

# Building Software and Communities With Peer Review: rOpenSci, pyOpenSci, and Beyond

Noam Ross  
PyData NYC, 2019-11

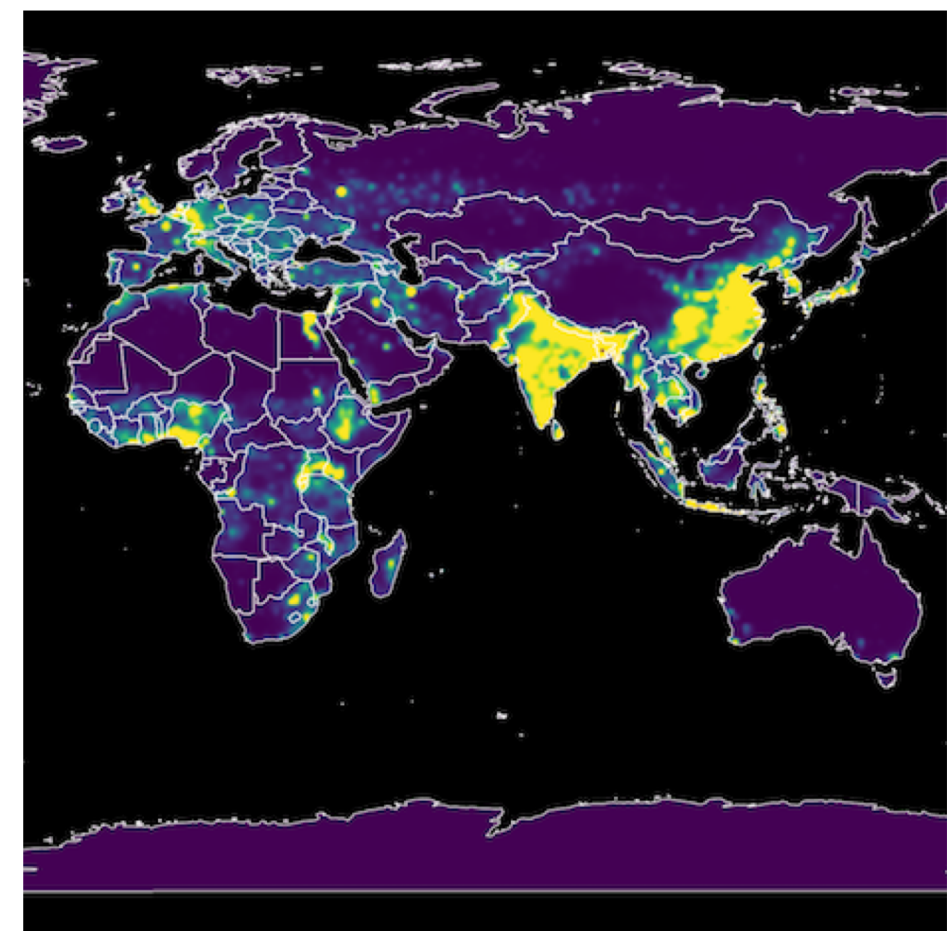
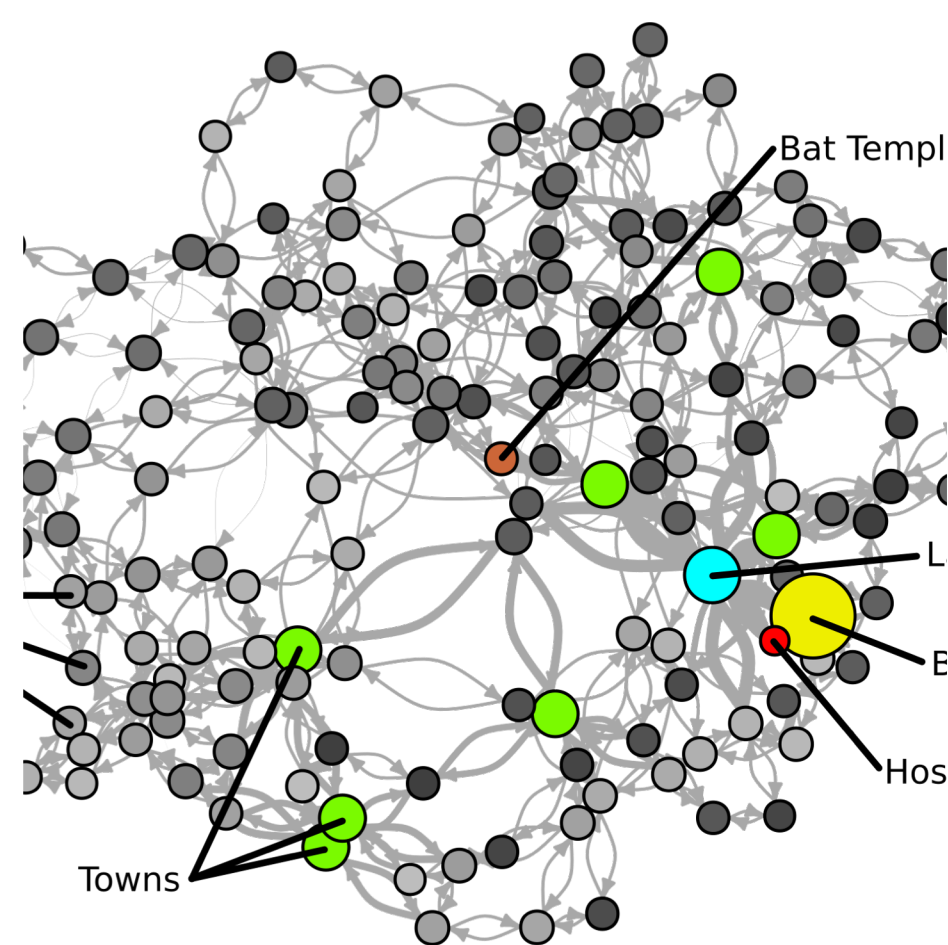
noamross @   



**EcoHealth  
Alliance**







EcoHealth Alliance



@EcoHealthNYC

ecohealthalliance.org





Building technical and community  
infrastructure for R to support  
open, reproducible science

@rOpenSci  
ropensci.org





pyOpenSci

@pyOpenSci  
pyopensci.org



# Packages

Our packages are carefully vetted, staff- and community-contributed R software tools that lower barriers to working with scientific data sources and data that support research applications on the web. Read our [blog](#) to learn how to use specific packages or contribute to their improvement. Browse our [tutorials](#) and [usecases](#).

Curious about contributing your package? See [our Software Peer Review page](#) for details. We welcome [volunteers to review](#) packages submitted to our open peer review process.

## FILTERS

- All
- Altmetrics
- Data Publication
- Tools
- Visualization
- Databases
- Geospatial
- Web
- Images Processing
- Literature
- Computing Infrastructure
- Security
- Taxonomy
- CRAN / BIOC available
- Active repos

NAME	MAINTAINER	DESCRIPTION	DETAILS
------	------------	-------------	---------












Our packages are carefully vetted, staff- and community-contributed R software tools that lower barriers to working with scientific data sources and data that support research applications on the web. Read our [blog](#) to learn how to use specific packages or contribute to their improvement. Browse our [tutorials](#) and [usecases](#).

Curious about contributing your package? See [our Software Peer Review page](#) for details. We welcome [volunteers to review](#) packages submitted to our open peer review process.

#### FILTERS

- All
- Altmetrics
- Data Publication
- Tools
- Visualization
- Databases
- Geospatial
- Web
- Images Processing
- Literature
- Computing Infrastructure
- Security
- Taxonomy
- CRAN / BIOC available
- Active repos

	NAME	MAINTAINER	DESCRIPTION		DETAILS
	<a href="#">auk</a>	<a href="#">Matthew Strimas-Mackey</a>	eBird Data Extraction and Processing in R		
	<a href="#">genbankr</a>	<a href="#">Gabriel Becker</a>	Parsing GenBank files into semantically useful objects		
	<a href="#">treeio</a>	<a href="#">Guangchuang Yu</a>	Base Classes and Functions for Phylogenetic Tree Input and Output		



**(Production)  
Code Review**

granular

narrow scope

project collaborators

quality control

**(Academic)  
Peer Review**

project-wide

broad scope

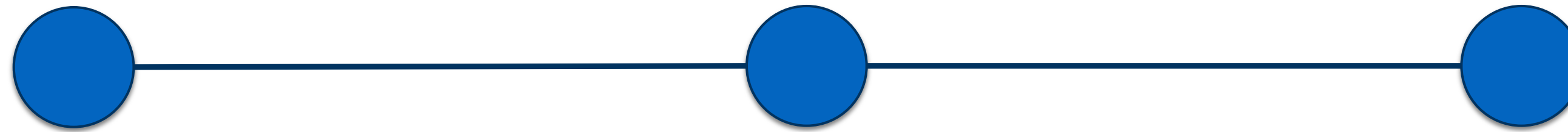
(anonymous) peers in  
your field

quality + legitimacy



Frequent  
iteration,  
High  
familiarity

Infrequent  
review,  
Low  
familiarity



Pair  
Programming

Pull Request  
Review

Project Peer  
Review



# Why peer review?

drive adoption of best practices and standards

increase quality in the long tail of applications

build a community of practice

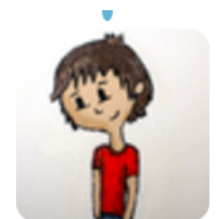


# Why peer review?

drive adoption of best practices and standards

increase quality in the long tail of applications

build a community of practice



**Manuel Ramón**

@manuramon

 Follow



Replying to [@noamross](#)

Thanks [@noamross](#). I really love the [@rOpenSci](#) package review process, especially its interactivity and how much you can learn from others

*"The review process taught me a lot about different tools available for making my code more robust and resilient to future changes."*

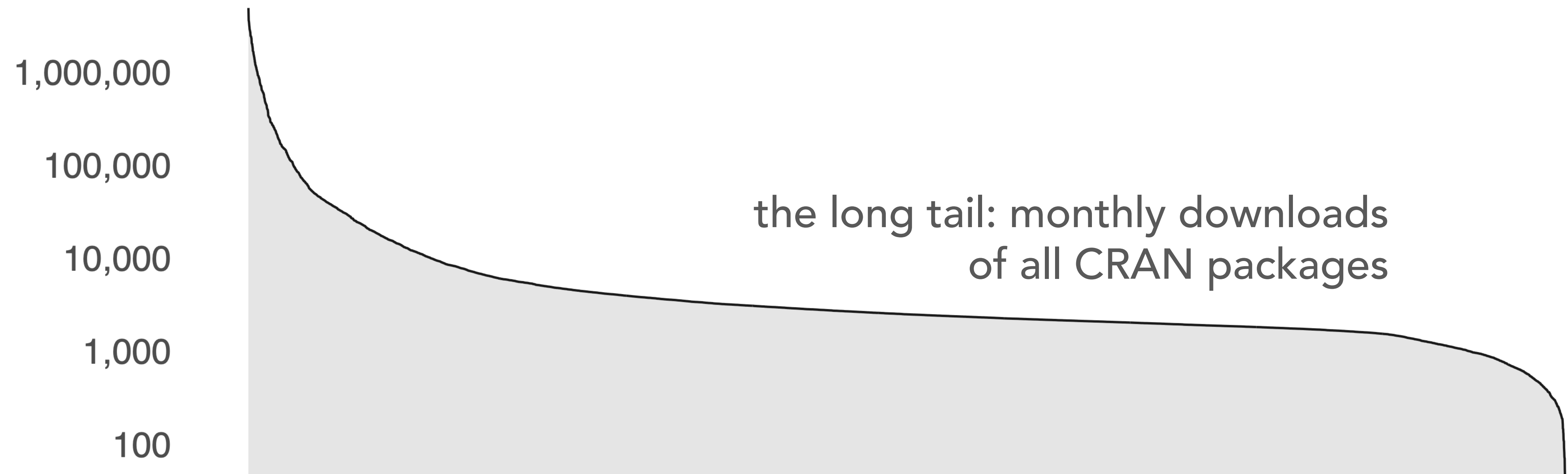
*"I learn a lot from closely reading other people's code and it is hard to do when I'm not forced to review so closely."*

# Why peer review?

drive adoption of best practices and standards

increase quality in the long tail of applications

build a community of practice





# Why peer review?

drive adoption of best practices and standards

increase quality in the long tail of applications

build a community of practice

The logo for DBHYDRO, consisting of the word "DBHYDRO" in a bold, sans-serif font. The letters are a light blue color and are set against a slightly darker blue rectangular background.

## Programmatic access to the South Florida Water Management District's [DBHYDRO database](#)

repo status **Active** build **passing** CRAN **0.2-2** downloads **353/month**

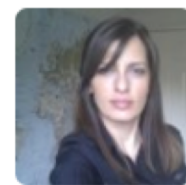
`dbhydroR` provides scripted access to the South Florida Water Management District's DBHYDRO database which holds over 35 million hydrologic and water quality records from the Florida Everglades and surrounding areas.

# Why peer review?

drive adoption of best practices and standards

increase quality in the long tail of applications

build a community of practice



cvitolo commented on Aug 28, 2016

Member



Thanks for the suggestion @jennybc! I have updated `rmarkdown` and `knitr` but that did not fix the problem. As a temporary workaround I manually edited the README.md file and removed the string .

I also noticed that the leaflet map is not rendered and on its place is the line , not sure what's going on there.



maelle commented on Aug 29, 2016

Member



@cvitolo great work! A few points

- There's no unit test for `ukair_get_coordinates` from what I see on codecov.io?
- In `ukair_get_hourly_data` you could output a warning if the side ID is not in the cached version of the catalogue. For instance imagine I enter a wrong ID, this is what I get:



# Why peer review?

drive adoption of best practices and standards

increase quality in the long tail of applications

build a community of practice

*"[I liked] meeting people in the R community who I didn't know before, and strengthening ties to the community."*



**Michael Koontz**

@\_mikoontz

Following



Such a cool process to do peer review of an [#Rstats](#) package for [@rOpenSci](#)! What a great group of folks to work with.



**Andy South**

@southmapr

+ Follow



Thankyou [@rOpenSci](#) for the great package review process & [@lincolnmullen](#) [@robinlovelace](#) [@sckottie](#) for reviewing [rnaturalearth](#)

# Why peer review?

training

quality control

team-building



# How to do a review?

Use a set of shared standards

Automate all you can

Do it their way: Run the author's workflow

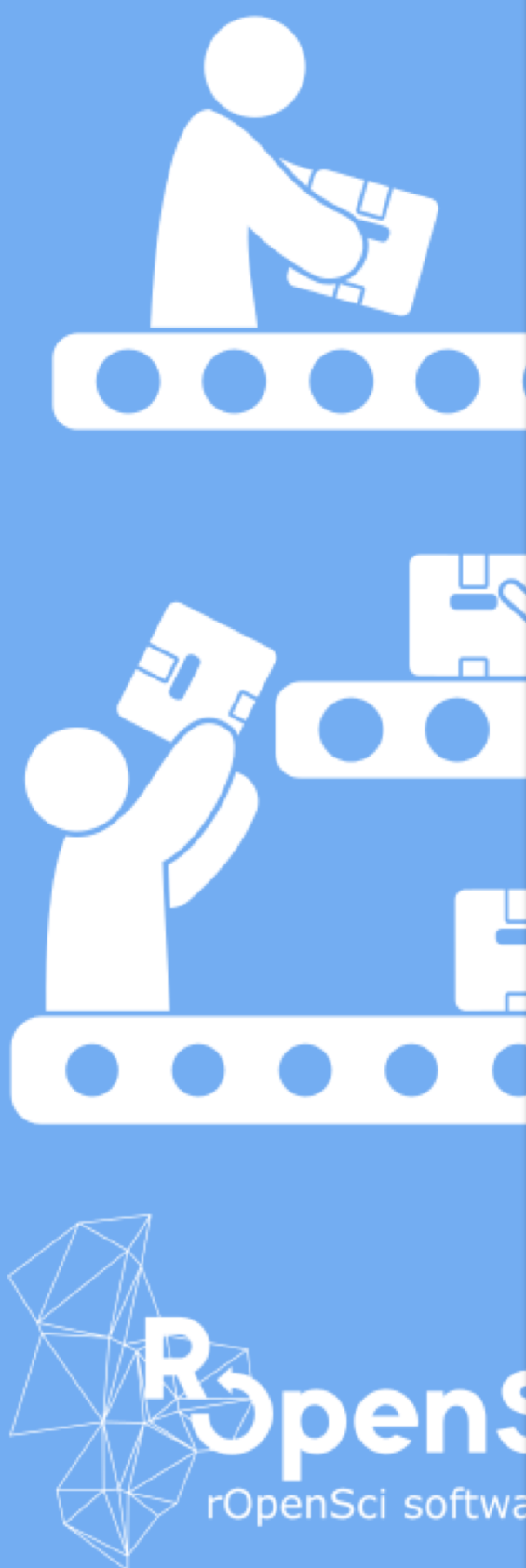
Do it your way: Break the author's workflow

Make a map of the source: Files, Functions, Tests, and Docs

Write as you follow your map

# rOpenSci Packages

*Development, Maintenance,  
and Peer Review*



## The pyOpenSci Developer Guide

[Home](#)

Search

1. Before Review - Packaging Guide

1.1 Questions and Help

2. Peer Review Process

2.1 Aims and Scope

2.2 Review Process and Guidelines

2.3 Guide for Authors

2.4 Guide for Reviewers

2.5 Guide for Editors

2.6 Community Code of Conduct

3. After Review - Maintenance

3.1 Collaboration



This guidebook contains information for pyOpenSci package authors, reviewers, and editors. It is organized into three sections:

### 1. Before Review - Packaging Guide

Contains our guidelines for creating and testing scientific python packages. Before submitting a package for review, check to be sure that your software meets the basic [requirements](#). The section also contains recommendations and best practices that might be helpful as you are writing and preparing your package.

### 2. Peer Review Process

Outlines the pyOpenSci peer review process. This includes guidelines for submitting and reviewing packages, as well as our [Code of Conduct](#). The [Aims and Scope](#) section lays out what types of packages we are able to review. If you are unsure whether your package fits, we encourage you to submit a [presubmission inquiry](#)

### 3. After Review - Maintenance

Provides tips and advice for maintaining and promoting your pyOpenSci package after the review process has finished.

### Improving this Guide

The pyOpenSci guidebook is a living document hosted on GitHub. If you have suggestions for improvements, create an issue or submit a pull request on [GitHub](#).

Have a set of  
common standards.  
Start small, steal  
most of them

[devguide.ropensci.org](https://devguide.ropensci.org)

[www.pyopensci.org/dev\\_guide](https://www.pyopensci.org/dev_guide)



- **Are functions as simple as possible?** We insisted in our submission guidelines that functions be short, but this is really a simple proxy for our true goal: to ensure that functions are modular. A modular function is one that does exactly one thing, and does it well - by insisting on breaking code up into the simplest possible functions, we make them easy to test, easy to understand, and easy to review. Can any of the functions in this new submission be broken up into simpler steps?
- **Is the code efficient?** If some really slow operation is performed in the code, it should be performed as infrequently as possible. For example, diagonalizing a matrix can be a slow computation. If the new code needs to do a slow operation like this, it should do it once only, and store the result for use, rather than recomputing it every time it needs to use it. Not sure which operations are slow? *Assume they all are!* Don't repeat any operation unnecessarily - we call this *well factored* code.
- **Is the usage of each function clear?** Try and put yourself in the mindset of someone coming to this code for the first time; how would they know how to re-use the existing code? The documentation for each function should have, at a minimum, a short description of the function, and a specification of its inputs and outputs.
- **Have edge cases been considered?** Users will invariably do all kinds of things with your code that they aren't supposed to. If the user does something unexpected (atypical inputs, clicking on things they're not supposed to...), will this new code break, or will it roll with the punches? For example, if a function takes a numerical input, is there a situation where division by 0 could be triggered?

Thanks Mozilla Science! →  
[mozillascience.github.io/codeReview/intro.html](https://mozillascience.github.io/codeReview/intro.html)



# Within `et.data.get_data()`, consider using Python's logging instead of `print()` #434

 Open

sgillies opened this issue 7 days ago · 8 comments



sgillies commented 7 days ago

Contributor



Context: [pyOpenSci/software-review#3](#)

When `get_data()` is called, text is printed to stdout.

```
>>> data_path = et.data.get_data("vignette-landsat")
Downloading from https://ndownloader.figshare.com/files/15197339
Extracted output to /Users/seang/earth-analytics/data/vignette-landsat/.
```

On the one hand, some feedback to users is a good thing. On the other hand, library functions should avoid printing to stdout and should use Python's logging facility instead so that users have more control over the output.



JOSS latest

Search docs

AUTHOR AND REVIEWER GUIDES

Submitting a paper to JOSS

Reviewing for JOSS

Review criteria

Review checklist

- Conflict of interest
- Code of Conduct
- General checks
- Functionality
- Documentation

### General checks

- **Repository:** Is the source code for this software available at the [repository url](#)?
- **License:** Does the repository contain a plain-text LICENSE file with the content [approved](#) software license?
- **Version:** Does the release version given match the GitHub release (v0.8)?
- **Authorship:** Has the submitting author made major contributions to the software? Do the list of paper authors seem appropriate and complete?

### Functionality

- **Installation:** Does installation proceed as outlined in the documentation?
- **Functionality:** Have the functional claims of the software been confirmed?
- **Performance:** If there are any performance claims of the software, have they been confirmed? (If there are no claims, please check off this item.)

### Documentation



- Installation
- Documentation
- Examples
- Tutorials
- Contributing

## The Matplotlib Developers' Guide

**Release:** 3.1.1

**Date:** October 28, 2019

- **Contributing**
  - [Submitting a bug report](#)
  - [Retrieving and installing the latest version of the code](#)
  - [Contributing code](#)
  - [Other ways to contribute](#)
  - [Coding guidelines](#)
- **Developer's tips for testing**
  - [Requirements](#)
  - [Running the tests](#)

### 1 Organize your document logically

- 2 Follow standard naming conventions
- 3 Use descriptive naming
- 4 Write simple, readable code
- 5 Code systematically
- 6 Show statistical variation / uncertainty
- 7 Always use relative paths
- 8 Version and archive documents
- 9 Prepare presentation-

## Package Guidelines

This page gives details concerning guiding principles and formatting required for *Bioconductor* packages. See also [Package Submission](#) for an overview of the submission process and what is expected as a *Bioconductor* maintainer.

- [Introduction](#)
- [General Package Development](#)
  - [Versions of R and Bioconductor](#)
  - [Correctness, Space, and Time](#)
  - [R CMD check environment](#)
- [DESCRIPTION](#)
- [NAMESPACE](#)
- [NEWS](#)
- [CITATION](#)
- [Including Data](#)
- [Package Documentation](#)
- [Vignettes](#)
- ['man' pages](#)
- [Unit Tests](#)
- [R Code](#)
- [C/Fortran Code](#)
- [.gitignore](#)
- [Conclusion](#)



## Golden Rules for Reproducible Statistical Analyses

Amy Gimma and Thibaut Jombart  
Monday 21 October 2019



### 1 Organize your document logically

- [1.1 Outline](#)
- [1.2 Data preparation](#)
- [1.3 Data analysis](#)
- [1.4 Export outputs](#)
- [1.5 System information](#)



# Automate, Automate!

- functionality
- implementation
- style + readability
- interface
- tests
- documentation

Parts of all these  
can be checked  
automatically

Let humans focus  
on what humans  
are best at

# Automate, Automate!

>R CMD check/BiocCheck #repository standards

>testthat::test\_package() #functionality

>covr::package\_coverage() #testing completeness

>devtools::spell\_check() #documentation

>lintr::lint\_package() #code style

>goodpractice::gp() #antipatterns/complexity

## — goodpractice::gp() report —

---

It is good practice to

write unit tests for all functions, and all package code in general. 86% of code lines are covered by test cases.

R/ccex.r:583:NA

... and 16 more lines

omit "Date" in DESCRIPTION. It is not required and it gets invalid quite often. A build date will be added to the package when you perform `R CMD build` on it.

use '<-' for assignment instead of '='. '<-' is the standard, and R users and developers are used it and it is easier to read your code for them if you use '<-'.

R/ccex.r:61:15

... and 128 more lines

avoid long code lines, it is bad for readability. Also, many people prefer editor windows that are about 80 characters wide. Try make your lines shorter than 80 characters

R/ccex.r:240:1

... and 2 more lines

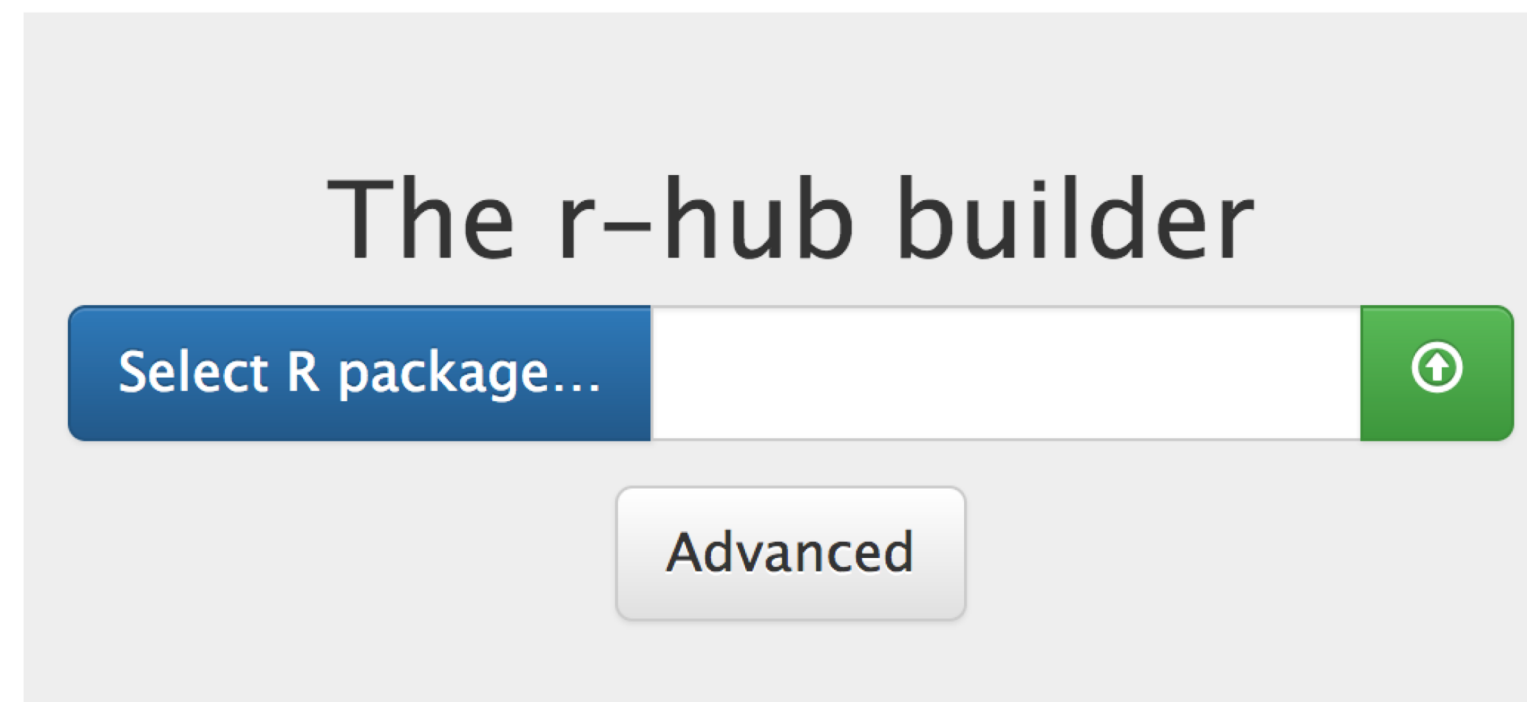
avoid 1:length(...), 1:nrow(...), 1:ncol(...), 1:NROW(...), 1:NCOL(...) expressions. They are error prone and result 1:0 if the expression on the right hand side is zero. Use seq\_len() or seq\_along() instead.

R/ccex.r:283:12

... and 3 more lines



# Automate, Automate!



```
>devtools::use_travis() #Linux + MacOS
```

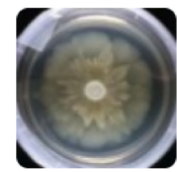
```
>devtools::use_appveyor() #Windows
```

```
>rhub::check() #Linux + Windows
```

# Run the Developer's Workflow

A screenshot of a GitHub comment. The comment is from user 'zkamvar' and is dated 'Sep 2'. The comment text reads: 'Walk through the [package vignette](#)  
The code from the vignette did not work out of the box for the following issues:  
1. mindate/maxdate needed the `_taken` suffix  
2. a comma was needed after `tags = "lake"`  
3. no one took any pictures of a lake with the text "mountain" on New Year's 2019  
4. I could not test the location services because they were down for flickr  
5. the function `related_terms()` was renamed to `related_tags()`'

# Break the Developer's Workflow with Your Own



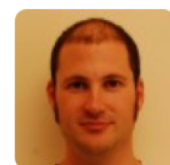
zkamvar commented on Sep 2

I wanted to see if I could show my spouse that yes, snakes did exist in the UK despite the fact that we haven't seen one while hiking in the last year and was able to produce a map of all photos tagged with "snake" in the last year.

When I tried to color the points by number of comments, I got a categorical variable instead of a continuous because all the columns were encoded as character. I found that this affected the SF object because it could not interpret the character coordinates as a bounding box for the print method:

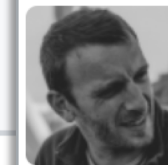
```
Simple feature collection with 148 features and 55 fields
```

```
geomet  
dimensi  
bbox:  
non-n
```



noamross commented on Oct 16, 2015

- When loading in a large corpus of documents, I got the error `Error: n not less than length(words)`, which was ultimately traced to `assert_that(n < length(words))` in `tokenize_ngrams()`. I found this was because I had a very short document, but also because the document was all whitespace characters. I wonder if it would make sense to check for empty documents.



davidgohel commented on Feb 20, 2018 • edited ▾

- ✓ **Performance:** Any performance claims of the software been confirmed.

*davidgohel:* I tested with 1204 documents (my admin folder). I got 20.354 sec. with 8 threads and 36.376 with 1 threads. This is really good (it tooks 20 second to get my whole admin directory scanned and returned as an R object).

I was expecting a faster execution with 8 threads (not 8 times faster but at least twice minimum). I used the profiling tools of RStudio and could see that the function spent 1/3 of time executing `.rtika_readFile()` and 1/3 executing `enc2utf8()`. I think that starting JVM with arg `-Dfile.encoding=UTF-8` instead of using `enc2utf8` could be a solution (not tested here, but was the solution in past experiences...).

Member





# Mapping the Source

Organize reading  
the source by

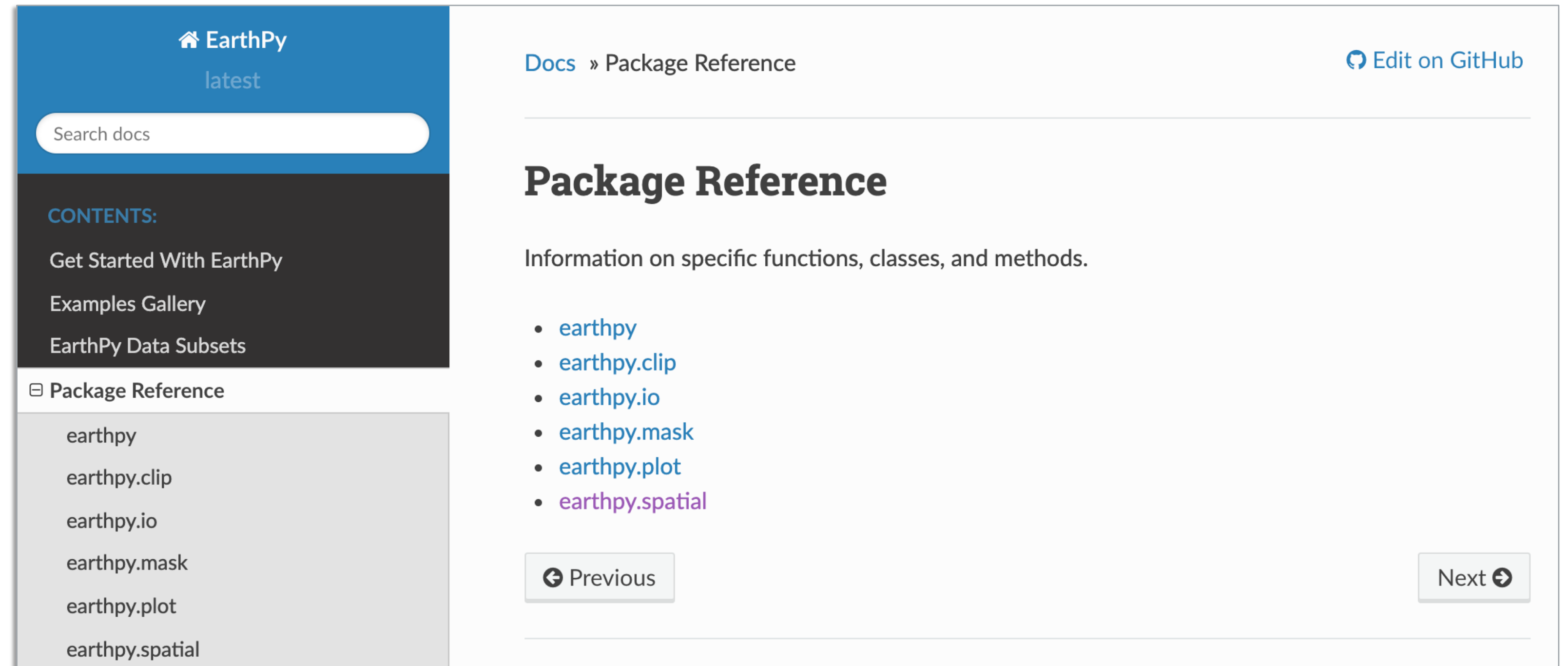
Files?

Docs?

Functions?

Objects?

Tests?



The screenshot displays the EarthPy documentation website. The top navigation bar is blue with the EarthPy logo and 'latest' version indicator. A search bar is present. The left sidebar contains a 'CONTENTS' section with links to 'Get Started With EarthPy', 'Examples Gallery', and 'EarthPy Data Subsets'. Below this is a 'Package Reference' section with a list of sub-packages: earthpy, earthpy.clip, earthpy.io, earthpy.mask, earthpy.plot, and earthpy.spatial. The main content area shows the 'Package Reference' page for 'earthpy.spatial', with a breadcrumb trail 'Docs » Package Reference' and an 'Edit on GitHub' link. The page title is 'Package Reference' and the subtitle is 'Information on specific functions, classes, and methods.' A list of sub-packages is shown, with 'earthpy.spatial' highlighted in purple. Navigation buttons for 'Previous' and 'Next' are located at the bottom of the page.

EarthPy  
latest

Search docs

CONTENTS:

- Get Started With EarthPy
- Examples Gallery
- EarthPy Data Subsets

Package Reference

- earthpy
- earthpy.clip
- earthpy.io
- earthpy.mask
- earthpy.plot
- earthpy.spatial

Docs » Package Reference [Edit on GitHub](#)

## Package Reference

Information on specific functions, classes, and methods.

- earthpy
- earthpy.clip
- earthpy.io
- earthpy.mask
- earthpy.plot
- earthpy.spatial

Previous Next

# Mapping the Source

```
covr::report()
```

arkdb coverage - 83.65%

Files	R/bulk_importers.R					
File	Lines	Relevant	Covered	Missed	Hits / Line	Coverage
<a href="#">R/assert.R</a>	127	59	26	33	4	44.07%
<a href="#">R/bulk_importers.R</a>	93	42	30	12	7	71.43%
<a href="#">R/ark.R</a>	238	85	79	6	13	92.94%
<a href="#">R/streamable_table.R</a>	184	65	61	4	12	93.85%
<a href="#">R/unark.R</a>	243	92	87	5	18	94.57%

# Mapping the Source

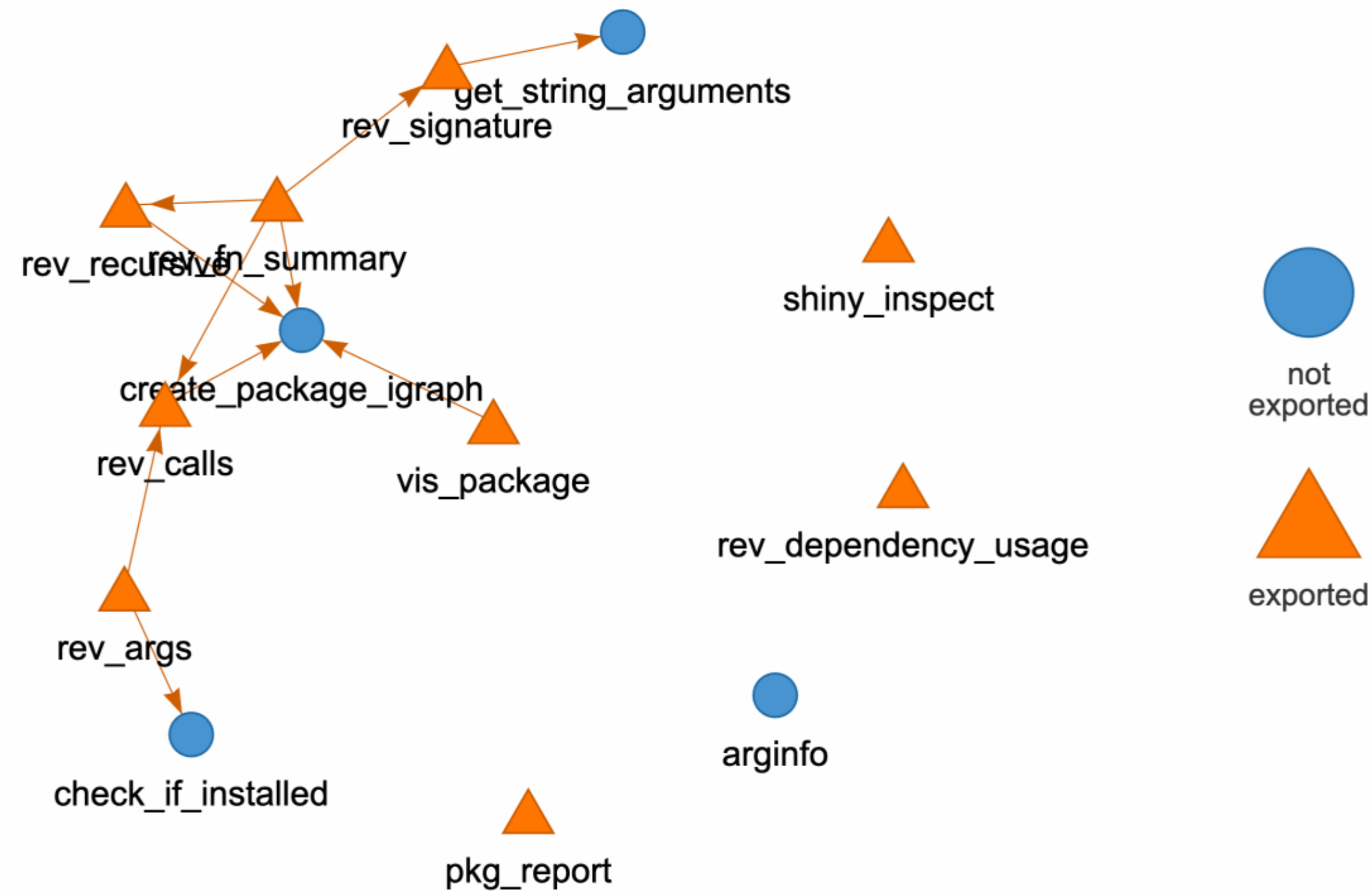
`cloc::cloc()`

language	file_count	file_count_pct	loc	loc_pct	blank_lines	blank_line_pct	comment_lines	comment_line_pct
:-----	-----	-----	-----	-----	-----	-----	-----	-----
HTML	26	0.2096774	3906	0.3859302	990	0.6269791	275	0.2160251
XML	18	0.1451613	2199	0.2172710	0	0.0000000	0	0.0000000
YAML	13	0.1048387	1509	0.1490959	25	0.0158328	7	0.0054988
R	22	0.1774194	1276	0.1260745	153	0.0968968	443	0.3479969
CSS	3	0.0241935	438	0.0432764	79	0.0500317	42	0.0329929
Markdown	9	0.0725806	375	0.0370517	113	0.0715643	0	0.0000000
Lua	7	0.0564516	213	0.0210454	22	0.0139329	15	0.0117832
JavaScript	2	0.0161290	144	0.0142278	40	0.0253325	16	0.0125687
Rmd	13	0.1048387	50	0.0049402	157	0.0994300	475	0.3731343
JSON	11	0.0887097	11	0.0010868	0	0.0000000	0	0.0000000

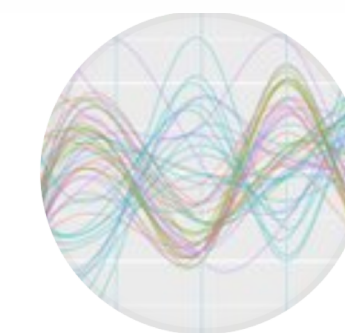


# Mapping the Source

Select by id ▾



[https://rpubs.com/jtr13/vis\\_package](https://rpubs.com/jtr13/vis_package)



**Joyce Robbins**  
@jtrnyc

# Following Your Map

Read the source! Run things as needed.

Keep your standards guide handy.

List big ideas and little notes separately.

Keep track of the good as well as bad!

Look for patterns and analogues.

# Writing it Up

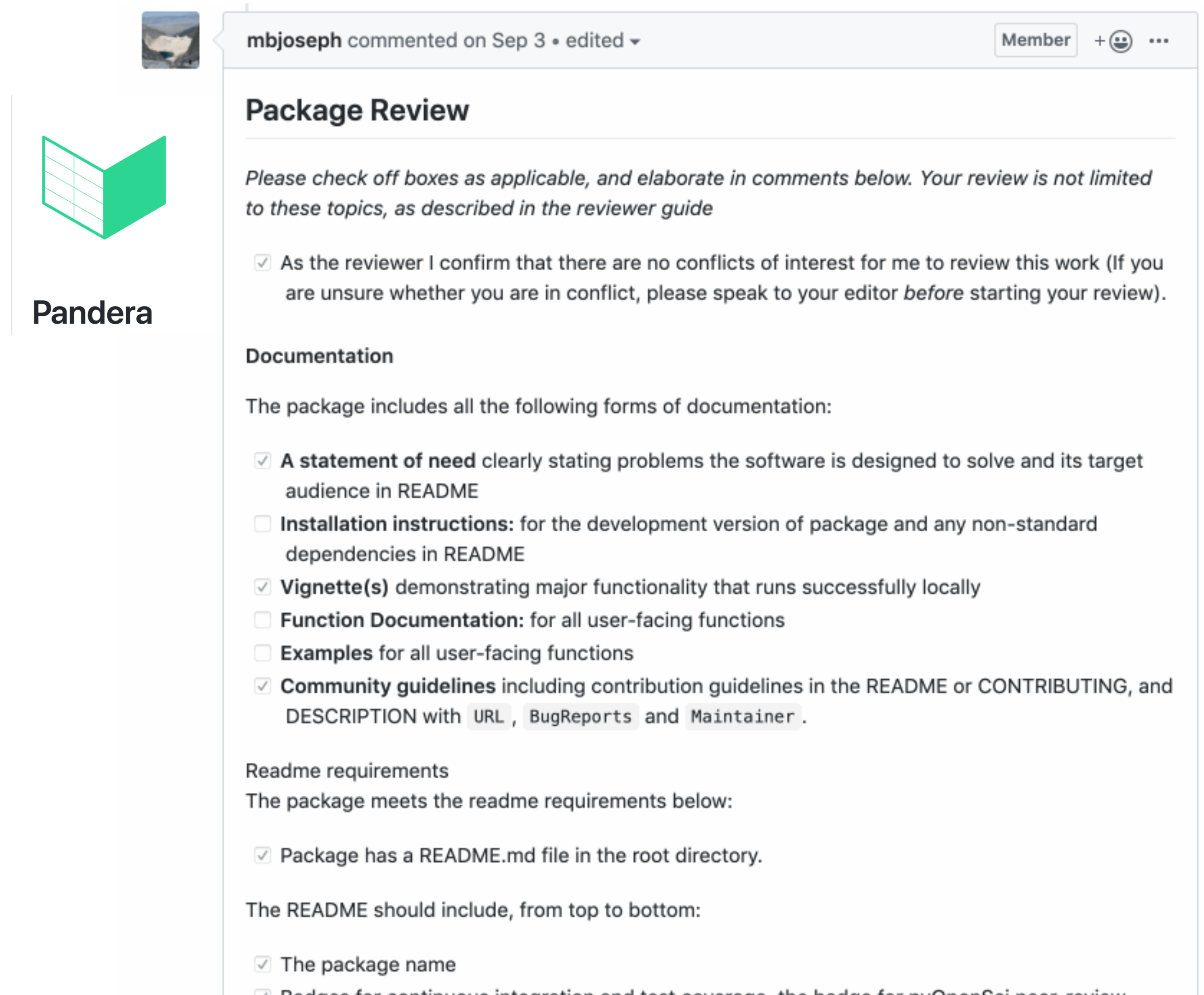
Set the context.

Go from big ideas to small.

Highlight the good parts as well as areas to improve.

Don't worry that you don't cover everything— cover your expertise.

**Be Nice.**



The screenshot shows a GitHub comment by user 'mbjoseph' on September 3rd, titled 'Package Review'. The review is for the 'Pandera' package, which is represented by a green logo of a grid and a chevron. The review text includes instructions for reviewers to check off boxes and elaborate in comments. The review is organized into sections: 'Documentation' and 'Readme requirements'. Under 'Documentation', there are several checklist items, some of which are checked. Under 'Readme requirements', there is one checked item. The review is marked as 'Member' and includes a '+😊' icon and a '...' menu icon.

mbjoseph commented on Sep 3 • edited ▾ Member +😊 ...

### Package Review

Please check off boxes as applicable, and elaborate in comments below. Your review is not limited to these topics, as described in the reviewer guide

- As the reviewer I confirm that there are no conflicts of interest for me to review this work (If you are unsure whether you are in conflict, please speak to your editor *before* starting your review).

#### Documentation

The package includes all the following forms of documentation:

- A statement of need** clearly stating problems the software is designed to solve and its target audience in README
- Installation instructions:** for the development version of package and any non-standard dependencies in README
- Vignette(s)** demonstrating major functionality that runs successfully locally
- Function Documentation:** for all user-facing functions
- Examples** for all user-facing functions
- Community guidelines** including contribution guidelines in the README or CONTRIBUTING, and DESCRIPTION with `URL`, `BugReports` and `Maintainer`.

#### Readme requirements

The package meets the readme requirements below:

- Package has a README.md file in the root directory.

The README should include, from top to bottom:

- The package name
- Badges for continuous integration and test coverage, the badge for pyOpenSci peer review



As **developers** **scientists** **analytics team** we want **good training** **quality code** **strong community** so we can do *awesome work*



# Thanks!

noamross @



Join us at:

[ropensci.org/software-review](https://ropensci.org/software-review)

[pyopensci.org/](https://pyopensci.org/)

Maëlle Salmon



Scott Chamberlain



Lincoln Mullen



Karthik Ram



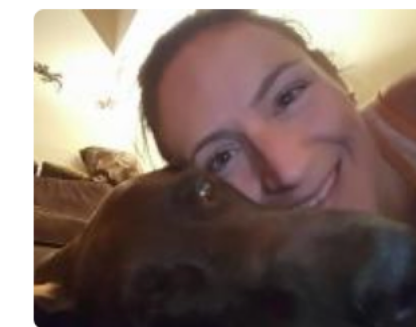
Anna Krystalli



Melina Vidoni



Brooke Anderson



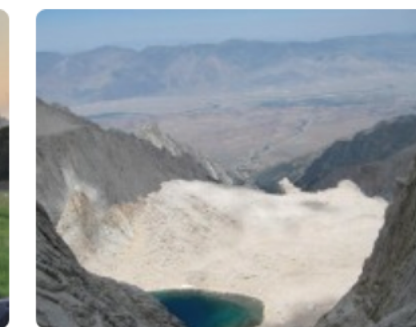
[Leah Wasser](#)

Earth Lab, University of Colorado - Boulder



[Chris Holdgraf](#)

Berkeley Bids, Project Jupyter, Binder



[Max Joseph](#)

Earth Lab, University of Colorado - Boulder



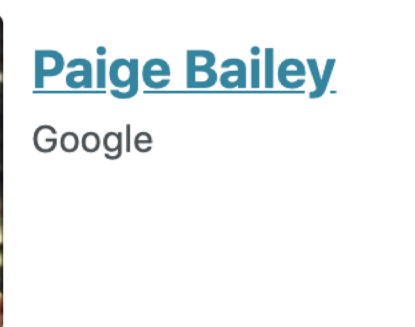
[Ivan Ogasawara](#)

SciPy Latin America, Ibis-framework



[Luiz Irber](#)

DIB Lab -- UC Davis



[Paige Bailey](#)

Google



[Filipe Fernandes](#)

IOOS, Conda Forge



[Jenny Palomino](#)

Earth Lab, University of Colorado - Boulder