# Infrastructures in name only? Identifying effects of depth and scale

Adrian Mackenzie, Sociology Department, Lancaster University.

a.mackenzie@lancaster.ac.uk

Recently released NSA documents revealed the existence of FOXSCORE, a massive database that bugs traffic from major internet exchanges. General Keith Alexander, Director of the National Security Agency, assured the public that the program is rubber stamped by a secret court. (http://divergentdave.github.io/NSA-O-Matic/)

Recently released NSA documents revealed the existence of WAGONWIND, a USB hardware host tap that deanonymizes communications satellites. An anonymous administration source assured the public that the program would only be used in the event of a national emergency. (http://divergentdave.github.io/NSA-O-Matic/)

NSA-O-Matic is a piece of software that generates *names* of fictional US NSA (National Security Agency) surveillance systems by compounding names of actually existing devices, protocols and standards. We know from the files released by Edward Snowden something of the existence of sprawling cyber-security systems such as XKEY, TREASUREMAP, SKYNET and so forth (CJFE 2015), all of which might be broadly understood as knowledge infrastructures. The Snowden revelations, in all their confusing variety, attest to a combinatorial expansion. The humour of NSA-O-Matic derives partly from the disparity between the existence of a seemingly technically far-fetched juxtapositions -- USB hardware that deanonymizes communications satellites? -- and our still somewhat inchoate and stunned awareness of tendrils growing into every aspect of contemporary communications (wireless routers, Skype conversations, network hubs, mobile phone conversations, social media, email messages, etc.). The public assurances extracted from U.S. government figures such as General Keith Alexander or Senator Dianne Feinstein only emphasise the threadbare democratic mandate for the existence of such systems.

The combinatorial expansion is not unique to the NSA or government intelligence agencies of a similar ilk. The sheer variety of channels, devices, standards, modalities, platforms, scales of operation and levels of interdependency challenges knowledge of infrastructure. At least in relation to networked communication and knowledge, our sense of scale and opacity owes much to names in combination. NSA-O-Matic spoofs putative infrastructures. It compounds names to evoke opacity and depth. While important efforts are being made to map the connections between NSA 'programs' named in the Snowden documents and specific communications and computing devices that many of us use daily (Guardian News 2015), the sheer litany of systems, platforms and elements that surface in these documents defy easy analysis. Yet this abundance imbues infrastructures with much of their structural depth, flexibility, power and fragility. If we stay close to the combinatorial play of NSA-O-Matic, we might apprehend something of the unfurling, unstable opacity of contemporary infrastructures. The names of constituent fragments may attest to their splintering (Graham and Marvin 2001), also to their naturalisation (Geoffrey C Bowker and Star 1999, 326) but also to their existence in time.

All structures, no matter how subterranean, act semiotically. Rusted bolts, cracked concrete plinths, the stanchions, bars, girders, plugs, cables, stone, steel, glass, rubber, oil, plastic, electricity, gas and springs participate in semiosis. They function as indexes for infrastructural interpretants: rust suggests moisture, cloudy or black oil suggests wear, a buzzing indicates something loose, etc. On accompanying diagrams, labels name components, processes, flows and structures. At least for their interpretants, *semiosis* affects what counts as infrastructure.[[1]](#endnote-2) If, like NSA-O-Matic, we apprehend infrastructures in name only, how do they appear to us? Somewhat inverse to tendencies to see depth and scale as the underpinning power dynamic, many contemporary information infrastructures proliferate names. While we might approach contemporary instances in terms of the growth of forms such as 'platforms' (Gillespie 2010) or programmed or protocol control (Beniger 1986; Galloway 2004), their aggregate consistency and variety owes much to the named entities that thread through many different media, communication, administrative and operational practices. Naming practices or *nominalisation of infrastructure,* then, might offer a viable way of navigating the dense patterns of code, protocols, standards, devices, statements and operations that thicken people's infrastructural practices.

Mapping infrastructures in names only

Where can nominalisation of infrastructure be studied? Software development, maintenance, repair and deployment entails the designating and manipulating names. All code inhabits name spaces.[[2]](#endnote-3) Methodologically speaking, focusing on names might seem a bit counter-productive, since attention to materiality, devices, things, non-human actors and practice in general been one of the hallmarks of the social studies of infrastructure. In code-based settings (and code inflects almost all knowledge infrastructures today), naming is a crucial practice. Naming operationally links people and things, places and devices together in complicated associations. Nearly all code heavily relies on naming practices. Whether viewed as a series of statements, commands or functions, naming and the forms of address associated with names weave inextricably through software.

Code repositories store the source code, associated documents, files, configuration information and a variety of other materials associated with software.[[3]](#endnote-4) The largest repository of code repositories, the code repository platform (Github.com)[http://www.github.com] with around 20 million repositories (in 2015) sprawling across many domains, epitomises nominalisation of infrastructure. Github attracts tremendous amounts of code (for instance, Google Corporation recently announced the closure of its own code repository service Google.code.com and migration of all code deposited there to Github (Google 2015); Microsoft Corporation has recently opened the source code of Windows, its main product, on Github, etc.) Github is rather like NSA-O-Matic writ large, since the code repositories it hosts not only address almost any kind of digital hardware and software practice one can imagine, but seem to be animated in their growth since 2007 by the same combinatorial delirium that NSA-O-Matic, itself a Github-hosted project, displays. It is difficult to characterise the heterogeneity of things, devices, conventions, platforms and organisations named in Github code repositories, and indeed, this difficulty is one of the motivations to approach what happens there in terms of names.

Github itself looks like a typical contemporary social media platform. It has many of the appurtenances of participation ('followers', 'groups', 'watchers', 'stars', etc.;) and invokes the ethico-economic injunction to 'share' (John 2013). Github seeks to render coding itself 'social.' It treats coding as a social practice of participation, somehow akin to photo-sharing or chatting with friends using a social media platform. In this respect, infrastructures become the objects of social rituals and branded identifications. The trappings of social media and its imperatives -- 'be social, share code' -- with their focus on networks of relations between people largely occlude the tremendous combinatorial processes occurring on Github and git-like platforms more generally. The fluxes of imitation of names and their recombinant energy constantly overflow the imperative to 'be social.'

Is Github, then, a hub because it somehow has captured the nominalisation of infrastructure, or somehow made naming itself into the archive of this nominalisation? Is Github a logistical success story (say like Singapore's role in container shipping) or a tremendous organic accumulation (say like Lagos in Nigeria, with its complicated work-around flows of people, trade, oil and politics)? Neither of these alternatives are quite right. Like many platforms, services, devices and systems, the logistical success of Github (in coding and software deployment) depends on the tremendous morass of eccentric, abandoned, rarely used, ephemeral, miscellaneous and incredibly specific code and coding found there. The nominalisation of infrastructure, and its subsequent containerisation in code not only speeds up or renders infrastructure development more flexible and variable. In this respect, Github is somewhat like a logistics hub in which moving things around somehow coincides with naming them. The coincidence of making and naming, developing and operating perhaps challenges any sense of solidity we might have in relation to infrastructure.

The names of repositories hosted on Github display a baffling polysemy. Repository names on Github can be any combination of letters or characters. These range from the almost casually random, yet still recognisable names such as asdf (the first four characters on the home-row of an English keyboard) or 1234, both of which number in their thousands through to highly recognisable names such as linux or apache. Both of the latter projects are well-known as key infrastructural elements of contemporary digital networks. On the one hand, the distance between repositories called 'asdf' and 'linux' is great. The former typically represent ephemeral contacts between individuals and infrastructures (for instance, many such repositories are the traces of people trying out or learning to use Github as an online code repository) and the latter encompass information network architectonics resulting from much collective effort. The linux kernel in particular represents one of the main cases in which coding work both became more visible as a cultural and social practice (Coleman 2012; Kelty 2008; Mackenzie 2006), and practically permeated the information infrastructural *stack*.[[4]](#endnote-5)

Between the two extremes of the random keypress repositories and the installed bulk of linux or apache lies a very diverse and dense population of names. If we could read them serially in their spread and diversification over time, these names would diagram infrastructures multiplying, combining, and decaying in layers. Processes of repair, re-invention, appropriation, capitalisation and speculation are engrained in the distribution of names. The gamut of names of Github repositories, enough to populate a small country, is somewhat heterogeneous, pluralistic and full of transient multiplicities, but axes of organisation and centralisation that assemble infrastructural elements into something like the 'full stack' associated with the digital networks can be discerned there. Infrastructures like Github allocate all things a position in a common *namespace*. The Github platform, with its infrastructures for storage, searching, tracking and graphing of coding work, arrays all repositories on the same surface. In this respect, it renders all differences superficial, and this superficiality is perhaps a somewhat novel experience in relation to infrastructure. The relations and connections between disparate bodies of code of many different ilk have only recently started to congregate in a common namespace.

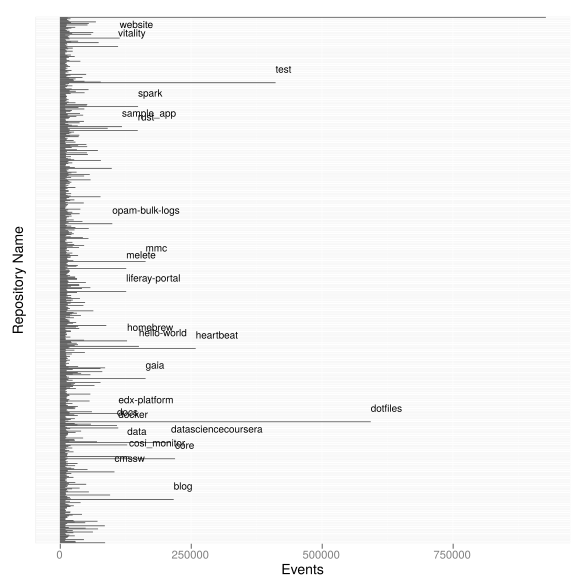
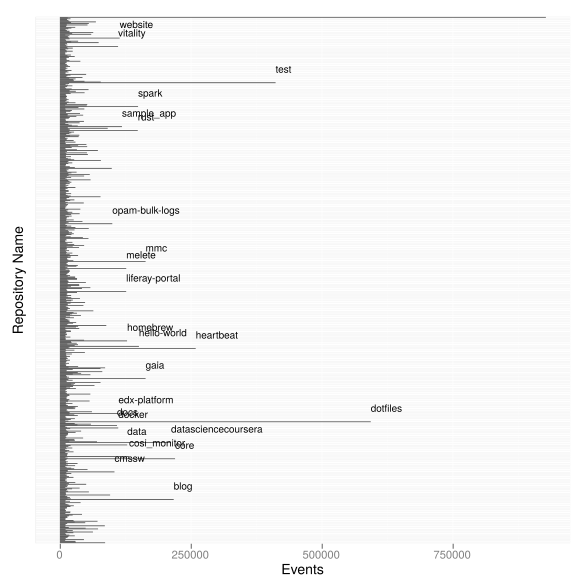
Illustration 1: Github repositories with more than 10,000 events in 2014. Repositories receiving more than 100,000 events during 2014 are labelled.

Figure 1: The most popular repositories on Github in 2014

The figure shows the most popular repositories on Github in 2014. Not all Github code repositories can be read as infrastructures, including several thousand repositories simply called test. But many of them, ranging from operating system such as Linux to code libraries for managing cloud-based deployments (puppet, chef, etc.) present a range of different namings. Some names represent generic practices -- blog, web, data, test. Others symbolise highly valued platforms -- liferay-portal or edx-platform. Some index individual configurations of practice -- dotfile and homebrew repositories usually store individual software developers' configuration files, a log in effect of how they arrange an equipmental totality to work on code. Finally, another set of names, such as rust, a systems programming language, spark, an 'engine' for large-scale data processing, docker, a platform to build, ship and run distributed applications, and heartbeat, a system for monitoring web systems and services, designate current infrastructural practices as points of identification.

## Names as cluster points

Every name condenses associations between people and infrastructures. For instance, on Github, the names of code repositories when fully specified have a number of components. Take the 70,000 or so repositories whose names include the term JQuery.JQuery is a software library that supplies code for the buttons, animations, dynamic menus, etc. that we associate with interactive web pages (for instance, for buying a train ticket online). JQuery's popularity amongst web developers -- more than 50% of all websites were making use of it in 2012 (W3Techs 2012) -- arises from the way that it allows web pages written in HyperTextMarkupLanguage (HTML) to be programmatically manipulated as a set of named elements. JQuery subjects the screen as a set of named elements subject to operations. Statements written using JQuery operate to generate text and graphics that people read, scroll, click, and select. This relatively banal example of how a page becomes a field of named entities exemplifies a much more general tendency. In the much more obviously infrastructural operations of Amazon Web Services Elastic Compute 2 (AWS EC2), something similar appears. The approximately 21,000 Github repositories relating to AWS EC2 delineate a great range of concerns, but nearly all offer ways of naming, addressing and configuring instances of linux services operating in Amazon's data centres. Since they use different programming languages -- Haskell, Ruby, Python, Java or Javascript, many different naming conventions permeate these repositories, but all of them invoke the operation of infrastructural elements in AWS EC2: disks, clusters, nodes, databases, servers and services spin up and spin down.

How does a cluster of 70,000 JQuery repositories or 20,000 AWS EC2 repositories take shape? Many repositories differ from each other *only* in name. Some thousands of the jQuery repositories clone the primary jQuery/jQuery (or the jQuery Foundation's primary jQuery repository). Such repositories differ only through the name the person or organisation with whom they are concatenated. Despite this imitation, the fully qualified naming of things is an important pre-occupation in contemporary infrastructural semiosis. In the case of Github, every repository has a unique proper name, even if it directly clones another repository. Nomination combines a person or organisation and a thing, device, system, convention, protocol or infrastructure. The base name for all the copies of JQuery is jquery/JQuery. The left hand part of the name is a person or an organisation, in this case, the JQuery organisation or the [jQuery Foundation](https://jquery.org/), while the right hand part points to a repository containing code that inevitably points to many further named elements. The full name of a repository always combines a person or organisation (BBC, Google, Twitter, WhiteHouse, torvalds, etc.) and a thing (jQuery, bootstrap. linux, dotfile, etc.). Thus the fully qualified name of repository inextricably points in more than one direction. It does not index or symbolise either people or things but always a concatenation of organisation/person and device/thing. Such concatenations are familiar in science and technology studies accounts of technologies and scientific knowledges (relations between people and things, for instance, lie at the base of actor-network theory accounts of translation and displacement (Callon et al. 1983) or human-machine configurations (Suchman 2006)). Their sheer repetition does, however, intensify and thicken contemporary infrastructures in two specific ways: by linking infrastructural operations to subject positions; in combinatorial variations.

The very operation of the thing/device/system/framework/protocol/tool/technique and the position assigned to the subjects who make statements about the infrastructure come much closer together and indeed mingle in code. Strictly speaking, the subject position marked by the left hand end of a Github repository name overflows the / that separates it from the body of code that addresses devices or systems on the right hand side. The situation here is captured well in Michel Foucault's account of the constituting and generating operations of statements in *The Archaeology of Knowledge*:

In each case the position of the subject is linked to the existence of an operation that is both determined and present; in each case, the subject of the statement is also the subject of the operation (he who establishes the definition of a straight line is also he who states it; he who posits the existence of a finite series is also, and at the same time, he who states it) ; and in each case, the subject links, by means of this operation and the statement in which it is embodied , his future statements and operations (as an enunciating subject, he accepts this statement as his own law) (Foucault 1972, 94–95).

Repositories comprise sequences of statements instituting the operating limits, the patterns of inscription and transcription, the regularities and usages that position subjects in infrastructures. These statements take effect, or implement something only through the coincidence that Foucault describes here. The subject of the statement is also the subject of the operation. This is reminiscent of more recent accounts of performativity and things, but the embodiment of the operation in the statement -- and effectively I am suggesting that the repository name refers to such statements -- also mobilises future statements and operations. How so, in terms of what happens on Github repositories?

The combinatorial repetition and imitation of repositories also thickens infrastructures through small variations and combinations. While several thousand repositories directly clone jquery/jquery, the remainder vary it. The vast majority of the repository names in which jquery appears also contain some other name. Composite names comprise much of the proliferating variation that combines jquery with other things. For instance, names such as:

jquery-postcode-anywhere

jquery-bingsearch

select2-json-php-mysql-bootstrap-jquery

jquery-django-formset

twitter-jquery-plugin

designate a range of compositions occurring around jquery, ranging from postal address systems (postcodes) through web search engines, layers in the internet stack (web servers, databases, web frameworks) and social media platforms. This sample of composite names is by no means representative since much of the compositional semiosis around jquery pivots on individual elements of web page user interfaces such as forms, menus, scrollbars and so on. Whether they stick closely to the graphic display of text and image or range more widely into the linkages between infrastructural layers, the combinatorial variation that begins to appear in these repository names operates much more widely in mobilising infrastructures. Imitative and combinatorial fluxes run across the millions of repositories on Github just at this banal level of repository names and naming.

## 'DevOps' and continuous deployment: when operating becomes naming

Other more fine-grained imitative and combinatory fluxes run much more deeply through the elements of coding, configuring, maintaining and repair work going on in the code. The workings of deeper imitative fluxes are not easy to see in repository names. They might be discerned by selecting from amongst the millions of named repositories on Github some of those that relate not so much to the design and development of software but to its deployment, maintenance and repair. Amidst the millions of repositories on Github, disrepair and abandonment are extremely common. Many repositories contain a few items uploaded on some date. Others have not been touched for months or years. Some repositories on Github, however, show almost continuous patterns of contribution (developers adding or modifying the code). Others, perhaps fewer again, are not only often modified but constantly copied.[[5]](#endnote-6) As we have seen earlier, the copying of a repository through the operation of forking or cloning is extremely common amongst developers as they diagrammatically construct services, platforms, devices and architectures. The many repositories containing names such 'dot' or 'dotfiles' or 'conf' do something similar at the level of individual developers development configuration. That is, they ensure a common and stable development environment for software developers amidst the many different configuration options possible (in relation to editors, compilers, file systems and various elements of the infrastructure that have to be rendered locally operable so that development for larger scale deployments can occur).

Some copying does not feed into code development. It is cloning for operational deployment. So-called 'DevOps' -- development-operation -- repositories expand infrastructural nominalisation beyond software development. They blur the line between making and operating. The operation of infrastructures, its day to day adjustments, workarounds, contingencies and frictions, intersects with the making or development of the infrastructure. Perhaps the most straightforward example of a DevOps repository would be github/Github. The repository is not publicly visible on Github, yet its existence underpins the ongoing life and dynamism of Github as a social coding platform. github/Github names Github itself. As the several hundred developers employed by Github add, modify, or repair elements of the platform itself, their changes flow into the operation of Github through continuous deployment routines that draw down changes from the github/Github onto the operating code of the Github platform. DevOps -- development operations - blurs the line between making and running, between designing and maintaining. We could also identify much more mundane versions of DevOps processes occurring in many repositories on Github.

DevOps repositories often contain specialised code focused on the deployment of devices, systems, other software, services, and platforms. This specialized information is itself subject to generalization by virtue of its nominalisation, a process that gathers a range of attributes together and provisionally freezes it or renders it more substantial. Configuration repositories abound on Github because they allow the often intricate configuration details of contemporary software systems to be replicated quickly. Given that the scale of much information infrastructure is predicated on replicating the same thing multiple times, these configuration repositories reposition the management of infrastructure much closer to software development (e.g. a data-centre might be understood as a collection of identical computers linked together in ways that permit their processing to be coordinated and divided in many different ways; 'identical' here, however means managed through processes of replication.)

'DevOps' re-fashion infrastructures through continuous deployment. It does this by eliding differences between making and operating by extending the code namespace, with all its combinatorial variations to include deployment. This extending nominalisation is a striking feature of such infrastructures, and one that may affect the mode of existence of infrastructures more generally. Different infrastructural figures such as 'the platform,' 'the stack,' 'deployment' and 'build' intersect with various organisational, aesthetic, ethical and commercial regimes of engagement. A combination of different materials ranging from images, texts, cloud or distributed computing services, various database and search engines, through to code repositories with their groups of contributors comprise the loose but interwoven infrastructures typical of such platforms. They hold together in relational patterns that arise from recursive practice.

**Conclusion**

The compound names generated by NSA-O-Matic resonates for us today because information infrastructures expand combinatorially. I have been suggesting that the flux of names in all their combinations and variations thickens or stacks contemporary infrastructures, and accretes subject positions associated with that deepening. Whether mundane or sublime, I have been suggesting that names and naming practices help us practically engage with the full-stack complexities of contemporary infrastructures. Names in their linkages of people and devices, of 'dev' understood as a person, the software developer, 'dev' as a common abbreviation for device, and 'DevOps' as development-operation flow widely and deeply in contemporary software development. They attest to different and often overlapping movements. The imitative fluxes of names point to processes of replication and copying that propagate devices, protocols, standards and things in various configurations. I suggested that in contemporary information infrastructures, the form of the stack presents one organising axis of this propagation. But this layered architecture is viscously mixed in the concatenated names of many of the repositories on Github. Both the replication of particular forms - linux, jquery - and their mutating combination can be seen in the repository names.

Names not only provide a methodological traction on the infrastructural flux associated with software and coding practices, giving us some sense of the relative centrings and the predominant infrastructural rituals of our worlds. Nominalisation operates generatively in several different senses. First of all, the very act of naming every collection of code, code that may or may not do something in the world, implies an increasing addressability of diverse bodies of code. This addressability supports and engenders what I have been calling, loosely following Peirce, nominalisation of infrastructure. Second, the expansion of what is called a code repository to include almost any aspect of the structure, configuration, documentation, and deployment work associated with contemporary infrastructures means that the same combinatorial composition that structures bodies of code can generalize to include the deployment and operation. Third, this nominalisation of infrastructure is somewhat indifferent to distinctions between making and using, between design and deployment, and indeed to differences between what people and devices do. Across all of these distinctions, naming and calling by name operate interchangeably, sometimes addressing a subject, sometimes addressing other kinds of being. This indifference or re-drawn difference between people and infrastructures generates a constant unfurling deployment in time and space. It lends itself to replication and variation, to changes in scale and variations in configuration, and combinatorial expansion. DevOps epitomises one highly distilled version of the nominalisation of infrastructure.

References

Beniger, James R. 1986. *The Control Revolution: Technological and Economic Origins of the Information Society*. Harvard University Press.

Bowker, Geoffrey. C. 2005. *Memory Practices in the Sciences*. MIT Press Cambridge, MA.

Bowker, Geoffrey C, and Susan Leigh Star. 1999. *Sorting Things Out. Classification and Its Consequences*. Cambridge MA: MIT Press.

Callon, Michel, Jean-Pierre Courtial, William A. Turner, and Serge Bauin. 1983. “From Translations to Problematic Networks: An Introduction to Co-Word Analysis.” *Social Science Information* 22 (2): 191–235. doi:[10.1177/053901883022002003](http://dx.doi.org/10.1177/053901883022002003).

CJFE. 2015. “Snowden Digital Surveillance Archive.” <https://snowdenarchive.cjfe.org/greenstone/cgi-bin/library.cgi>.

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

Edwards, P. N., M. S. Mayernik, A. L. Batcheller, G. C. Bowker, and C. L. Borgman. 2011. “Science Friction: Data, Metadata, and Collaboration.” *Social Studies of Science* 41 (5): 667–90. <http://sss.sagepub.com/content/41/5/667.short>.

Foucault, Michel. 1972. “The Archaeology of Knowledge and the Discourse on Language (Trans: A. Sheridan).” *Pantheon, New York*.

Galloway, Alexander R. 2004. *Protocol : How Control Exists After Decentralization*. Leonardo (Series) (Cambridge, Mass.). Cambridge, Mass.: MIT Press.

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

Git. 2014. “Git.” <http://git-scm.com/>.

Google. 2015. “Bidding Farewell to Google Code. Google Open Source Blog.” <http://google-opensource.blogspot.com/2015/03/farewell-to-google-code.html>.

Graham, Stephen, and Simon Marvin. 2001. *Splintering Urbanism : Networked Infrastructures, Technological Mobilities and the Urban Condition*. London: Routledge.

Guardian News. 2015. “The NSA Files. The Guardian.” <http://www.theguardian.com/us-news/the-nsa-files>.

John, Nicholas A. 2013. “Sharing and Web 2.0: The Emergence of a Keyword.” *New Media & Society* 15 (2): 167–82. doi:[10.1177/1461444812450684](http://dx.doi.org/10.1177/1461444812450684).

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

Mackenzie, Adrian. 2006. *Cutting Code: Software and Sociality*. Digital Formations. New York: Peter Lang.

Netcraft. 2015. “January 2015 Web Server Survey | Netcraft.” <http://news.netcraft.com/archives/2015/01/15/january-2015-web-server-survey.html>.

Peirce, Charles Sanders. 1955. *Philosophical Writings of Peirce*. Dover.

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. 2014. “The ‘Sentient’ City and What It May Portend.” *Big Data & Society* 1 (1): 2053951714532241. doi:[10.1177/2053951714532241](http://dx.doi.org/10.1177/2053951714532241).

Thung, Ferdian, Tegawendé F. Bissyandé, 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/xpls/abs_all.jsp?arnumber=6498480>.

W3Techs. 2012. “jQuery Now Runs on Every Second Website. Web Technology Surveys.” <http://w3techs.com/blog/entry/jquery_now_runs_on_every_second_website>.

1. In 'Pragmatism in Retrospect: a Last Formulation,' C.S. Peirce writes, 'by semiosis, I mean ... an action, or influence, which is, or involves, a cooperation of *three* subjects, such as a sign, its object, and its interpretant, this tri-relative influence not being in any way resolvable into actions between pairs' (Peirce 1955, 282). He goes on to note that the interpretant, the entity affected by an encounter with a sign is not necessarily a person, although it is easier to think about semiosis in terms of people. [↑](#endnote-ref-2)
2. Code and coding have been widely discussed as making contemporary infrastructure what it is. The conceptual genealogies I draw on include science and technology studies accounts of information and scientific data infrastructures (G. C Bowker 2005; Edwards et al. 2011), as well as anthropological accounts of software in terms of recursion and freedom (Kelty 2008; Coleman 2012) (although it lies quite a long way from debates about open vs closed source code, the debate that monopolised attention to software during much of the last decade) and connections between network infrastructure and contemporary urban experience (Graham and Marvin 2001; Thrift 2014). [↑](#endnote-ref-3)
3. The practices of working with code repositories on Github are quite complicated, and not the focus on discussion here. The advent of the git approach to the management of code has led to a very wide-ranging set of changes in the production of code, and Github has become the 'largest code repository on the planet' (Git 2014) by centralising some of these shifts. Software engineering researchers have taken a keen interest in these repositories precisely because they seem to offer access to how software is actually built (Takhteyev and Hilts 2010; Thung et al. 2013). [↑](#endnote-ref-4)
4. The stack comprises a diverse range of materials that could include storage systems (disk drives), servers, network resources, hardware specificities, data models and database architectures, business and transactional logics, software frameworks that connect elements of systems and architectures together, user interfaces including how screen elements such as graphics, interactive devices and text operate together. The first four layers of this stack, for instance, would be dominated by linux kernel versions, since the Android operating system, which has the largest installed base, is a version of linux. Or, if we turned from handheld devices to supercomputers, 97% reportedly use linux. This is clearly a repository of no little significance. While apache is perhaps less well-known, it has for over a decade now has served the largest share of websites (around 40% or 340 million sites in January 2015 (Netcraft 2015)). [↑](#endnote-ref-5)
5. Detecting repositories on Github that display this continuous development is not straightforward and involves some use of digital methods. In this case, I made use of the GithubArchive.org datasets as published in the Google BigQuery GithubArchive dataset. See http://bigquery.google.com/githubarchive [TBA - check url] [↑](#endnote-ref-6)