



EXASCALE COMPUTING PROJECT



Patterns in Scientific Software Development

Boyana Norris, norris@cs.uoregon.edu

Bosco Ndemeye, Stephen Fickas, University of Oregon

Armando Acosta and Kanika Sood, California State University, Fullerton

Anshu Dubey, Argonne National Laboratory

<https://github.com/HPCL/ideas-uo>

SIAM Computational Science & Engineering
Mar 3, 2021

Background: IDEAS Productivity Project

The IDEAS Productivity project is part of the DOE Exascale Computing Program.



Improve scientific productivity by qualitatively improving developer productivity.



Improve software sustainability: reduce the cost of maintaining and evolving software capabilities.



Overview ▾

IDEAS-Classic ▾

IDEAS-Watersheds ▾

IDEAS-ECP ▾

Events ▾

Resources ▾

Getting Involved

About IDEAS

Project Vision

Advances in next-generation computational science and engineering (CSE) require the development of applications that can fully exploit emerging extreme-scale architectures for optimal performance and provide high-fidelity multiphysics and multiscale capabilities. To help address overwhelming complexity, the IDEAS family of projects focuses on *improving scientific productivity* by qualitatively improving *developer productivity* (positively impacting product quality, development time, and staffing resources) and *software sustainability* (reducing the cost of maintaining, sustaining, and evolving software capabilities in the future)—thereby enabling a fundamentally different attitude to creating and supporting CSE applications.

We are creating an extreme-scale scientific software development ecosystem composed of high-quality, reusable CSE software components and libraries; a collection of best practices, processes, and tools; and substantial outreach mechanisms for promoting and disseminating productivity improvements. We intend to improve CSE productivity by enabling better, faster *and* cheaper CSE application capabilities for extreme-scale computing.

Objectives

IDEAS projects address productivity concerns that are emerging from important trends in extreme-scale computing for science and engineering. In particular, the projects:

- Address a confluence of trends in hardware and increasing demands for predictive multiscale and multiphysics simulations, analysis, and design.
- Respond to a trend of continuous refactoring with efficient, agile software engineering methodologies and improved software design.









The goals of this work

High-level goal: We want to determine whether analyzing development data can lead to positive change in HPC software development.

Subgoal (this talk): Automatically identify software development patterns that have been shown to have impact on productivity and/or code quality.

Approach: Analyze available data, such as git activity, issues, mailing lists to help formulate and answer questions about development processes and their impact on projects.

Questions that can be answered (in part) with the help of data analysis

-  How active is the developer community? (git, issues, emails)
-  What parts of the code base could benefit from review or refactoring? (git, issues)
-  What is the project's reliance on individual developers? (git)
-  How are developers' contributions split among different categories? (git, manual labeling required)
-  How engaged are the user and developer communities? (PRs, issues, mailing lists)
-  What are some hot topics of discussion? (issues, mailing lists)
-  How do developers collaborate? (git, issues, mailing lists)
- 

Common patterns with known implications

Pluralsight book (2019)¹:

“20 patterns is a collection of work patterns we’ve observed in working with hundreds of software teams. Our hope is that you’ll use this field guide to get a better feel for how the team works, and to recognize achievement, spot bottlenecks, and debug your development process with data.”

What we do:

- 🦀 Identify patterns that are relevant to HPC (open-source) software development
- 🦀 Characterize each pattern using data from revision control systems and developer communications



¹https://www.pluralsight.com/content/dam/pluralsight2/landing-pages/offers/flow/pdf/Pluralsight_20Patterns_ebook.pdf

Example metrics

Metric	Description
Monthly bug fix rate	Computes cumulative count of bugs closed each month, starting from the beginning of the project. A higher value is not necessarily good, trends are more informative than individual values.
Monthly feature request rate	Number of new feature requests made by users. A higher value may indicate popularity/importance of the feature or missing functionality in the current software version.
Correlation of the number of issues with project age	Gives a good insight into the life cycle of the project by mapping the trend of issues raised over the lifetime of the project.
Commits, derived metrics	Characterizes some aspects of developer activity.
Number of issues	Project-related communications can be used to indicate community involvement and rates at which issues are resolved. For example, fast-growing projects can have significantly more activity in issues than more mature ones.
Issue categories	Identifies the types of issues that are reported in the repository. This information will be useful to derive correlation with other metrics.
Number of followers and watchers	Reports the code maturity and popularity. In addition, the number of followers can be correlated with the time it takes to fix reported issues.
Mailing list metrics	Relates to average time in discussion, most popular topic of interest, product activity based on type of emails, etc.
Number of contributors	Size of development community. When analyzed over time, we can estimate project turnover and identify different types of contributors.
Code complexity	Compute changes to code complexity metrics, e.g., cumulative size or cyclomatic complexity, over time.

Example pattern: **Domain Champion**

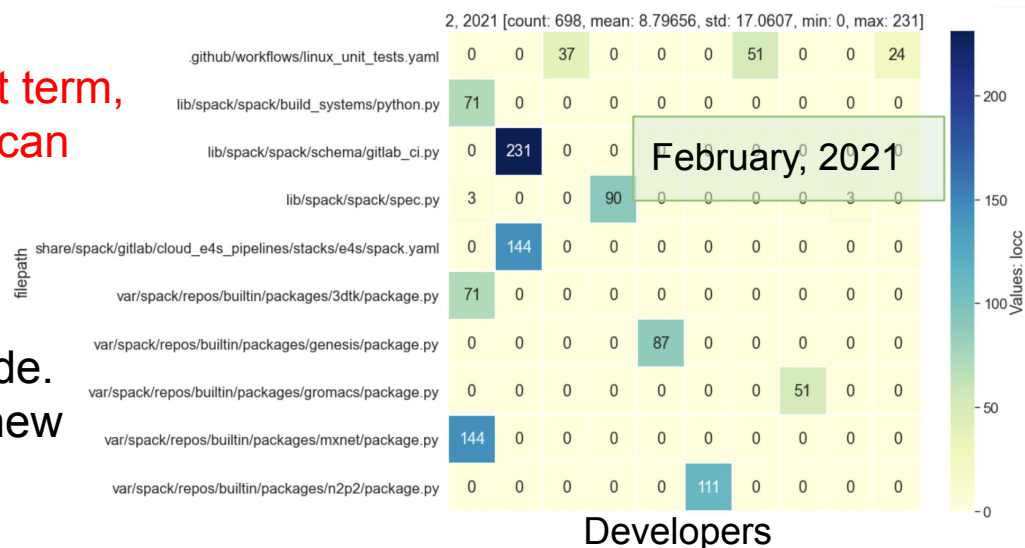
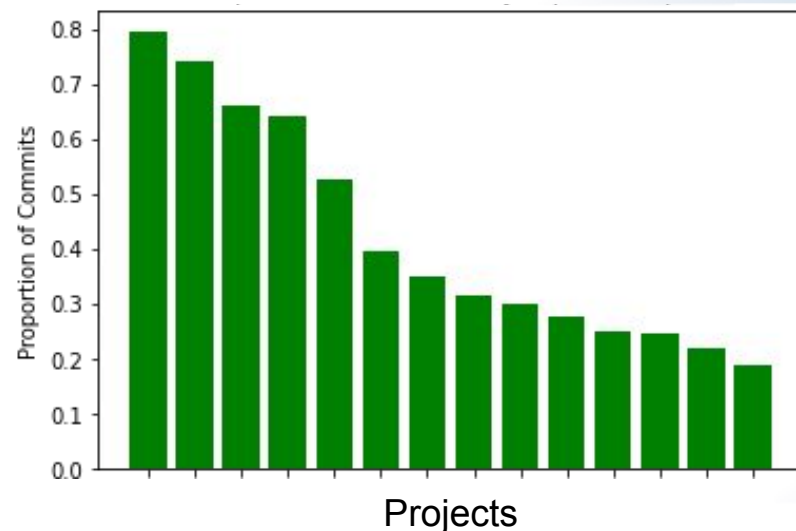
“The Domain Champion is an expert in a particular area of the codebase.”

- + typically submit their work in small, frequent commits and will show a sustained above average impact; highly productive pattern in the short term.
- there's usually very little actionable feedback that others can provide in code reviews.
- while highly productive in the short term, the pattern is not sustainable and can lead to stagnation.

☺ Actions:

Assign tickets for other areas of the code.
Make an effort to involve others (e.g., new developers) in work in that domain.

Proportion of commits by the most active developer



Example pattern: **Unusually high churn**

Unusually large amount of changes to single files or components may lead to development inefficiencies.

- + could be a sign of normal productive development
- may indicate need for more developer resources
- high conflict potential

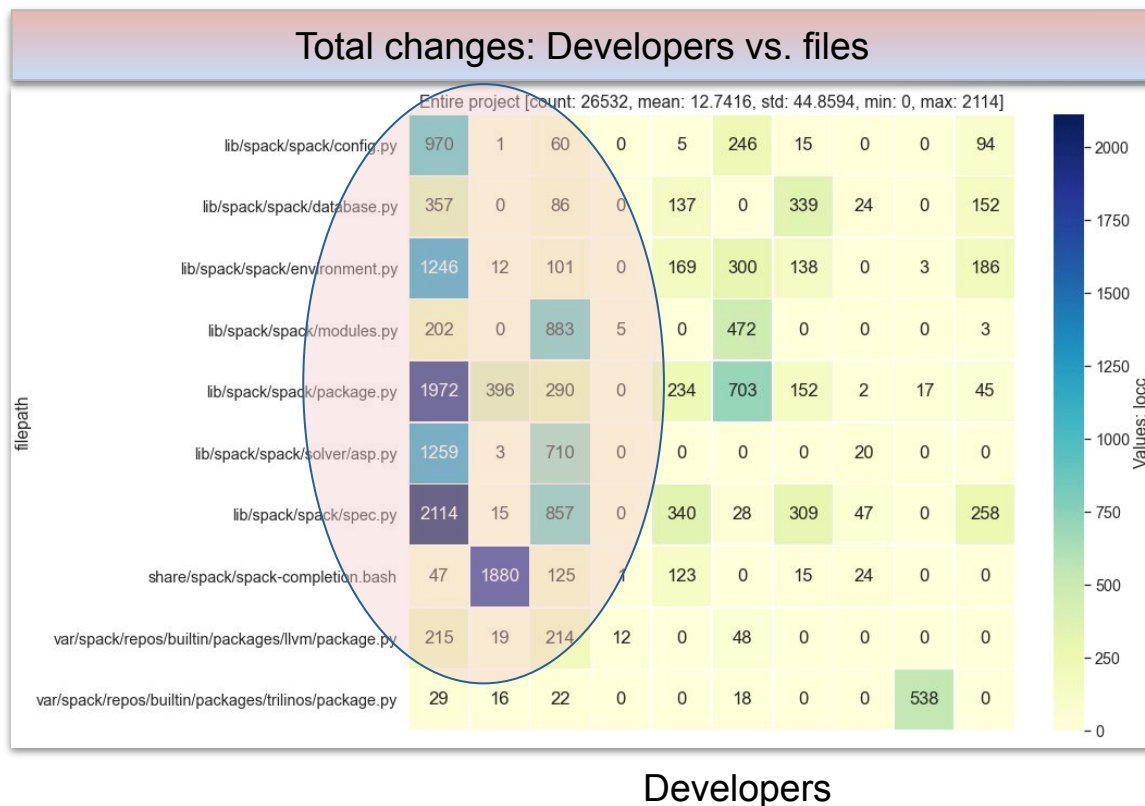
☺ Actions:



Consider refactoring the high-churn project components



Consider involving more developers in high-churn areas that are dominated by a single person



Pattern: In the zone

When are top contributors most productive?

- + Consistent high productivity during certain times of day
- Burnout, work-life balance

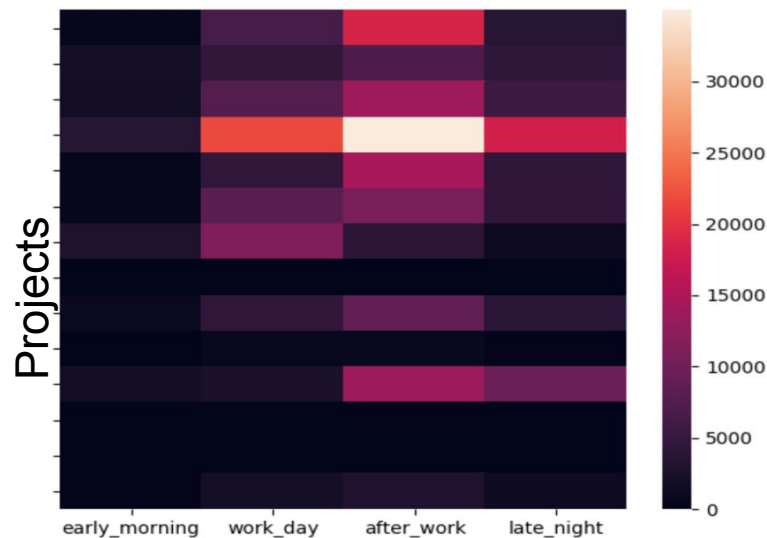
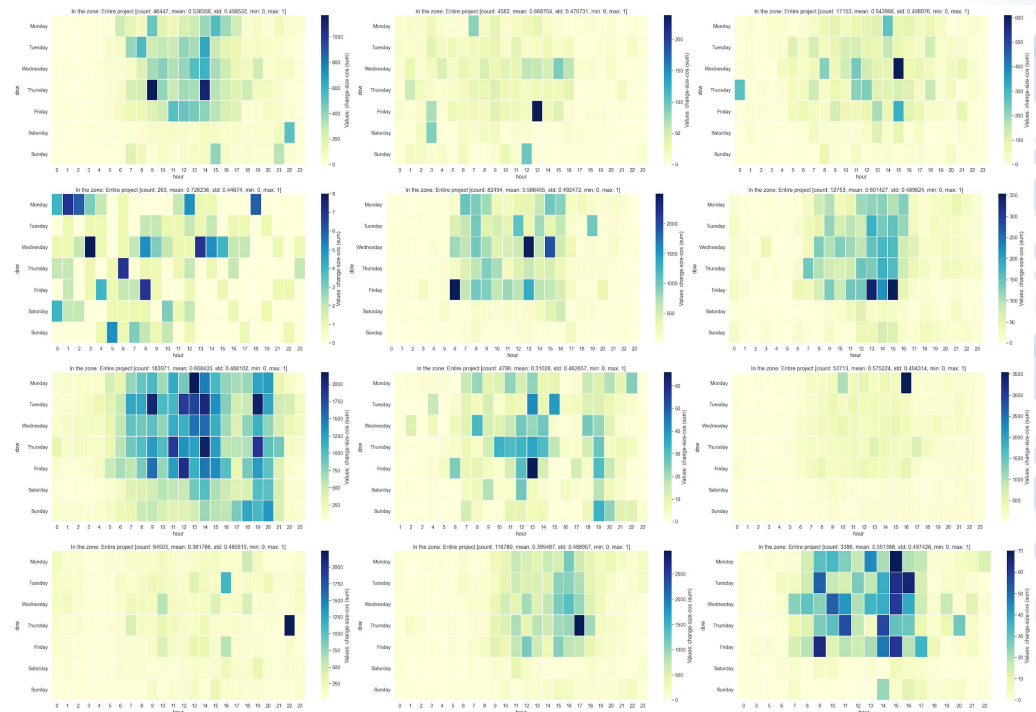
😊 Actions:



Acknowledge the consistently high-performing developers.

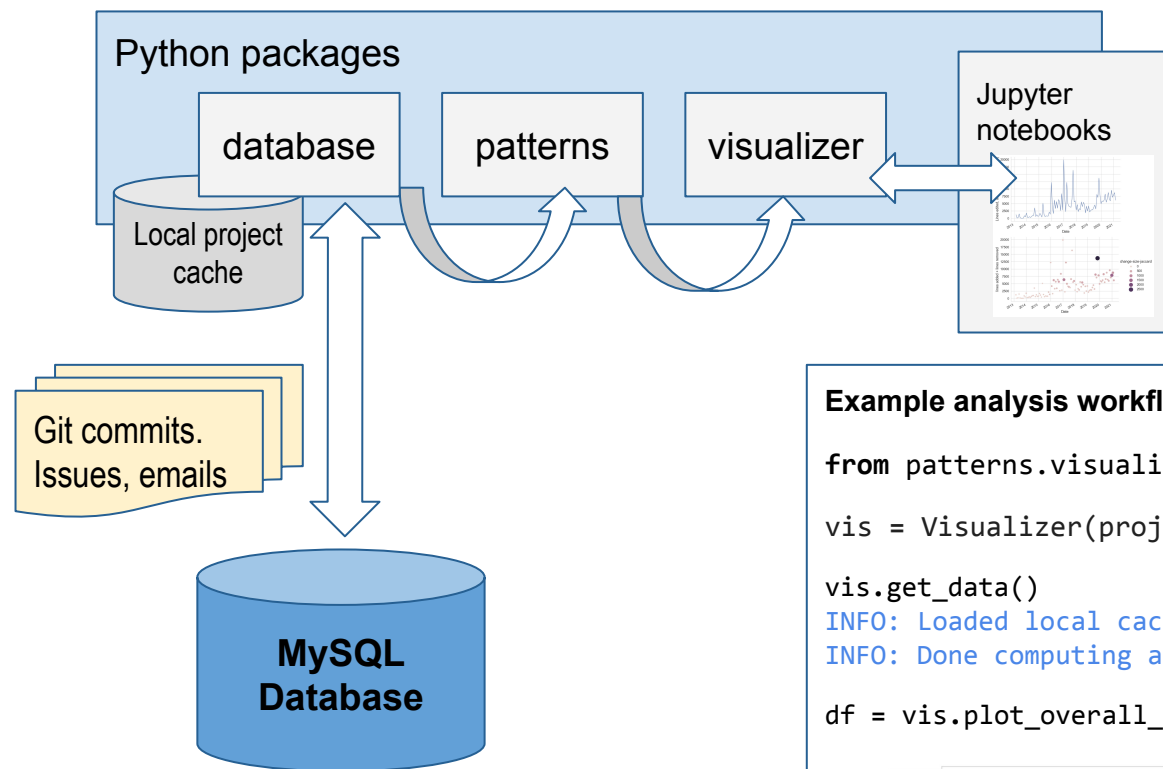


Consider the timing of and number of meetings during developers' most productive times.



Implementation

<https://github.com/HPCL/ideas-uo>



Some currently available projects (more are being added constantly): LAMMPS, Spack, PETSc, Nek5000, NWChem, E3SM, QPMCPACK, qdpxx, NWChem, TAU,,, Initial import can take up to 36 hours for some larger projects, but subsequent updates are fast and automated.

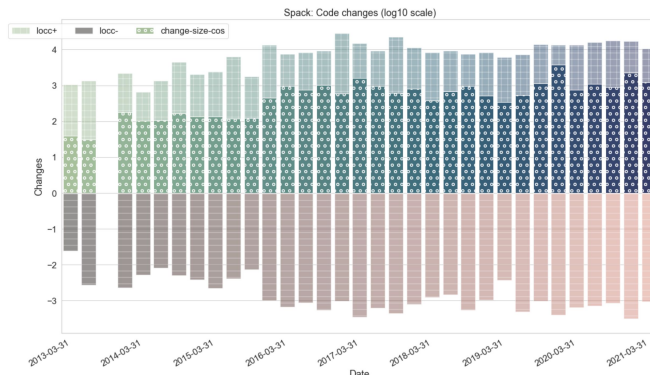
Example analysis workflow

```
from patterns.visualizer import Visualizer

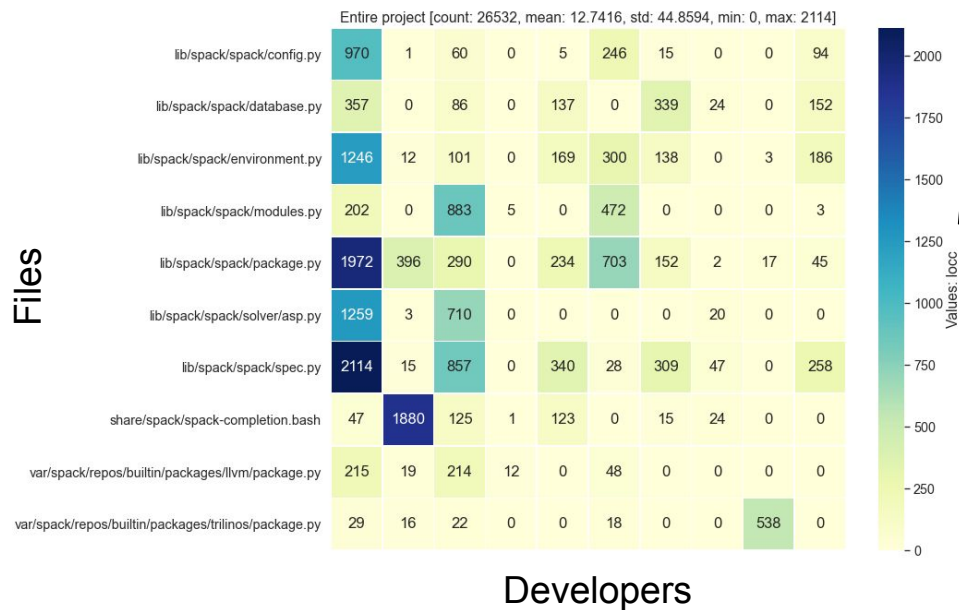
vis = Visualizer(project_name='spack')

vis.get_data()
INFO: Loaded local cached copy of spack data.
INFO: Done computing averages. 64909 commits (code only)

df = vis.plot_overall_project_locc(time_range=None, log=True)
```



Impact of different “change” estimates

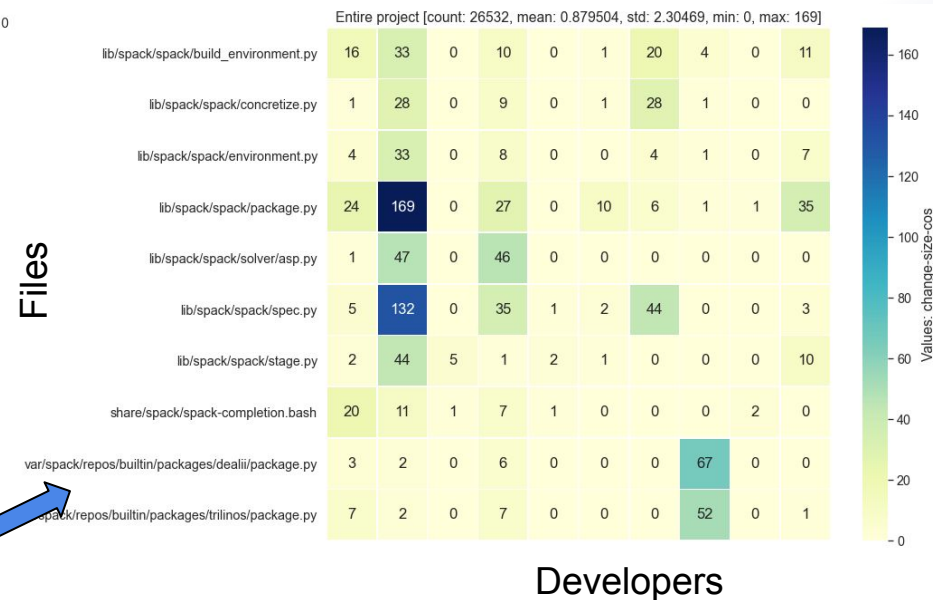


Simple line counts in git commit diffs:

- Patterns such as ‘---+++’ are counted as *edited* lines, e.g., 3 in this example
- Unmatched ‘-’ and ‘+’ lines counted as lines *deleted* and *added*, respectively
- LOCC = edited + deleted + added

More “intelligent” estimate of the magnitude of changes:

- Collect the *old* and *new* strings corresponding to each commit’s diffs
- Apply text distance metrics (based on the textdistance Python page; ~30 methods)
- E.g., change-size-cos is the cos distance between the vector embeddings of *old* and *new*



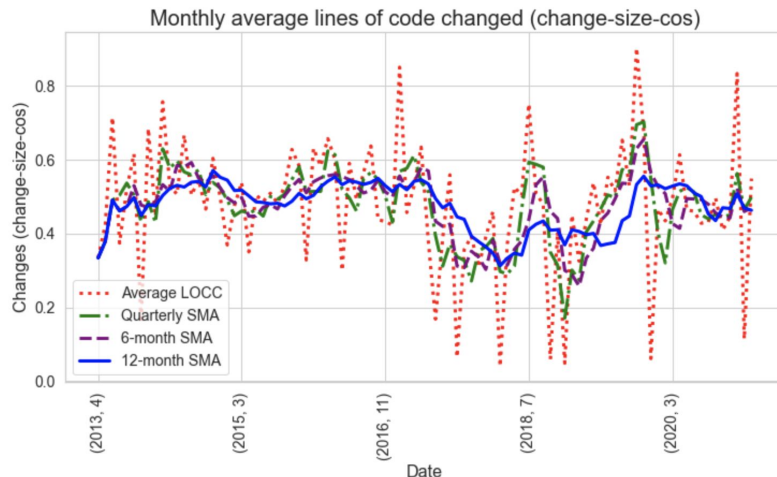
Impact of different “change” estimates (cont.)



Time-series git data is messy



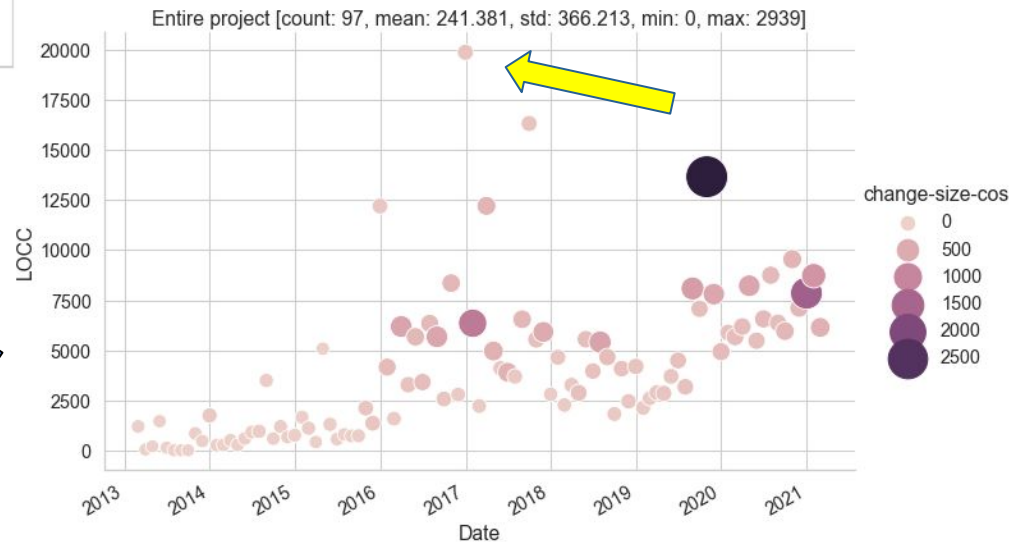
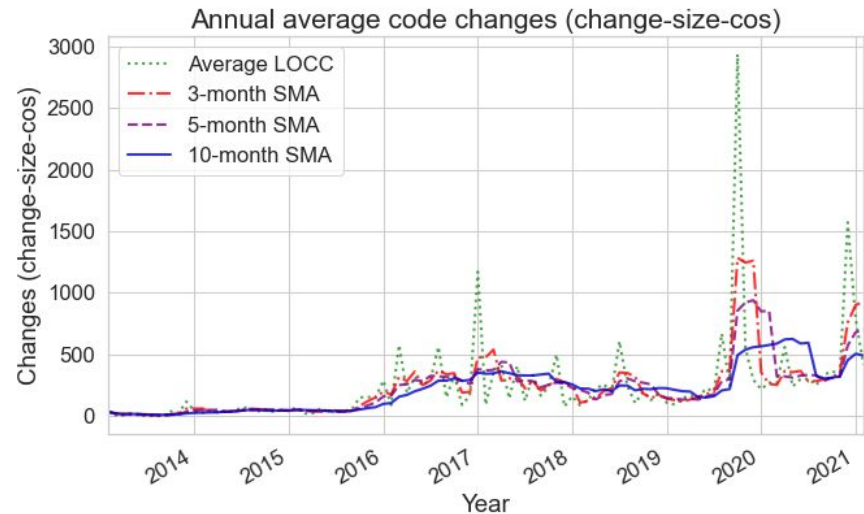
Moving averages help see trends for any time period



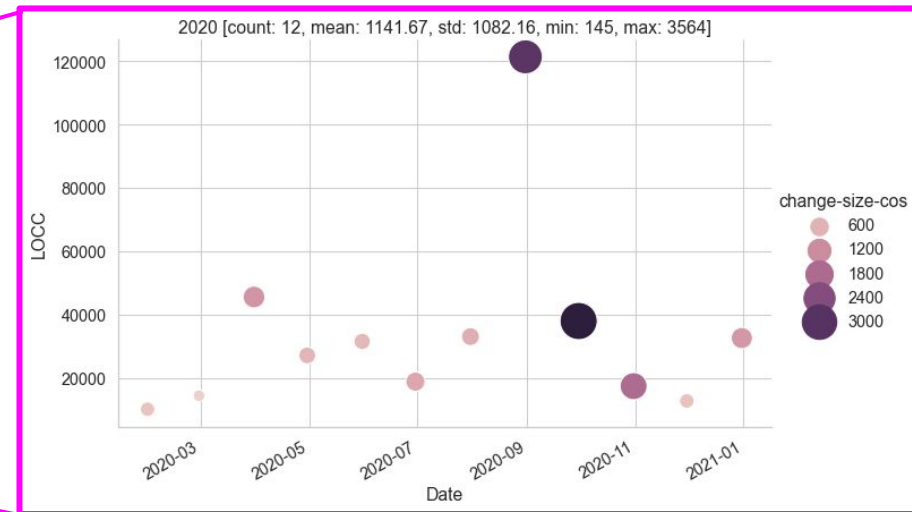
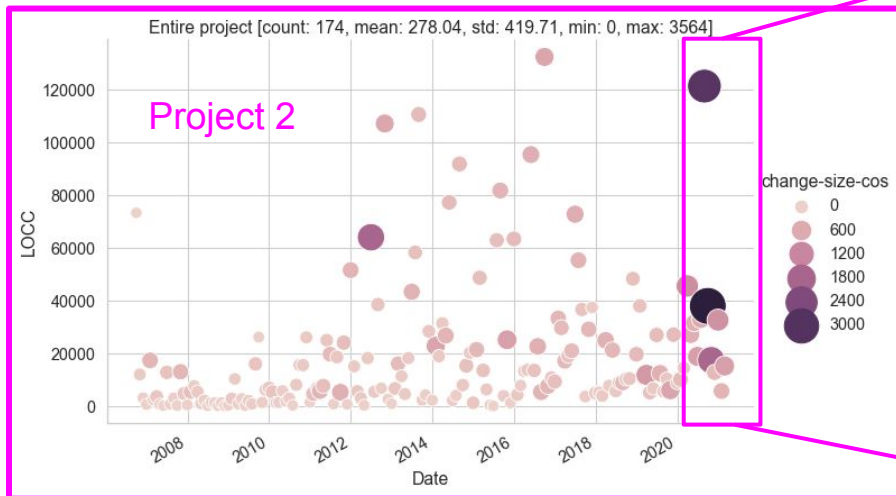
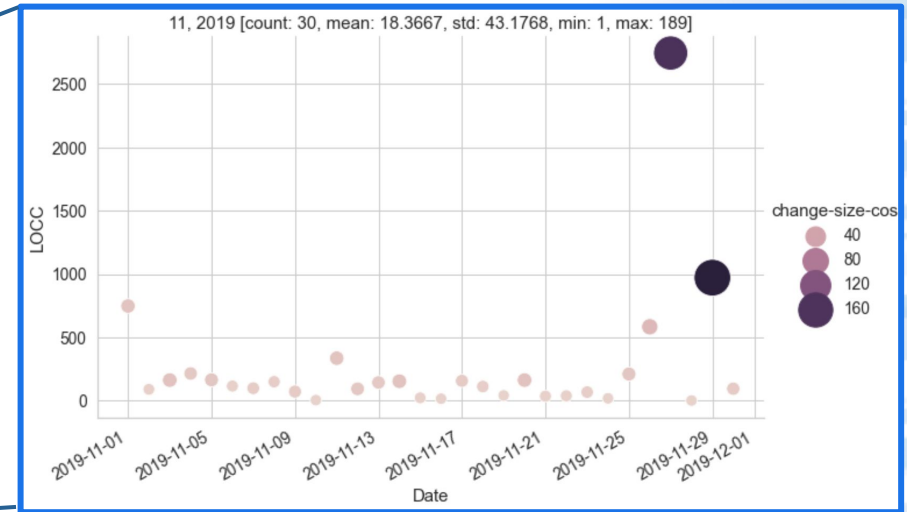
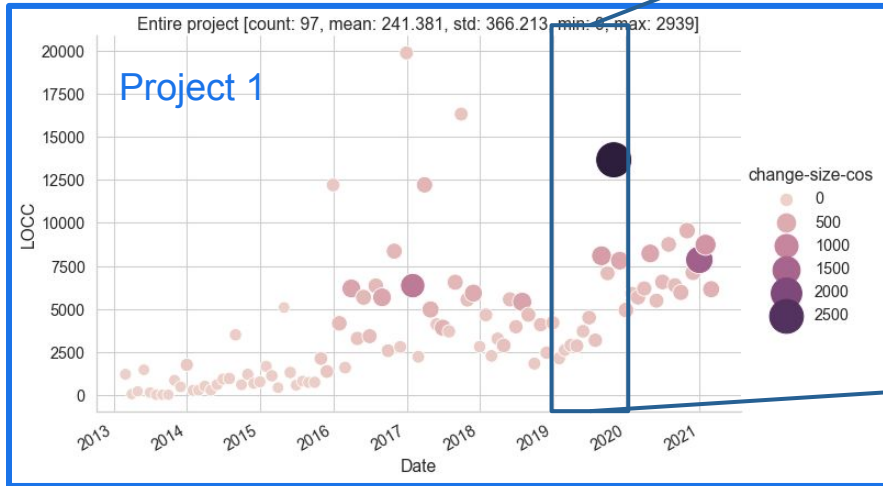
Combining different “change size” metrics



Example on right: simple LOCC totals and cos distance



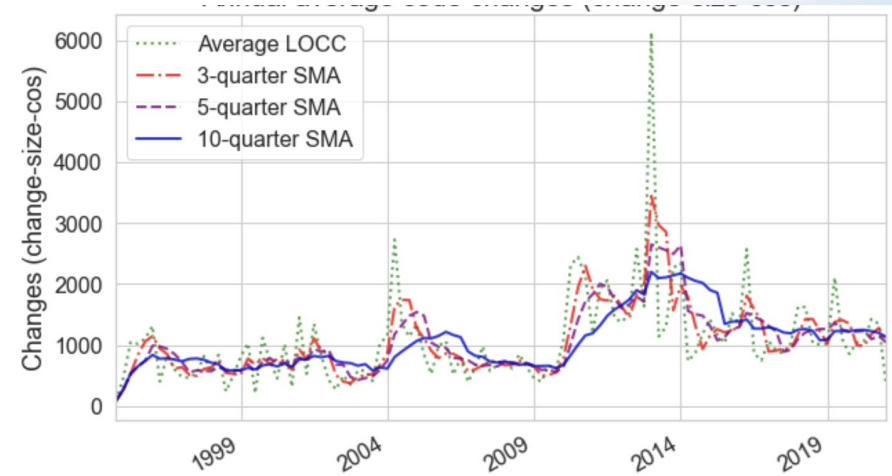
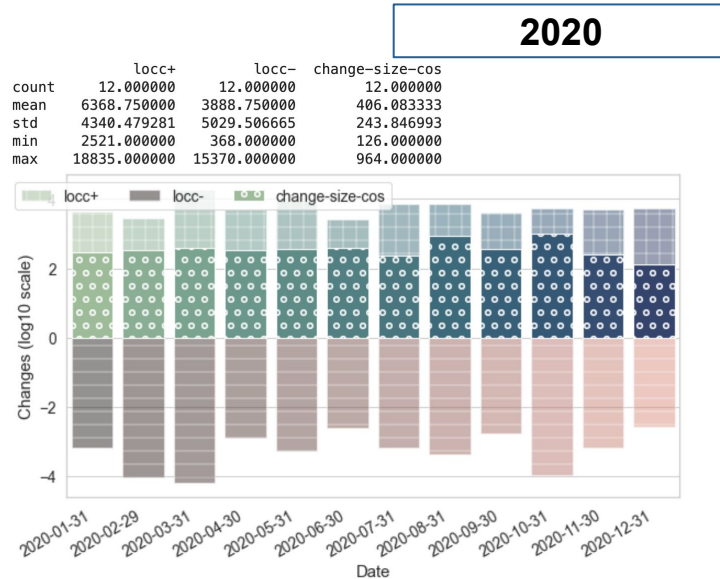
Significant events



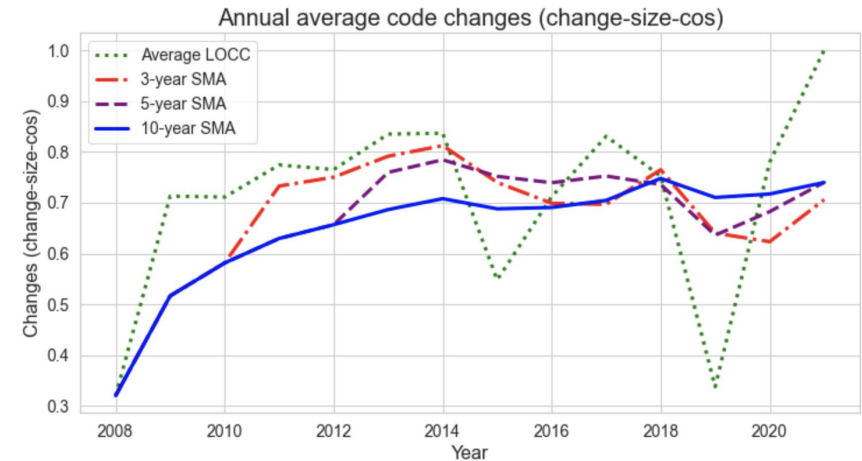
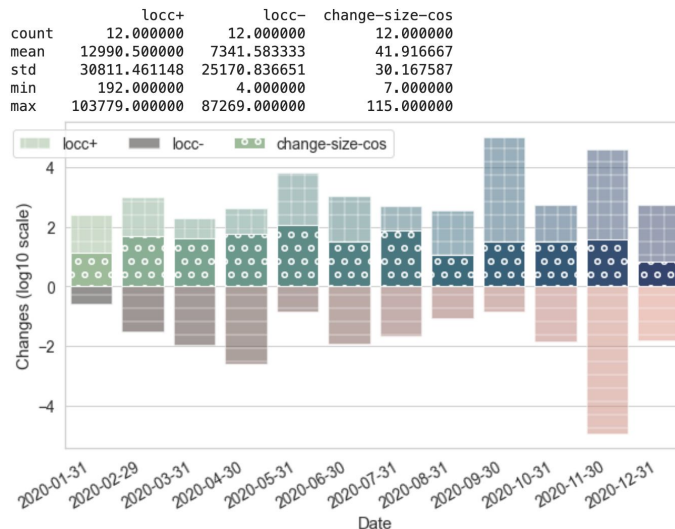
No single view tells the whole story

(project names omitted on purpose)

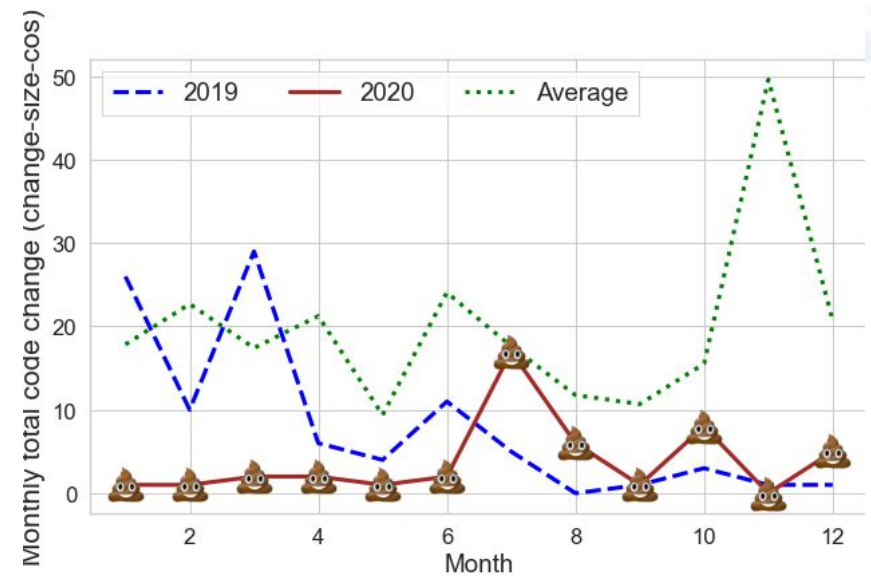
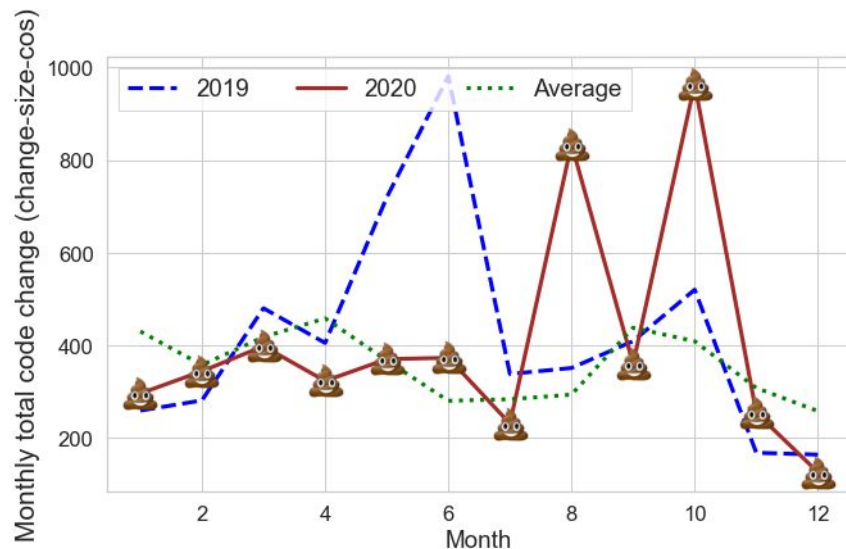
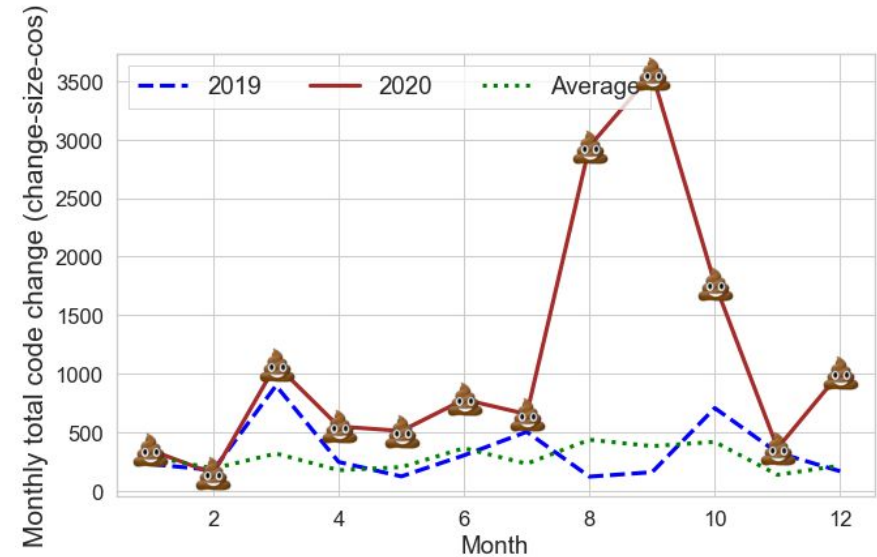
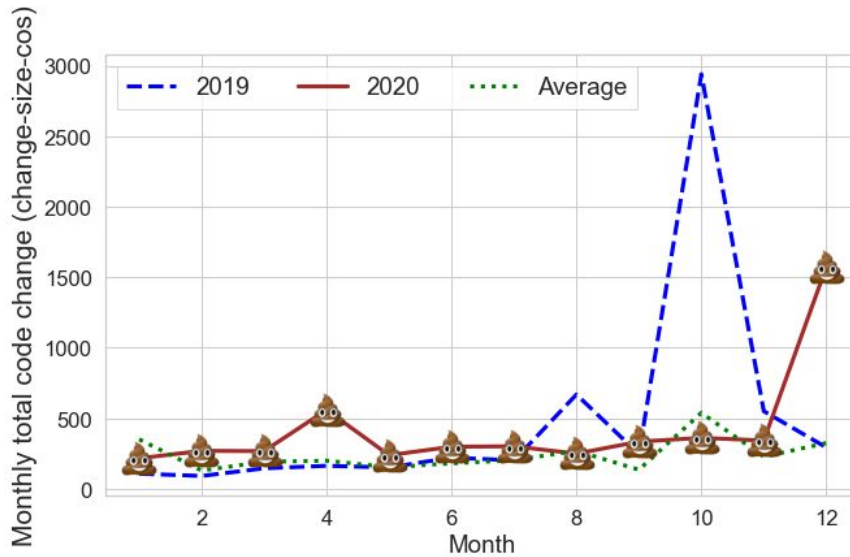
Project 1



Project 2



How do projects weather interesting times?



Conclusions



We have created a
flexible,
efficient,
and usable
software framework for
acquiring,
storing,
manipulating,
and visualizing
development-related data.



We demonstrated a few of its capabilities here; an ever growing
number of patterns and other types of analysis are constantly being
developed

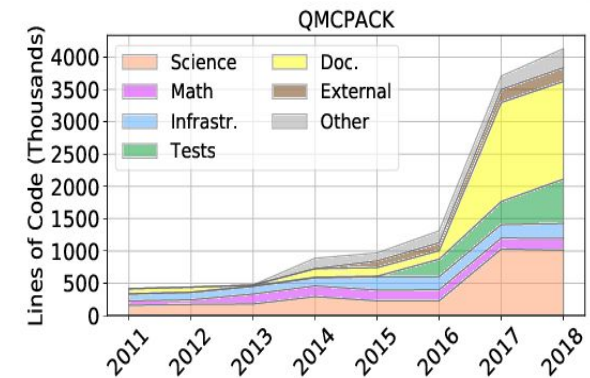
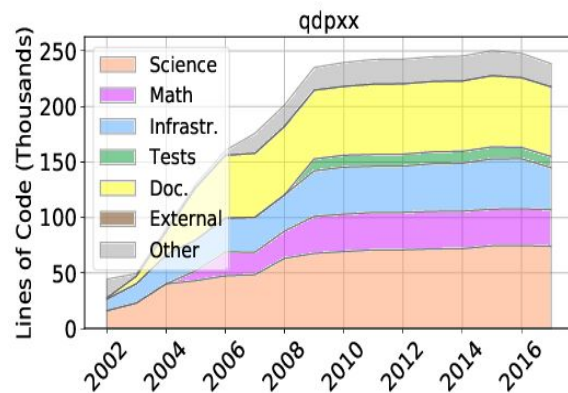
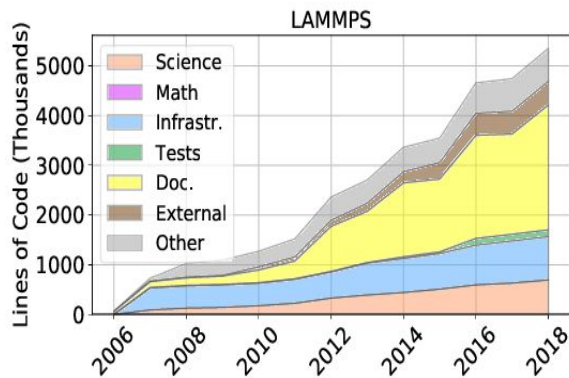
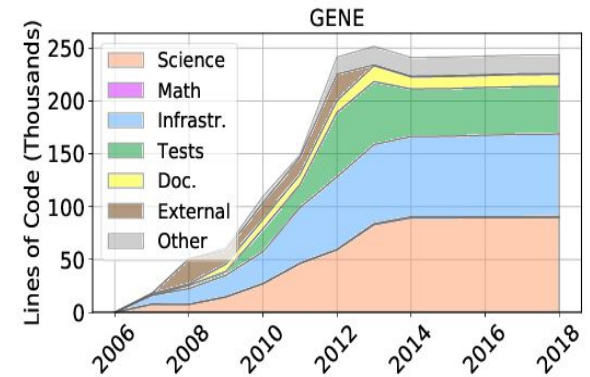
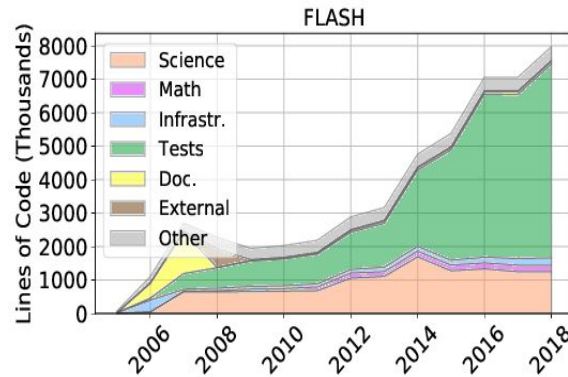
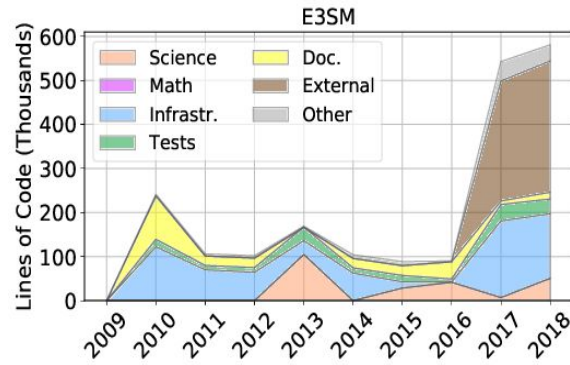


Requests/challenges welcome!

Thank you!

Projects that may or may not have been represented in this talk: Spack, LAMMPS, PETSc, Nek5000
E3SM, QMCPACK, QDPXX, LATTE NAMD, fast-export, Enzo, TAU2, xpress-apex, LATTE, NWChem

Where is development effort going?



Grannan, A., Sood, K., Norris, B., & Dubey, A. (2020). Understanding the landscape of scientific software used on high-performance computing platforms. *The International Journal of High Performance Computing Applications*. <https://doi.org/10.1177/1094342019899451>