

# A scientific approach to studying opensource communities

Nelle Varoquaux

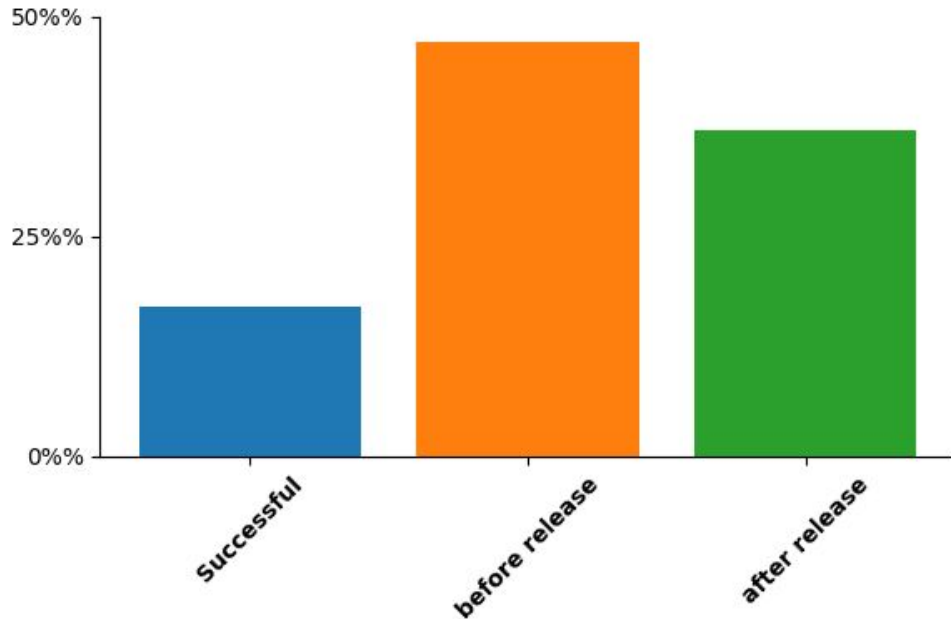
The opensource  
community is a big  
part of my success in  
academia



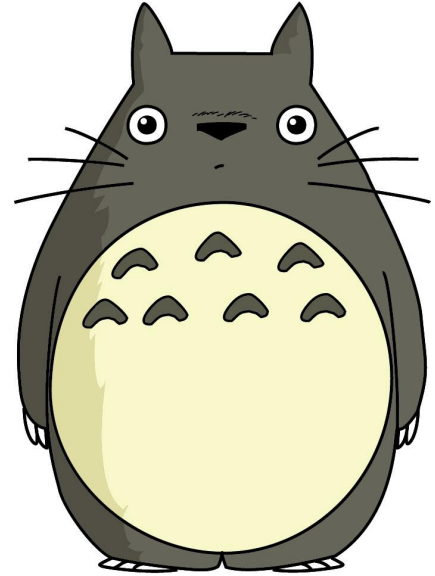


# But, most opensource software are not successful

Less than one out of six open source projects are successful.



So why is that?



“Come for the code  
stay for the  
community.”

--Brett Canon



“A FOSS project is only  
as successful as its  
community.”

--Pieter Hintjens



So how can we built an efficient community?





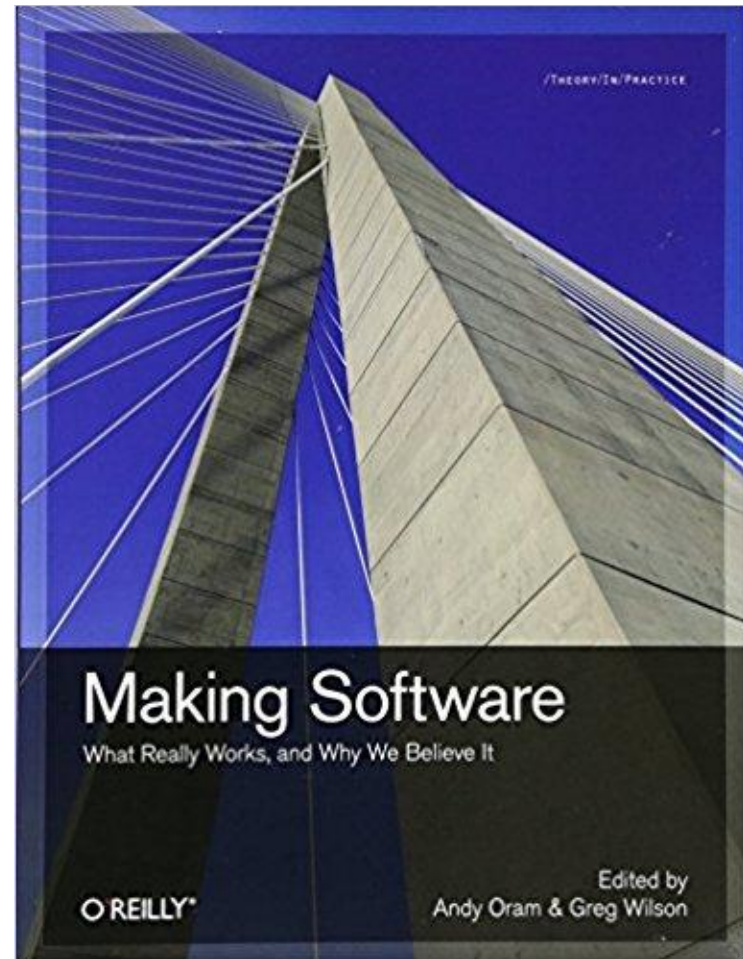
Can we tackle this question through a  
rigorous scientific approach?



Why is a scientific approach necessary?



“Which claims  
are true and  
which are mere  
wishful thinking?”



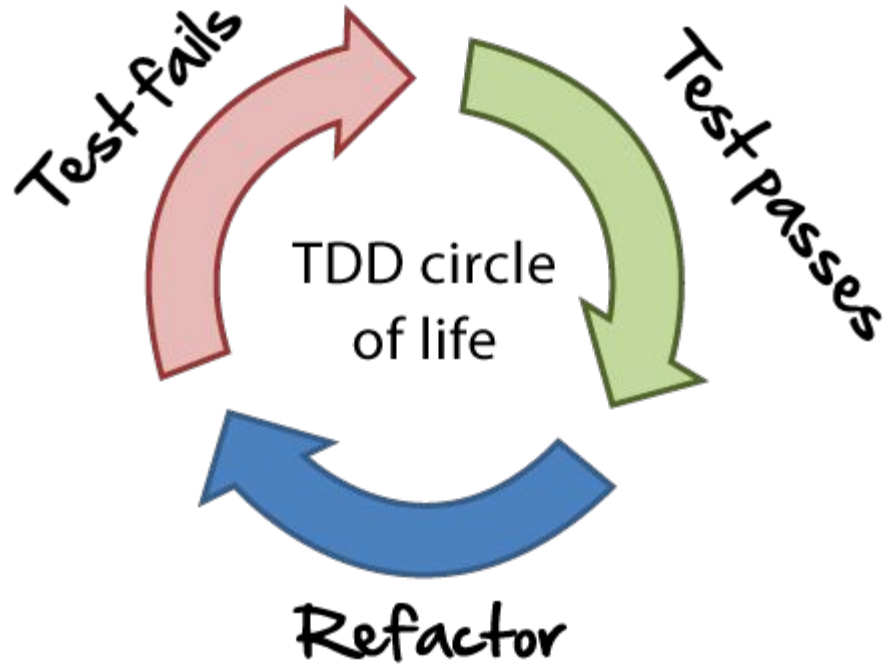
# Treating breast cancer



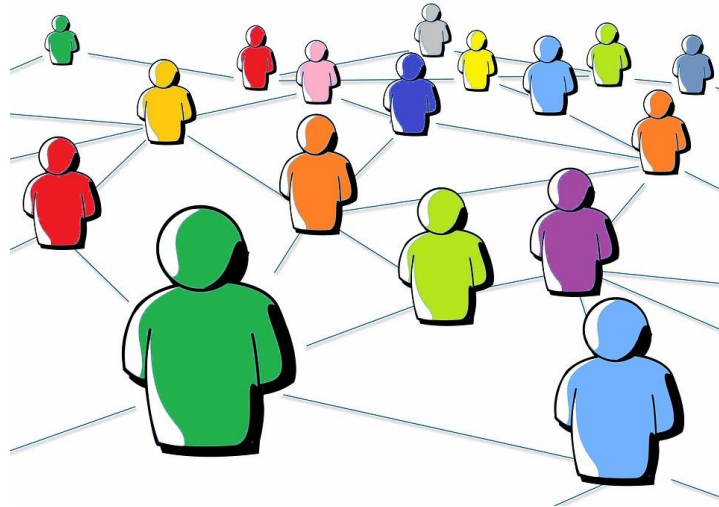
# Brain surgery



Does  
test-driven  
development  
really work?



# How can we study communities using scientific approaches?



# The Berkeley Institute for Data Science





# The Berkeley Institute for Data Science



“Ethnography is an approach to understanding cultural life that is founded not just on witnessing but on participation, with the goal of understanding not simply what people are doing, but how they experience what they do.”

--Paul Dourish, *Reading and Interpreting Ethnography*



# Ethnography methods

- Interviews (un/semi-structured)
- Focus groups
- Field observation
- Participant-observation
- Case studies
- Oral histories
- Archival research
- Surveys\*



# Ethnography of computations

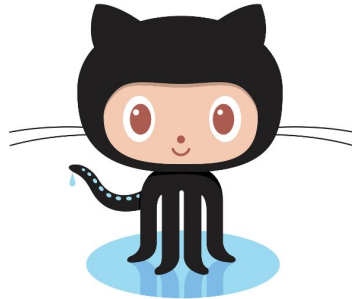
Using traditional ethnographic methods **to study how people in particular cultural contexts relate to computation and/or relate to each other through computation.**

How do people design, develop, deploy, document, debate, maintain, manage, use, not use, learn, or teach computation and computational systems in their everyday life and work?

What are the different perspectives and attitudes people have towards computational systems, artifacts, and methods?

# What kind of questions can we answer?

- Why are some programmers 10 times more efficient than others?
- Does TDD really work?
- Has new infrastructure impacted the way opensource communities interact?
  - Has Github changed anything?
  - Does travis allow to reduce the number of bugs?
- What is the motivation/incentive behind contributing to opensource software?
- Why do/don't people write documentation?



# A scientific approach to understanding open source communities

*The faces, uses, practices, and tensions around documentation in  
opensource software communities*

# Why look at documentation?

“Successful projects have some common characteristics [..]  
**Good project communication** -- a quality website, good documentation, a bug-tracking system and a communication system such as an email list or forum.”

--Rich Gordon, summarizing Schweik & English



# Why look at documentation?

In a 2017 GitHub survey of OSS contributors, 93% reported that “incomplete or outdated documentation is a pervasive problem”

Yet...

“60% of contributors say they rarely or never contribute to documentation”



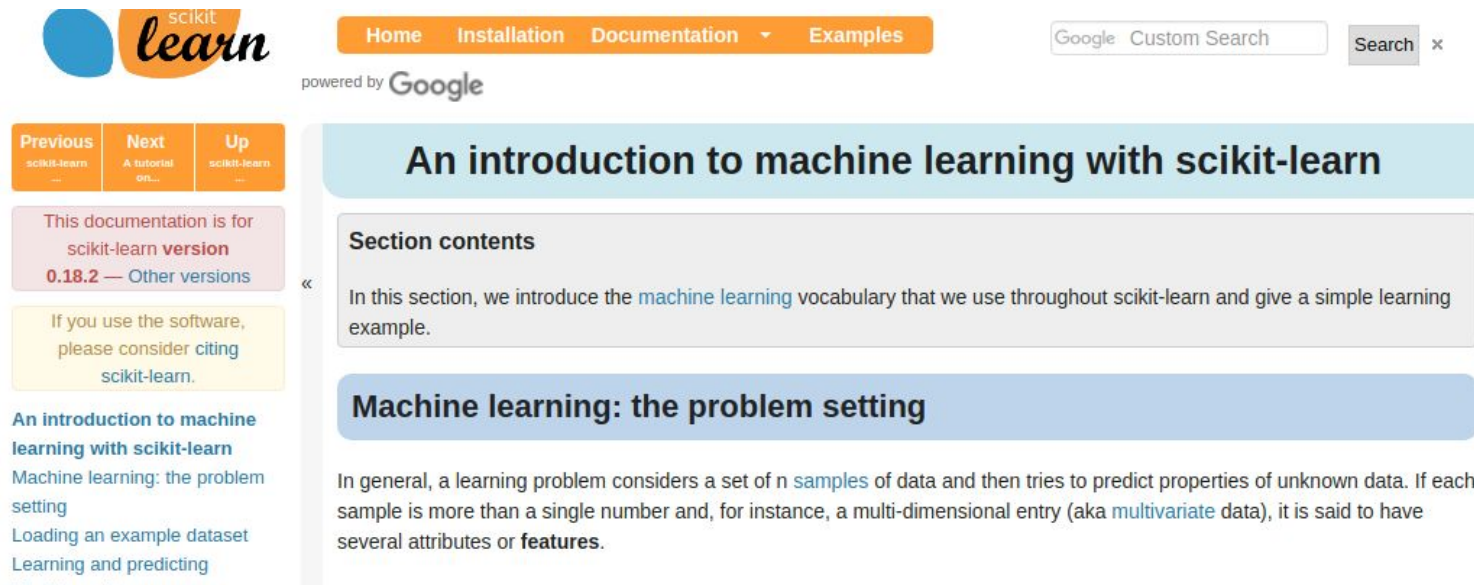
Why is that?



# Types of documentation

# Types of documentation

- Literal documentation



The screenshot shows the scikit-learn documentation website. At the top left is the scikit-learn logo. A navigation bar contains links for Home, Installation, Documentation (with a dropdown arrow), and Examples. To the right is a search box with the text 'Google Custom Search' and a 'Search' button with a close icon. Below the navigation bar, it says 'powered by Google'. On the left side, there are three buttons: 'Previous' (scikit-learn), 'Next' (A tutorial on...), and 'Up' (scikit-learn). Below these is a pink box stating 'This documentation is for scikit-learn version 0.18.2 — Other versions'. Below that is a yellow box with the text 'If you use the software, please consider citing scikit-learn.' At the bottom left, there is a list of links: 'An introduction to machine learning with scikit-learn', 'Machine learning: the problem setting', 'Loading an example dataset', and 'Learning and predicting'. The main content area has a light blue header for the page title 'An introduction to machine learning with scikit-learn'. Below this is a 'Section contents' section with a left-pointing arrow and the text 'In this section, we introduce the machine learning vocabulary that we use throughout scikit-learn and give a simple learning example.' Below that is another light blue header for 'Machine learning: the problem setting', followed by the text 'In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or features.'

# Types of documentation

- Literal documentation
- API documentation

pandas 0.20.2 documentation » API Reference » pandas.Series »

## Table Of Contents

- What's New
- Installation
- Contributing to pandas
- Package overview
- 10 Minutes to pandas
- Tutorials
- Cookbook
- Intro to Data Structures
- Essential Basic Functionality
- Working with Text Data
- Options and Settings
- Indexing and Selecting Data
- Multindex / Advanced Indexing
- Computational tools
- Working with missing data
- Group By: split-apply-combine
- Merge, join, and concatenate
- Reshaping and Pivot Tables

## pandas.Series.value\_counts

`Series.value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)` [\[source\]](#)

Returns object containing counts of unique values.

The resulting object will be in descending order so that the first element is the most frequently-occurring element. Excludes NA values by default.

### Parameters:

**normalize** : *boolean, default False*

If True then the object returned will contain the relative frequencies of the unique values.

**sort** : *boolean, default True*

Sort by values

**ascending** : *boolean, default False*

Sort in ascending order

**bins** : *integer, optional*



But there are many other types of documentation...





# Roles of documentation

# Roles that documentation plays...

- Learning/Education

*“You can imagine a user, with some sort of need, Googling around trying to find some sort of software to do what they want to do. Then they happen upon software and try it. There’s this patience period that probably is something like five minutes, during which they may try a software. Then it might not work, probably won’t work. Then if there’s no documentation to help, that user is basically lost for that software project and will say, ‘I tried that but it didn’t work.’ **You need documentation, like ideally of everything but especially of the very beginning of, to create a minimal user experience and have it in the documentation how to set the thing up and how to do the thing that it’s supposed to do”***

# Roles that documentation plays...

- Learning/Education
- Publicity / signal of health

*“it’s certainly **the case I decide whether to use a project or not based on the quality of the documentation** [...] If I’m looking for a library that does something and I have, you know, five libraries, there are different criteria that I use to decide which one I’m going to use but **quality of the documentation** is certainly one of them [...] “*

# Roles that documentation plays...

- Learning/Education
- Publicity / signal of health
- Institutional memory

***“I care more and more because I come across more and more things I’ve written a couple of years ago, and I have no clue what I fricking wrote”***

# Roles that documentation plays...

- Learning/Education
- Publicity / signal of health
- Institutional memory
- Testing verification

*“It also allows me to make sure that I understand exactly when the method works and when it doesn’t work. [...] it also allows us to check that the API is nice, and it’s also a very simple way to check that the method works.”*

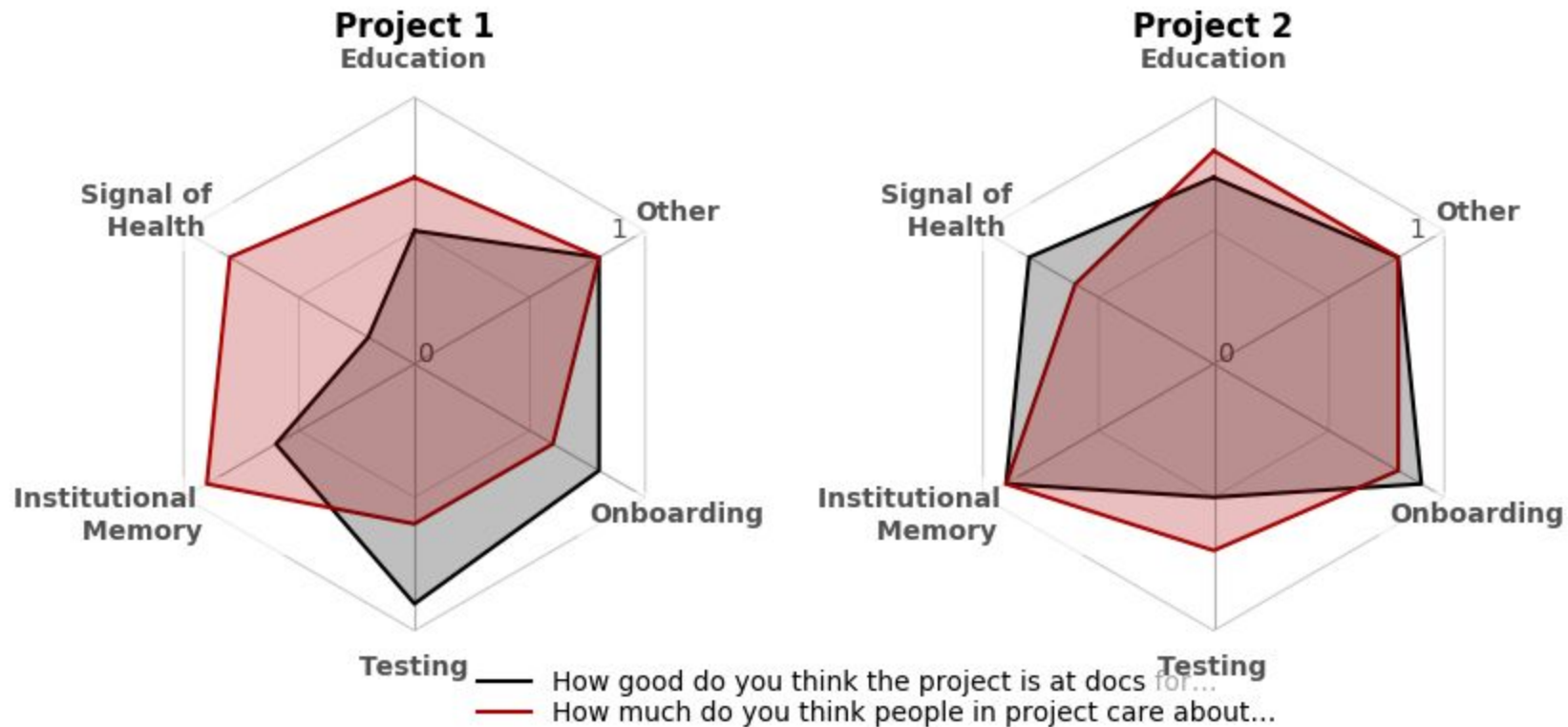
# Roles that documentation plays...

- Learning/Education
- Publicity / signal of health
- Institutional memory
- Testing verification
- Onboarding newcomers

*“docs are something that anyone can sort of critique and improve, even if they don't necessarily have a deep knowledge about the code base.*

“

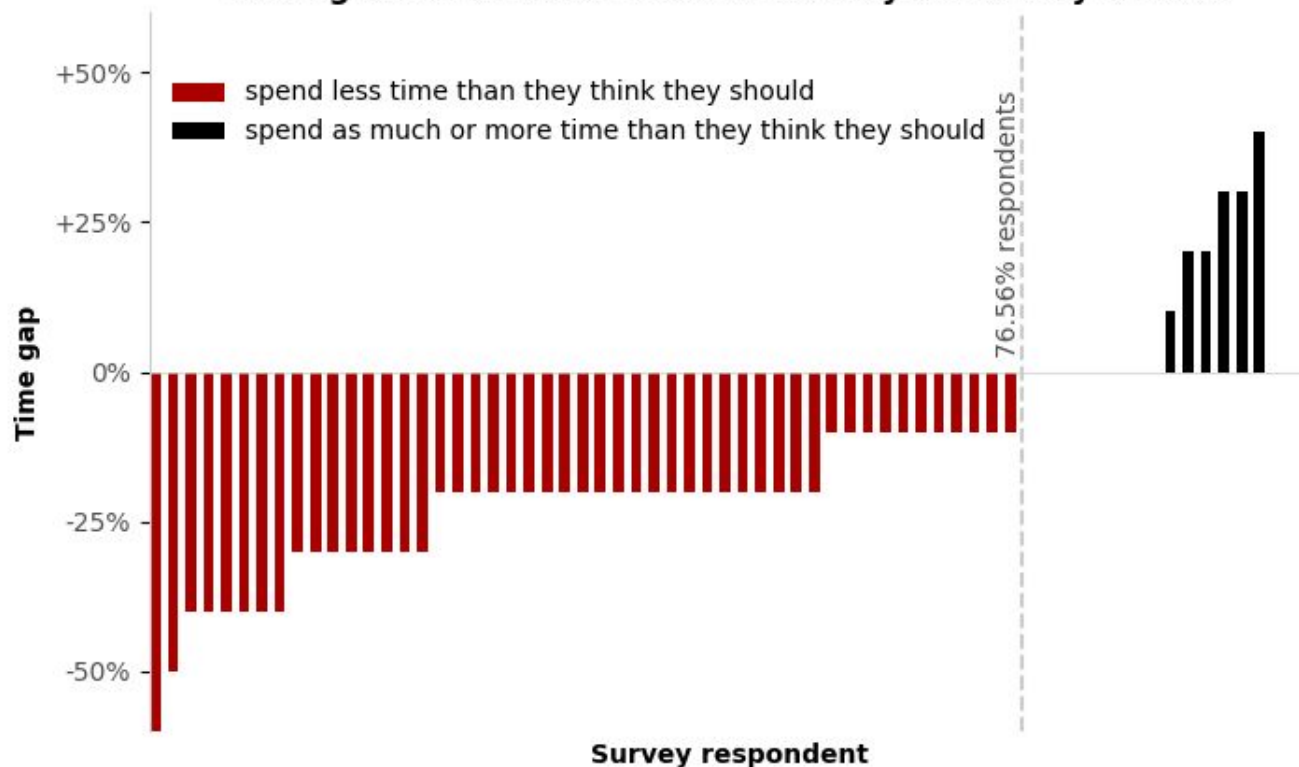
## A contributor's ratings of documentation's qualities and values in two FOSS projects



# Barriers to writing documentation



## Less than 25% of respondents spend as much or more time writing documentation than what they think they should.



“Everyone hates writing documentation”

--Anonymous Docathon participant



# Lacking the skills

- Self-efficacy

*“I don’t know many people who enjoy writing documentation. I think one of the reasons being it’s not a skill that we learn very well, so I think a **lot of us feel that it’s not something we’re good at.** If we have been feeling different, that we’re good at it, probably we would enjoy it more, but it’s sort of a painful process to do.”*

# Lacking the skills

- Self-efficacy
- Empathy

***“The biggest problem is that what I need in documentation is not necessarily what someone coming to the library using documentation does. I may be lacking sufficient empathy to write what newcomers need. Whereas a newcomer probably still remembers what they didn’t know yesterday and can write the docs with that in mind.”***

# Lacking the skills

- Self-efficacy
- Empathy
- Language proficiency

*“You need to be able to form complete English sentences, which is challenging for some of us.”*



# Lacking the skills

- Self-efficacy
- Empathy
- Language proficiency
- Communication skills
- Knowledge of the software to be documented

***“Creative writing is important, to enable search to boil down whatever are the key features of the software, and also what the science of the software is doing, down to clear explanations.”***

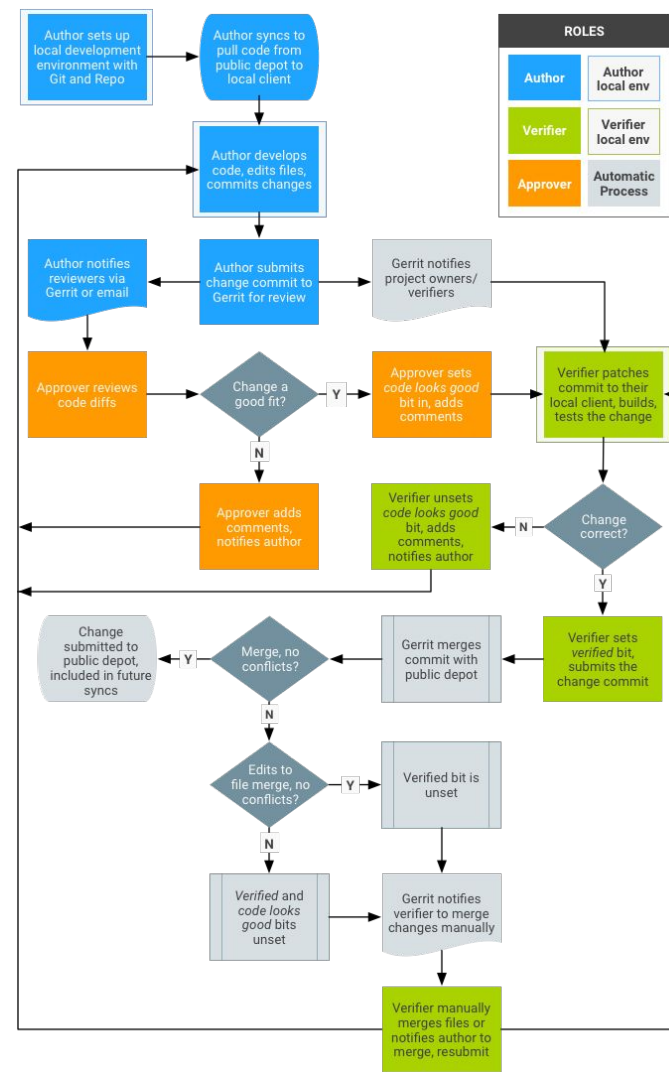
# Lacking the skills

- Self-efficacy
- Empathy
- Language proficiency
- Communication skills
- Knowledge of the software to be documented



# Technical barriers

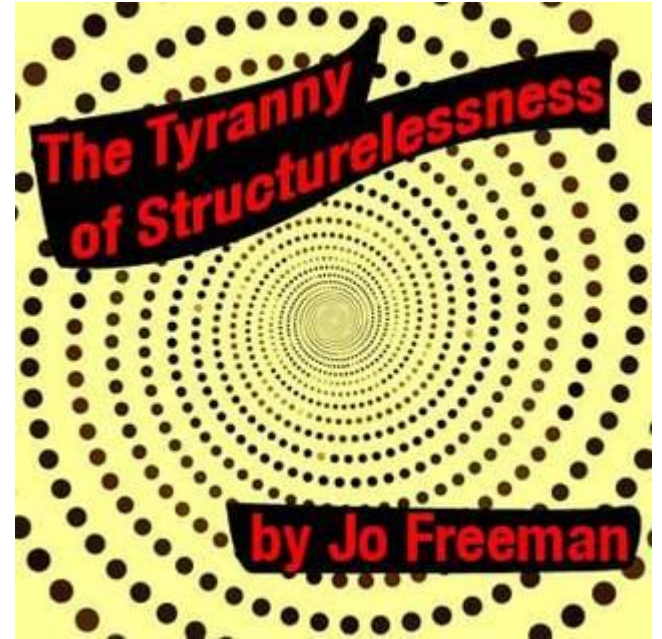
*“Projects use many platforms, tools, and practices to manage their workflow of contributing code and documentation, each of which has its own learning curve.”*





# Tyranny of structurelessness

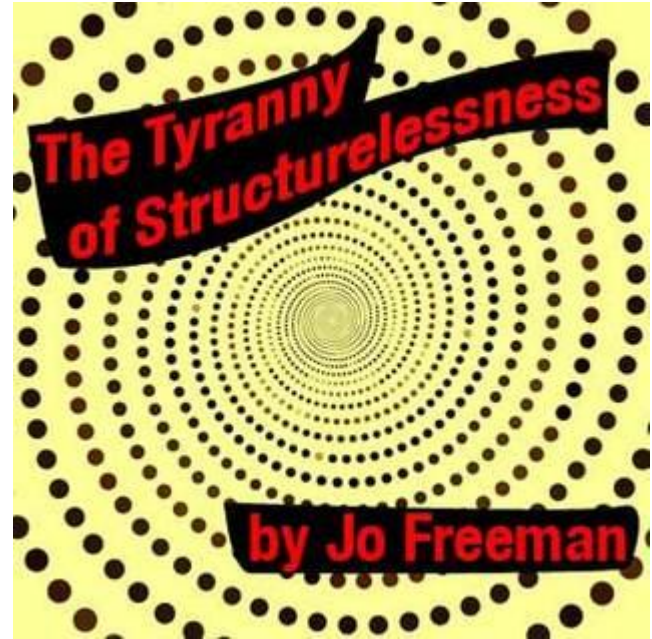
*“Docstrings are supposed to be pretty terse and straightforward and those I’m not worried about doing on volunteer effort. Because again, basically, you take all the voice out and they say this is what it does, these are the parameters, this is what it returns.”*



# Tyranny of structurelessness

*“Sorry, this is somewhat annoying. Writing documentation isn't the most fun part, but I thought I could give a little help here. But **it's no use if there are no clear rules.**”*

*<http://website.org/documentation>  
referenced in #PR  
and your comments contradict  
each other in parts.”*



# Lessons learned

1. Make it easy to contribute to your software.
  - a. Follow conventions
  - b. Use standard tools
2. Make writing documentation part of your contribution pipeline
3. Automate verification as much as possible.
  - a. Remove as much subjectiveness as possible

What do scientific python packages do in practice?

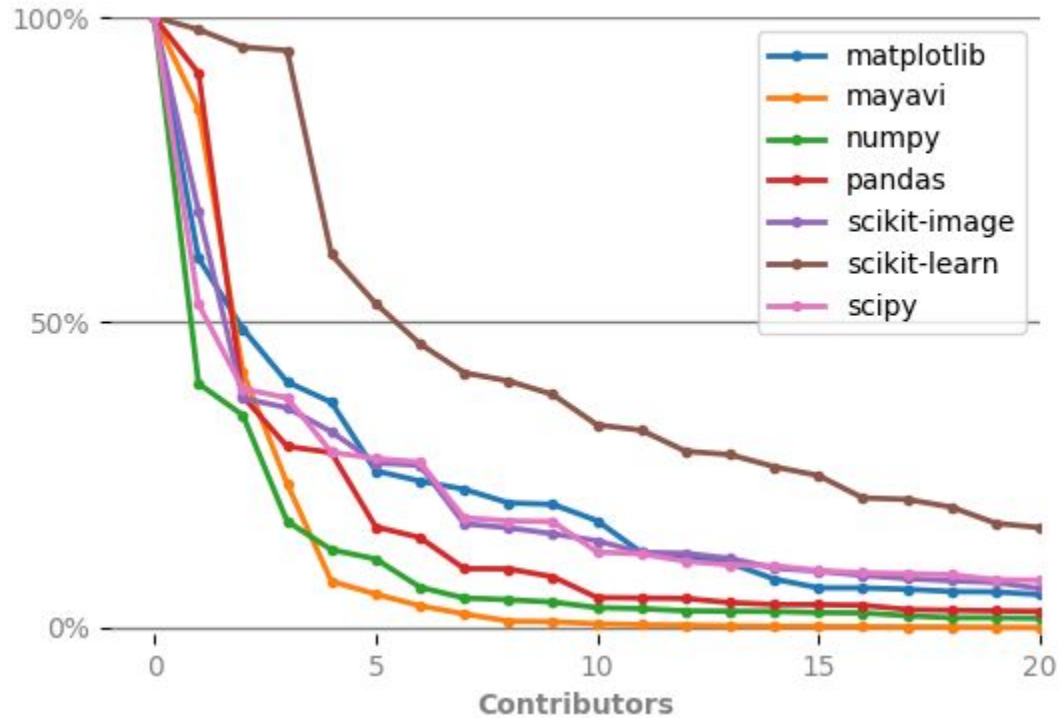
# In practice...

	Literal documentation	API	Gallery
numpy	✓	✓✓	
scipy	✓	✓✓	
pandas	✓	✓✓	
matplotlib	✓✓	✓	✓✓
mayavi	✓✓	✓	✓✓
scikit-learn	✓✓	✓✓	✓✓
scikit-image	✓✓	✓✓	✓✓

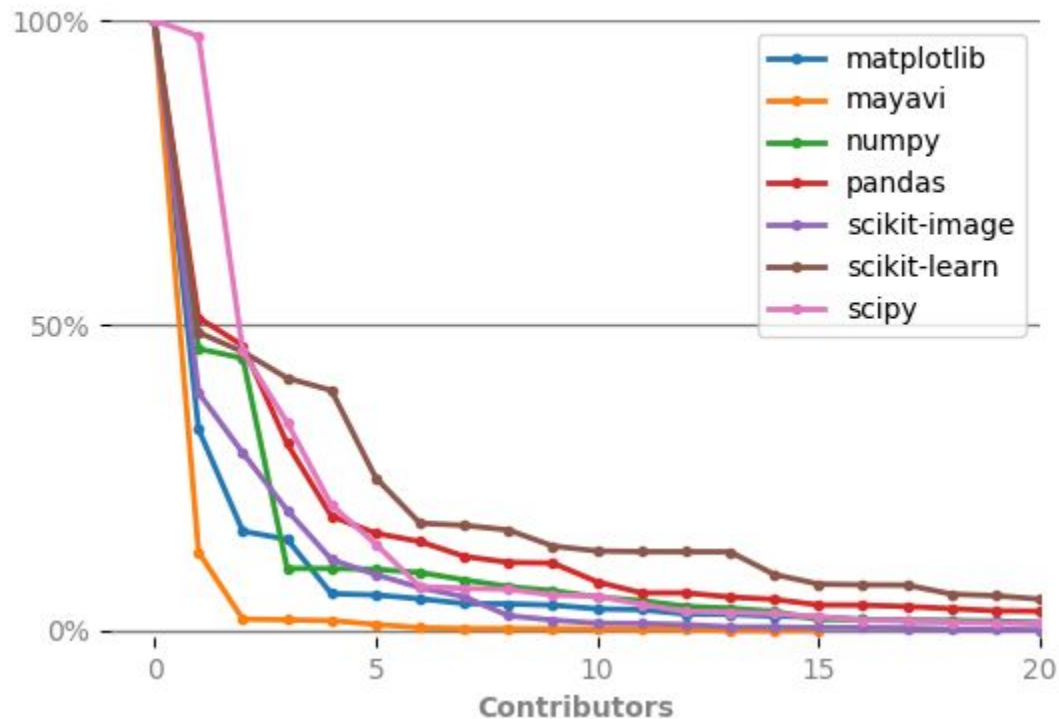
# Measuring the bus factor



# Code bus factor of major Python FOSS

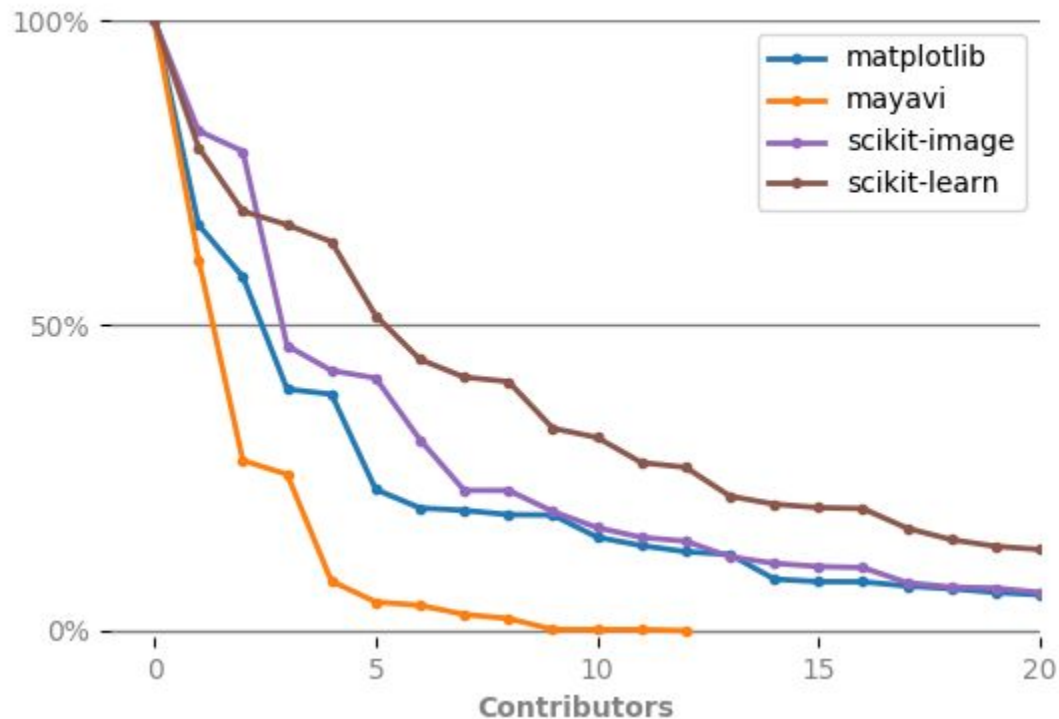


# Documentation bus factor of major Python FOSS

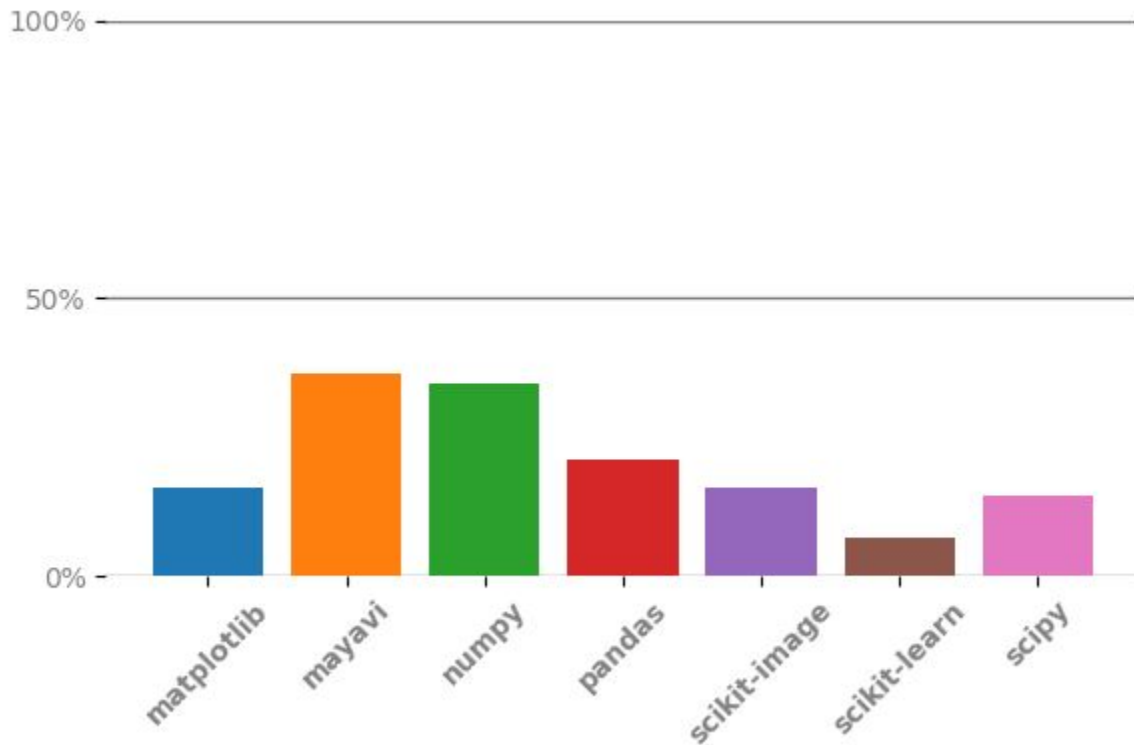




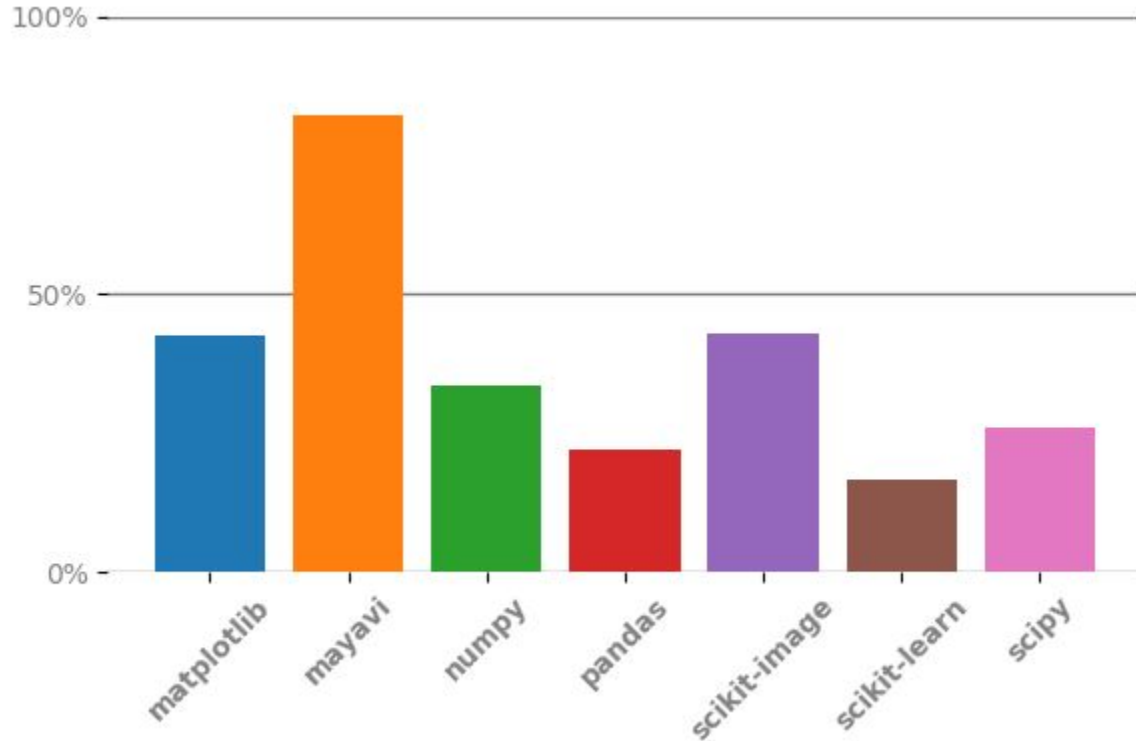
# Gallery bus factor of major Python FOSS



# Top contributor's code contribution



# Top contributor's documentation contribution



Thank you!



# Top contributor's documentation contribution

