

PROOF COVER SHEET

Author(s): Adrian Mackenzie

Article Title: 48 million configurations and counting: platform numbers and their capitalization

Article No: RJCE1393443

Enclosures: 1) Query sheet
2) Article proofs

Dear Author,

1. Please check these proofs carefully. It is the responsibility of the corresponding author to check these and approve or amend them. A second proof is not normally provided. Taylor & Francis cannot be held responsible for uncorrected errors, even if introduced during the production process. Once your corrections have been added to the article, it will be considered ready for publication.

Please limit changes at this stage to the correction of errors. You should not make trivial changes, improve prose style, add new material, or delete existing material at this stage. You may be charged if your corrections are excessive (we would not expect corrections to exceed 30 changes).

For detailed guidance on how to check your proofs, please paste this address into a new browser window: <http://journalauthors.tandf.co.uk/production/checkingproofs.asp>

Your PDF proof file has been enabled so that you can comment on the proof directly using Adobe Acrobat. If you wish to do this, please save the file to your hard disk first. For further information on marking corrections using Acrobat, please paste this address into a new browser window: <http://journalauthors.tandf.co.uk/production/acrobat.asp>

2. Please review the table of contributors below and confirm that the first and last names are structured correctly and that the authors are listed in the correct order of contribution. This check is to ensure that your name will appear correctly online and when the article is indexed.

Sequence	Prefix	Given name(s)	Surname	Suffix
1		Adrian	Mackenzie	

Queries are marked in the margins of the proofs, and you can also click the hyperlinks below. Content changes made during copy-editing are shown as tracked changes. Inserted text is in **red font** and revisions have a red indicator **▲**. Changes can also be viewed using the list comments function. To correct the proofs, you should insert or delete text following the instructions below, but **do not add comments to the existing tracked changes**.

AUTHOR QUERIES

General points:

1. **Permissions:** You have warranted that you have secured the necessary written permission from the appropriate copyright owner for the reproduction of any text, illustration, or other material in your article. Please see <http://journalauthors.tandf.co.uk/permissions/usingThirdPartyMaterial.asp>.
2. **Third-party content:** If there is third-party content in your article, please check that the rightsholder details for re-use are shown correctly.
3. **Affiliation:** The corresponding author is responsible for ensuring that address and email details are correct for all the co-authors. Affiliations given in the article should be the affiliation at the time the research was conducted. Please see <http://journalauthors.tandf.co.uk/preparation/writing.asp>.
4. **Funding:** Was your research for this article funded by a funding agency? If so, please insert ‘This work was supported by <insert the name of the funding agency in full>’, followed by the grant number in square brackets ‘[grant number xxxx]’.
5. **Supplemental data and underlying research materials:** Do you wish to include the location of the underlying research materials (e.g. data, samples or models) for your article? If so, please insert this sentence before the reference section: ‘The underlying research materials for this article can be accessed at <full link>/ description of location [author to complete]’. If your article includes supplemental data, the link will also be provided in this paragraph. See <<http://journalauthors.tandf.co.uk/preparation/multimedia.asp>> for further explanation of supplemental data and underlying research materials.
6. The **CrossRef database** (www.crossref.org/) has been used to validate the references. Changes resulting from mismatches are tracked in **red font**.

QUERY NO.	QUERY DETAILS
AQ1	The abstract is currently about 226 words. Please edit the abstract down to no more than 200 words.
AQ2	Please note that the keywords were not supplied with the manuscript for this paper, so they have been extracted from the files accompanying the manuscript. Please check and ensure this version is correct.
AQ3	Please note that the Funding section has been created from information provided through CATS. Please correct if this is inaccurate.
AQ4	There is emphasis given in the quoted text “technology into an asset that has ...”. Please specify if we need to add either “emphasis in the original” or “my emphasis” after the quote.
AQ5	Please replace “???” with the appropriate text here and subsequent occurrence in the text, and also in Note 3.

QUERY NO.	QUERY DETAILS
AQ6	The year for “Verran 2012” has been changed to “Verran 2011” after CrossRef validation of the references list entry. Please provide revisions if this is incorrect.
AQ7	The sentence “Formatting action in named events ...” has been changed. Please check the change conveys the intended meaning or amend.
AQ8	Please check the page number “p. 3.3” here.
AQ9	Please provide the missing acknowledgements.
AQ10	The disclosure statement has been inserted. Please correct if this is inaccurate.
AQ11	The CrossRef database (www.crossref.org/) has been used to validate the references. Mismatches between the original manuscript and CrossRef are tracked in red font. Please provide a revision if the change is incorrect. Do not comment on correct changes.
AQ12	Please provide the date that the web page was accessed for the references where the accessed date has not been provided.
AQ13	Please provide the missing place of publication for the reference “Campbell-Kelly, 2003”.
AQ14	Please provide the missing place of publication for the reference “Chun, 2011”.
AQ15	Please provide the missing place of publication for the reference “Doganovam and Muniesa, 2015”.
AQ16	Please provide the missing place of publication for the reference “Kelty, 2008”.
AQ17	Please provide the missing place of publication for the reference “Lovink <i>et al.</i> , 2015”.
AQ18	Please check the date “44 August” for the reference “Meyer, 2013”.
AQ19	Please check the date “45 December” for the reference “Newcomer, 2016”.
AQ20	Please check “1461444816661553” in the reference “Plantin et al., 2016”.
AQ21	Please check the date “39 December” for the reference “Plassnig, 2016”.
AQ22	Please provide the missing journal title for the reference “Star and Ruhleder, 1996”.
AQ23	Please provide the missing place of publication for the reference “Suchman, 2006”.
AQ24	Please provide the missing place of publication for the reference “Suchman, 2012”.
AQ25	The text is blurred in figure [4]. Please resupply the figure with clearer text.

How to make corrections to your proofs using Adobe Acrobat/Reader

Taylor & Francis offers you a choice of options to help you make corrections to your proofs. Your PDF proof file has been enabled so that you can edit the proof directly using Adobe Acrobat/Reader. This is the simplest and best way for you to ensure that your corrections will be incorporated. If you wish to do this, please follow these instructions:

1. Save the file to your hard disk.
2. Check which version of Adobe Acrobat/Reader you have on your computer. You can do this by clicking on the “Help” tab, and then “About”.

If Adobe Reader is not installed, you can get the latest version free from <http://get.adobe.com/reader/>.

3. If you have Adobe Acrobat/Reader 10 or a later version, click on the “Comment” link at the right-hand side to view the Comments pane.
4. You can then select any text and mark it up for deletion or replacement, or insert new text as needed. Please note that these will clearly be displayed in the Comments pane and secondary annotation is not needed to draw attention to your corrections. If you need to include new sections of text, it is also possible to add a comment to the proofs. To do this, use the Sticky Note tool in the task bar. Please also see our FAQs here: <http://journalauthors.tandf.co.uk/production/index.asp>.
5. Make sure that you save the file when you close the document before uploading it to CATS using the “Upload File” button on the online correction form. If you have more than one file, please zip them together and then upload the zip file.

If you prefer, you can make your corrections using the CATS online correction form.

Troubleshooting

Acrobat help: <http://helpx.adobe.com/acrobat.html>

Reader help: <http://helpx.adobe.com/reader.html>

Please note that full user guides for earlier versions of these programs are available from the Adobe Help pages by clicking on the link “Previous versions” under the “Help and tutorials” heading from the relevant link above. Commenting functionality is available from Adobe Reader 8.0 onwards and from Adobe Acrobat 7.0 onwards.

Firefox users: Firefox’s inbuilt PDF Viewer is set to the default; please see the following for instructions on how to use this and download the PDF to your hard drive: http://support.mozilla.org/en-US/kb/view-pdf-files-firefox-without-downloading-them#w_using-a-pdf-reader-plugin



48 million configurations and counting: platform numbers and their capitalization

Adrian Mackenzie

Department of Sociology, Lancaster University, Lancaster, UK

ABSTRACT

Platforms are important actors in contemporary cultural-economic processes. They include social network sites, online content management systems, streaming media platforms, mobile communication infrastructures, supply chain logistics management solutions, and cryptocurrency systems. This paper argues that analysis of platforms and their capitalization should take into account both the ways they structure social practice as assets and the constitutive opacity of platforms as configured realities. It explores this opacity by focusing on the problems of counting people and things on platforms. Via a case study of the software code repository platform Github.com, it analyses how the counting of 'platform numbers' participates in their capitalization. It describes 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 encounter unstable aggregates, forms of association, duplication, combination, imitation, and configuration that are crucial to the ensemble but remain refractory to asset-structuring capitalization and its enactments. It proposes configurative enumeration of the platform numbers as a way of conceptualizing 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 configurations crucial to the ongoing formation of platforms come into view. Configurative enumerations engage the inventive realities of platformization, realities that precede and sometimes overflow their capitalization.

ARTICLE HISTORY

Received 27 January 2017
Accepted 13 October 2017

KEYWORDS

Number; configuration; platform; capitalization; social media

AQ2

Introduction

This paper describes and theorizes 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 is a threat that connects what happens on Github (coding or software development: 'Github is how people build software' as Github rather tersely puts it (Github 2015)) to what happens on platforms more generally.

Platforms, as many authors (Gillespie 2010, Van Dijck 2013, Lovink *et al.* 2015, Langley and Leyshon 2016, Plantin *et al.* 2016, Srnicek 2016, McAfee and Brynjolfsson 2017) have suggested, stand at the centre of the processes of economic and cultural life. What are platforms? Nick Srnicek's *Platform Capitalism*, for instance, defines platforms as 'digital infrastructures that enable two or more groups to interact' (Srnicek 2016, p. 25). Srnicek proceeds to categorize platforms according to

the types: advertising, cloud, industrial, product, and lean (p. 49). Economists such as Andrew McAfee and Erik Brynjolfsson understand platforms as ‘a digital environment characterized by near zero marginal cost of access, reproduction and distribution’ (McAfee and Brynjolfsson 2017, p. 137). Their account in *Machine, Platform, Crowd* is strongly oriented to the economically disruptive development of platforms.

Neither Srnicek or McAfee focus on the making of platforms. They view them from the perspective of business models, assets, and ownership rather than in terms of an ongoing process of development and organization. The ongoing making of platforms, or the *platformizing* practices, do not figure much (or even at all) in these accounts, even as they stress the centrality of platforms to ‘our digital future’ (McAfee and Brynjolfsson 2017). The singularity of Github is that it is a platform where platforms are assembled and configured. Many of the most well-known platforms are themselves interwoven with code drawn from Github and the like.

Large numbers such as Github’s 48 million present a cultural-analytical artefact of the dynamism of technical platforms. They attest to ‘the scope and range of an operation that typically connects distant situations and configures large social realities’ (Muniesa *et al.* 2017, p. 17). The associative, imitative, and often highly layered fabric of technical platforms gives rise 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). 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 analysed. 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 concretization and abstraction, and distributed operational realities, that cannot be easily enumerated or conceptualized.¹ 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). Platform analysis has highlighted the entanglement of capitalization and platformization.

With respect to capitalization, 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), 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, p. 110). In many platforms, the constructed asset will largely comprise the dynamics or behaviours of social networks, or what Nathan Tcaz and Geert Lovink term ‘the platformed masses’ (Lovink *et al.* 2015, p. 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. Platformizing refers to the process of making platforms by placing people and things in specific kinds of relations. Platformizing configures people and things in constantly varying and experimentally modulated relations. If much recent literature on platforms tends to treat platforms as fully accomplished or completely built, the idea of platformizing shifts the emphasis to their ongoing genesis. 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, p. 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’ (p. 15).

The difficulty of engaging with platform capitalization has deep roots. If the philosopher Gilbert Simondon is right, technological cultures have increasingly arranged collections of technical elements such as tools and machines in *ensembles* that existing conceptual framings such as tool,

machine, device, or system struggle to hold on to (Simondon 1989, p. 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). Platforms, however, cannot be explained in terms of machines or technologies, because they cohere in ways that have not yet been fully conceptualized. The term *configuration* might go some way in addressing this difficulty.

Configuration, as Lucy Suchman suggests (Suchman 2012), concerns both the technical arrangements and significance of platforms. A configuration refers first of all the composition and joining of technical elements. **Comprising** largely generic commodity hardware and widely circulated software elements, the ensemble reality or specific texture and composition of platforms inheres in its configuration. Configurations in this sense range from technical details such as the specific key shortcuts a software developer sets up in their code editor to the broad architecture of data-flows between database, web-server, and mobile messaging system in a high-volume social media platform. **The** configuration also refers, much less overtly but directly, to the narration of the significance of platforms. The senses of configuration – composition and narration – play out together, at least for some of the actors concerned. If we are interested in both the composition and significance of platforms, then configurative analysis, with its entwining of technical practice and narratives of significance might be valuable. As I will suggest, despite their seeming remoteness from operational realities, platform numbers such as ‘48 million’ offer scope for sociological engagement with the configurative 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 aggregate, arrange, concretize, group, and connect existing technical elements), how are they made? How are machines, technologies, tools, and networks platformized?

Github lies at the heart of contemporary platformizing practice. On Github, a social network layer of programmers is immersed in making platforms. Github is a platformizing platform, or a hub of platformizing practice. Its mundane operational reality concerns how platforms (among other things; laws, recipes, plans, policies, scientific articles, books, etc. can also be found there) are made, maintained, and deployed. Github documents, directly and indirectly, platformizing processes.

Platformization depends on programming. Coding practices have changed over the last decade or so in ways profoundly entangled with technical platforms. First, coding has shifted away from relatively closed proprietary machine-centred software development (the kind epitomized by Microsoft Windows or IBM mainframe software during the latter part of last century (Campbell-Kelly 2004, Ensmenger 2010)) to a much more legally and technically open, de-centralized practice of varying scope threaded across communication and network infrastructures.² 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 (???). Similarly, the geography of coding work still centres on well-established urban centres such as San Francisco and London (Mackenzie 2016), but by virtue of the networked distribution of coding tools and skills, occurs in many other places as well.

Across all of these changes, coding has become a much more *public* economic and cultural activity (see Christopher 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

have 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. They signal some of the complexities involved in contemporary technical platforms.

Platform numbers and aggregating practices

Github is a typical technical platform because it intensifies all of these tendencies and tensions around the ensemble, distribution, and publicness/participation in coding practice. Github started in late 2007 and has since become a hub of coding. Like some other social media platforms, it has grown tremendously in the last 10 years. Github’s growth flows from a variety of processes that are difficult to summarize or classify partly because the actors, topics, or domains of coding are so diverse. At first glance, it appears as an online file store where, initially for free, coders can publish, store, and find the code. Some 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, p. 72). Such growth is not incidental or extrinsic to platform capitalization. As Liliana Doganova and Fabian Muniesa (2015) argue, 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, p. 12). While online code repositories have existed for several decades (most famously, SourceForge or GoogleCode that migrated to Github in 2015), more recent code repository platforms, such as Github, BitBucket, or GitLab, 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 describes and performs its own growth. It presents growth 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 concretize 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 the 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). 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; <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 (DoS) attack on Github, allegedly by the Chinese government³).

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) or investment (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). 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 (Ver-
 205 AQ6 ran 2011, 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, Meyer 2013, Gage 2015, Newcomer 2016, 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,
 210 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. They 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.

215

Aggregating events

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. 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 Dijk 2010)). The Github projects counted in 48 million
 220 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 (???). 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:
 225 ‘2,641,337 people hosting over 4,442,708 repositories’ announces the Github home page on 27 November 2012 (Github 2012).

230

How is the number 48 million actually produced? The count of repositories, or ‘projects’ as Github more recently terms them, is an artefact of the way that the flow of code into and out of Github is configured on 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 as people develop them. Control of the merging and branching of versions of code allows platforms to be updated, reconfigured, integrated, expanded, deployed,
 235 and localized. 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.

240

Code version control is premised on tracking every single action on Github, even those that appear to be quite elementary. Actions or ‘events’ are published by Github on its API or Application Programmer Interface. The act of publishing events on an 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). An extract from a single ForkEvent – the act of creating a new version of an existing repository – presents some of the unique naming practice that renders platform numbers countable:
 245
 250

```

{
  "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 themselves ntd.⁴ 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 connect the event to other groups of people, organizations, repositories, and images (gravatar_id). The intricate syntax of this data (many brackets, inverted commas, colons, commas) attests to configuration practices 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' permits precise enumeration of human and non-human elements, and opens the possibility of enumerating 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 of or around all of this.

Differentiating the flux of events into asset geographies

AQ7 Formatting action in named events not only provides momentary apprehensions of the platform numbers, it also constitutes a space for what Leyshon and Thrift term 'asset geographies' (Leyshon and Thrift 2007, p. 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 shows a more differentiated view of events on Github.⁵ Github itself presents all the public events on the platform in various ways: through the platform-defining Application Programmer Interfaces (APIs; Github 2016), through online archives such as GithubArchive or GHTorrent (GHTorrent 2017) and through cloud-based data analytics platforms such as GoogleBigQuery (Inc 2016). These data-streams embody the changes, aggregations, and commutations occurring as technical platforms take shape, concretize, 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 summarized 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 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) 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 potentializing uncertainty in its own right if techniques or forms of realization can be applied to it. During 2012–2014 Github organized three ‘Data Challenges’ (Github 2013) focused on the same 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) concerned with making sense of platform numbers or engaged in practical re-countings of data generated by their own actions.

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) 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 these 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.

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	5628
Total	289,195,208

In response to the Github Data Challenge in 2012, contestants also 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 messages 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 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).

The platform numbers associated with Github also began 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 emphasizes the geography of the ‘top’ repositories, coders, and their programming languages (Fischer 2013–2013-05-05T02:47:58+00:00).

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) (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

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!


OK. I promise!

THE
OPEN SOURCE
REPORT CARD

Elite developers,
hand-picked to match your
needs. Startups depend on
TopTale.

ads via Carbon

Hire The Best



Developers

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

How about 'mojombo', for example...

HOW IT WORKS

Figure 1. The open source report card.

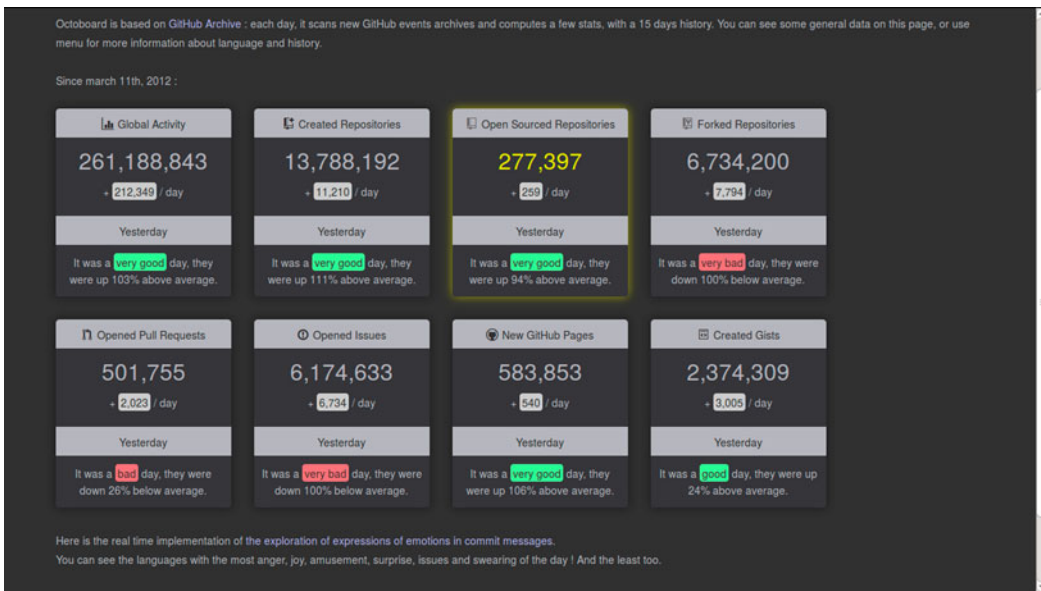


Figure 2. Octoboard: Github activity dashboard.

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, localize, and qualify platform numbers such as 48 million. They potentialize 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

Enumerations of people and things play a vital role in platform capitalization. They show how what is enumerated might be re-counted or re-grouped as asset forms. These enumerations are synthetic, platform-specific counts. They can draw on reputational measures such as 'likes', 'watching', and in the case of Github, the act of forking or 'sharing' code.

It is not, however, the case that platform capitalization always results in stable asset forms. Like Karl Marx's decomposition of the commodity form into shifting ratios of forms of capital (fixed, relative, absolute surplus value, etc. (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 stabilize as assets; and (b) gauging whether platformization, or the process of rendering social life through platforms heeds something other than asset forms. More specifically, I would suggest that platform numbers do not simply act performatively (as most current understandings of capitalization and related cultural-economic processes presume; see Mohamed *et al.* 2011, Cardwell 2015, Paudyn 2015, Jerne 2016, Yarrow and Kranke 2016), but derive from pre-capitalizing processes specific to the platform in question and animated directly by processes of configuration vital to the platform's ensemble reality.⁷

The configurative modality of platform numbers stems from the most abundant and elementary entity in code and on Github more generally, the *name*. Ludwig Wittengstein suggests that 'only in the nexus of a proposition does a name have a meaning' (2003, p. 3.3). Paraphrasing, we might say:

only in **the** nexus of a configuration does a name operate. ‘Naming’ lies at the heart of the massive systems of addressability associated with software (see Thrift and French 2002). The sample data shown in Data Extract 1 suggest that events in the Github timeline data and on the platform more generally can be understood as **a** nexus of named entities. The event data format is a nexus of names referring to people, organizations, 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 to address multitudes of platform elements – powerfully informs both the platform number 48 million, and the configurative dynamism associated with it. Github’s platform numbers count people and repositories because entities (people, devices, locations, etc.) have been named.

What would happen if we approach the platform number 48 million with an eye on the diverse temporal, associative, and configurative composition of named entities in the timeline event data (as published on GoogleBigQuery)?⁸ 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 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 **1000** events.

The 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 in the event of their naming 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) 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. Many repositories come into existence through an act forking another repository. A fork associates a named user with a named repository. Each time a repository is forked, a ForkEvent appears in the timeline event data. Figure 3 separates fork and non-fork repositories. 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 repositories created through forking or copying have some of the highest event counts.

A final example of the configurative growth on Github can be seen in how repositories combine with each other. If forking powerfully drives the growth of platform numbers, but repositories also derive from commutations of existing repositories. Both Figures 4 and 5 count the number of repositories that commute or recombine existing repositories. Rather than counting overall or fork events, these plots show 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 concretize 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 Each day, however, new combinations of existing repositories appear 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

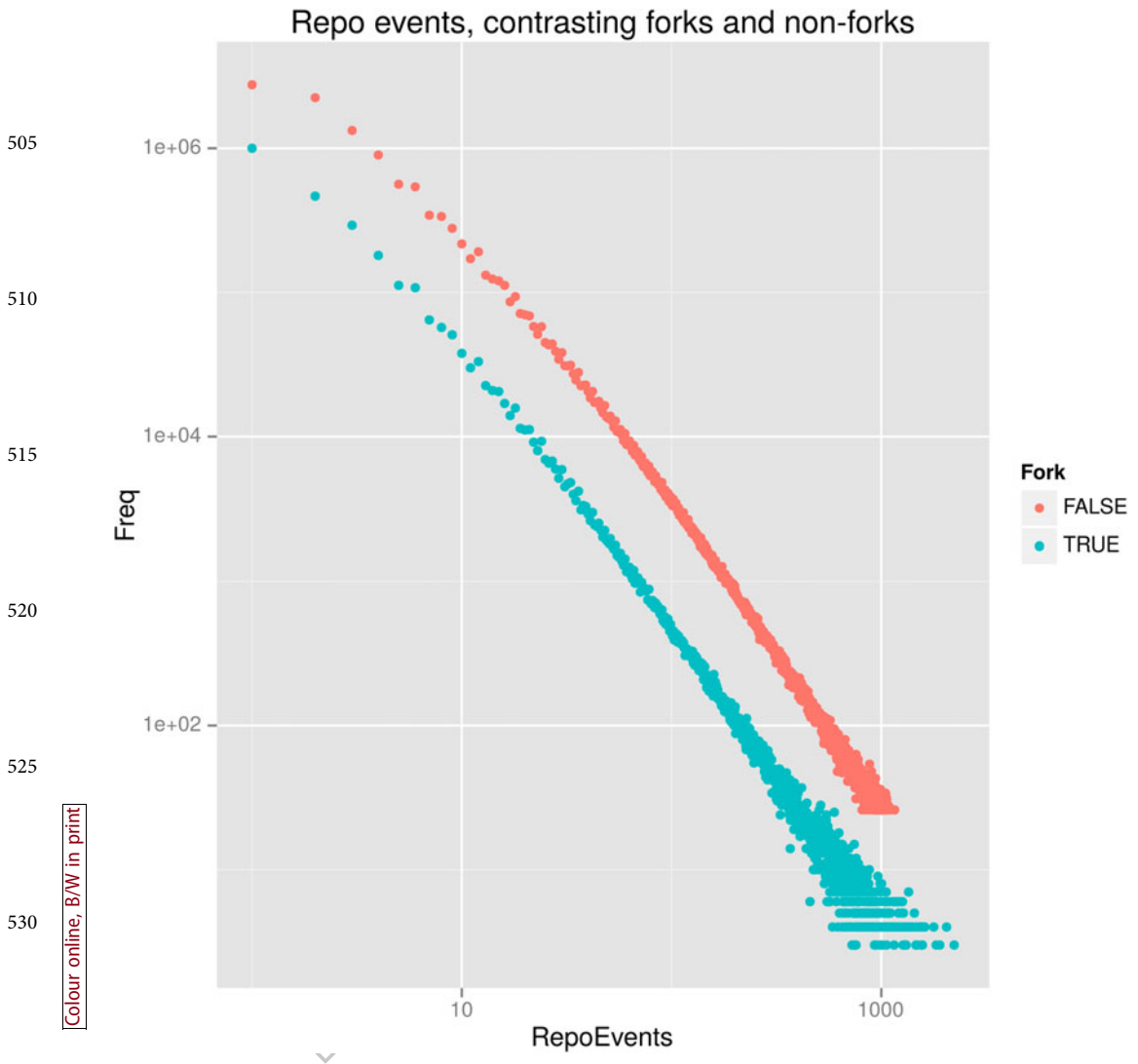
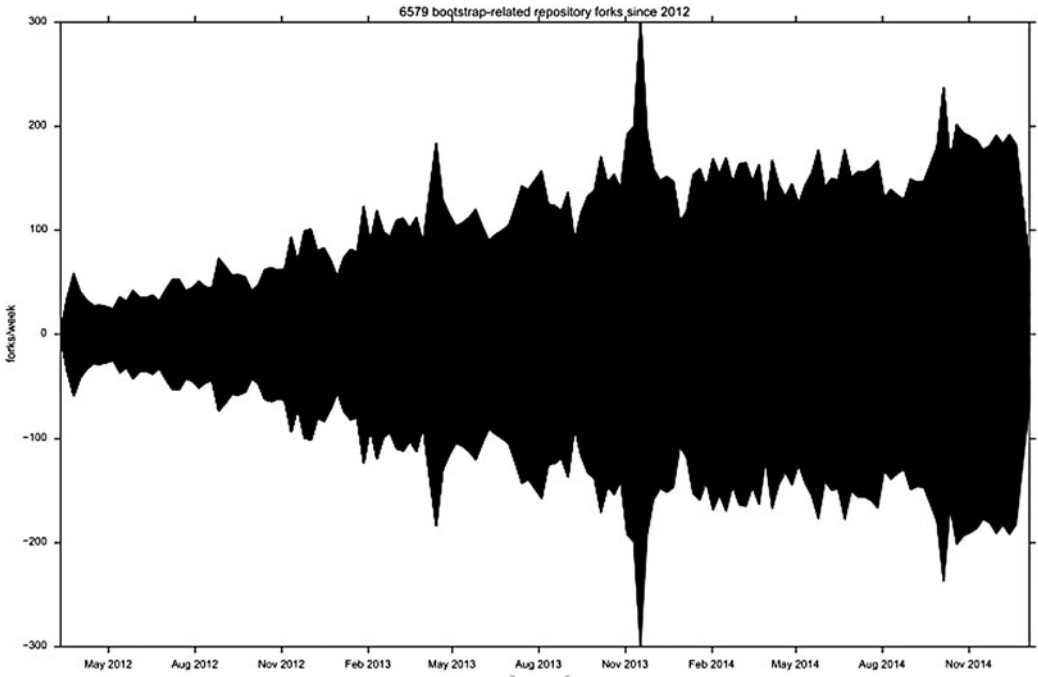


Figure 3. The power of events over time.

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 combination of repositories generate large numbers of events on Github, and hence strongly animate platform numbers. Recombining repositories will never have prominence comparable to the base repositories they relate to. These striations mark processes – an incessant configuring of technical elements in the ensembles – that 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, forked, and re-combined repositories is not inimical to growth and future-revenue streams, but it suggests that any such growth is rooted in 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



AQ25 Figure 4. Bootstrap repository forks.

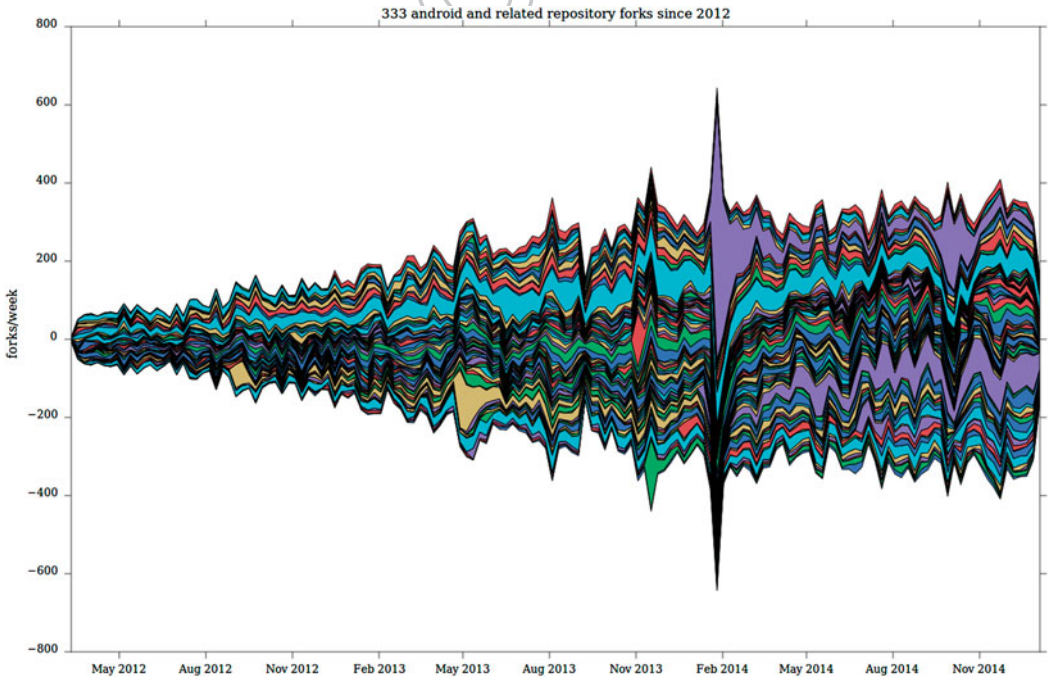


Figure 5. Android repository forks.

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 Two different tendencies stand out in this subset of the 48 million.

The small number of 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) 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 stabilize, to hybridize, or even infrastructuralize (Plantin *et al.* 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`. The development and much of the associative fabric of `eclipse` does not reside on Github, but is copied there.¹² In other words, the many events in the `eclipse.platform.common` repository on Github echo actions elsewhere. For many other significant repositories (`linux`, `android`, `mozilla`, `apache`), high event counts attest here to Github’s function as an element in a wider platform configuration.

A final feature in high event-count repositories is vital to the ongoing genesis of technical platforms as configured operational realities. The list of names of high-event count repositories cluster often include `test`, `hello`, `config`, `doc`, `build`, `setting`, or `demo`. 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 the configuration as the composition of a technical platform. These repositories store configurations that sustain arrangements of technical elements in the making of platforms. In the platform number 48 million, the proportion of events absorbed by repositories explicitly concerned with storing configurations is substantial. Of the events received by the top 1000 repositories, 18% belong to configuratively named repositories, repositories that explicitly focus on setting up and maintaining the small but crucial linkages that connect code libraries, editors, build tools, databases, network connections, online access, and authorizations, thereby allowing developers to position themselves within the operational ensemble of platforms.¹³

Conclusion

Platform capitalization can be understood as an 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 social network behaviours associated with coding and programming. It configures coding as a ‘social’ practice, replete with organizational or collaborative affordances. Both code itself and coding as a form of work are tangible assets here in all their practical varieties and intangible dimensions. Processes of collaboration, and programmers’ identification with specific bodies of code are structured as assets alongside the code itself.

Platform capitalization enacts high-level metrics – 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 configurations that assemble technical platforms. They evoke growth, a form of change vital to capitalization in its promissory modality, but they cannot subsume or fully digest it. In key respects, Github is a singular case of platform capitalization because it makes this aspect of platformization more obvious. It is a platform where platforms are assembled and configured. Because it is a platformizing platform, a place where the coding masses technically and socially negotiate platform configurations, Github presents the possibility of observing some of the dynamics of platformization. Github presents, almost inadvertently, 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, typical of platform capitalization more generally if, as I have been suggesting, all platforms have a constitutive opacity rooted in their configured ensemble reality.

In their recent work, Fabian Muniesa and co-authors advocate direct engagement with capitalization: 'studying capitalization is also redefining it, engaging with it, against it, alongside it' (Muniesa *et al.* 2017, p. 127). In this paper, I have re-counted platform numbers as they derive from names native to code. This approach is platform-specific and it attends to 'contingent configuration' (Langley and Leyshon 2016, p. 15). Configurative enumerations re-do platform numbers in ways that pay attention to their 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 of platform numbers – almost half of the repositories are forks; the vast majority of repositories only ever receive one to two events; a substantial proportion of code repositories have no code in them; recombination of projects heavily contribute to the repository count; many repositories mirror work done elsewhere; configurative repositories hold things together – are limited in their grasp to that aspect of the platform 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 concretization that technical platforms undergo as part of their genesis, and to the configurative processes that in many ways precede and sometimes de-stabilize asset-structuring value in capitalization.

Notes

1. Concepts of infrastructure (Star and Ruhleder 1996), sociotechnological project (Latour 1996), global assemblage (Ong and Collier 2005), configuration (Suchman 2006), interface (Marres and Gerlitz 2015), machinic assemblage (Lazzarato 2014), and more recently, 'stack' (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 conceptualize.
2. I do not 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) and Coleman (2012) for discussion of these discourses.
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 materializing 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). In early 2015, Github itself was subject to a massive DoS 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 (??).

4. In what follows, I use a monospace font like Github when referring to entities named in the Github data. The typography highlights entities subject to enumeration by platform numbers.
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). Grigorik, or `igrigorik` not 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) 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 flow 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.
6. Looking slightly more widely, the Github timeline data have quickly become a favourite training tool for data mining textbooks 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). 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 organization 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). Like the many Github-hosted projects discussed above, they analyse sentiment (Guzman et al. 2014), collaboration and productivity (Dabbish et al. 2012), and geography (Takhteyev and Hilts 2010).
7. For the beginnings of discussion moving beyond the performative, see (Cooper and Konings 2016, Pellandini-Simányi 2016). In this 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 2011, 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.
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.
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).
10. The ForkEvents are not included in Figures 4 and 5.
11. The GoogleBigQuery query de-capitalizes 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`.
12. The presence of `eclipse` on Github suggests another important analytical problem to consider when enumerating: how do we know for a given setting or place that what we count belongs there? Sometimes repositories on Github mention they are 'mirrors' of something else. Other times they may not.

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 repository for typical contents (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 10 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 11 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.

Acknowledgements

AQ9 xxx.

Disclosure statement

AQ10 No potential conflict of interest was reported by the author.

Funding

AQ3 This work was supported by the Economic and Social Research Council [ES/K007912/1].

Notes on contributor

Adrian Mackenzie (Professor in Technological Cultures, Department of Sociology, Lancaster University) researches cultural intersections in science, media and technology. His most recent project is *Machine Learners: Archaeology of a Data Practice* (MIT Press 2017).


References

- Bratton, B.H., 2016. *The stack: on software and sovereignty*. 1st ed. Cambridge: MIT Press.
- Bucher, T., 2013. Objects of intense feeling: the case of the Twitter API: computational culture. *Computational Culture* [online], 3. Available from: <http://computationalculture.net/article/objects-of-intense-feeling-the-case-of-the-twitter-api>.
- Bynens, M., 2016. Mathiasbynens/Dotfiles. *GitHub* [online]. Available from: <https://github.com/mathiasbynens/dotfiles>.
- Campbell-Kelly, M., 2003. *The history of mathematical tables*. Oxford University Press.
- Campbell-Kelly, M., 2004. *From airline reservations to sonic the hedgehog: a history of the software industry*. New ed. Cambridge: MIT Press.
- Cardwell, E., 2015. Power and performativity in the creation of the UK fishing-rights market. *Journal of Cultural Economy* [online], 8 (6), 705–720. Available from: <http://www.tandfonline.com/doi/abs/10.1080/17530350.2015.1050441>.
- Chun, W., 2011. *Programmed visions: software and memory*. The MIT Press.
- Coleman, G., 2012. *Coding freedom: the ethics and aesthetics of hacking*. Princeton, NJ: Princeton University Press.
- Cooper, M. and Konings, M., 2016. Pragmatics of money and finance: beyond performativity and fundamental value. *Journal of Cultural Economy* [online], 9 (1), 1–4. Available from: <http://www.tandfonline.com/doi/abs/10.1080/17530350.2015.1117516>.
- Dabbish, L., et al., 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*. ACM, 1277–1286 [online]. Available from: <http://dl.acm.org/citation.cfm?id=2145396>.
- Doganovam, L. and Muniesa, F., 2015. Capitalization devices: business models and the renewal of markets. In: M. Kornberger, L. Justesen, A.K. Madsen, and J. Mouritsen, eds. *Making things valuable*. Oxford University Press, 109–125.
- Eclipse Foundation, 2016. *Eclipse – the eclipse foundation open source community website* [online]. Available from: <https://www.eclipse.org/>.

- Ensmenger, N., 2010. *The computer boys take over: computers, programmers, and the politics of technical expertise* [online]. MIT Press. Available from: <http://www.jstor.org.ezproxy.lancs.ac.uk/stable/j.ctt5hhjdj>.
- Fischer, D., 2013–2013-05-05T02:47:58+00:00., GitHub data challenge II. *David Fischer Dot Name* [online]. Available from: <https://www.davidfischer.name/2013/05/github-data-challenge-ii/>.
- Foreman-Mackay, D., 2014. *The open source report card* [online]. 14 February. Available from: <https://web.archive.org/web/20140214105201/http://osrc.dfm.io/>.
- Gage, D., 2015. GitHub raises \$250 million at \$2 billion valuation. *Wall Street Journal* [online], 29 July. Available from: <http://www.wsj.com/articles/github-raises-250-million-at-2-billion-valuation-1438206722>.
- Gerlitz, C. and Lury, C., 2014. Social media and self-evaluating assemblages: on numbers, orderings and values. *Distinktion: Journal of Social Theory*, 15 (2), 174–188. doi:10.1080/1600910X.2014.920267.
- GHTorrent., 2017. *The GHTorrent project* [online]. Available from: <http://ghtorrent.org/>.
- Gillespie, T., 2010. The politics of ‘platforms’. *New Media & Society*, 12 (3), 347–364. doi:10.1177/1461444809342738.
- GitHub, 2012. *GitHub social coding* [online]. Available from: <http://wayback.archive.org/web/20121127113950/https://github.com/>.
- GitHub, 2013. Data challenge II results. *GitHub* [online], 26 June. Available from: <https://github.com/blog/1544-data-challenge-ii-results>.
- GitHub, 2014. Results of the GitHub investigation. *GitHub* [online], 21 April. Available from: <https://github.com/blog/1823-results-of-the-github-investigation>.
- GitHub, 2015. About GitHub. *Internet Archive* [online], 16 December. Available from: <https://web.archive.org/web/20151216055610/https://github.com/about>.
- GitHub, 2016. *GitHub events application programmer interface* [online]. Available from: <https://api.github.com/events>.
- Grigorik, I., 2012. *GitHub archive* [online]. Available from: <http://www.githubarchive.org/>.
- Guyer, J.L., 2014. Percentages and per chance: archaic forms in the twenty-first century. *Distinktion: Scandinavian Journal of Social Theory*, 15 (2), 155–173. doi:10.1080/1600910X.2014.920268.
- Guzman, E., Azócar, D., and Li, Y., 2014. Sentiment analysis of commit comments in GitHub: an empirical study. In: *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR. New York, NY, USA: ACM, 352–55. doi:10.1145/2597073.2597118.
- Hallinan, B. and Striphas, T., 2014. Recommended for you: the netflix prize and the production of algorithmic culture. *New Media & Society*, 1–21. doi:10.1177/1461444814538646.
- Hardy, Q., 2012. Dreams of ‘open’ everything. *Bits Blog* [online], 28 December. Available from: <http://bits.blogs.nytimes.com/2012/12/28/github-has-big-dreams-for-open-source-software-and-more/>.
- Inc, Google, 2016. GitHub repositories on google bigquery. *Google BigQuery* [online]. Available from: https://bigquery.cloud.google.com/dataset/bigquery-public-data:github_repos.
- Jerne, C., 2016. Performativity and grassroots politics: on the practice of reshuffling mafia power. *Journal of Cultural Economy* [online], 9 (6), 541–554. Available from: <http://www.tandfonline.com/doi/abs/10.1080/17530350.2016.1214849>.
- Kelty, C., 2005. Geeks, social imaginaries, and recursive publics. *Cultural Anthropology*, 20 (2), 185–214. doi:10.1525/can.2005.20.2.185.
- AQ16** Kelty, C., 2008. *Two bits: the cultural significance of free software*. Duke University Press.
- ▲ Knorr-Cetina, K. and Bruegger, U., 2002. Traders’ engagement with markets. A postsocial relationship. *Theory, Culture and Society*, 19 (5/6), 161–185.
- Lampland, M., 2010. False numbers as formalizing practices. *Social Studies of Science*, 40 (3), 377–404. doi:10.1177/0306312709359963.
- Langley, P. and Leyshon, A., 2016. Platform capitalism: the intermediation and capitalisation of digital economic circulation. *Finance and Society* [online]. Available from: <http://dro.dur.ac.uk/19828/>.
- Latour, B., 1996. *Aramis, or the love of technology*. Translated by Catherine Porter. Cambridge, MA: Harvard University Press.
- Lazzarato, M., 2014. *Signs and machines: capitalism and the production of subjectivity* [online]. Cambridge, MA: Semiotext (e). Available from: <http://mitpress.mit.edu/books/signs-and-machines>.
- Leyshon, A. and Thrift, N., 2007. The capitalization of almost everything: the future of finance and capitalism. *Theory, Culture & Society*, 24 (7–8), 97–115. doi:10.1177/0263276407084699.
- Lovink, G., Tkacz, N., and de Vries, P., eds., 2015. *Moneylab reader: an intervention in digital economy*. Institute of Network Cultures.
- AQ17** ▲ Mackenzie, A., 2006. *Java™: the practical virtuality of internet programming*. *New Media & Society*, 8 (2), 441–466.
- Mackenzie, A., 2016. Kittydar: detecting edges in the world – cultural anthropology. *Theorizing the Contemporary, Cultural Anthropology* [online]. Available from: <https://culanth.org/fieldsights/824-kittydar-detecting-edges-in-the-world>.
- Marres, N., and Gerlitz, C., 2015. Interface methods: renegotiating relations between digital social research, STS and sociology. *Sociological Review* [online]. Available from: <http://research.gold.ac.uk/11343/>.
- Marx, K., 1986. *Capital a critique of political economy. The process of production of capital*. Moscow: Progress.


McAfee, M. and Brynjolfsson, E., 2017. *Machine, platform, crowd: harnessing our digital future*. New York: W. W. Norton.

Meyer, R., 2013. Github, object of nerd love, makes play for non-programmers. *The Atlantic* [online], 44 August. Available from: <http://www.theatlantic.com/technology/archive/2013/08/github-object-of-nerd-love-makes-play-for-non-programmers/278971/>.

855 **AQ18**  Mohamed, A., et al., 2011. Deep belief networks using discriminative features for phone recognition. *IEEE*, 5060–5063. doi:10.1109/ICASSP.2011.5947494.

Muniesa, F., et al., 2017. *Capitalization: a cultural guide* [online]. Paris: Presses de l'Ecole des Mines. Available from: <https://ideas.repec.org/p/hal/journal/halshs-01426044.html>.

Newcomer, E., 2016. GitHub is building a coder's paradise. It's not coming cheap. *Bloomberg.com* [online], 45 December. Available from: <https://www.bloomberg.com/news/articles/2016-12-15/github-is-building-a-coder-s-paradise-it-s-not-coming-cheap>.


860 **AQ19**  Niederer, S. and van Dijck, J., 2010. Wisdom of the crowd or technicity of content? Wikipedia as a socio-technical system. *New Media & Society* [online], 12. Available from: <http://dare.uva.nl/cgi/arno/show.cgi?fid=337567>.

Ong, A. and Collier, S.J., 2005. *Global assemblages: technology, politics, and ethics as anthropological problems*. Malden, MA: Blackwell.

865 Paudyn, B., 2015. The struggle to perform the political economy of creditworthiness: European union governance of credit ratings through risk. *Journal of Cultural Economy* [online], 8 (6), 655–672. Available from: <http://www.tandfonline.com/doi/abs/10.1080/17530350.2015.1062792>.

Pellandini-Simányi, L., 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* [online], 9 (6), 570–586. Available from: <http://www.tandfonline.com/doi/abs/10.1080/17530350.2016.1214614>.

870 Plantin, J.-C., et al., 2016. *Infrastructure studies meet platform studies in the age of Google and Facebook* [online]. 1461444816661553. Available from: <http://nms.sagepub.com/content/early/2016/08/02/1461444816661553.abstract>.

AQ20  Plassnig, M., 2016. GitHub is doing much better than bloomberg thinks – here is why. *Medium* [online], 39 December. Available from: <https://medium.com/@moritzplassnig/github-is-doing-much-better-than-bloomberg-thinks-here-is-why-a4580b249044>.


AQ21  Roussel, D., 2015. Octoboard. *Github Activity Dashboard* [online], 1 August. Available from: <https://web.archive.org/web/20150801193208/http://octoboard.com/>.

875 Russell, M.A., 2013. *Mining the social web: data mining Facebook, Twitter, LinkedIn, Google+, GitHub, and more* [online]. O'Reilly Media, Inc. Available from: http://books.google.co.uk/books?hl=en&lr=&id=_VkrAQAAQBAJ&oi=fnd&pg=PR4&dq=github&ots=JiqitzTxmK&sig=sfea4ce1ue2XYt_dERD41VpSTS4.


Simondon, G., 1989. *Du Mode d'existence Des Objets Techniques* [online]. Paris: Aubier. Available from: <https://philpapers.org/rec/SIMDM2>.


880 Srnicek, N., 2016. *Platform capitalism*. Cambridge, UK: Polity Press.

Star, S.L and Ruhleder, K., 1996. Steps toward an ecology of infrastructure: design and access for large information spaces. 7 (1), 111–134.

AQ22  Statista, 2017. Number of Facebook users worldwide 2008–2017 | statistic. *Statista* [online], 12 June. Available from: <https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/>.

Suchman, L., 2006. *Human and machine reconfigurations: plans and situated actions*. 2nd ed. Cambridge University Press.

885 **AQ23**  Suchman, L., 2012. Configuration. In: Celia Lury and Nina Wakeford, eds. *Devices and the happening of the social*. Routledge, 48–60.

AQ24  Takhteyev, Y. and Hilts, A., 2010. *Investigating the geography of open source software through GitHub* [online]. Available from: <http://takhteyev.org/papers/Takhteyev-Hilts-2010.pdf>.

Thrift, N. and French, S., 2002. The automatic production of space. *Transactions of the Institute of British Geographers* [online], 27 (3): 309–335. Available from: <http://onlinelibrary.wiley.com/doi/10.1111/1475-5661.00057/abstract>.

890 Thung, F., et al., 2013. Network structure of social coding in Github. In: Software Maintenance and Reengineering (Csmr), 2013 17th European conference on [online], IEEE, 323–326. Available from: <http://ieeexplore.ieee.org/abstract/document/6498480/>.

Van Dijck, J., 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* [online], 19 (2), 141–155. Available from: <http://con.sagepub.com/content/early/2012/09/17/1354856512457548.abstract>.

895 Verran, H., 2011. 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.

Wittgenstein, L., 2003. *Tractatus logico-philosophicus: volume 123*. 2nd ed. London: Routledge.

900 Yarrow, D. and Kranke, M., 2016. The performativity of sports statistics: towards a research agenda. *Journal of Cultural Economy* [online], 9 (5), 445–457. Available from: <http://www.tandfonline.com/doi/abs/10.1080/17530350.2016.1202856>.