# Visualizing Collaboration and Influence in the Open-Source Software Community

Brandon Heller, Eli Marschner, Evan Rosenfeld, Jeffrey Heer
Stanford University
{brandonh, emrosenf}@stanford.edu, {eli,jheer}@cs.stanford.edu

## ABSTRACT

We apply visualization techniques to user profiles and repository metadata from the GitHub source code hosting service. Our motivation is to identify patterns within this development community that might otherwise remain obscured. Such patterns include the effect of geographic distance on developer relationships, social connectivity and influence among cities, and variation in project-specific contribution styles (e.g., centralized vs. distributed). Our analysis examines directed graphs in which nodes represent users' geographic locations and edges represent (a) follower relationships, (b) successive commits, or (c) contributions to the same project. We inspect this data using a set of visualization techniques: geo-scatter maps, small multiple displays, and matrix diagrams. Using these representations, and tools based on them, we develop hypotheses about the larger GitHub community that would be difficult to discern using traditional lists, tables, or descriptive statistics. These methods are not intended to provide conclusive answers; instead, they provide a way for researchers to explore the question space and communicate initial insights.

## Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Programming teams*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Computer-supported cooperative work*; K.1 [**Computing Millieux**]: The Computer Industry

## General Terms

Experimentation, Measurement

## Keywords

Visualization, mapping, data exploration, open source, collaboration, GitHub, geoscatter, social graph

## 1. INTRODUCTION

Looking into source code repository data to discover patterns is by no means a new phenomenon. Early work looked deep into *single* repositories, hoping to use the knowledge uncovered to inform and improve software practices. Examples include studies of code decay [9], taxonomies of commits [14], and tools to help developers write better code by identifying changes that may have been missed and suggesting opportunities for refactoring [20]. Others have used visualization techniques to make sense of and support exploration of code structure, change histories, and social relationships [3, 8, 11, 19], as well as animate a project's history [6, 15].

As centralized software hosting services such as SourceForge, Google Code, and GitHub began to appear, the scope of analysis expanded to *multiple* repositories. With data from a few hundred thousand developers, up from a few hundred, the set of questions we can ask changes. One natural question is "Where are the developers?"; multiple studies have looked into developer geo-distribution [7]. One study [12] investigated demographics and economic influence, correlating developer data with other sources to show the geographic distribution of developers by country, both in raw form and scaled to internet usage and GDP/capita. A similar study [10] combined email, time zone, and IP address information, along with mailing-list message data, to view the world-wide allocation of open source activity. In the study closest to ours, the authors focus on continent-level comparisons and filtering for accurate geolocations [16], concluding that North America receives disproportionately more attention and contributions.

Our preferred questions are why and how, in addition to where. With enough data, we can start to examine higher-level aspects of collaboration, like coding relationships, social relationships, and the balance of influence. To form hypotheses for these questions, and to consider how the answers vary by project and over time, we apply visualization techniques to data collected from GitHub. We employ three techniques: **linked geo-scatter maps** depicting interactions among locales, **small multiple displays** combining multiple views to enable visual comparisons, and **matrix diagrams** that reduce clutter to reveal pairwise relationships among metropolitan areas. After introducing each technique, we discuss observations uncovered from the visualization and future directions for analysis.

Our application of these tools reveals patterns that communicate the underlying data more directly than purely statistical analysis. These methods make the data more accessible to a wider audience of both researchers and developers. They augment, but do not replace, analytic approaches.

## 2. DATA

We use data collected from GitHub [1], a hosted source code repository with an integrated social network and detailed user profiles. This data set includes the complete social graph of 500,000 follow links as well as over 1,000,000 commits and 50,000 users.

**Data Collection -** GitHub provides public access to its social graph, source repository content, and user profiles. Fortunately, a large fraction of users provide a location in their profile, which we can turn into geographic coordinates using a geocoding API


*MSR* '11, May 21–22, 2011, Waikiki, Honolulu, HI, USA


**Figure 1: Build-Upon Graph for October 2010.**

like PlaceFinder [2]. Location strings with single geocode results are considered valid and are used, whereas those with none, or more than one (such as the case of identically-named cities) are ignored. GitHub location strings were also geocoded by Takhteyev and Hilts, who improved accuracy with a string-parsing and validation approach [16]. Like them, we found many examples of ambiguous or non-specific locations (e.g., "Earth").

To acquire this data, we built a custom GitHub crawler. The crawler uses single-letter queries to build a seed group of usernames, then fetches the set of repositories watched by each user in this group. For each repository, we extract the owner, collaborator, and contributor usernames, plus branch names. New usernames help to find new repositories, while branch names are used to fetch commit metadata. Using this method, the crawler uncovered 40,860 code repositories, representing 33,388 unique project names and 1,219,872 individual commits.

In addition to crawled data, we use the complete GitHub user follower graph from Jan 19, 2011. This graph includes 452,248 links connecting 106,247 unique users, 47% (49,500) of which could be geocoded with the PlaceFinder API. Given GitHub's total membership of approximately 550,000 users, the majority of users must be disconnected from the follow graph, due to not using the service after registration or following only repositories and not other users.

**Graphs -** We constructed three types of graphs from our collected data. All have users as nodes, but the meaning of edges varies. These graphs include (1) the *build-upon graph*: if user B creates a commit that immediately succeeds user A's, then we add a link from B to A to represent the build-upon relationship; (2) the *follower graph*: on GitHub, following a user causes their updates (commits, forks, etc.) to appear in your news stream; and (3) the *contributor graph*: contributors have either (a) pushed code directly into a repository or (b) had it pulled in by a project maintainer. Other link types to consider in future work include developers watching the same repository, forking a repository, and modifying the line, file, or block that another developer last modified. For space reasons, we show only graphs generated from types 1 and 2.

## 3. GEO-SCATTER MAP WITH LINKS

A **linked geo-scatter map** (a node/link diagram overlaid on a map) is a natural fit for graphs with geo-located nodes. To show build-upon relationships, we connect the points representing the locations of a parent commit's user and its child commit's user with a semi-transparent line, as shown in Figure 1. As more of these build-upon connections are found between nearby locations, the lines become increasingly opaque. The number of commits made at a location is represented by a circle, log-scaled to highlight location diversity at the expense of comparison accuracy. For a more detailed discussion of the visual encoding choices in the geoscatter map style and their tradeoffs, we refer the reader to [13]. This same general approach has recently been used to visualize Facebook friendships and professional citation networks [4, 5].

First, we see heavy development occurring between the US and Europe along an "Atlantic rail", which may be an artifact of high levels of Internet connectivity and/or cultural overlap between those regions. We also see that South American countries (Brazil, Argentina, Uruguay) connect more with Europe than North America; is this due to language effects? Links to Asia, South America and Africa are less prevalent. Perhaps development using GitHub is less prevalent on those continents? Alternatively, perhaps developers on those continents are less willing to advertise their whereabouts for cultural, political, or other reasons. Or perhaps there are more widely accepted alternatives to GitHub? The strange spot in the southern Atlantic is an erroneous geocoding of "New Zealand", showing visualization's power to expose data flaws.

## 4. SMALL MULTIPLES

To add depth to our analysis we next generate a matrix of maps, where each map is filtered by time range and project. This visual technique of repetition is called **small multiples**: "At the heart of quantitative reasoning is a single question: *Compared to what?* Small multiple designs, multivariate and data bountiful, answer directly by visually enforcing comparisons of changes" - E. Tufte [17]. Figure 2 represents five data dimensions: (1) *projects:* one per row, (2) *quarterly time periods:* one per column, (3) *developer locations:* coordinates of nodes on each mini map, (4) *commits at each location:* size of nodes, and (5) *collaboration between developers:* edges between nodes.

We see a significant contrast between projects with distributed development patterns, versus those that are more centralized. Looking at centralized examples, we find that the visual hub corresponds to project creators or maintainers. For example, the primary developer on the Redis project, Salvatore Sanfilippo, lives in Sicily, which is evident from the concentration of links leading there, as seen in Figure 3(a). The Git project shows a concentration around LA (Figure 3(c)), where its maintainer Junio C. Hamano lives.

Some projects' patterns stand out because they hint at unforeseen influences, and others because they buck established trends. Figure 4(a) highlights developer distribution patterns for the Homebrew project (a software package manager for Mac OS X) that might reflect where Apple has more market share. Figure 4(b) shows that Mono (an open implementation of the .NET platform) is characterized by distinctly distributed development, with a surprising dearth of developers on the west coast of the US.

## 5. MATRIX DIAGRAMS

Linked geo-scatter maps help to identify patterns of interest in an intuitive manner. However, more subtle patterns among the connections may be hard to discern. Overlapping edges obscure underlying details, and accurate quantitative comparisons become difficult, if not impossible. We now switch contexts to the GitHub follower graph and examine social links using **matrix diagrams**, a more abstract visualization technique that reduces clutter and helps to perceive edge metrics, using visual encodings inspired by the Honeycomb project [18]. Matrix diagrams are grids in which each cell represents a link metric, while rows and columns represent nodes.

Our metrics include (1) *followers*: the number of follow links, (2) *asymmetry*: the relative difference between follower totals in each direction, and (3) *deviation from expected*: the relative difference of actual link totals compared to those expected from sampling the node distribution at random. Figure 5 shows these matrix diagrams.

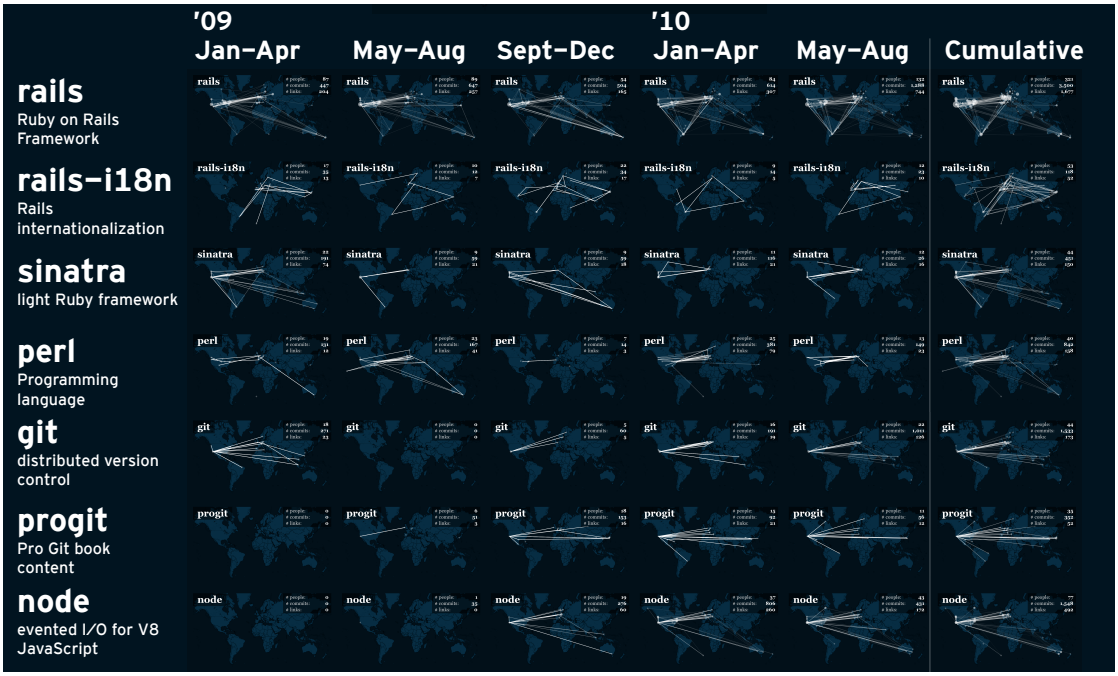
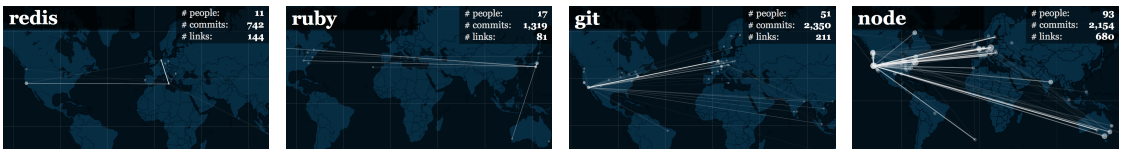



**Figure 2: Map matrix showing collaboration maps of different projects over quarterly time periods, with a cumulative view at right.**

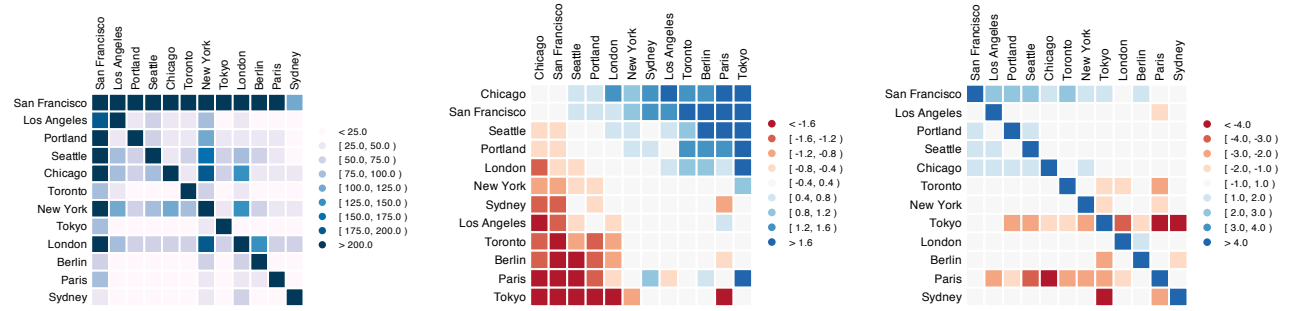



(a) Redis (b) Ruby (c) Git (d) Node

**Figure 3: Developer distribution and collaboration patterns for projects with a central maintainer, whose location is pronounced.**


(a) Homebrew (b) Mono (c) Rails

**Figure 4: Developer distribution and collaboration patterns for different projects.**




(a) **Followers:** follow link totals between areas, sorted by geographic proximity. Column nodes follow row nodes.

(b) **Asymmetry:** relative following link vs followed links, sorted from most to least influential. Positive (row, col) values indicate that a row is more influential (i.e., column follows row more often).

(c) **Deviation from expected:** relative actual links vs. expected links, assuming geography had no effect, sorted by geographic proximity. Positive (row, col) values indicate that a column follows a row more than expected.

**Figure 5: Matrix diagrams comparing major metropolitan areas on various metrics of the GitHub user-follow graph.**

To employ more links to make more statistically valid pairwise comparisons, we clustered the graph by metro area, aggregating users within a radius of each city. For example, the label San Francisco includes the Bay Area cities of Oakland and San Jose. Looking at Figure 5(a), the most prominent visual feature is the diagonal, suggesting that most collaboration occurs between developers near each other. We also see distinct horizontal and vertical bands for cities with large high-tech industries, such as Chicago, Seattle, New York, London, and especially San Francisco, as expected.

The total number of links to and from a city is a useful starting point for comparison, but taken alone it can fail to reveal more subtle patterns. We next look at follower asymmetry by comparing counts of in- versus out- links. If the number of in- and out-links are the same between nodes, the asymmetry is zero. For twice as many in- as out-links, the asymmetry is +1, and for the reverse, -1.

The diagram in Figure 5(b) is ordered by decreasing asymmetry, starting at the top left. In this ordering, Chicago comes before San Francisco, indicating that developers in its metropolitan area are followed more extensively than they follow people in other places. This may have something to do with the fact that the Ruby on Rails project, one of the first and most prominent GitHub-hosted projects, was started in Chicago. We also see some outlier patterns, particularly for Paris, whose developers seem to mostly follow rather than be followed, except when it comes to their relationships with developers in Sydney, Berlin, and especially Tokyo. This may indicate that there is a popular project, influential developer, or niche community centralized in Paris that many Australian, German and Japanese developers work with. The links between Tokyo and Paris constitute a noticeable outlier, but one based on few links (4 vs 14).

Now, we seek a different insight: which cities are more or less influential when compared to a baseline prediction that is independent of geographic proximity? We use knowledge of how many users are in each city, along with the total number of that city's links, to predict links – by selecting nodes from the real-world graph to connect with each other, uniformly and at random. Geography has no effect on the structure of this graph; we use it to compute expectations for the connectivity between each city pair.

The expected links diagram in Figure 5(c) shows the difference between the computed link-count expectations and observed counts. It reinforces observations made from the other two matrix diagrams by showing that Paris and Tokyo each have either many fewer incoming "followed" links than would be expected, or many more outgoing "following" links, and that San Francisco consistently has a surplus of "followed" links. The cluster of blue cells in the upper-left corner suggests that higher-than-average follow totals occur between developers in the top US cities, though it may be due to many of these cities being geographically close, on the west coast. By comparison, the top European cities show fewer unexpected links, which may suggest a community with a higher degree of internal balance, or one with less interest in collaboration.

# 6. DISCUSSION

Intriguing patterns emerge when we apply visualization techniques to source repository data. The resulting displays enable rapid hypothesis formation and are accessible to a wide audience. However, it is important to consider the incomplete, opt-in nature of the data source; visual patterns do not conclude, but instead point to aspects of data worthy of further, more traditional analysis.

We invite the reader to explore our collected data and find their own patterns – using the same interactive web-based visualization that helped the authors – at: http://gothub.stanford.edu. Users can filter by project name, date range, and geographic region. A play/pause button animates the history of a project, adding nodes and links as they appear. In fact, each map in this paper is simply a screenshot from the interactive website. Full source code is available, naturally, on GitHub: github.com/emarschner/gothub

# 7. ACKNOWLEDGEMENTS

We wish to thank the GitHub team for providing open access to their repository data. The visualizations in this work were made possible by Protovis, Polymaps, and CloudMade.

# 8. REFERENCES

[1] GitHub API. http://develop.github.com.

[2] Yahoo! PlaceFinder API. http://developer.yahoo.com/geo/placefinder/.

[3] T. Ball and S. Eick. Software visualization in the large. *Computer*, 29(4):33–43, 2002.

[4] O. H. Beauchesne. Map of scientific collaboration. http://olihb.com/2011/01/23/map-of-scientific-collaboration-between-researchers/.

[5] P. Butler. Visualizing friendships. https://www.facebook.com/note.php?note_id=469716398919.

[6] A. Caudwell. Gource software version control visualization. http://code.google.com/p/gource/.

[7] F. Cuny. Github communities poster. http://fr.linkfluence.net/wp-content/misc/github.pdf, 2010.

[8] C. De Souza, J. Froehlich, and P. Dourish. Seeking the source: software source code as a social and technical artifact. In *SIGGROUP Conference on Supporting Group Work, November*, pages 06–09. ACM, 2005.

[9] S. Eick, T. Graves, A. Karr, J. Marron, and A. Mockus. Does code decay? assessing the evidence from change management data. *IEEE Transactions on Software Engineering*, 27(1):1–12, 2002.

[10] A. Freytag, S. von Engelhardt, and C. Schulz. On the Geographic Allocation of Open Source Software Activities. *Jena Economic Research Papers in Economics*, 2010.

[11] J. E. Froehlich. Unifying artifacts and activities in a visual tool for distributed software teams. Master's thesis, University of California, Irvine, 2004.

[12] J. Gonzalez-Barahona et al. Geographic origin of libre software developers. *Information Economics and Policy*, 20(4):356–363, 2008.

[13] B. Heller, E. Marschner, and E. Rosenfeld. Geocoding github project report. https://graphics.stanford.edu/wikis/cs448b-10-fall/FP-RosenfeldEvan.

[14] A. Hindle, D. German, and R. Holt. What do large commits tell us?: a taxonomical study of large commits. In *Mining software repositories*, pages 99–108. ACM, 2008.

[15] M. Ogawa. code_swarm: An experiment in organic software visualization. http://www.michaelogawa.com/code_swarm/.

[16] Y. Takhteyev and A. Hilts. Investigating the Geography of Open Source Software through Github.

[17] E. Tufte. Envisioning Information. 1990.

[18] F. van Ham, H. Schulz, and J. Dimicco. Honeycomb: Visual analysis of large scale social networks. *Human-Computer Interaction–INTERACT 2009*, pages 429–442, 2009.

[19] F. Van Rysselberghe and S. Demeyer. Studying software evolution information by visualizing the change history. *Software Maintenance*, 2004.

[20] T. Zimmermann et al. Mining version histories to guide software changes. *IEEE Transactions on Software Engineering*, pages 429–445, 2005.