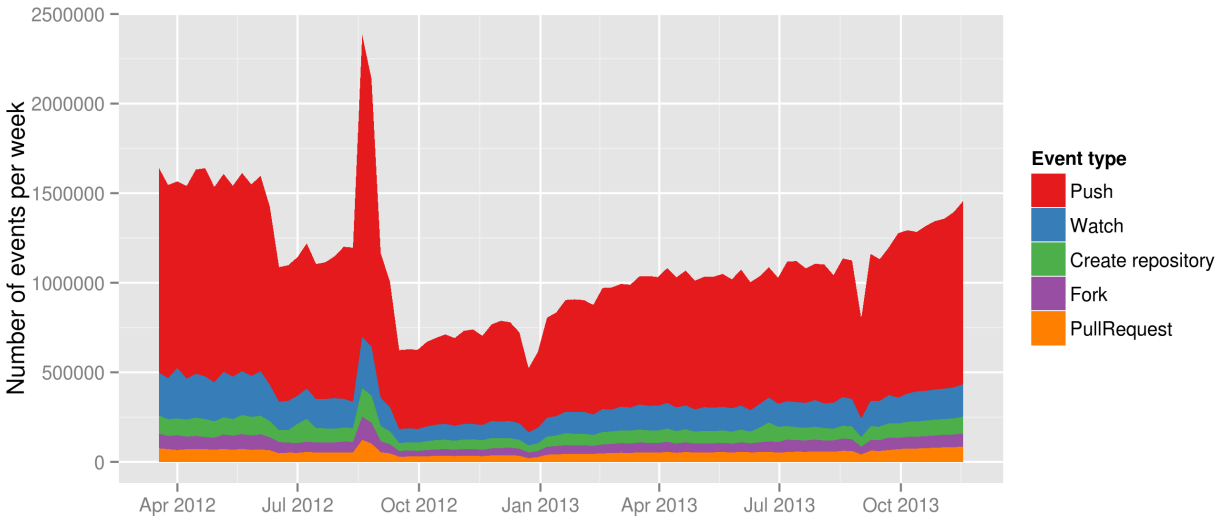
We signed on to BigQuery to access the timeline. The timeline is around 112Gb, and is, as I mentioned already, updated hourly. Our intention was to track the movement of coding practices between different software projects to gain some sense of the deep sharing in the sense of holding in common being done today.

We should not BigQuery is not the only way this data is shared. We could have downloaded it all from githubarchive.org and stored it in our own databases. A command like this:

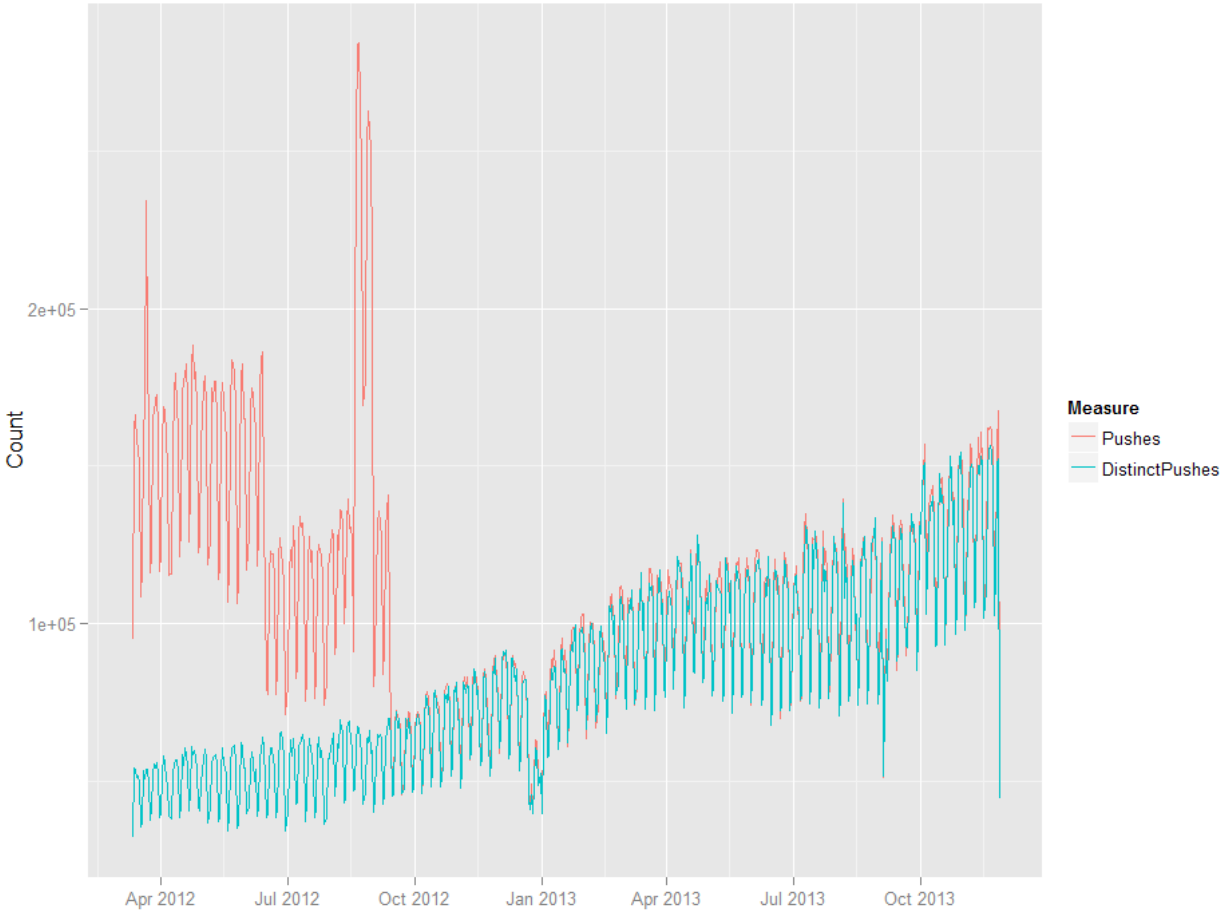
```
wget http://data.githubarchive.org/2011-04-{01..30}-{0..23}.json.gz
```

brings down all the hourly data files for 1-30 April 2011. But in the spirit of contemporary data analytics and its increasingly cloud-centric infrastructures, we decided to use Google's BigQuery system just as competitors in the Github data competition had.

But the power of highend data analytics comes at a cost. For instance, if we just trusted what is in the timeline dataset on BigQuery, we might think that something amazing was happening on Github in 2012.



Not only was there more activity on Github in April 2012 than 2014, there was a massive spike in activity during September 2012. This is completely counter to normal trends in growth. But a quick look at the data around September shows there is heavy duplication of events recorded on the timeline. Something happened here in the pipeline between Github and githubarchive.org and the bigquery.cloud.google.com.



There are couple of possible responses to this spike. We could say the data is very messy data and needs to be cleaned up. Yes, that is possible, but it is quite hard to actually eliminate duplicate events in the timeline dataset. Another response is to see this dirtiness as part of the sharing, and therefore not something to be cleaned away, but to be explored carefully. Here I am suggesting that we should see datasets like this and databases more generally not as as inert objects but as open, processual and emergent forms with many of the attributes and propensities of human informants that we might more typically work with in ethnographic research. Just as it can be hard to get to know human informants and to establish good dialogues with them, databases and datasets need to be treated less automatically if we want to make sense of what is happening in them.

7 The events types on the timeline and how they do not map to git events: tensions between git mechanisms and github mechanisms

But this is just a standard problem in work with data – the need to de-duplicate and clean data is always there. There is a greater difficulty in that events on the timeline do not always map well to the git events that actually add to code. That mapping is difficult maintain because Github tries to turn work onto code repositories into a social media style form of sharing participation. This means that it can be very difficult to track what happens because the social media event structure tends to cover over the coding event structure.

8 The big cost of query – contacted by Google Marketing team; drop of 80% in costs announced March 2014 – SLIDE of this

Much of our work with the Github event timeline has involved trying to extract from the social media event structures the more substantial events – those with longer duration, those that go further.

Conclusion

Github is interesting I have been suggesting because it combines a couple of different symptomatic features.

Given its concern with practices of making and distributing software, the ways of making and working taking shape act as models or exemplars of sharing and coordinated work. Actually, I have been suggesting that there are blockages and stoppages in trying to map coding work onto social media. The kinds of participation are the not the same. Coding is not necessarily social. Much of the e**mphemerality, and disconnection between repositories suggests that sharing is not the fundamental social practice** that social media platforms sometimes imagine. This means that sharing-based accounts of events or what happens might miss something. I guess OpenSSL and the HeartBleed problem illustrate this disconnect. But more fundamentally, the difficulty of mapping git events to the social media style events

One of the most powerful and well-developed conceptualisations of contemporary open source software is also slightly complicated by what I have been saying. The recursive publics of code, the groups who modify their own conditions of existence, are **pervasively present and expanding** on Github. Organisations of many different kinds

adopt this modus operandi, but at the cost of an **extra level of recursion**. The platform on which they remake their own conditions of existence is remade.

Finally, given the way in which software imbues our worlds with structures and relations of control and interaction, the pattern of practices on Github matters. It tells us something about what is happening and often about to happen in loaded parts of the world – science, business, etc. **The vastness of Github is like a monadic view on the world**. As Latour, Jensen, Venturini and ?? (Latour et al., 2012) suggest, we can build social theories by following the chains of links that compose the fabric of the social. Database or dataset navigation suggests a different kind of social theory. But I am somewhat sceptical about the smoothness of this navigation. I have been seeking to demonstrate and corroborate something of what Marres and Weltevrede write when they suggest that we 'render traffic between the object and process of social research analytically productive' (Marres and Weltevrede, 2013: 3). The particular forms of traffic I have focused on concern how data moves between different platforms. Marilyn Strathern writes that there needs to be 'alterity to structure the moment of validation. .. the producers of creations need witnesses to confirm their value' (Strathern, 2006, p. 23). This interplay between platforms and creations suggest such moments of validation.

Acknowledgments

Did I mention that this research is funded under a 'Google Data Analytics' research-program?

References

Grigorik I (2012) GitHub Archive. Available from: http://www.githubarchive.org/ (accessed 31 March 2014).

Hannerz U (1992) *Cultural complexity : studies in the social organization of meaning*. New York ; Oxford: Columbia University Press.

Hardy Q (2012) Dreams of ‘Open’ Everything. *Bits Blog*, Available from: http://bits.blogs.nytimes.com/2012/12/28/github-has-big-dreams-for-open-source-software-and-more/ (accessed 27 March 2014).

Kelty CM (2008) *Two bits : the cultural significance of free software*. Durham: Duke

University Press.

Latour B, Jensen P, Venturini T, et al. (2012) The Whole is Always Smaller than its Parts. How Digital Navigation May Modify Social Theory. *British Journal of Sociology*, 63(4), 590–615.

Marres N and Weltevrede E (2013) SCRAPING THE SOCIAL? Issues in live social research. *Journal of Cultural Economy*, (ahead-of-print), 1–23.

Uprichard E (2012) Being stuck in (live) time: the sticky sociological imagination. *Sociological Review*, 60, 124–138.