Metrics

Release 2021-03

https://chaoss.community/metrics

Contents

WG-COMMON

WHAT

Technical Fork

Question: What are a number of technical forks of an open source project on code development platforms?

Description A technical fork is a distributed version control copy of a project. The number of technical forks indicates the number of copies of a project on the same code development platform.

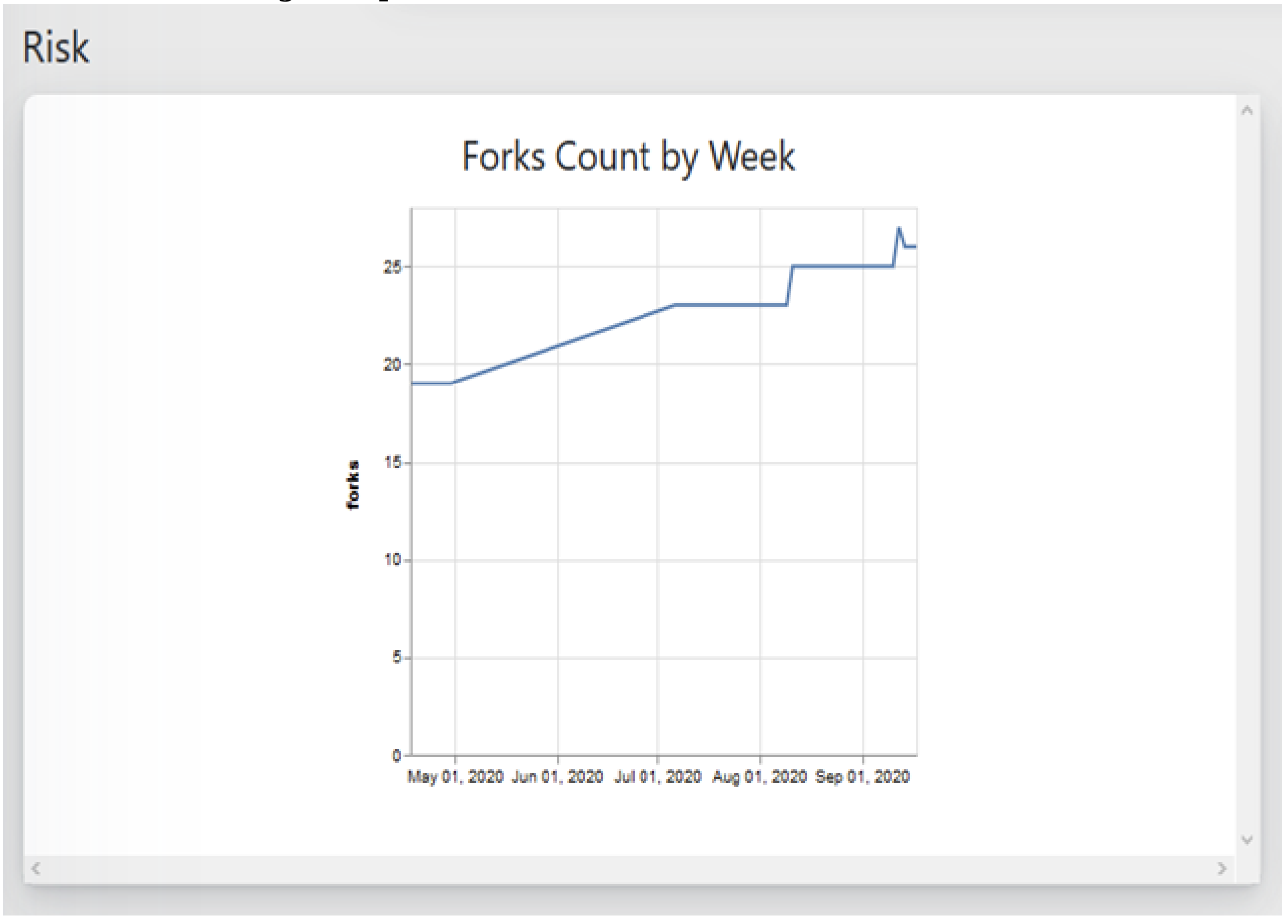
Objectives The objective of the Technical Fork metric is to ascertain how many copies of a project exist on a code development platform. Analysis of technical forks may provide insight into forking intentions (different types of forks such as contributing, and non-contributing forks).

Implementation

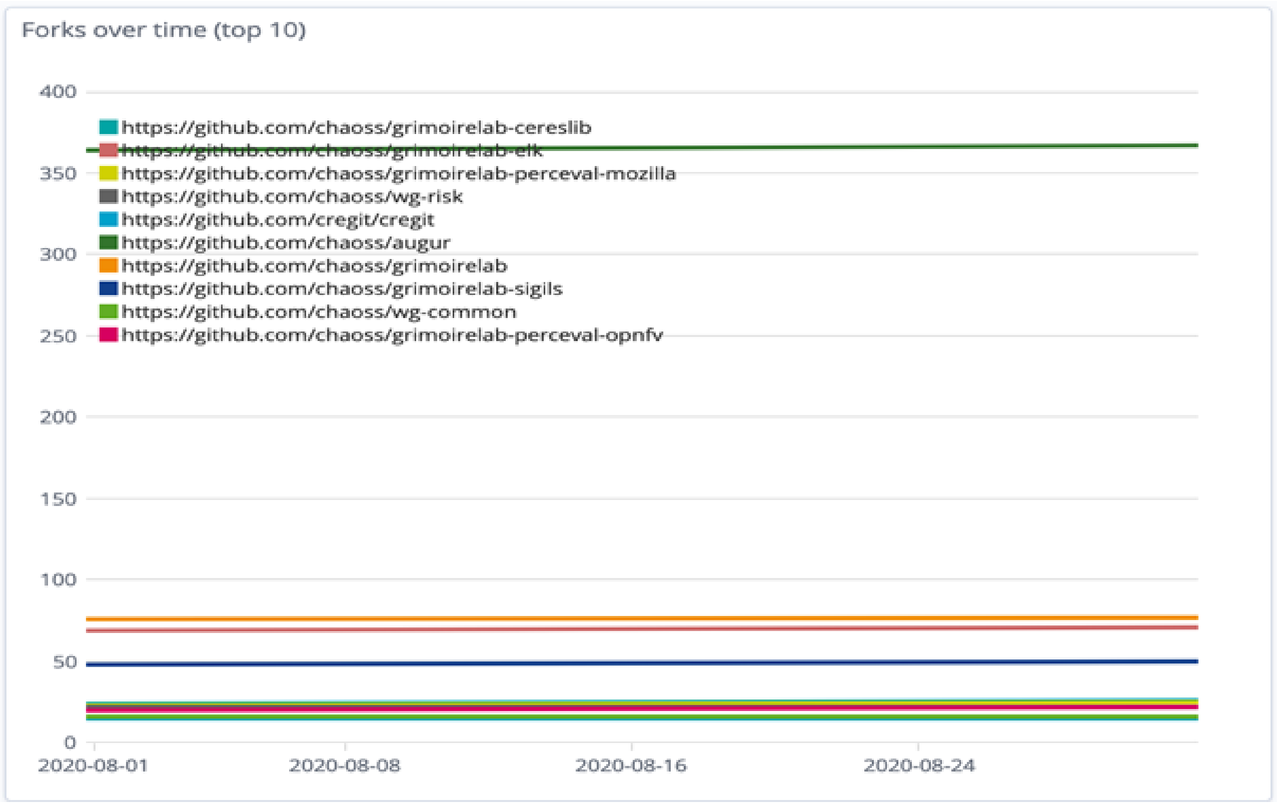
Filters

- Time Period (e.g., Weekly, Monthly, Annually)
- Ratio of contributing fork to total forks (A contributing fork is a fork that has opened a change request against the original repository.)
- Ratio of non-contributing fork to total forks (A non-contributing fork is a fork that has never opened a change request against the original repository.)

Visualizations **Augur Implementation**



GrimoireLab Implementation



Tools Providing the Metric

- Augur
- GrimoireLab

Data Collection Strategies **Github API**
https://developer.github.com/v3/repos/forks/#list-forks

GitLab API
https://docs.gitlab.com/ee/api/projects.html#list-forks-of-a-project

Bitbucket API
https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Bre

References https://help.github.com/en/enterprise/2.13/user/articles/fork-a-repo https://opensource.com/a
clone-difference

Types of Contributions

Question: What types of contributions are being made?

Description Multiple, varied contributions make an open source project healthy. Many projects have community members who do not write code but contribute in equally valuable ways by managing the community, triaging bugs, evangelizing the project, supporting users, or helping in other ways.

Objectives A variety of contribution types can demonstrate that a project is mature and well-rounded with sufficient activity to support all aspects of the project, and enable paths to leadership that are supportive of a variety of contribution types and people with varying expertise beyond coding.

Implementation How contributions are defined, quantified, tracked and made public is a challenging question. Answers may be unique to each project and the following suggestions are a starting point. As a general guideline, it is difficult to compare different contribution types with each other and they might better be recognized independently.

- The following list can help with identifying contribution types:
  - Writing Code
  - Reviewing Code
  - Bug Triaging
  - Quality Assurance and Testing
  - Security-Related Activities
  - Localization/L10N and Translation
  - Event Organization
  - Documentation Authorship
  - Community Building and Management
  - Teaching and Tutorial Building
  - Troubleshooting and Support
  - Creative Work and Design
  - User Interface, User Experience, and Accessibility
  - Social Media Management
  - User Support and Answering Questions
  - Writing Articles
  - Public Relations - Interviews with Technical Press
  - Speaking at Events
  - Marketing and Campaign Advocacy
  - Website Development
  - Legal Council
  - Financial Management

Data Collection Strategies

- **Interview or Survey:** Ask community members to recognize others for their contributions to recognize contribution types that have previously not been considered.
  - Who in the project would you like to recognize for their contributions? What did they contribute?
- **Observe project:** Identify and recognize leads of different parts of the project.

  - What leaders are listed on the project website or in a repository?
- **Capture Non-code Contributions:** Track contributions through a dedicated system, e.g., an issue tracker.
  - Logging can include custom information a project wants to know about non-code contributions to recognize efforts.
  - Proxy contributions through communication channel activity. For example, If quality assurance members (QA) have their own mailing list, then activity around QA contributions can be measured by proxy from mailing list activity.
- **Collect Trace Data:** Measure contributions through collaboration tool log data.
  - For example, code contributions can be counted from a source code repository, wiki contributions can be counted from a wiki edit history, and email messages can be counted from an email archive
- **Automate Classification:** Train an artificial intelligence (AI) bot to detect and classify contributions.
  - An AI bot can assist in categorizing contributions, for example, help requests vs. support provided, or feature request vs. bug reporting, especially if these are all done in the same issue tracker.

Other considerations:

- Especially with automated reports, allow community members to opt-out and not appear on the contribution reports.
- Acknowledge imperfect capture of contribution types and be explicit about what types of contributions are included.
- As a project evolves, methods for collecting types of contributions will need to adapt. For example, when an internationalization library is exchanged, project activity around localization conceivably produces different metrics before and after the change.
- Account for activity from bots when mining contribution types at large scale.

References

- https://medium.com/@sunnydeveloper/revisiting-the-word-recognition-in-foss-and-the-dream-of-open-credentials-d15385d49447
- https://24pullrequests.com/contributing
- https://smartbear.com/blog/test-and-monitor/14-ways-to-contribute-to-open-source-without-being/
- https://wiki.openstack.org/wiki/AUCRecognition
- https://www.drupal.org/drupalorg/blog/a-guide-to-issue-credits-and-the-drupal.org-marketplace

WHEN

Activity Dates and Times

Question: What are the dates and timestamps of when contributor activities occur?

Description Individuals engage in activities in open source projects at various times of the day. This metric is aimed at determining the dates and times of when individual activities were completed. The data can be used to probabilistically estimate where on earth contributions come from in cases where the time zone is not UTC.

Objectives

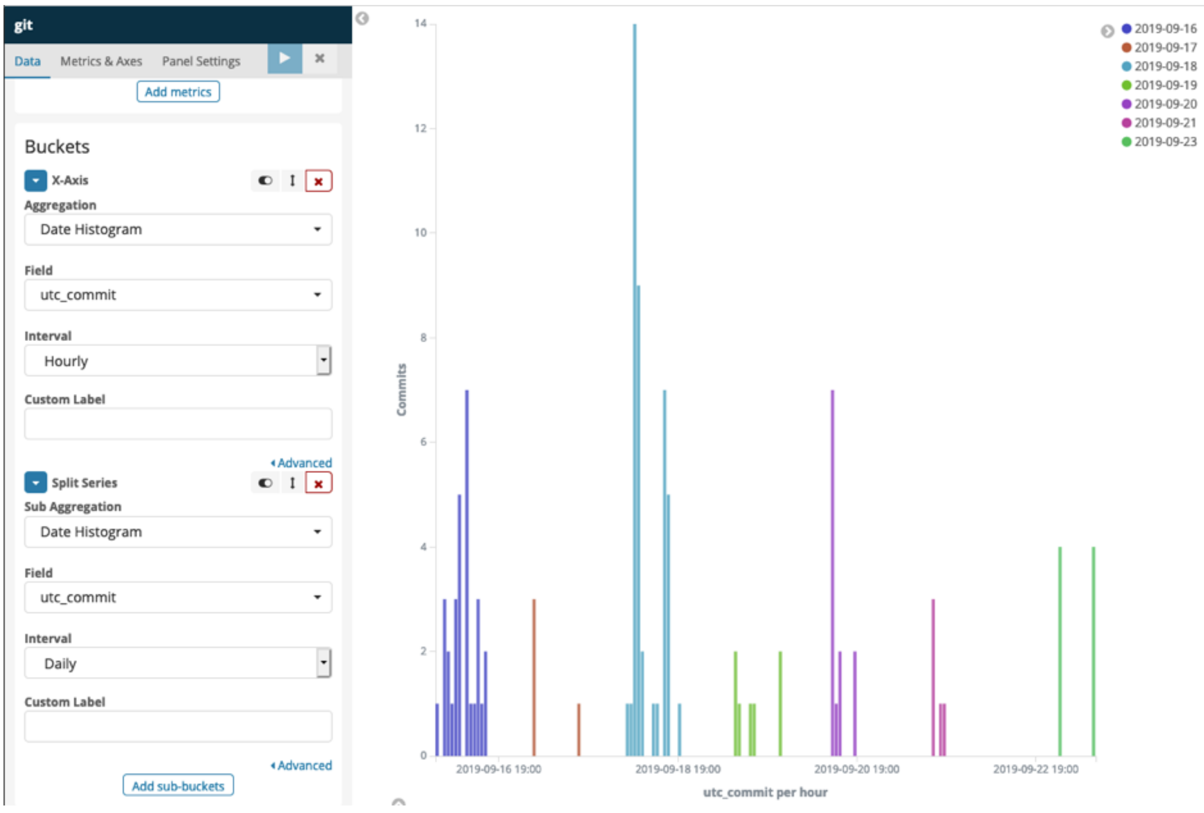
- Improve transparency for employers about when organizational employees are engaging with open source projects
- Improve transparency for open source project and community managers as to when activity is occurring

Implementation

Filters

- Individual by Organization
- Aggregation of time by UTC time
    - Can show what times across the globe contributions are made; when the project is most active.
- Aggregation of time by local time
    - Can show what times of day in their local times they contribute. Conclusions about the If contributions are more during working hours, or if contributions are more during evening hours.
- Repository ID
- Segment of a community, (e.g., GrimoireLab has more EU time zones activity and Augur more US time zones activity)

Visualizations



git

Buckets

X-Axis

Aggregation: Date Histogram

Field: utc_commit

Interval: Hourly

Custom Label:

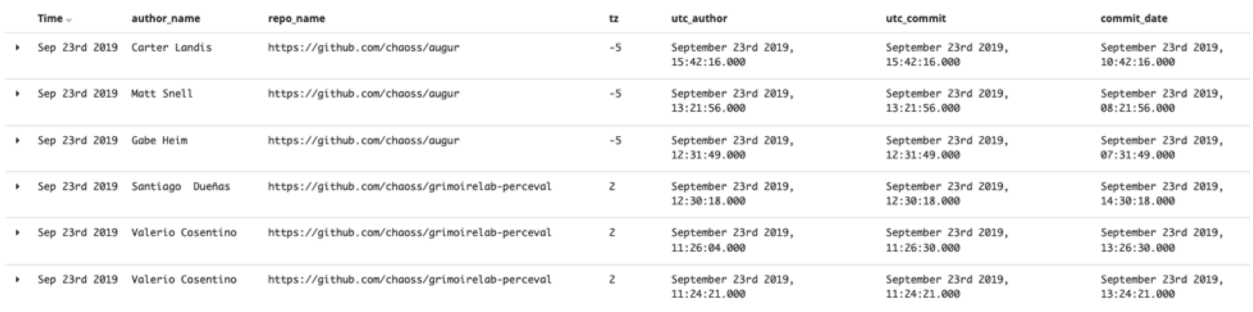
Split Series

Sub Aggregation: Date Histogram

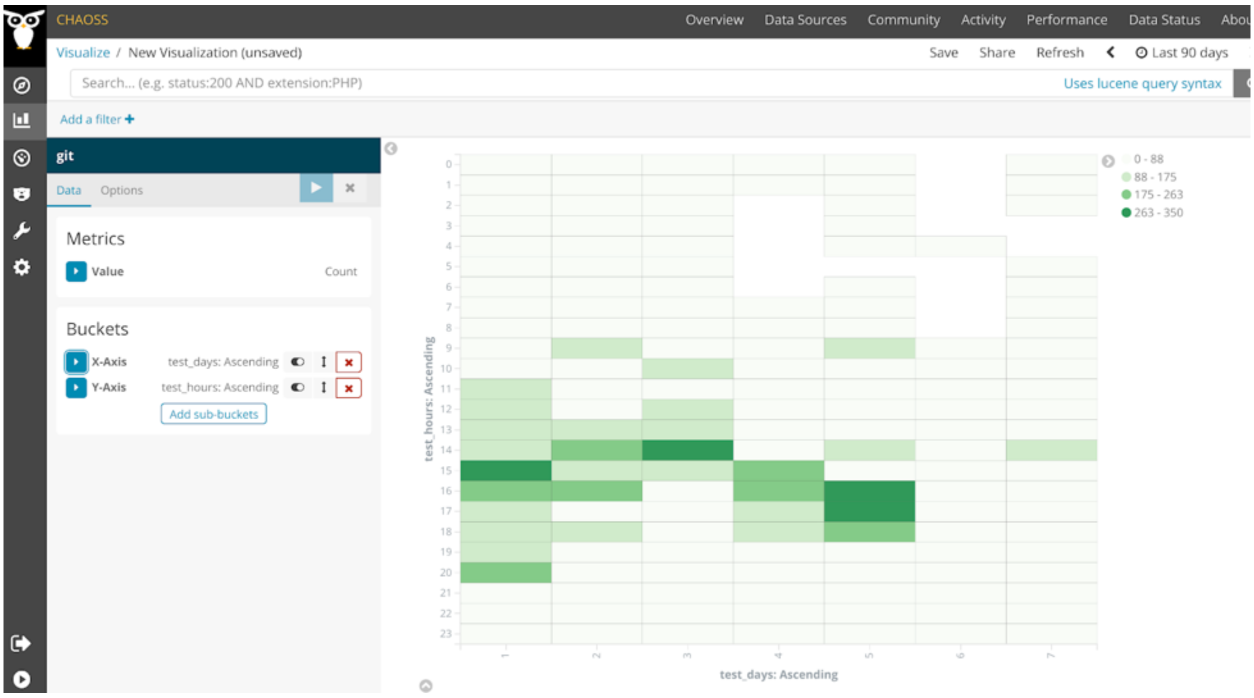
Field: utc_commit

Interval: Daily

Custom Label:



	Time	author_name	repo_name	tz	utc_author	utc_commit	commit_date
▸	Sep 23rd 2019	Carter Landis	https://github.com/chaoss/augur	-5	September 23rd 2019, 15:42:16.000	September 23rd 2019, 15:42:16.000	September 23rd 2019, 10:42:16.000
▸	Sep 23rd 2019	Matt Snell	https://github.com/chaoss/augur	-5	September 23rd 2019, 13:21:56.000	September 23rd 2019, 13:21:56.000	September 23rd 2019, 08:21:56.000
▸	Sep 23rd 2019	Gabe Heim	https://github.com/chaoss/augur	-5	September 23rd 2019, 12:31:49.000	September 23rd 2019, 12:31:49.000	September 23rd 2019, 07:31:49.000
▸	Sep 23rd 2019	Santiago Dueñas	https://github.com/chaoss/grimoirelab-perceval	2	September 23rd 2019, 12:30:18.000	September 23rd 2019, 12:30:18.000	September 23rd 2019, 14:30:18.000
▸	Sep 23rd 2019	Valerio Cosentino	https://github.com/chaoss/grimoirelab-perceval	2	September 23rd 2019, 11:26:04.000	September 23rd 2019, 11:26:30.000	September 23rd 2019, 13:26:30.000
▸	Sep 23rd 2019	Valerio Cosentino	https://github.com/chaoss/grimoirelab-perceval	2	September 23rd 2019, 11:24:21.000	September 23rd 2019, 11:24:21.000	September 23rd 2019, 13:24:21.000



Visualize / New Visualization (unsaved)

Search... (e.g. status:200 AND extension:PHP)

git

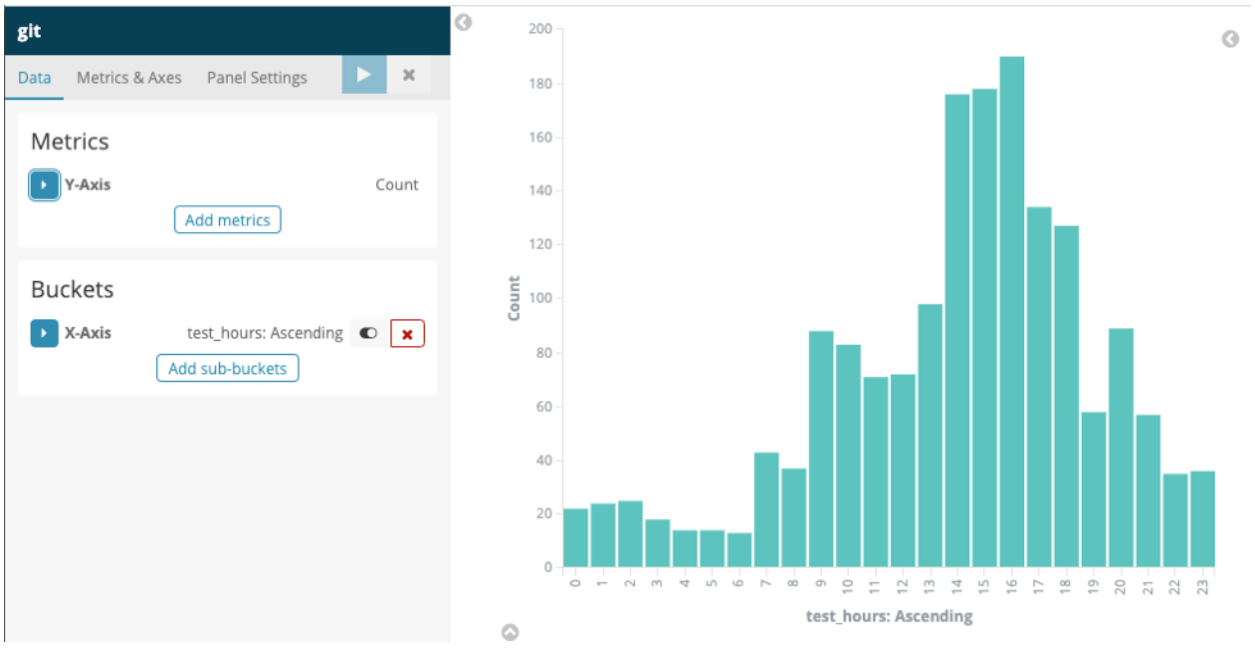
Metrics

Value: Count

Buckets

X-Axis test_days: Ascending

Y-Axis test_hours: Ascending



Tools Providing Metric GrimoireLab

Augur Date/Timestamps

References Coordinated Universal Time

Burstiness

Question: How are short timeframes of intense activity, followed by a corresponding return to a typical pattern of activity, observed in a project?

Description There are a number of reasons that may prompt a sudden increase or decrease in the amount of activity within a repository. These increases and decreases appear both as a sudden change in activity against the average amount of activity. Burstiness is a way of understanding the cycle of activity in existing metrics, like issues, merge requests, mailing lists, commits, or comments. Examples of root causes for bursts in activity include:

- Release cycles
- Global pandemics
- Hackathon activities
- Mentorship programs
- Conferences, meetups, and other events where tools are presented
- Conventional and social media announcements and mentions
- Critical bugs as raising awareness and getting people's attention
- Community design meetings or brainstorming meetings to address a particular issue
- Community members show up from another community that is relying on your project (e.g., dependencies)

Objectives

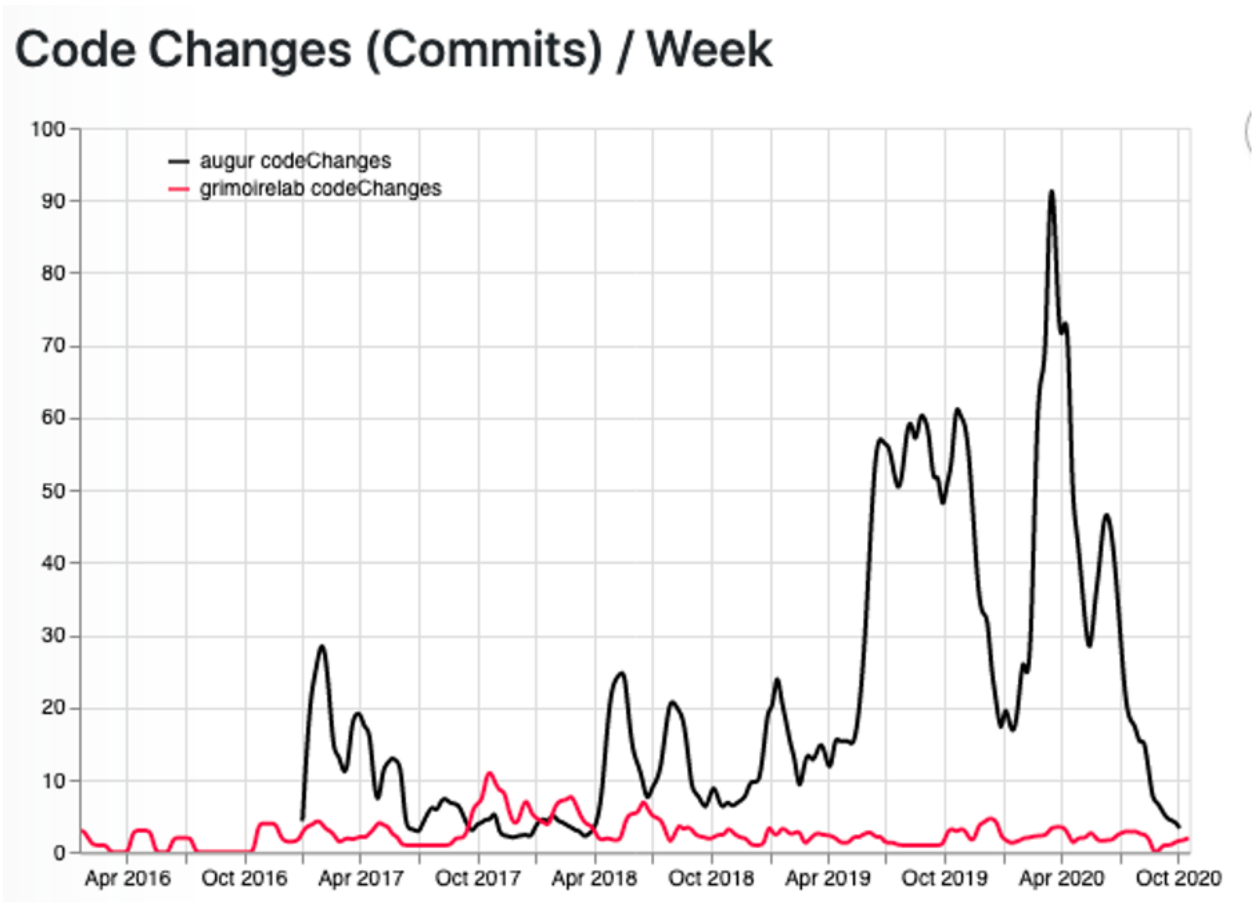
- To identify impacts of root causes of a burst in activity
- To provide awareness when project activity unknowingly goes up
- To help capture the meaningfulness of increases or decreases in project activity
- To help the community and maintainers prepare for future bursts that follow a pattern
- To help measure the impact of influential external activities
- To differentiate skewed activity versus normal activity

Implementation

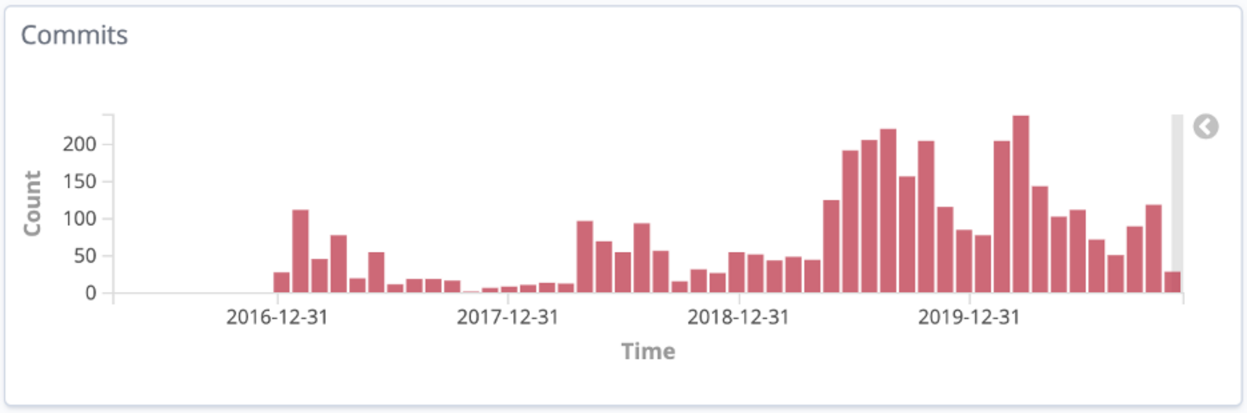
Filters

- Stars
- Forks
- Issues or bug reports
- Labels
- Downloads
- Release Tags
- Change Requests
- Mail List Traffic
- Documentation additions or revisions
- New Repositories
- Feature Requests
- Messaging Conversations
- Conventional and Social Media Activity
- Conference Attendance and Submissions

Visualizations Augur:



GrimoireLab:



Tools Providing the Metric

- Grimoire Lab
- Augur

Data Collection Strategies

- Quantitative
  - Time box activities identifying deviations away from some norm
  - Outliers for certain thresholds, using statistics like Bollinger Bands to measure stability or volatility: https://en.wikipedia.org/wiki/Bollinger_Bands
- Qualitative Interview Questions

- Why do you contribute more during a period of time?
- What do you believe to be the root cause for particular bursts?
- What impact do different events (e.g., hackathons, mentorship program, or conferences) have on project activity?

References This metric was inspired by the work of Goh and Barabasi (2008): https://arxiv.org/pdf/physics/

Review Cycle Duration within a Change Request

Question: What is the duration of a review cycle within a single change request?

Description A change request is based on one or more review cycles. Within a review cycle, one or more reviewers can provide feedback on a proposed contribution. The duration of a review cycle, or the time between each new iteration of the contribution, is the basis of this metric.

Objectives This metric provides maintainers with insight on: Code review process decay, as there are more iterations and review cycle durations increase. Process bottlenecks resulting in long code review iterations. Abandoned or semi-abandoned processes in the review cycles, where either the maintainer or the submitter is slow in responding. Characteristics of reviews that have different cyclic pattern lengths.

Implementation Review Cycle Duration is measured as the time length of one review cycle within a single change request. The duration can be calculated between: The moment when each review cycle begins, defined as the point in time when a change request is submitted or updated. The moment when each review cycle ends, either because the change request was updated and needs a new review or because it was accepted or rejected.

Filter Average or Median Duration, optionally filtered or grouped by: Number of people involved in review Number of comments in review Edits made to a change request Project or program Organization making the change request Time the change request was submitted Developers who contributed to a change request Change request Number of review cycle on a change request (e.g., filter by first, second, ... round)

Visualizations

Tools Providing the Metric

References Example of data that could be used to develop the metric: https://gerrit.wikimedia.org/r/c/med

Time to Close

Question: How much time passes between creating and closing an operation such as an issue, change request, or support ticket?

Description The time to close is the total amount of time that passes between the creation and closing of an operation such as an issue, change request, or support ticket. The operation needs to have an open and closed state, as is often the case in code review processes, question and answer forums, and ticketing systems.

Related metric: Issue Resolution Duration

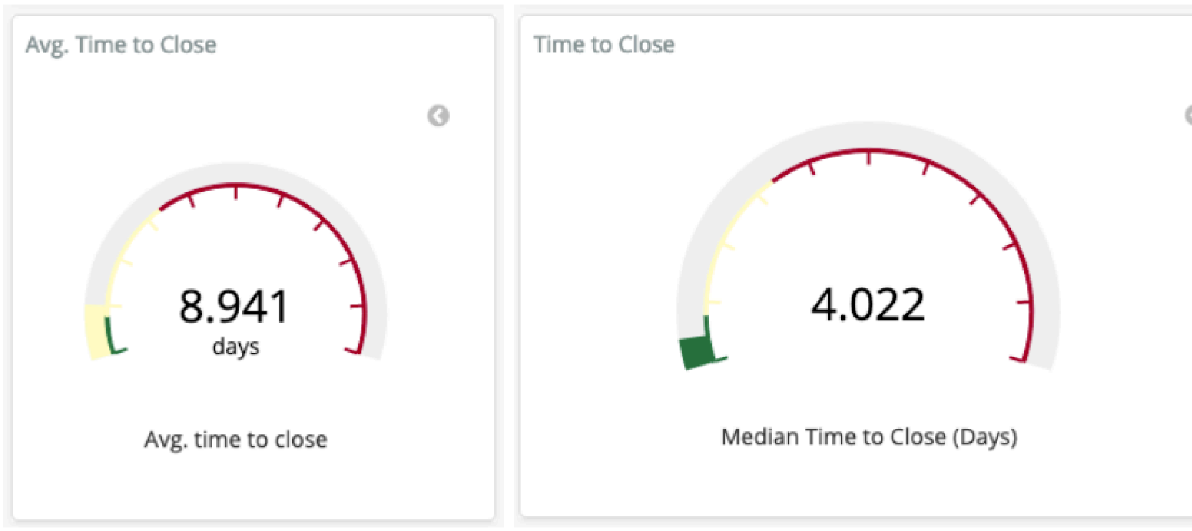
Objectives

1. Determining how responsive a community is can help efforts to be inclusive, attract, and retain new and existing contributors.
2. Identifying characteristics of operations that impact an operation closing quickly or slowly (e.g., finding best practices, areas of improvement, assess efficiency).
3. Identifying bias for timely responses to different community members.
4. Detecting a change in community activity (e.g., to indicate potential maintainer burnout, reduction in the diversity of contributions)

Implementation

Filters

- Creator of operation (e.g., new contributor vs. maintainer)
- First closed, final closed
- issue labels (e.g., bug vs. new feature)



Visualizations

Tools Providing the Metric Augur implementation:

- Issue Close Duration
- Issue Duration
- Issue Response Time

GrimoireLab implementation:

- Pull Requests Efficiency
- Issues Efficiency
- Efficiency:TimingOverview

Data Collection Strategies The time to close metric may be contextual based on the project activity and objectives. For example, the time to close a bug report may provide different information than the time to close a new feature request. Data collection strategies should address different project objectives. Other variables that may influence these processes are:

- Issue Tracking Systems: the type of issue such as bug report, blueprint (OpenStack nomenclature), user story, feature request, epic, and others may influence how long this event takes to be closed. Other variables, such as the priority or severity may help to advance how quickly this event will be closed.
- Change Request Processes: this depends on the change request infrastructure, as Gerrit, GitHub or mailing lists (as in the Linux Kernel) and may differ depending on how complicated the process is. For example, newcomers or advanced and experienced developers will proceed in different ways and with more or less time required.
- Question and Answer Forum: this depends on the quality of the answer and the opinion of the person asking the question. A valid answer is marked, and the process is closed once the person questioning has successfully found a correct answer to their question.

References

- "Practice P.12: Respond to all submissions" from "Appendix to: Managing Episodic Volunteers in Free/Libre/Open Source Software Communities" by Ann Barcomb, Klaas-Jan Stol, Brian Fitzgerald and Dirk Riehle: https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/13519

Time to First Response

Question: How much time passes between when an activity requiring attention is created and the first response?

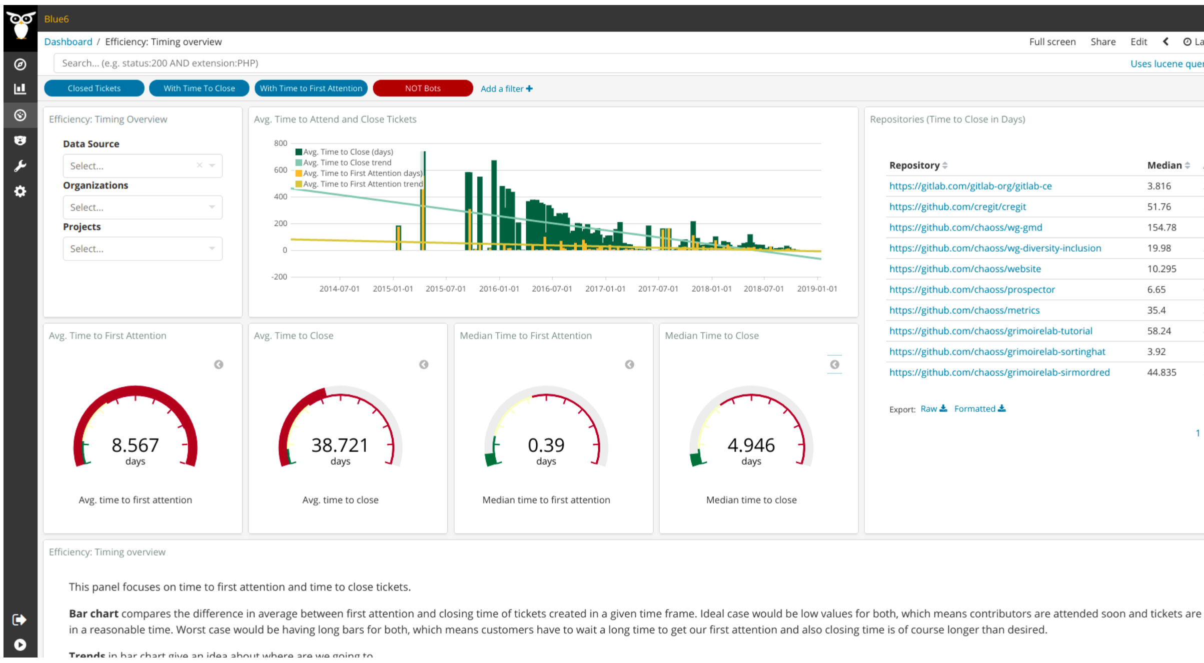
Description The first response to an activity can sometimes be the most important response. The first response shows that a community is active and engages in conversations. A long time to respond to an activity can be a sign that a community is not responsive. A short time to respond to an activity can help to engage more members into further discussions and within the community.

Objectives Identify cadence of first response across a variety of activities, including PRs, Issues, emails, IRC posts, etc. Time to first response is an important consideration for new and long-time contributors to a project along with overall project health.

Implementation Time to first response of an activity = time first response was posted to the activity - time the activity was created.

Filters

- Role of responder, e.g., only count maintainer responses
- Automated responses, e.g., only count replies from real people by filtering bots and other automated replies
- Type of Activity, e.g., issues (see metric Issue Response Time), emails, chat, change requests



Repositories (Time to Close in Days)

Repository	Median
https://gitlab.com/gitlab-org/gitlab-ce	3.816
https://github.com/cregit/cregit	51.76
https://github.com/chaoss/wg-gmd	154.78
https://github.com/chaoss/wg-diversity-inclusion	19.98
https://github.com/chaoss/website	10.295
https://github.com/chaoss/prospector	6.65
https://github.com/chaoss/metrics	35.4
https://github.com/chaoss/grimoirelab-tutorial	58.24
https://github.com/chaoss/grimoirelab-sortinghat	3.92
https://github.com/chaoss/grimoirelab-sirmordred	44.835

8.567 days — Avg. time to first attention

38.721 days — Avg. time to close

0.39 days — Median time to first attention

4.946 days — Median time to close

Efficiency: Timing overview

This panel focuses on time to first attention and time to close tickets.

Bar chart compares the difference in average between first attention and closing time of tickets created in a given time frame. Ideal case would be low values for both, which means contributors are attended soon and tickets are in a reasonable time. Worst case would be having long bars for both, which means customers have to wait a long time to get our first attention and also closing time is of course longer than desired.

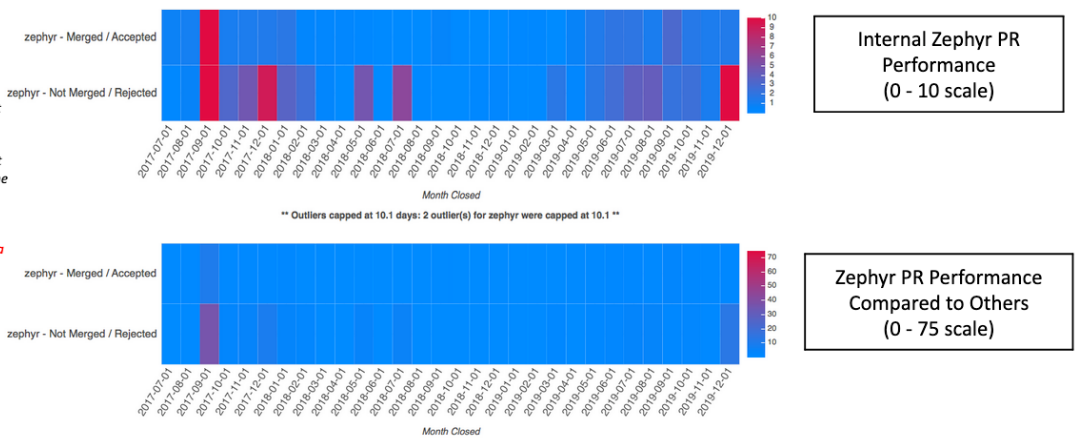
Trends in bar chart give an idea about where are we going to

Visualizations

Mean Days to First Response for Closed Pull Requests

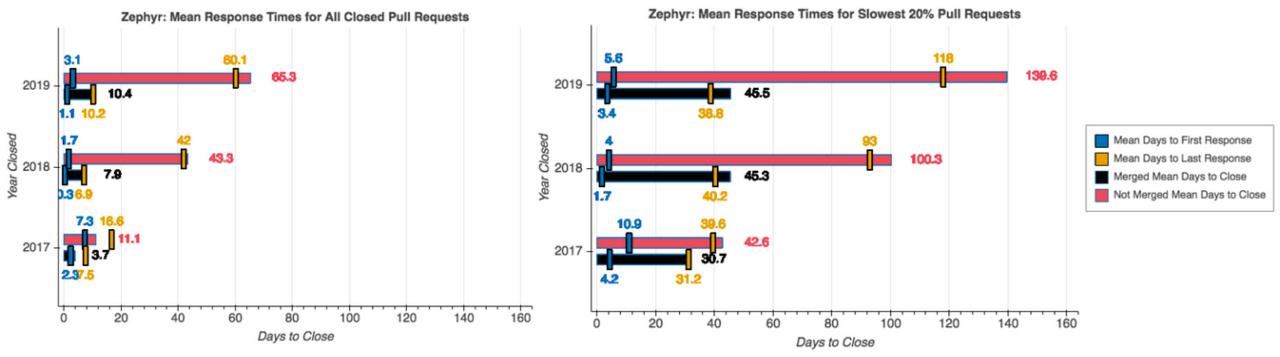
Some Internal Slowing, But Outperforming Other Repositories

The heat maps illustrate variance in mean pull request duration to the first response for Zephyr over time. The left side compares Zephyr against itself, and the heat map on the right side is on a scale more aligned with the competition. Note especially that the internal Zephyr gradient has a max of 10 days, while the external comparisons have a max of 75 days.



Mean Response Times (Days) For Closed Pull Requests

Long Running Pull Requests Are Usually Rejected



The length of the black bar illustrates the total number of days that rejected and merged pull requests were open. The blue bar shows that, for the Zephyr project, we see the mean time to first response improving significantly for both rejected and merged pull requests from 2017 - 2019. The orange bar shows the last response or event associated with a pull request.

Tools Providing the Metric

- GrimoireLab Panel: Efficiency Timing Overview
- Kata Containers dashboard efficiency panel

References

WHO

Contributor Location

Question: What is the location of contributors?

Description Geographical location from which contributors contribute, where they live, or where they work.

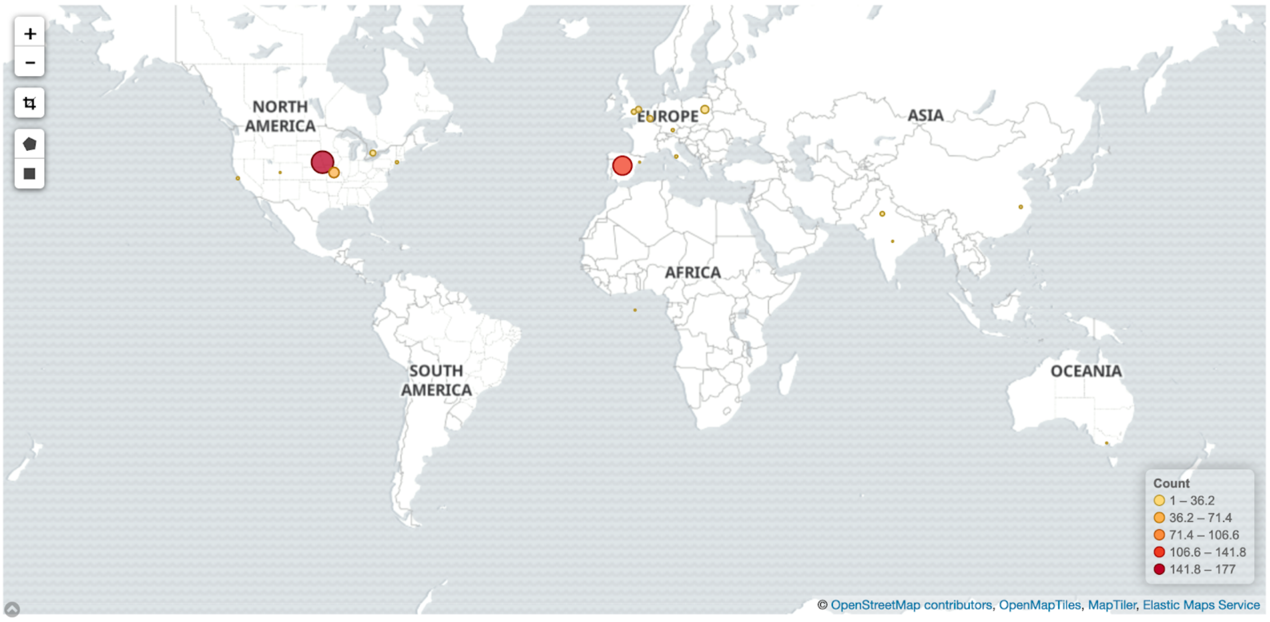
Objectives To determine global locations of contributors in an effort to understand work practices and times zones. To identify where contributions do not come from in an effort to improve engagement in these areas.

Implementation

Filters Filter contributions by:

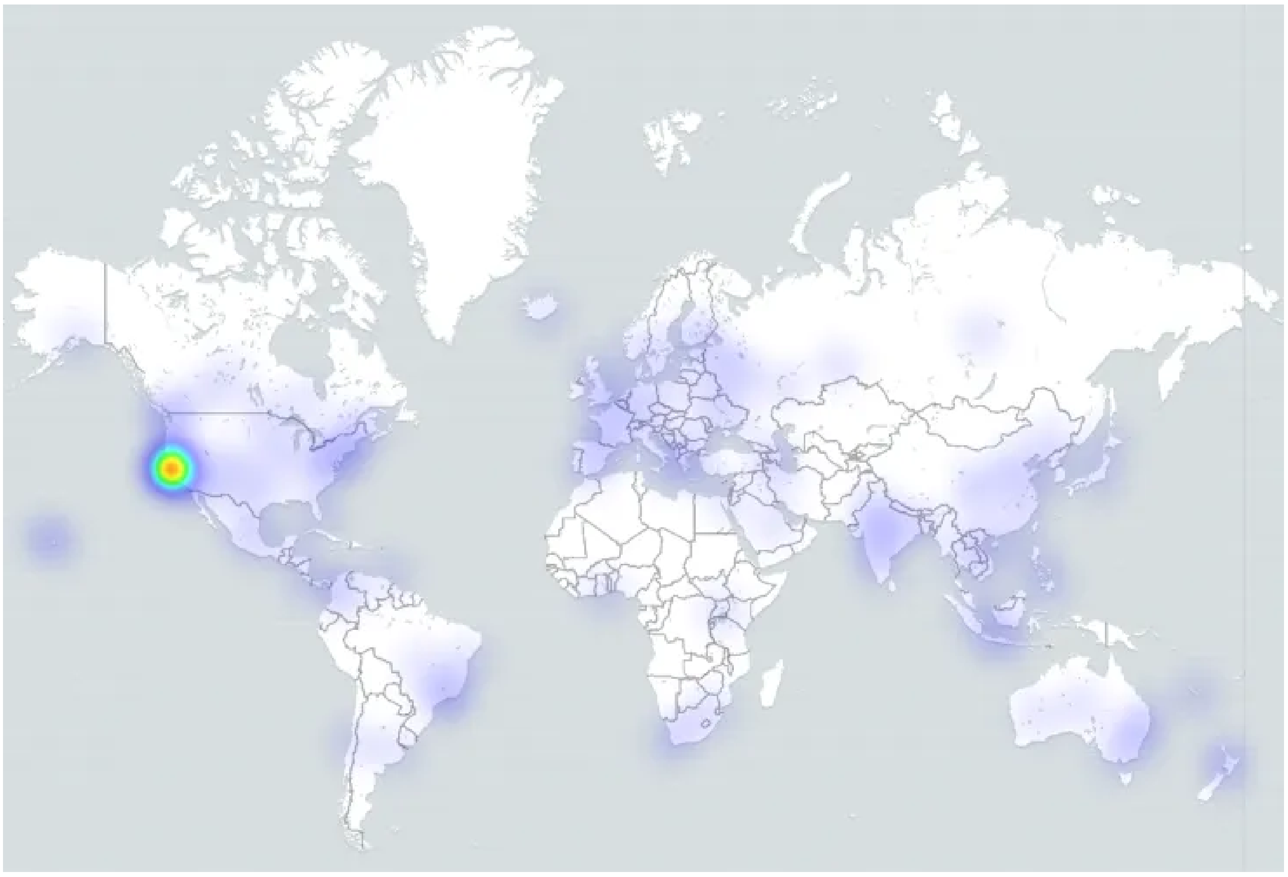
- **Location.** Attempt to group locations in regions to have multiple levels of reporting. Location is a purposely ambiguous term in this context, and could refer to region, country, state, locale, or time zone.
- **Period of time.** Start and finish date of the period. Default: forever. Period during which contributions are counted.
- **Type of contributor**, for example:
  - Repository authors
  - Issue authors
  - Code review participants
  - Mailing list authors
  - Event participants
  - IRC authors
  - Blog authors
  - By release cycle
  - Programming languages of the project
  - Role or function in project

Visualizations Dot Density Map:



Source: https://chaoss.biterg.io/goto/a62f3584a41c1c4c1af5d04b9809a860

Visual heat map:



Source: https://blog.bitergia.com/2018/11/20/ubers-community-software-development-analytics-for-open-source-offices

Tools providing the metric

- GrimoireLab
- Augur

Data Collection Strategies Different approaches can be used to collect information about location:

- Collect the location information from a contributor's profile in the system of engagement.
- Use IP address geolocation of the most frequent locations that contributions are made.
- Infer geographical location from the timestamp in contributions.
- Survey contributors.

The key challenge for collecting data is determining the location of the contributor. Best practice would be to leverage any profile information available from the system of engagement, and if that is not available then use IP geolocation to determine the most frequent location of contribution from that individual. Note that contributors may enter in their profile information false or nonsensical location information (e.g., "Earth" or "Internet"). Note that IP geolocation can provide large numbers of false positives due to use of VPNs or other IP masking tools.

An additional consideration would be the use of external data collection tools such as community surveys or event registration data that could cross reference systems of engagement profiles. Contributor location data could be collected inline with event attendee demographics and speaker demographics.

References

- Gonzalez-Barahona, J. M., Robles, G., Andradas-Izquierdo, R., & Ghosh, R. A. (2008). Geographic origin of libre software developers. *Information Economics and Policy*, *20*(4), 356-363.

Contributors

Question: Who are the contributors to a project?

Description A contributor is defined as anyone who contributes to the project in any way. This metric ensures that all types of contributions are fully recognized in the project.

Objectives Open source projects are comprised of a number of different contributors. Recognizing all contributors to a project is important in knowing who is helping with such activities as code development, event planning, and marketing efforts.

Implementation Collect author names from collaboration tools a project uses.

Aggregators:

- Count. Total number of contributors during a given time period.

Parameters:

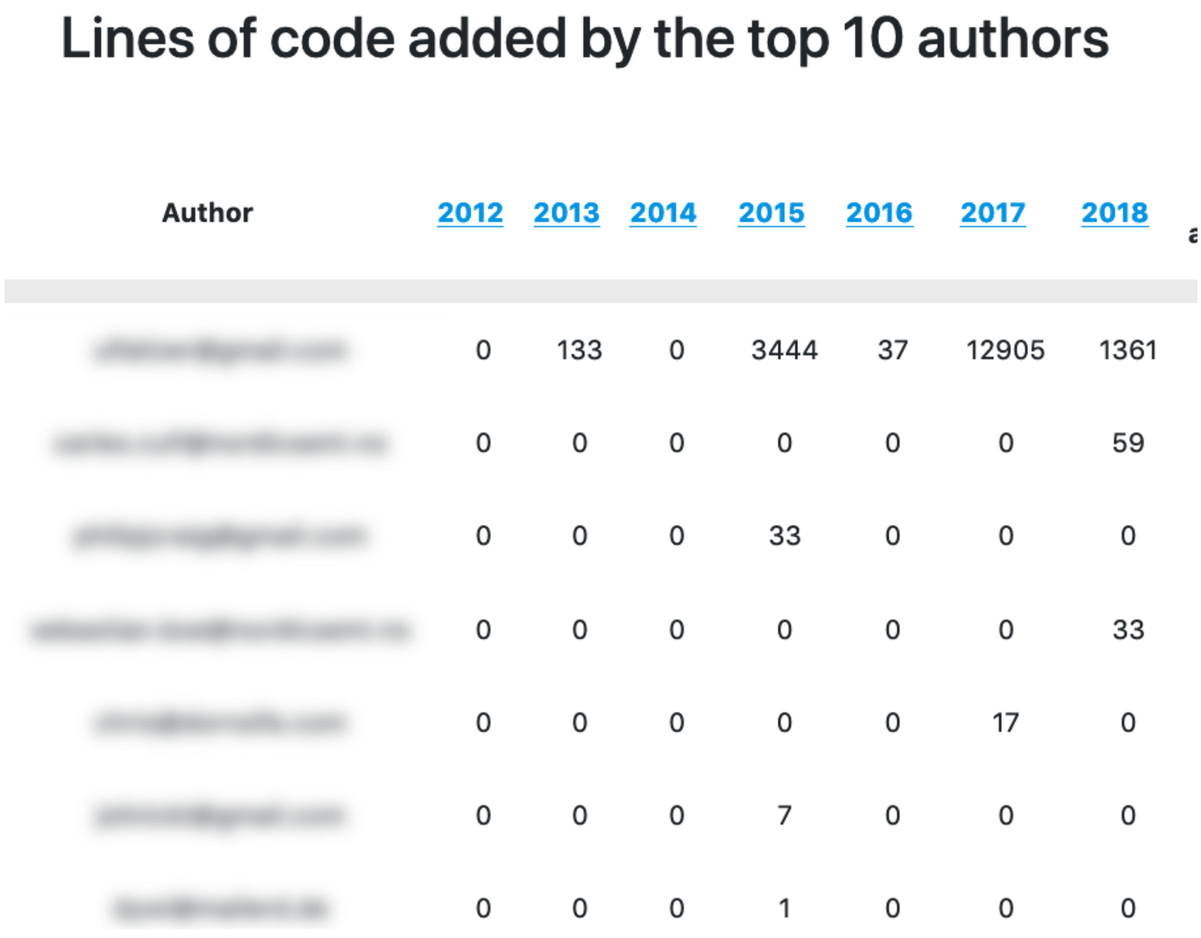
- Period of time. Start and finish date of the period. Default: forever. Period during which contributions are counted.

Filters By location of engagement. For example:

- Commit authors
- Issue authors
- Review participants, e.g., in pull requests
- Mailing list authors
- Event participants
- IRC authors
- Blog authors
- By release cycle
- Timeframe of activity in the project, e.g, find new contributors
- Programming languages of the project
- Role or function in project

Visualizations

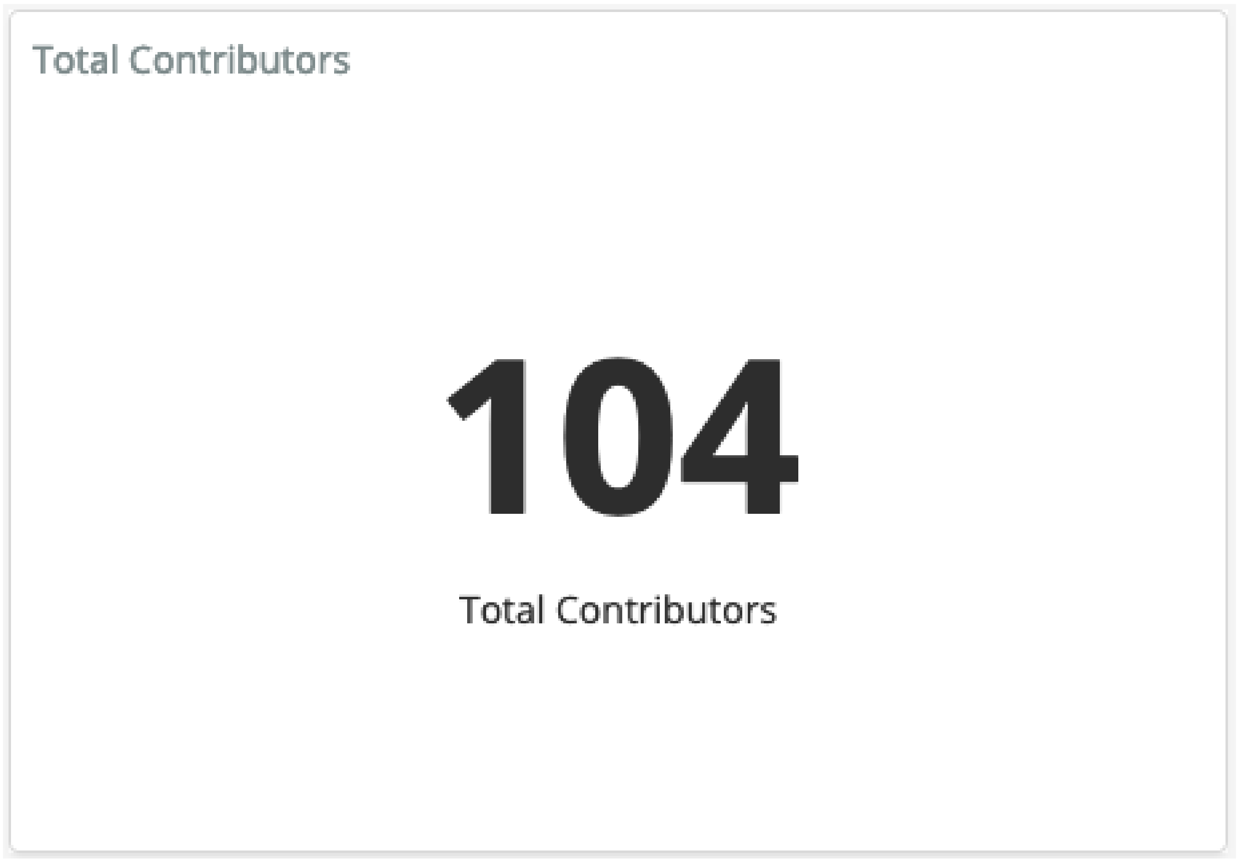
1. List of contributor names (often with information about their level of engagement)



Lines of code added by the top 10 authors

Author	2012	2013	2014	2015	2016	2017	2018
[redacted]	0	133	0	3444	37	12905	1361
[redacted]	0	0	0	0	0	0	59
[redacted]	0	0	0	33	0	0	0
[redacted]	0	0	0	0	0	0	33
[redacted]	0	0	0	0	0	17	0
[redacted]	0	0	0	7	0	0	0
[redacted]	0	0	0	1	0	0	0

2. Summary number of contributors

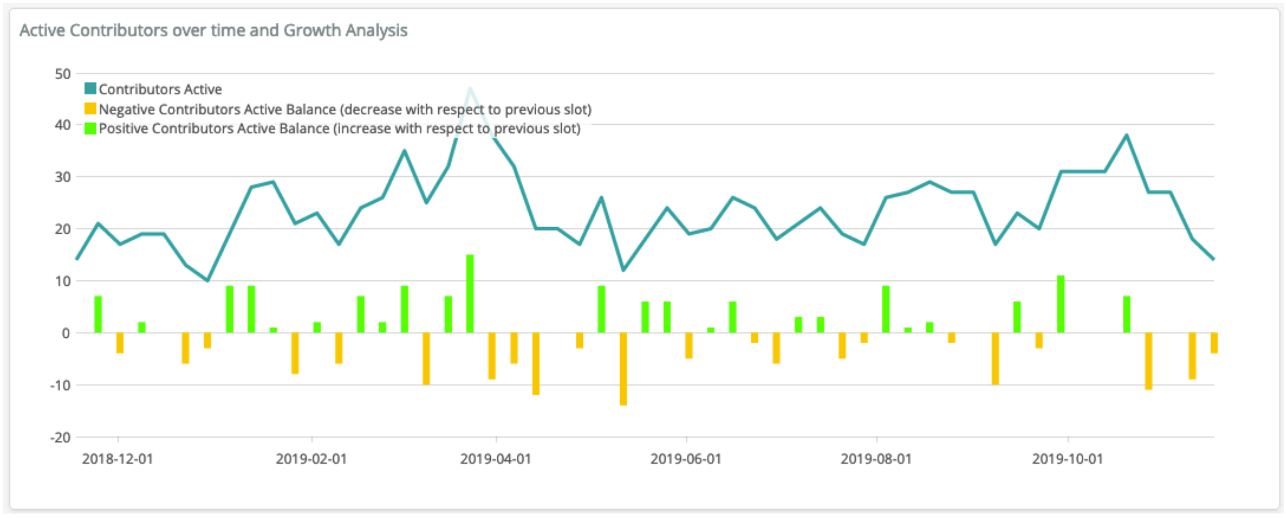


Total Contributors

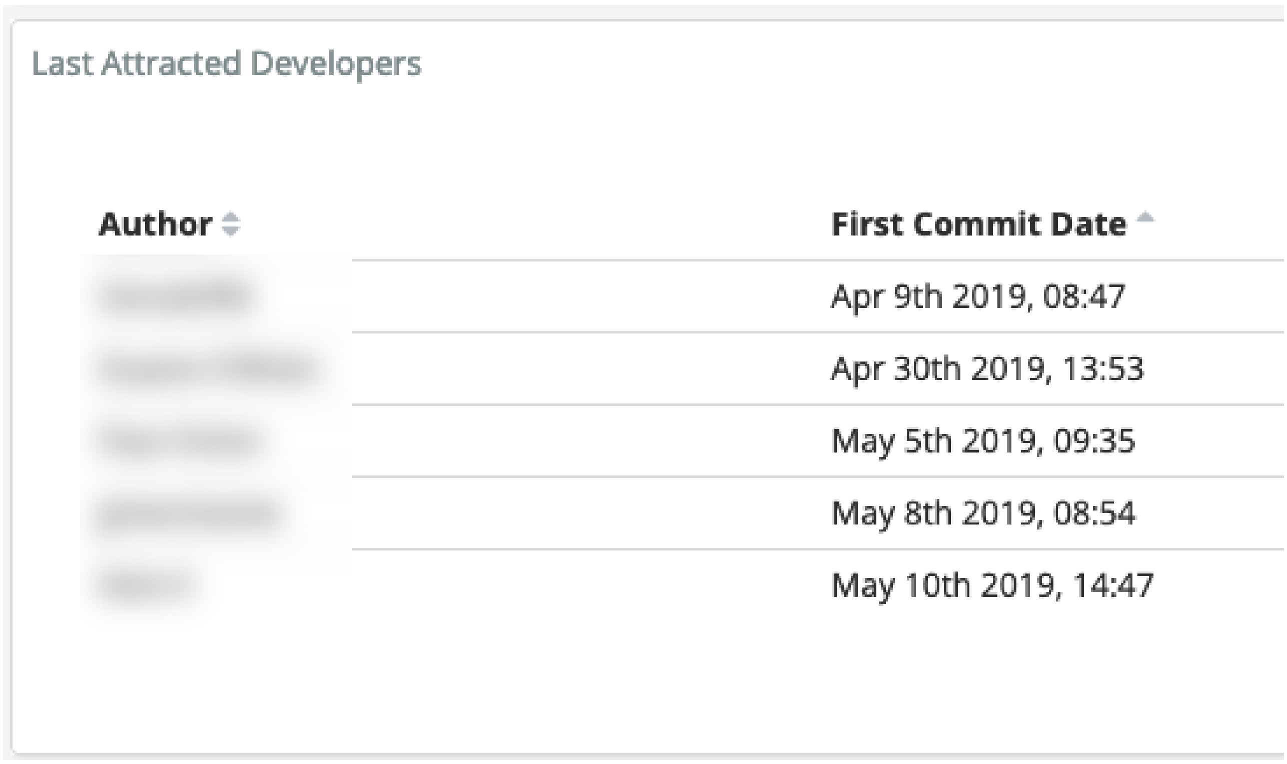
104

Total Contributors

3. Change in the number of active contributors over time



4. New contributors (sort list of contributors by date of first contribution)


Last Attracted Developers

Author	First Commit Date
[redacted]	Apr 9th 2019, 08:47
[redacted]	Apr 30th 2019, 13:53
[redacted]	May 5th 2019, 09:35
[redacted]	May 8th 2019, 08:54
[redacted]	May 10th 2019, 14:47

Tools Providing the Metric

- GrimoireLab
- Augur

Data Collection Strategies As indicated above, some contributor information is available via software such as GrimoireLab and Augur. However, some contributor insights are less easily obtained via trace data. In these cases, surveys with community members or event registrations can provide the desired information. Sample questions include:

- Interview question: Which contributors do not typically appear in lists of contributors?
- Interview question: Which contributors are often overlooked as important contributors because their contributions are more "behind the scenes"?
- Interview question: What other community members do you regularly work with?

Additionally, surveys with community members can provide insight to learn more about contributions to the project. Sample questions include:

- Likert scale [1-x] item: I am contributing to the project
- Matrix survey item: How often do you engage in the following activities in the project?
  - Column headings: Never, Rarely(less than once a month), Sometimes (more than once a month), Often(once a week or more)
  - Rows include: a) Contributing/reviewing code, b) Creating or maintaining documentation, c) Translating documentation, d) Participating in decision making about the project's development, e) Serving as a community organizer, f) Mentoring other contributors, g) Attending events in person, h) Participating through school or university computing programs, i) Participating through a program like Outreachy, Google Summer of Code, etc., j) Helping with the ASF operations (e.g., board meetings or fundraising)

References

Organizational Diversity

Question: What is the organizational diversity of contributions?

Description Organizational diversity expresses how many different organizations are involved in a project and how involved different organizations are compared to one another.

Objectives

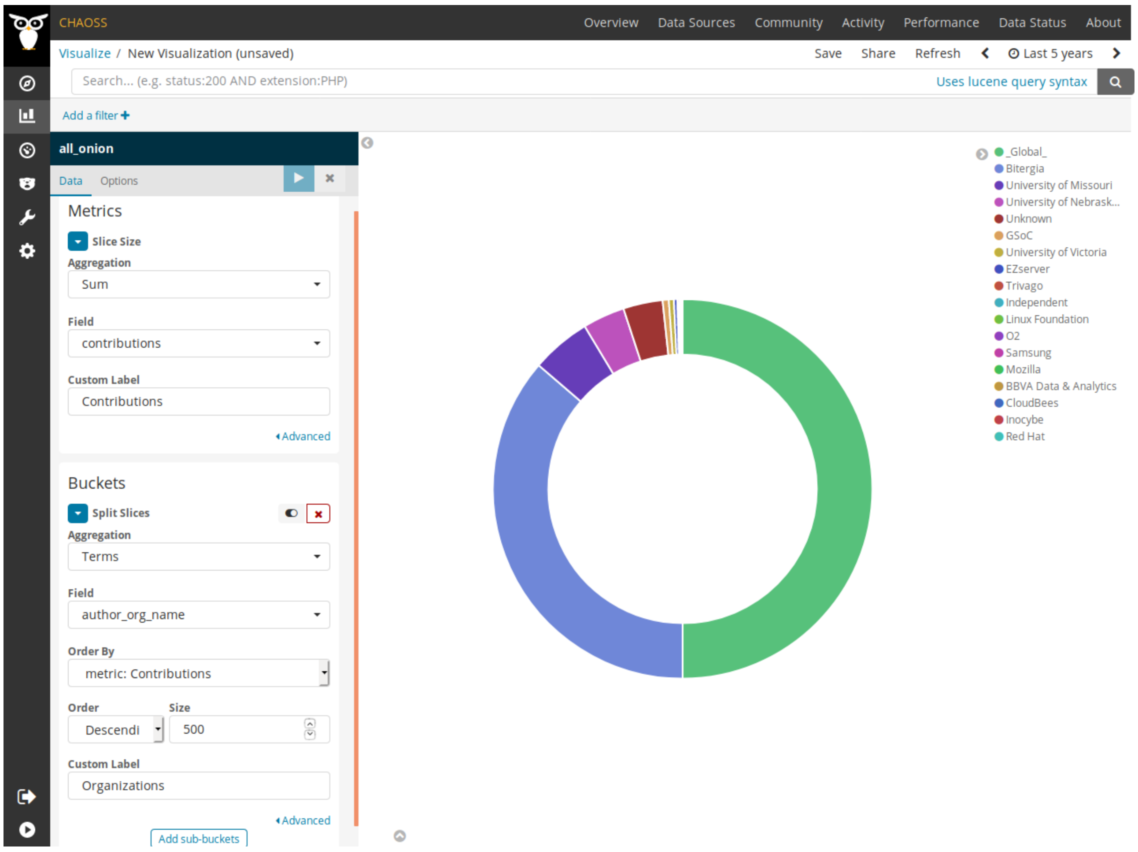
- Get a list of organizations contributing to a project.
- See the percentage of contributions from each organization within a defined period of time.
- See the change of composition of organizations within a defined period of time.
- Get a list of people that are associated with each organization.

Implementation

- Collect data from data sources where contributions occur.
- Identify contributor affiliations to get a good estimate of which organizations they belong to.
- Correlate information about contributions, assigning each to appropriate organization.
- Depending on the needs of the project, you may want to consider such issues as how to handle multiple email addresses, affiliation changes over time, or contractor vs. employee.

Tools Providing the Metric

- GrimoireLab supports organizational diversity metrics out of the box. The GrimoireLab SortingHat manages identities. The GrimoireLab Hatstall user interface allows correcting organizational affiliation of people and even recording affiliation changes.
  - View an example visualization on the CHAOSS instance of Bitergia Analytics.
  - Download and import a ready-to-go dashboard containing examples for this metric visualization from the GrimoireLab Sigils panel collection.
  - Add a sample visualization to any GrimoreLab Kibiter dashboard following these instructions:
    - Create a new Pie chart
      - Select the `all_onion` index
      - Metrics Slice Size: `Sum` Aggregation, `contributions` Field, `Contributions` Custom Label
      - Buckets Split Slices: `Terms` Aggregation, `author_or_name` Field, `metric: Contributions` Order By, `Descending` Order, `500` Size, `Organization` Custom Label
    - Example Screenshot



CHAOSS Overview Data Sources Community Activity Performance Data Status About

Visualize / New Visualization (unsaved) Save Share Refresh Last 5 years

Search... (e.g. status:200 AND extension:PHP) Uses lucene query syntax

Add a filter +

all_onion

Data Options

Metrics

Slice Size

Aggregation: Sum

Field: contributions

Custom Label: Contributions

Advanced

Buckets

Split Slices

Aggregation: Terms

Field: author_org_name

Order By: metric: Contributions

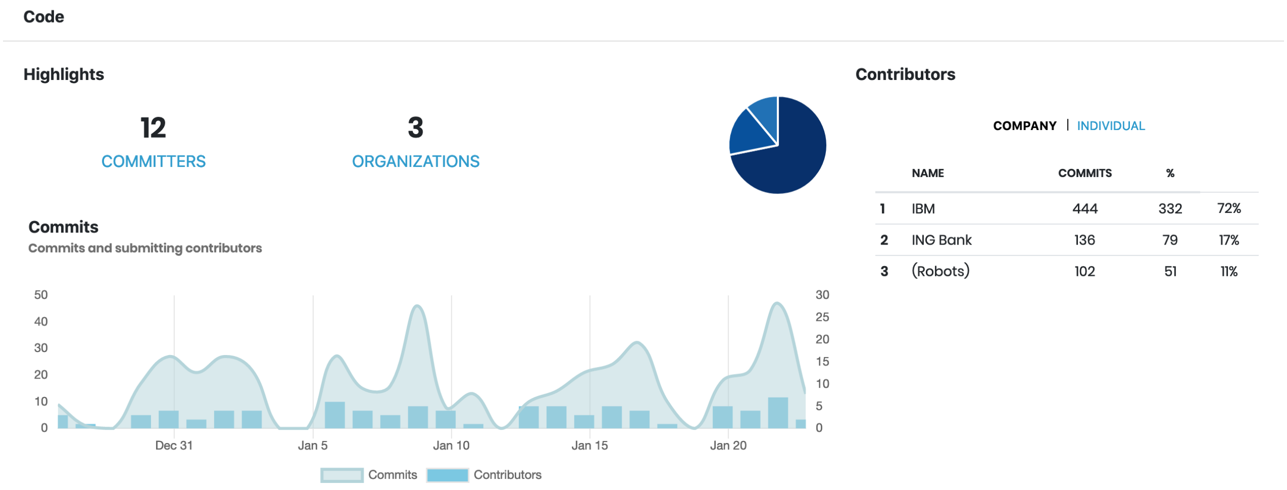
Order: Descendi Size: 500

Custom Label: Organizations

Advanced

Add sub-buckets

- LF Analytics provides organization diversity metrics in the primary view for commits, issues filed, and communication channels (current support for Slack and groups.io)



Code

Highlights

12 COMMITTERS

3 ORGANIZATIONS

Contributors

COMPANY | INDIVIDUAL

	NAME	COMMITS		%
1	IBM	444	332	72%
2	ING Bank	136	79	17%
3	(Robots)	102	51	11%

Commits

Commits and submitting contributors

Data Collection Strategies Qualitative

- Footprint of an organization in a project or ecosystem
- Influence of an organization in a project or ecosystem
- Affiliation diversity in governance structures.

Quantitative

- % of commits by each organization
- % of merges/reviews from each organization
- % of any kind of contributors from each organization

- % of lines of code contributed by each organization
- % issues filed by each organization
- Contributing Organizations - What is the number of contributing organizations?
- New Contributing Organizations - What is the number of new contributing organizations?
- New Contributor Organizations - New organizations contributing to the project over time.
- Number of Contributing Organizations - Number of organizations participating in the project over time.
- Elephant Factor - If 50% of community members are employed by the same company, it is the elephant in the room. Formally: The minimum number of companies whose employees perform 50% of the commits
- Affiliation Diversity - Ratio of contributors from a single company over all contributors. Also described as: Maintainers from different companies. Diversity of contributor affiliation.
- In projects with the concept of code ownership, % of code owners affiliated with each organization weighed by the importance/size/LoC of the code they own and the number of co-owners.

References

- Potential implementations and references:
  - https://bitergia.gitlab.io/panel-collections/open_source_program_office/organizational-diversity.html
  - Kata Containers dashboard entry page (bottom of this)
  - Augur

WG-VALUE

COMMUNAL-VALUE

Project Popularity

Question: How popular is an open source project?

Description Project popularity can be measured by how much activity is visible around a project. Popularity has a positive feedback loop in which more popular projects get more attention, attract more users or developers, and see increases in popularity, spinning the popularity wheel.

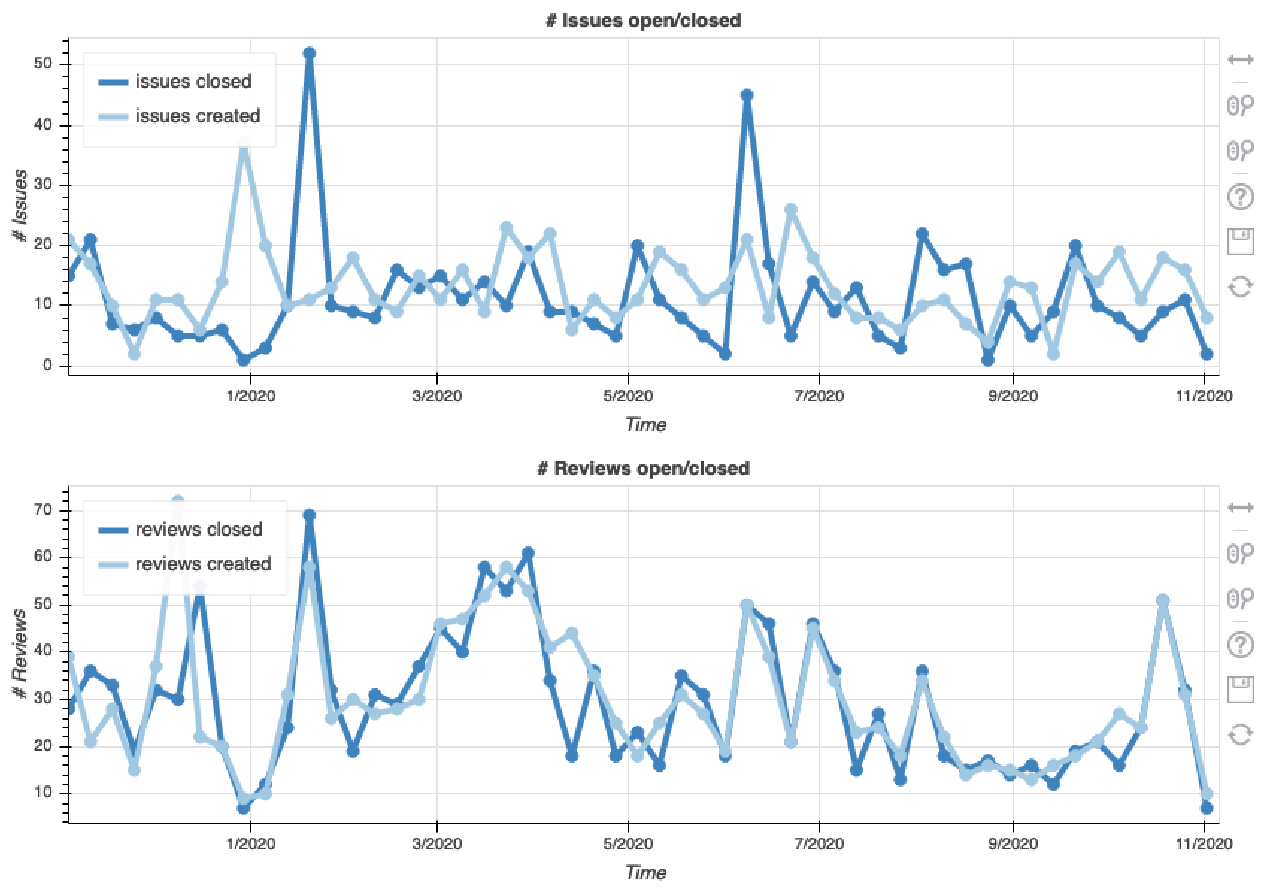
Project popularity may be used as a proxy for understanding project value because open source project economic value is hard to measure, due to a lack of available usage or sales information for open source projects.

Objectives In a quest to earn a living wage, and to maximize future employment opportunities, workers may be interested in knowing which projects are growing and are underserved. Similarly, from an organizational perspective, knowing which projects are highly used can be helpful in knowing which projects might be worth investing in. The Project Popularity metric can be used to identify the trajectory of a project's development.

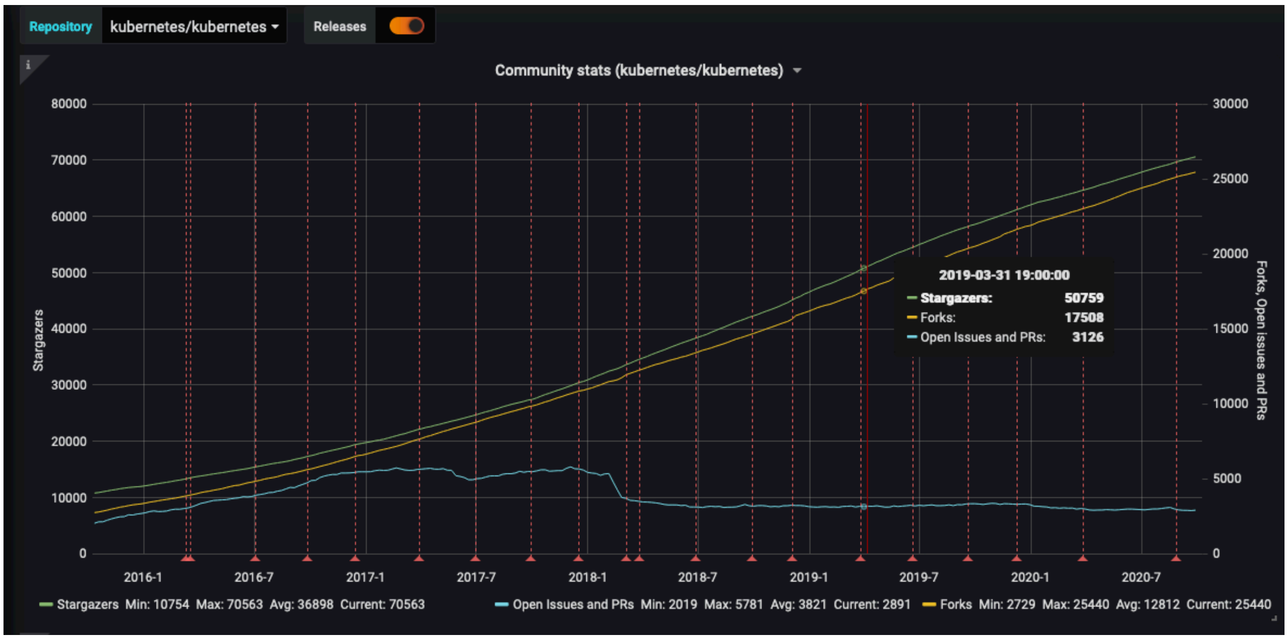
Implementation The project popularity metric is often considered with changes over time. There are numerous example vectors to consider when measuring project popularity based on the number of:

1. Social media mentions
2. Forks
3. Change requests
4. New Issues
5. Stars, badges, likes
6. New contributors
7. Organizational Diversity
8. Job postings requesting skills in project
9. Conversations within and outside of project
10. Clones
11. Followers
12. Downstream dependencies
13. People attending events that focus on a project

Visualizations Issues and reviews (change requests) visualization from Cauldron (GrimoireLab):



Kubernetes project popularity statistics from DevStats:



Stargazers Min: 10754 Max: 70563 Avg: 36898 Current: 70563
Open Issues and PRs Min: 2019 Max: 5781 Avg: 3821 Current: 2891
Forks Min: 2729 Max: 25440 Avg: 12812 Current: 25440

Tools Providing the Metric

- Augur
- GrimoireLab
- Cauldron

References

- Popular OpenSource Projects
- Is It Maintained?
- Open Source Project Trends
- Kubernetes Salary

Project Velocity

Question: What is the development speed for an organization?

Description Project velocity is the number of issues, the number of pull requests, volume of commits, and number of contributors as an indicator of 'innovation'.

Objectives Gives an Open Source Program Office (OSPO) manager a way to compare the project velocity across a portfolio of projects.

The OSPO manager can use the Project Velocity metric to:

- Report project velocity of open source projects vs in-house projects
- Compare project velocity across a portfolio of projects
- Identify which projects grow beyond internal contributors (when filtering internal vs. external contributors)
- Identify promising areas in which to get involved
- Highlight areas likely to be the successful platforms over the next several years

See Example

Implementation Base metrics include:

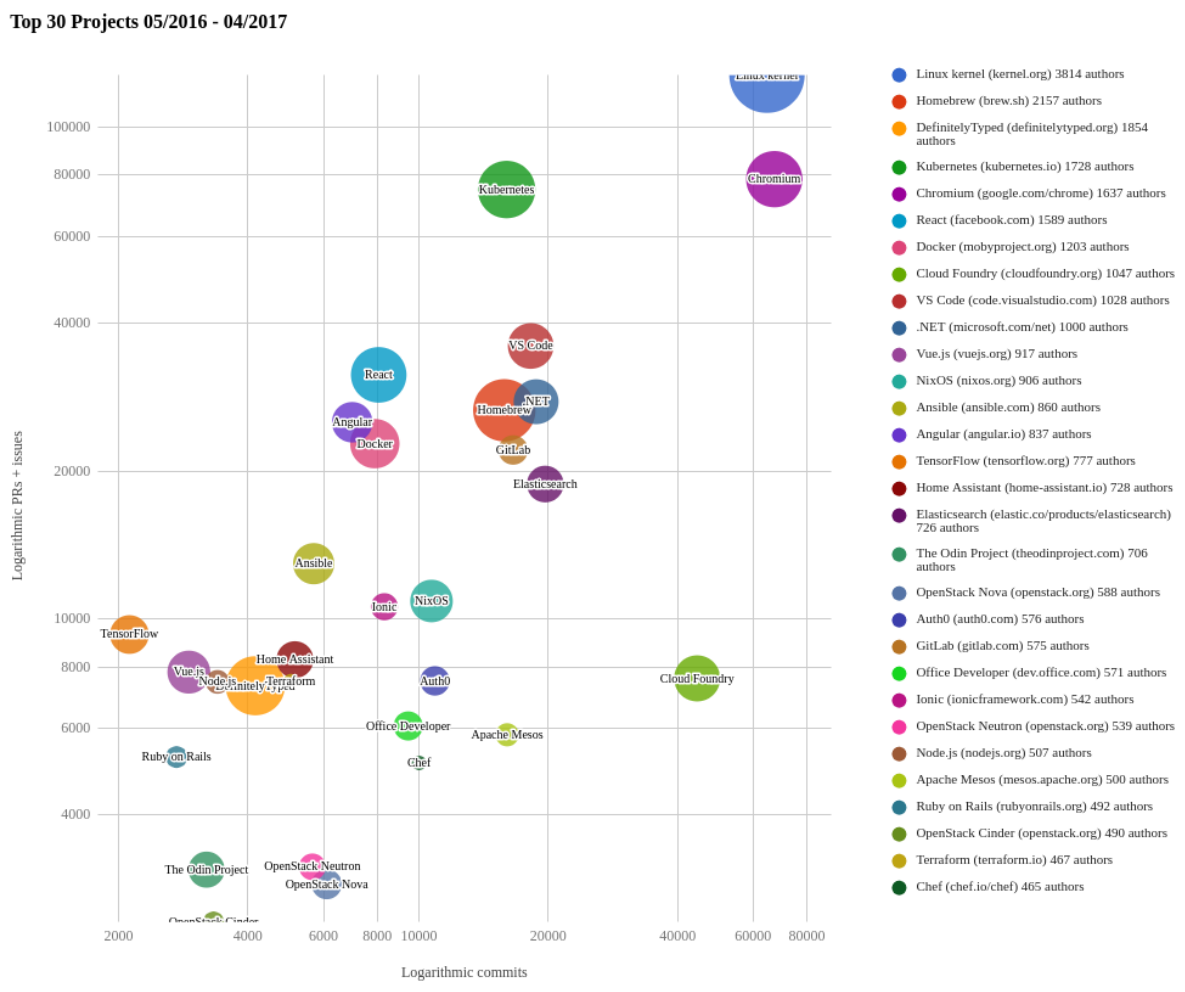
- issues closed
- number of reviews
- # of code changes
- # of committers

Filters

- Internal vs external contributors
- Project sources (e.g., internal repositories, open-source repositories, and competitor open-source repositories)
- Time

Visualizations

- X-Axis: Logarithmic scale for Code Changes
- Y-Axis: Logarithmic scale of Sum of Number of Issues and Number of Reviews
- Dot-size: Committers
- Dots are projects



From CNCF

Tools providing the Metric

- CNCF - https://github.com/cncf/velocity

References

- Can Open Source Innovation work in the Enterprise?
- Open Innovation for a High Performance Culture
- Open Source for the Digital Enterprise
- Highest Velocity Open Source Projects

Social Listening

Question: How does one measure the value of community interactions and accurately gauge "reputation" of a community as evident from qualitative sentiment?

Note: This metric synthesizes several other metrics that can be derived from trace data, and several process-oriented metrics. Embedded footnotes annotate areas planned for later clarification, and questions for later resolution.

Description Social Listening is a combination of social listening practices across multiple channels along with a meaningful set of categorizations. The combination of these tactics can lead to systematic community analysis and can inform a community strategy that leads to measurable business value. 1

Theory and Origin

Social currency or social capital is a social scientific theory. It broadly considers how human interactions build relationships and trust in a community. The Social Listening metric represents the reputation of a community as measured via community trust, transparency, utility, consistency, and merit.

Interpersonal relationships are the social fabric of communities. This is shown in the Levinger's Relationship Model and Social Penetration Theory. Community members' sense of personal and group identity grows as they interact. Members build shared values, accumulate a sense of trust, encourage cooperation, and garner reciprocity through acts of self-disclosure. These interactions build an increased and measurable sense of connection. The measure of these characteristics is called social currency. 2

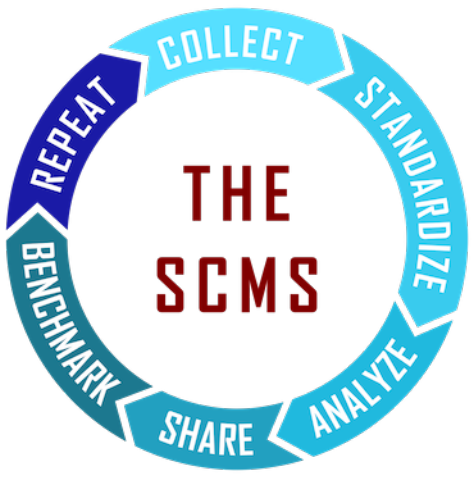
Results

The Social Listening metric is a way to sort through a fire hose of qualitative data from community interactions. A central premise of this approach is that community members' interactions have an impact on the community. The Social Listening metric continually measures the sentiment 3 from those interactions. It illustrates the reputation and level of trust between community members and leaders. 4

Objectives Analyze the qualitative comments in community interactions. Gain an overview of sentiment in a community. Get metrics that show at a glance how a community is and was doing. Use lead metrics from continuous measurements for proactive community strategy development. Instill trust in community members that their thoughts and opinions are valued.

Implementation The Social Listening requires the collection of community comments (communication traces), the definition of a codex, and the on-going review of the communication traces. 5

Set up a Data Collection Platform of your choice as described in the "Tools" section below. Ensure it has a minimum of 4 dimensions and 3 communication channels. Once it is set up, the following method is used to collect, analyze, and interpret results:



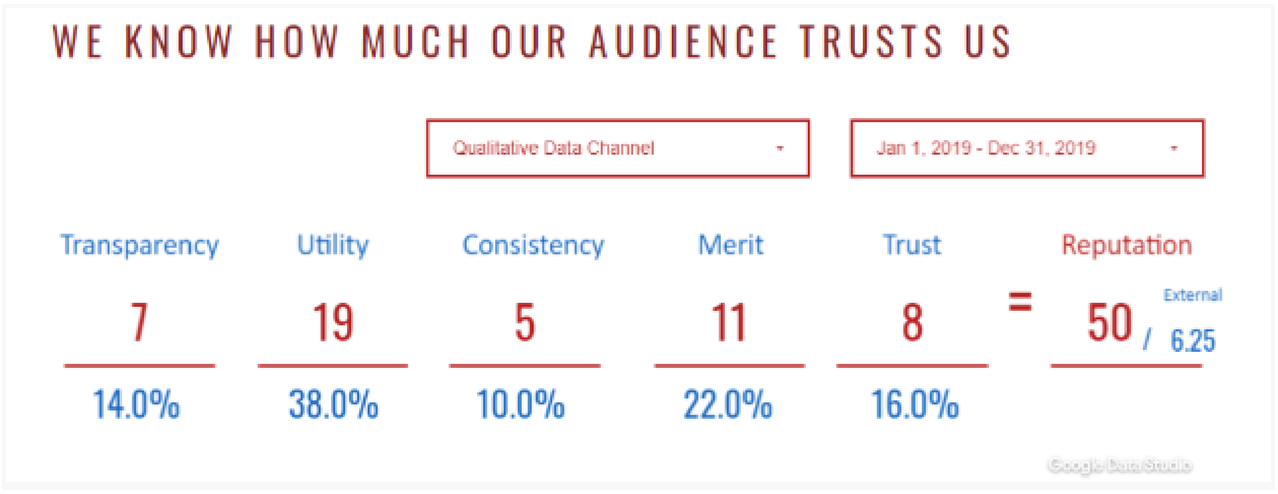
1. **Collect Communication Traces** -- Identify online platforms that your community is communicating on. Set up data funnels from the primary platform to your Social Listening tool. The critical data for the system is user generated content.
2. **Standardize How Communication Traces Should Be Assessed** -- Use a codex to define important concepts as a "tracking keyword" or "category" in the focal community. This unified codex of terms ensures consistent analysis as different people read and tag community sentiment. Formalizing the revision and addition structure to this codex on a regular basis is a must. 5
3. **Analyze the Communication Traces** -- Community sentiment is analyzed in the Social Listening tool by tagging data with codex terms. If the tagging is done by a team of people, it is recommended that everyone gets together regularly to discuss trends and ensure consistent tag use. If the tagging is done by an artificial intelligence algorithm, then a human team should supervise and retrain the AI as necessary. 5
4. **Share and Visualize the Aggregated Analysis** -- Visualize the quantitative count of codex terms over time, e.g., in a dashboard. This is where the qualitative analysis results produce an easy to observe dashboard of trends. Share analysis with team members. 6
5. **Benchmark, Set Goals & Predict Future Growth** -- After getting enough data to form a benchmark, take stock of where your community stands. What are its strengths and weaknesses? What actions can be taken to make the community healthier and more robust? Then form community initiatives with well-defined goals and execute on these projects to affect the social currency metrics for next week. 6
6. **Repeat the Process** -- In regular evaluation meetings, discuss the shortcomings of the dataset or collection methods. Come up with methods to address these shortcomings in the future. Work solutions into the system and move forward. Truth is in the trend, power is in the pattern.7

Filters

1. **Channel**: Sort by where the data was collected from.
2. **Tag**: Show data based on what codex tags were used to identify sentiment in comments.
3. **Time**: Show trends in the data over time and pull specific data-sets.
4. **Most impactful comments**: Sort and filter by flags that can be placed in the data to highlight specific data points and explain their importance.

5. **AI vs. Human tagged**: Filter by whether tags were applied programmatically or by a person.
6. **Weighted currency:** Weight the "importance" of certain comments based on any one individually selected criteria. A resulting weighted view is simply a re-order of information based on weight.

Visualizations Dashboard visualizing the aggregate metrics:

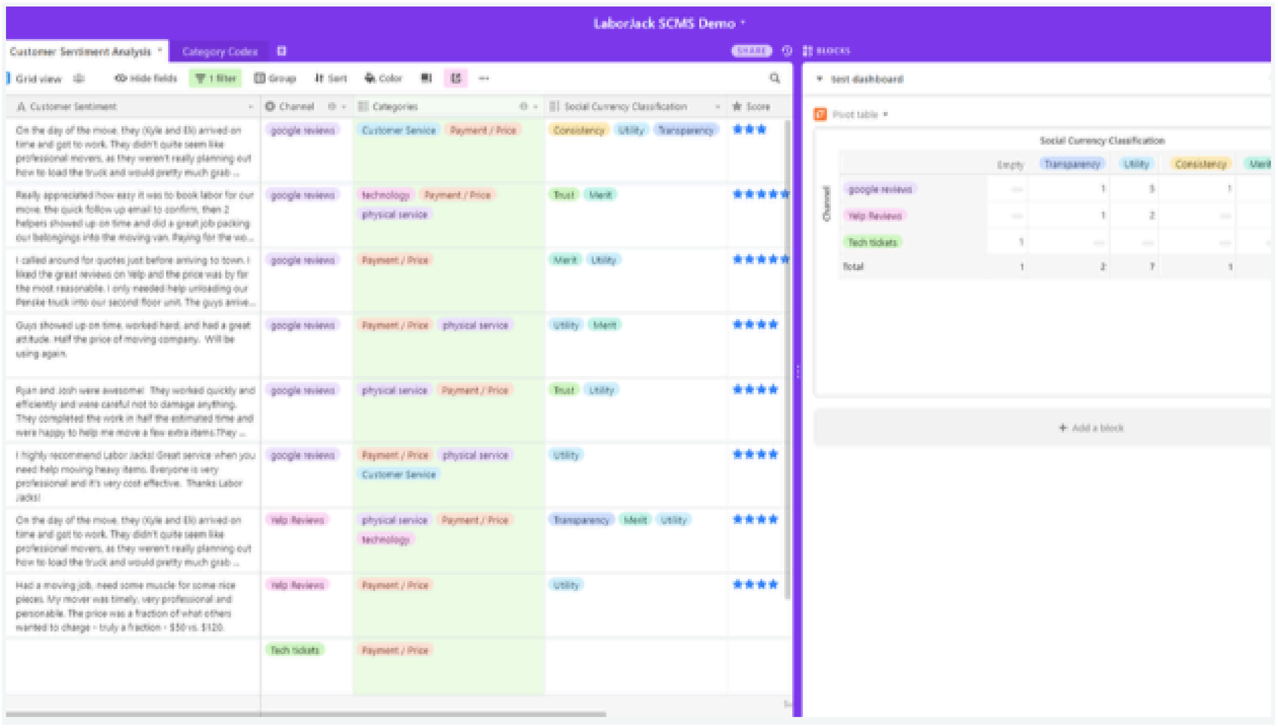


WE KNOW HOW MUCH OUR AUDIENCE TRUSTS US

Qualitative Data Channel ▾ | Jan 1, 2019 - Dec 31, 2019 ▾

Transparency	Utility	Consistency	Merit	Trust		Reputation
7	19	5	11	8	=	50 / External 6.25
14.0%	38.0%	10.0%	22.0%	16.0%		

Example Social Listening tool: On the left, raw community comments are shown and tags are added in columns immediately to the right. On the right, a pivot table shows in numbers how often tags occurred in combination with other tags.

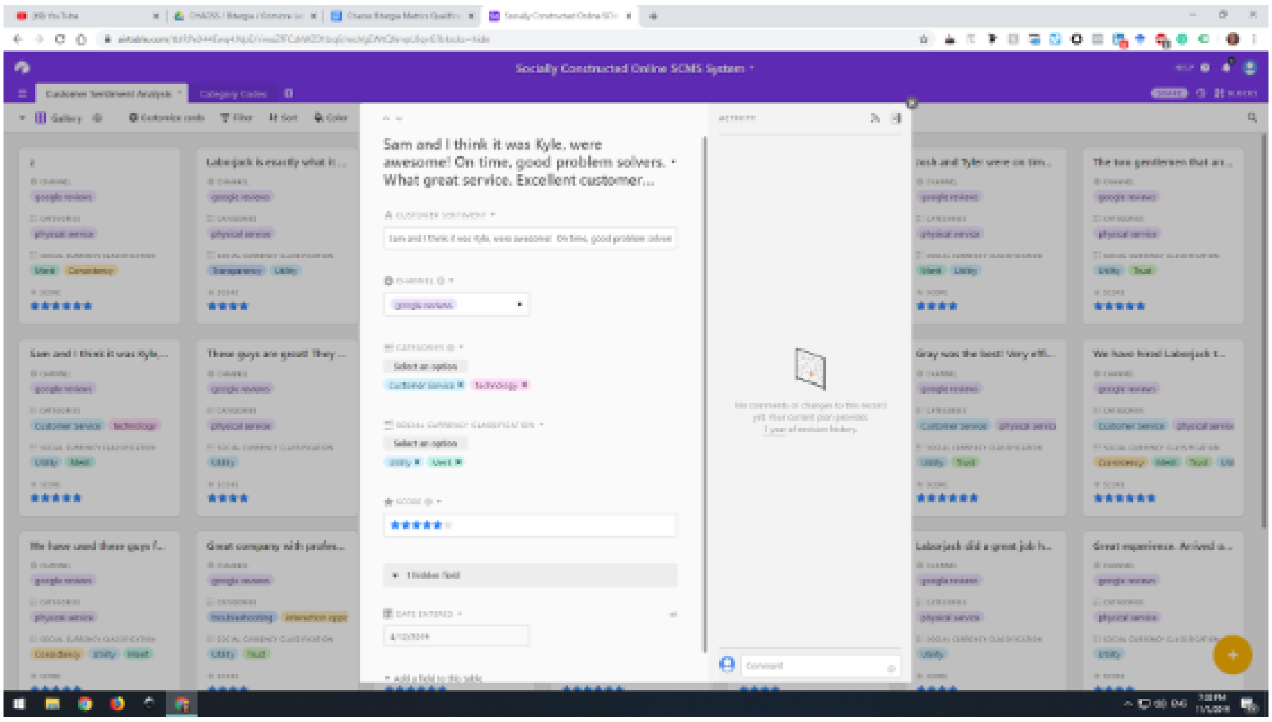


Customer Sentiment	Channel	Categories	Social Currency Classification	Score
On the day of the move, they (Kyle and Eli) arrived on time and got to work. They didn't quite seem like professional movers, as they weren't really planning out how to load the truck and would pretty much grab ...	google reviews	Customer Service, Payment / Price	Consistency, Utility, Transparency	★★★
Really appreciated how easy it was to book labor for our move, the quick follow up email to confirm, then 2 helpers showed up on time and did a great job packing our belongings into the moving van. Paying for the wo...	google reviews	technology, Payment / Price, physical service	Trust, Merit	★★★★★
I called around for quotes just before arriving to town. I liked the great reviews on Yelp and the price was by far the most reasonable. I only needed help unloading our Penske truck into our second floor unit. The guys arrive...	google reviews	Payment / Price	Merit, Utility	★★★★★
Guys showed up on time, worked hard, and had a great attitude. Half the price of moving company. Will be using again.	google reviews	Payment / Price, physical service	Utility, Merit	★★★★
Ryan and Josh were awesome! They worked quickly and efficiently and were careful not to damage anything. They completed the work in half the estimated time and were happy to help me move a few extra items.They ...	google reviews	physical service, Payment / Price	Trust, Utility	★★★★
I highly recommend Labor Jacks! Great service when you need help moving heavy items. Everyone is very professional and it's very cost effective. Thanks Labor Jacks!	google reviews	Payment / Price, physical service, Customer Service	Utility	★★★★
On the day of the move, they (Kyle and Eli) arrived on time and got to work. They didn't quite seem like professional movers, as they weren't really planning out how to load the truck and would pretty much grab ...	Yelp Reviews	physical service, Payment / Price, technology	Transparency, Merit, Utility	★★★★
Had a moving job, need some muscle for some nice pieces. My mover was timely, very professional and personable. The price was a fraction of what others wanted to charge - truly a fraction - $50 vs. $120.	Yelp Reviews	Payment / Price	Utility	★★★★
	Tech tickets	Payment / Price		

Social Currency Classification

Channel	Empty	Transparency	Utility	Consistency	Merit
google reviews	—	1	5	1	
Yelp Reviews	—	1	2	—	
Tech tickets	1	—	—	—	
Total	1	2	7	1	

Expanded comments view: remove the "quantitative" from the fields and provide the best possible way to read the different comments.



Socially Constructed Online SCMS System

Sam and I think it was Kyle, were awesome! On time, good problem solvers. What great service. Excellent customer...

Tools Providing the Metric To implement the metric any MySQL, smart-sheet, excel, or airtable-like excel datasheet program works fine. This data should be simplified enough to interact with other data interfaces to ensure that data migration is simple, straightforward, and can be automated (such as google data studio). This requires that systems used to implement the Social Listening metric work with CSV and other spreadsheet files, and we heavily recommend open source programs for its implementation.

Once you have this, create a data set with the following data points: 8

Data Points	Description
Date of entry	Date data was imported to Social Listening tool
Date of comment	Date comment was made on original platform
Comment Text	Qualitative data brought in. Decide on how large you want thes
Data channel	Originating data channel the comment came from
Tags (created on codex document below)	Based on the unified codex of terms, decide what tags to track.
Social Currency Metric	The social currency being awarded or demerited in the system.
Weighted Score	Once you've decided what your "weight" will be, you can assign

Create a second sheet for the Unified Codex of Terms which will define terms. It should look like this: 8

Category Term	Definition
[Custom Tags - themes and categories]	
[Community specific jargon]	
Social Currency Dimensions:	
TRANSPARENCY	Do people recognize and feel a connection to your community?
UTILITY	Is your community doing something useful or is it contributing va
CONSISTENCY	Do you have a history of being reliable and dependable?
MERIT	Does your community merit respect and attention for your accom

Category Term	Definition
TRUST	Can people trust that your community will continue to provide va
INTERNAL REPUTATION 9	Do people believe these things strongly enough to warrant conve
EXTERNAL REPUTATION 9	What amount of your reputation in your community is transferabl

The codex is filled in by stakeholders on a regular basis by specific communities and forms the basis for analysis of the data. This is the MOST IMPORTANT part. Without this the subjectivity of qualitative data does not follow the rule of generalization: 9

> “A concept applies to B population ONLY SO FAR AS C limitation.”

Data Collection Strategies Community member comments are available from trace data. The Social Listening metric ideally imports the comment text automatically into a tool for tagging. Trace data can be collected from a communities' collaboration platforms where members write comments, including ticketing systems, code reviews, email lists, instant messaging, social media, and fora.

Legal and Regulatory Considerations

Points of destruction: Detailed data is destroyed after *xx* months has passed. Quantitative calculations are kept for up to 250 weeks per GDPR compliance. Data older than 250 weeks becomes archived data you cannot manipulate but can see. Users can negotiate the primary statistic.

References

- An example implementation on airtable
- An example implementation in data studio(report)
- An example implementation in data studio (data source)
- An example implementation in google sheets
- Implementation documentation (starts on page 33)

Annotated Footnotes 1 CHAOSS metrics historically is to create standard definitions that can be used reliably across projects to support comparisons. This metric may evolve into a project in the future.

2 What metrics emerge from this description? Likely included are: 1. community trust, 2. transparency, 3. utility, 4. consistency, and 5. merit

3 Analysis of sentiment suggests that metric (6) is likely ”Communications Sentiment”, and the definition may need to include references to common sentiment analysis tools, sometimes called ”bags of words”.

4 Measuring how trust trust is instilled in community members, such that their thoughts and opinions are valued is likely metric (7) that will define a process, and perhaps is not measurable via trace data.

5 A substantial portion of any codex for open source software will be common across projects, and each project is likely to have a set of particular interests that are a subset of that codex. In some cases, their main interests may not be present in an established codex component. In general, the codex, like the CHAOSS project itself, is open sourced as shared metadata to ensure shared understanding across open source communities.

6 This describes the evolution of a standard codex, and its elements through the process of CHAOSS working groups and projects, characterized in the previous footnote. Likely this will be a process metric (8).

7 Candidate process oriented metric (9).

8 Examples of data coded using the open sourced codex, as it evolves, will be essential components for advancing open source software through Social Listening. Implementations will require these examples, and their provision as open source assets of the CHAOSS project will return value as shared data.

9 Internal and external reputation are likely metrics (10), and (11) arising from the Social Listening metric.

INDIVIDUAL-VALUE

Job Opportunities

Question: How many job postings request skills with technologies from a project?

Description A common way for open source contributors to earn a living wage is to be employed by a company or be a self-employed or freelance developer. Skills in a specific project may improve a job applicant's prospects of getting a job. The most obvious indicator for demand related to a skill learned in a specific open source project is when that project or its technology is included in job postings.

Objectives The metric gives contributors a sense of how much skills learned in a specific open source project are valued by companies.

Implementation To obtain this metric on a job search platform (e.g., LinkedIn, Indeed, or Dice), go to the job search and type in the name of the open source project. The number of returned job postings is the metric. Periodically collecting the metric through an API of a job search platform and storing the results allows to see trends.

Filters

- Age of job posting; postings get stale and may not be removed when filled

Visualizations The metric can be extended by looking at:

- Salary ranges for jobs returned
- Level of seniority for jobs returned
- Availability of jobs like on-site or off-site
- Location of job
- Geography

References

- LinkedIn Job Search API: https://developer.linkedin.com/docs/v1/jobs/job-search-api#
- Indeed Job Search API: https://opensource.indeedeng.io/api-documentation/docs/job-search/
- Dice.com Job Search API: http://www.dice.com/external/content/documentation/api.html
- Monster Job Search API: https://partner.monster.com/job-search
- Ziprecruiter API (Requires Partnership): https://www.ziprecruiter.com/zipsearch

Note: This metric is limited to individual projects but engagement in open source can be beneficial for other reasons. This metric could be tweaked to look beyond a single project and instead use related skills such as programming languages, processes, open source experience, or frameworks as search parameters for jobs.

Organizational Project Skill Demand

Question: How many organizations are using this project and could hire me if I become proficient?

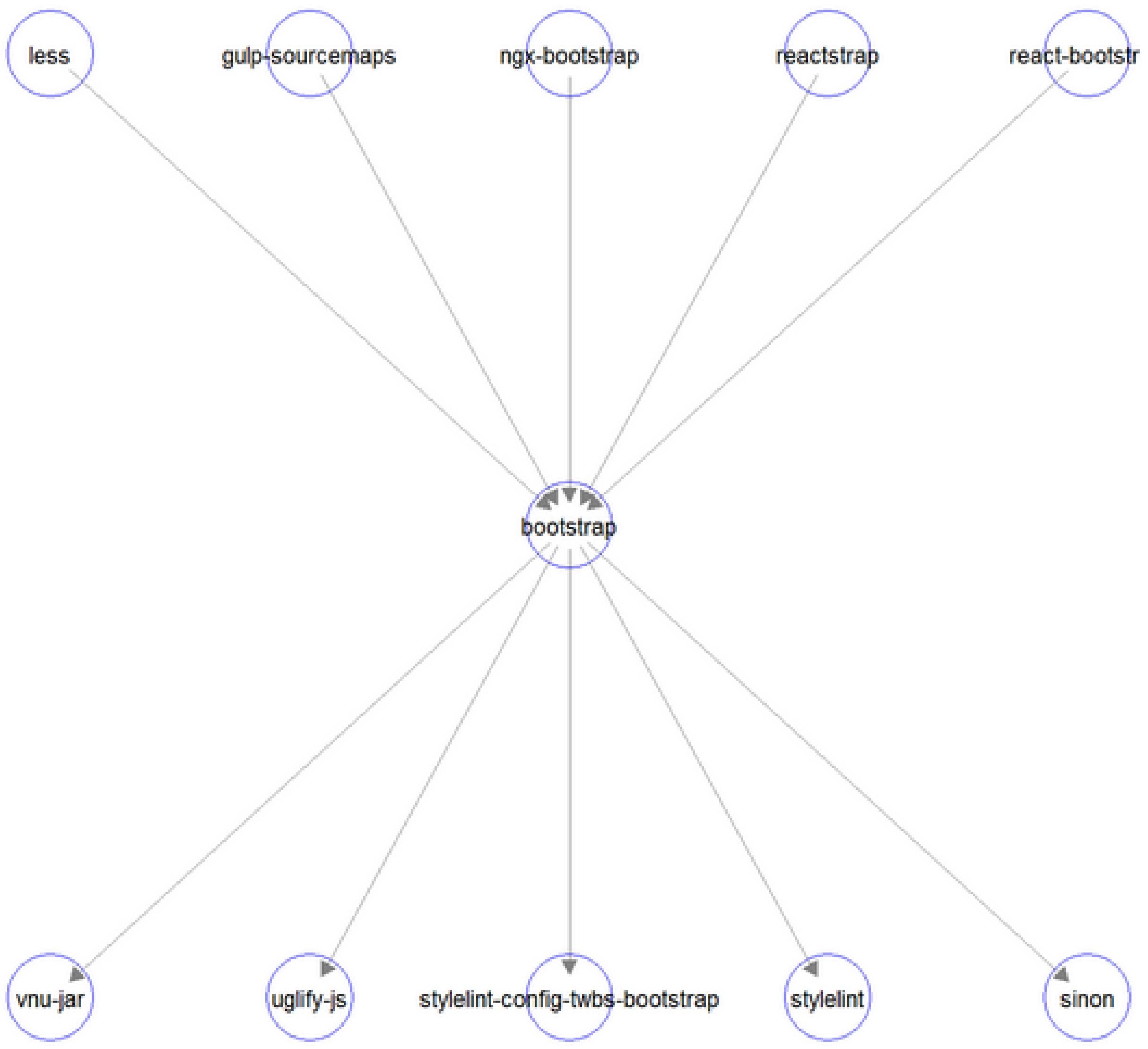
Description Organizations engage with open source projects through use and dependencies. This metric is aimed at determining downstream demand of skills related to an open source project. This metric looks at organizations that deploy a project as part of an IT infrastructure, other open source projects with declared dependencies, and references to the project through social media, conference mentions, blog posts, and similar activities.

Objectives As a developer, I'd like to invest my skills and time in a project that has a likelihood of getting me a decent paying job in the future. People can use the Downstream Organizational Impact of a Project Software metric to discover which projects are used by organizations, and they may, therefore, be able to pursue job opportunities with, possibly requiring IT support services.

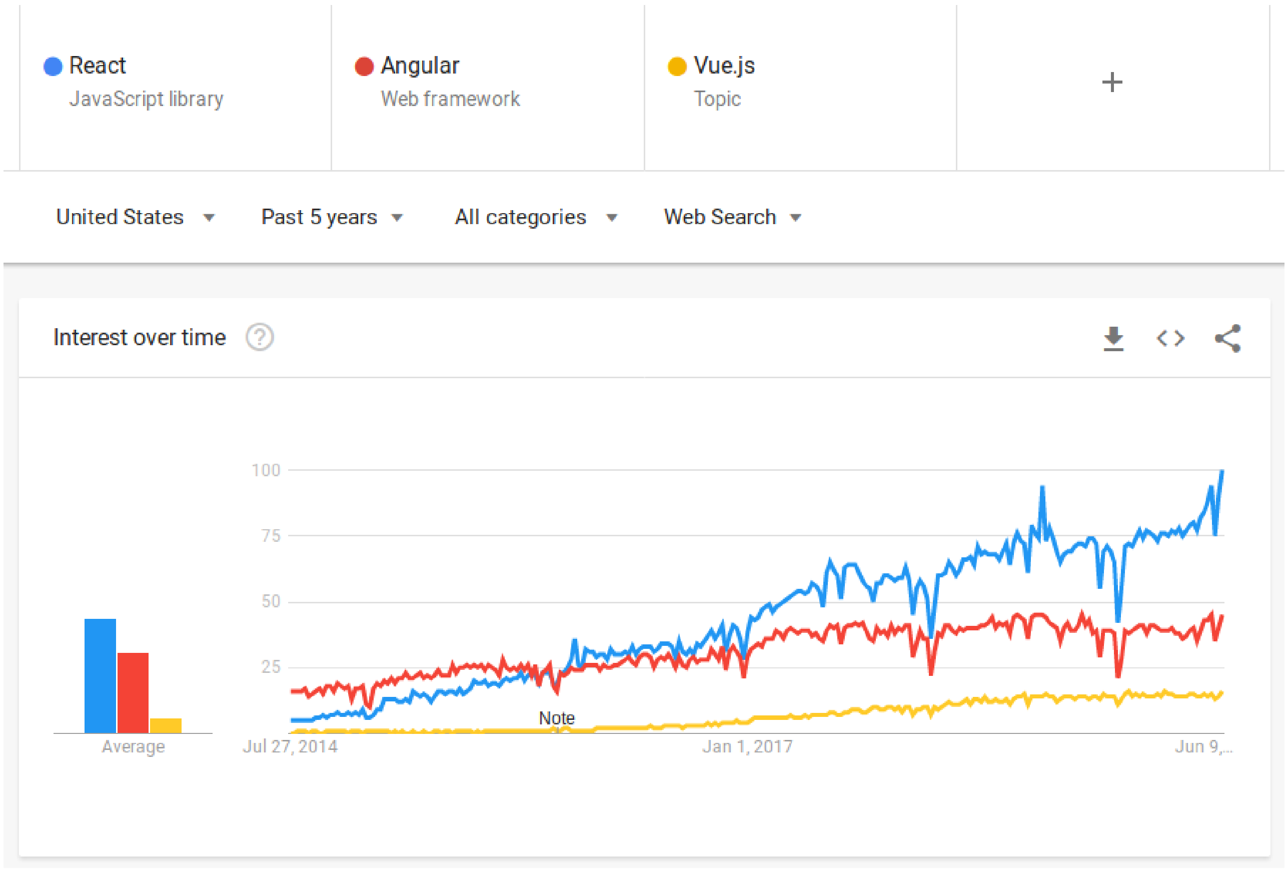
Implementation Base metrics include:

- Number of organizations that created issues for a project
- Number of organizations that created pull requests for a project
- Number of organizations that blog or tweet about a project
- Number of organizations that mention a project in open hiring requests
- Number of organizations that are represented at meetups about this project
- Number of other projects that are dependent on a project
- Number of books about a project
- Google search trends for a project

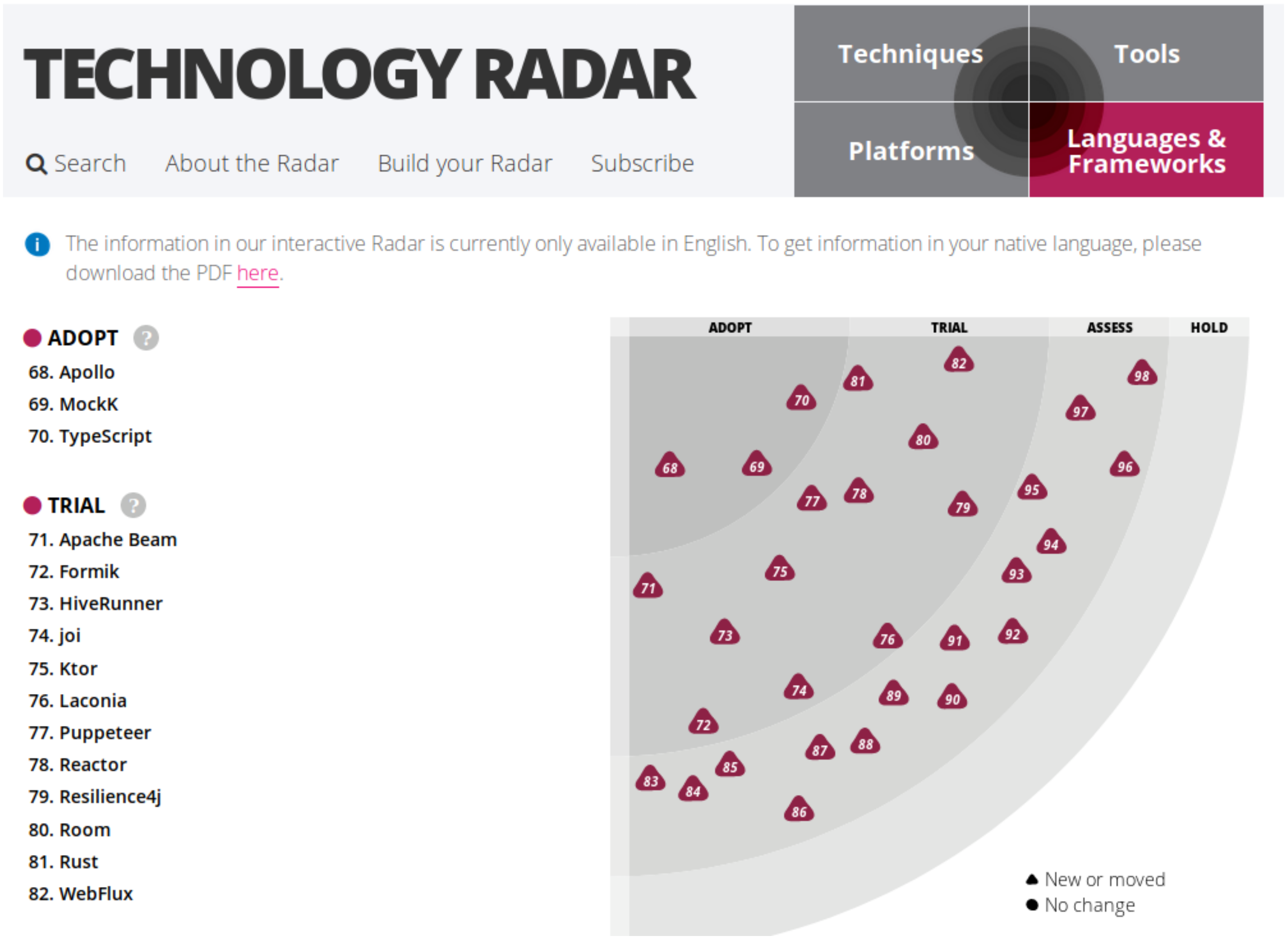
Visualizations The following visualization demonstrates the number of downstream projects dependendent on the project in question. While this visualization does not capture the entirety of the Downstream Organizational Impact of a Project Software metric, it provides a visual for a portion.



Other visualizations could include Google search trends (React vs. Angular vs. Vue.js)



ThoughtWorks publishes a series called 'Tech Radar' that shows the popularity of technologies.

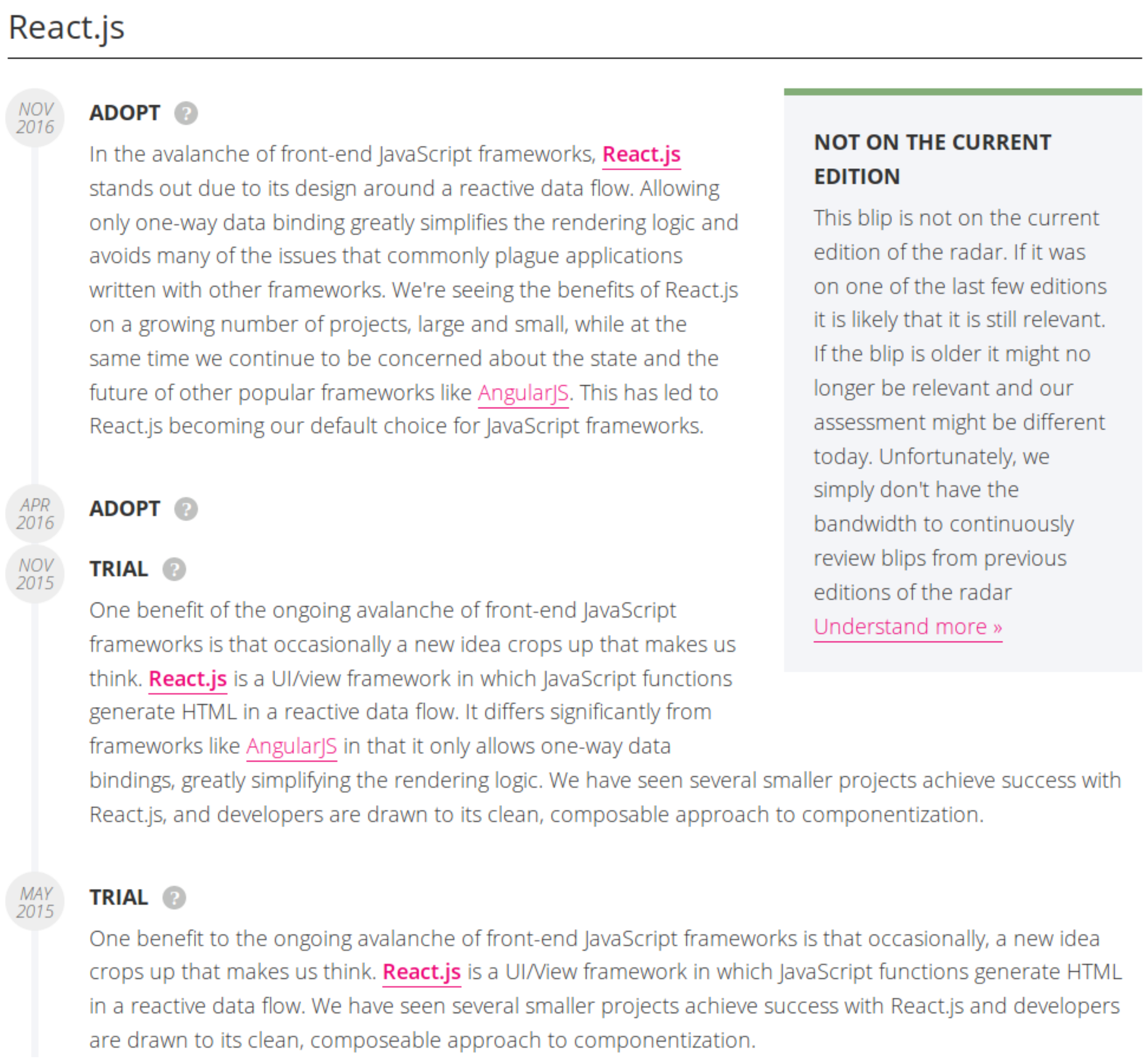


The information in our interactive Radar is currently only available in English. To get information in your native language, please download the PDF here.

● ADOPT
68. Apollo
69. MockK
70. TypeScript

● TRIAL
71. Apache Beam
72. Formik
73. HiveRunner
74. joi
75. Ktor
76. Laconia
77. Puppeteer
78. Reactor
79. Resilience4j
80. Room
81. Rust
82. WebFlux

Tech Radar allows you to drill down on projects to see how the assessment has changed over time.



React.js

NOV 2016 **ADOPT**

In the avalanche of front-end JavaScript frameworks, **React.js** stands out due to its design around a reactive data flow. Allowing only one-way data binding greatly simplifies the rendering logic and avoids many of the issues that commonly plague applications written with other frameworks. We're seeing the benefits of React.js on a growing number of projects, large and small, while at the same time we continue to be concerned about the state and the future of other popular frameworks like AngularJS. This has led to React.js becoming our default choice for JavaScript frameworks.

APR 2016 **ADOPT**

NOV 2015 **TRIAL**

One benefit of the ongoing avalanche of front-end JavaScript frameworks is that occasionally a new idea crops up that makes us think. **React.js** is a UI/view framework in which JavaScript functions generate HTML in a reactive data flow. It differs significantly from frameworks like AngularJS in that it only allows one-way data bindings, greatly simplifying the rendering logic. We have seen several smaller projects achieve success with React.js, and developers are drawn to its clean, composable approach to componentization.

MAY 2015 **TRIAL**

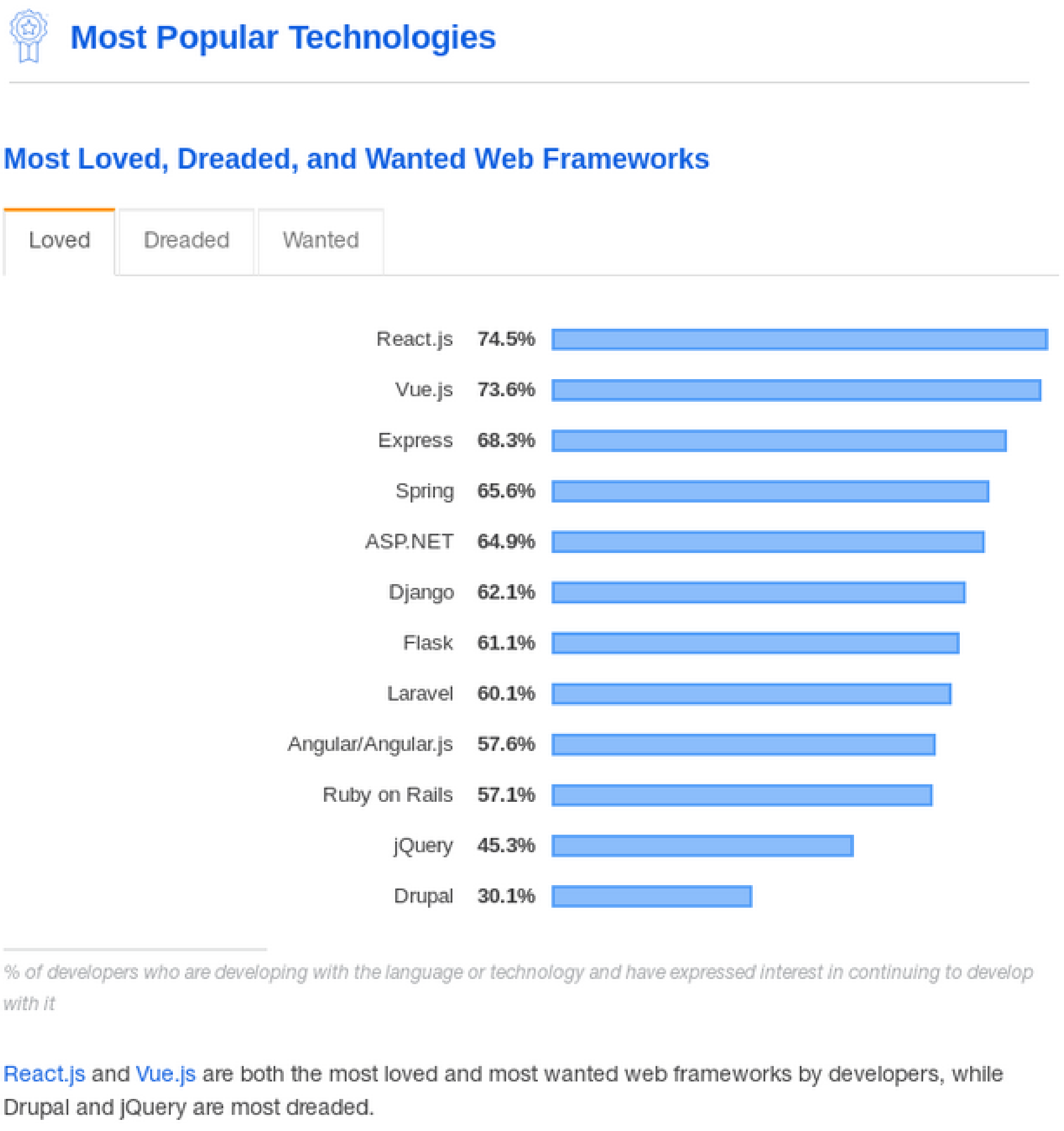
One benefit to the ongoing avalanche of front-end JavaScript frameworks is that occasionally, a new idea crops up that makes us think. **React.js** is a UI/View framework in which JavaScript functions generate HTML in a reactive data flow. We have seen several smaller projects achieve success with React.js and developers are drawn to its clean, composeable approach to componentization.

NOT ON THE CURRENT EDITION

This blip is not on the current edition of the radar. If it was on one of the last few editions it is likely that it is still relevant. If the blip is older it might no longer be relevant and our assessment might be different today. Unfortunately, we simply don't have the bandwidth to continuously review blips from previous editions of the radar

Understand more »

StackOverview publishes an annual developer's survey



Most Popular Technologies

Most Loved, Dreaded, and Wanted Web Frameworks

% of developers who are developing with the language or technology and have expressed interest in continuing to develop with it

React.js and Vue.js are both the most loved and most wanted web frameworks by developers, while Drupal and jQuery are most dreaded.

Tools Providing the Metric

- Google Trends - for showing search interest over time
- ThoughtWorks TechRadar - project assessments from a tech consultancy
- StackOverflow Developer's Survey - annual project rankings
- Augur; Examples are available for multiple repositories:
  - Rails
  - Zephyr
  - CloudStack

References

- Open Source Sponsors
- Fiscal Sponsors and Open Source
- Large Corporate OpenSource Sponsors

- Google Trends API
- Measuring Open Source Software Impact
- ThoughtWorks Tech Radar
- Stack Overflow Developer’s Survey

ORGANIZATIONAL-VALUE

Labor Investment

Question: What was the cost of an organization for its employees to create the counted contributions (e.g., commits, issues, and pull requests)?

Description Open source projects are often supported by organizations through labor investment. This metric tracks the monetary investment of organizations (as evident in labor costs) to individual projects.

Objectives As organizational engagement with open source projects becomes increasingly important, it is important for organization to clearly understand their labor investment. The objective of this metric is to improve transparency in labor costs for organizations engaged with open source projects. This metric gives an Open Source Program Office (OSPO) manager a way to compare contributed labor costs across a portfolio of projects. For example, the Labor Investment metric can be used to prioritize investment or determine return on investment such as:

- Labor Investment as a means of evaluating OSPO priorities and justifying budgets
- Labor Investment as a way to explain product/program management priority
- Labor Investment as an argument for the value of continued investing in OSPOs
- Labor Investment to report and compare labor costs of contributed vs in-house work
- Labor Investment to compare project effectiveness across a portfolio of projects

Implementation Base metrics include:

- Number of contributions
- Number of contributions broken out by contributor types (internal / external)
- Number of contributions broken out by contribution types (e.g., commits, issues, pull requests)

Parameters include:

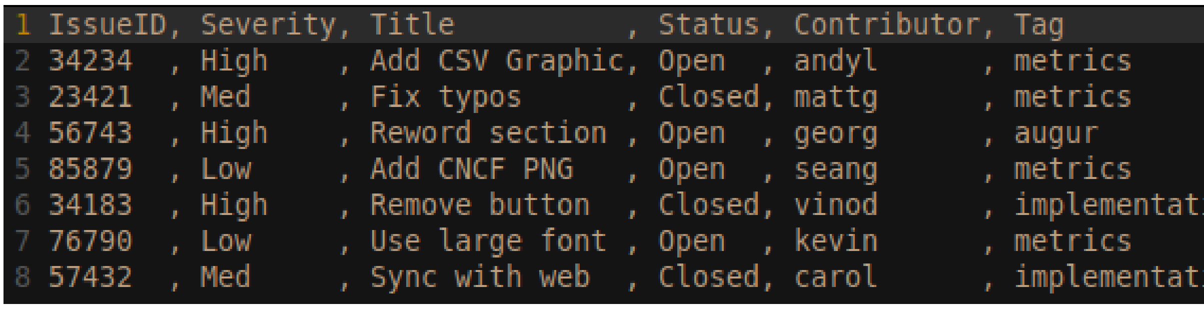
- Hourly labor rate
- Average labor hours to create contribution (by contribution type)

Labor Investment = For each contribution type, sum (Number of contributions * Average labor hours to create contribution * Average hourly rate)

Filters

- Internal vs external contributors
- Issue tags
- Project sources (e.g., internal, open-source repos, competitor open-source repos)

Visualizations



```
IssueID, Severity, Title           , Status, Contributor, Tag
34234   , High     , Add CSV Graphic, Open   , andyl       , metrics
23421   , Med      , Fix typos      , Closed, mattg       , metrics
56743   , High     , Reword section , Open   , georg       , augur
85879   , Low      , Add CNCF PNG   , Open   , seang       , metrics
34183   , High     , Remove button  , Closed, vinod       , implementat
76790   , Low      , Use large font , Open   , kevin       , metrics
57432   , Med      , Sync with web  , Closed, carol       , implementat
```

Our first visualization of parameterized metrics rely on CSV exports that can be made available from Augur. Spreadsheets are used for metric parameters and calculation formulas. Future implementations may add features for parameter manipulation directly in the webapp.

References

- Starting an Open Source Program Office
- Creating an Open Source Program Office
- Open Source in the Enterprise