A LIST OF QUESTIONS

For transparency and replicability, we list all evaluated questions of the survey including their exact phrasing. We exclude a small number of questions about power structures, community health, and motivation that we have not used in this paper.

Part I: Ecosystem.

- Please choose ONE software ecosystem* in which you publish a package**. If you don't publish any packages, then pick an ecosystem whose packages you use.
  * "Software ecosystem" = a community of people using and developing packages that can depend on each other, using some shared language or platform
  ** "Package": A distributable, separately maintained unit of software. Some ecosystems have other names for them, such as "libraries", "modules", "crates", "cocoapods", "rocks" or "goodies", but we'll use "package" for consistency.
  [selection or textfield, substituted for <ecosystem> in remainder of survey]

Ecosystem Role.

- Check the statement that best describes your role in this ecosystem.
  - I'm a founder or core contributor to <ecosystem> (i.e. its language, platform, or repository).
  - I'm a lead maintainer of a commonly-used package in <ecosystem>.
  - I'm a lead maintainer of at least one package in <ecosystem>.
  - I have commit access to at least one package in <ecosystem>.
  - I have submitted a patch or pull request to a package in <ecosystem>.
  - I have used packages from <ecosystem> for code or scripts I've written.
- About how many years have you been using <ecosystem> in any way?
  - < 1 year
  - 1 - 2 years
  - 2 - 5 years
  - 5 - 10 years
  - 10 - 20 years
  - > 20 years

Ecosystem values.

- "How important do you think the following values are to the <ecosystem> community? (Not to you personally; we'll ask that separately.)" — see Section 3.1 for the 11 value questions; **results shown in Figure 1**.
- How confident are you in your ratings of the values of <ecosystem> above?
  - Not confident
  - Slightly confident
  - Confident
  - Very confident
- "Is there some other value the <ecosystem> community emphasizes that was not asked above? If so, describe it here:"

Part II: Package.

- In the following we are going to ask about your experience working on one particular package. Please think of one package in <ecosystem> you have contributed to recently and are most familiar with. If you haven't contributed to a package in <ecosystem>, then name some software you've written that relies on packages in <ecosystem> packages. You may use a pseudonym for it if you are concerned about keeping your responses anonymous. — [text fields, substituted for <package> in remainder of survey]
- Do you submit the package you chose to a/the repository associated with <ecosystem>? (Choose "no" if the ecosystem does not have its own central repository.) — [yes/no]
- Is there any software maintained by other people that depends on the package you chose? — [yes/no]
- Is the package you chose installed by default as part of a standard basic set of packages or platform tools? — [yes/no]
- "How important are each of these values in development of <package> to you personally?" — see Section 3.1 for the 11 value questions; **results shown in Appendix B**.
- (OPTIONAL) Is there some other value important to you personally for <package> which was not mentioned? — [text fields]
- How often do you face breaking changes from any upstream dependencies (that require rework in <package>)? — **results shown in Figure 3a**
  - Never
  - Less than once a year
  - Several times a year
  - Several times a month
  - Several times a week
  - Several times a day
- How often do you make breaking changes to <package>? (i.e. changes that might require end-users or downstream packages to change their code) — [frequency scale as above] **results shown in Figure 2a**

Making changes to <package>.

- I feel constrained not to make too many changes to <package> because of
- potential impact on users. — **results shown in Figure 2b**
  - Strongly agree
  - Somewhat agree
  - Neither agree nor disagree
  - Somewhat disagree
  - Strongly disagree
  - I don't know
- I know what changes users of <package> want. — [agreement+don't know scale as above]
- If I have multiple breaking changes to make to <package>, I try to batch them up into a single release. — [agreement+don't know scale as above] **results shown in Figure 2d**
- I release <package> on a fixed schedule, which <package> users are aware of. — [agreement+don't know scale as above] **results shown in Figure 2j**

- Releases of <package> are coordinated or synchronized with releases of packages by other authors. — [agreement+don't know scale as above] **results shown in Figure 2i**
- When working on <package>, I make technical compromises to maintain backward compatibility for users. — [agreement+don't know scale as above] **results shown in Figure 2c**
- When working on <package>, I often spend extra time working on extra code aimed at backward compatibility. (e.g. maintaining deprecated or outdated methods) — [agreement+don't know scale as above]
- When working on <package>, I spend extra time backporting changes, i.e. making similar fixes to prior releases of the code, for backward compatibility. — [agreement+don't know scale as above]

Releasing Packages.

- A large part of the community releases updates/revisions to packages together at the same time. — [agreement+don't know scale as above]
- A package has to a meet strict standards to be accepted into the repository. — [agreement+don't know scale as above] **results shown in Figure 2k**
- Most packages in <ecosystem> will sometimes have small updates without changing the version number at all. — [agreement+don't know scale as above]
- Most packages in <ecosystem> with version greater than 1.0.0 increment the leftmost digit of the version number if the change might break downstream code. — [agreement+don't know scale as above]
- I sometimes release small updates of <package> to users without changing the version number at all. — [agreement scale, without 'don't know'] **results shown in Figure 2g**
- For my packages whose version is greater than 1.0.0, I always increment the leftmost digit if a change might break downstream code (semantic versioning). — [agreement as above] **results shown in Figure 2f**
- When making a change to <package>, I usually write up an explanation of what changed and why (a change log). — [agreement as above] **results shown in Figure 2e**
- When working on <package>, I usually communicate with users before performing a change, to get feedback or alert them to the upcoming change. — [agreement as above] **results shown in Figure 2h**
- When making a breaking change on <package>, I usually create a migration guide to explain how to upgrade. — [agreement as above]
- After making a breaking change to <package>, I usually assist one or more users individually to upgrade. (e.g. reaching out to affected users, submitting patches/pull requests, offering help) — [agreement as above]

Part IV: Dependencies.

- In the last 6 months I have participated in discussions, or made bug/feature requests, or worked on development of another package in <ecosystem> that one of my packages depends on. — [yes/no]
- Have you contributed code to an upstream dependency of one of your packages in the last 6 months (one where you're not the primary developer)? — [yes/no]
- About how often do you communicate with developers of packages you depend on (e.g. participating in mailing lists, conferences, twitter conversations, filing bug reports or feature requests, etc.)? — [frequency scale, as above] **results shown in Figure 3f**

For most dependencies that my packages rely on, the way I typically become aware of a change to the dependency that might break my package is:

- I read about it in the dependency project's internal media (e.g. dev mailing lists, not general public announcements) — [agreement scale, as above]
- I read about it in the dependency project's external media (e.g. a general announcement list, blog, twitter, etc) — [agreement scale, as above]
- A developer typically contacts me personally to bring the change to my attention — [agreement scale, as above] **results shown in Figure 3e**
- Typically I get a notification from a tool when a new version of the dependency is likely to break my package — [agreement scale, as above] **results shown in Figure 3f**
- Typically, I find out that a dependency changed because something breaks when I try to build my package. — [agreement scale, as above] **results shown in Figure 3g**
- How do you typically declare the version numbers of packages that <package> depends — **results shown in Figure 3i**
  - I specify an exact version number
  - I specify a range of version numbers, e.g. 3.x.x, or [2.1 through 2.4]
  - I specify just a package name and always get the newest version
  - I specify a range or just the name, but I take a snapshot of dependencies (e.g. shrinkwrap, packrat)
- What is the common practice in <ecosystem> for declaring version numbers of dependencies? — [same scale as previous + "don't know"]

Using or avoiding dependencies.

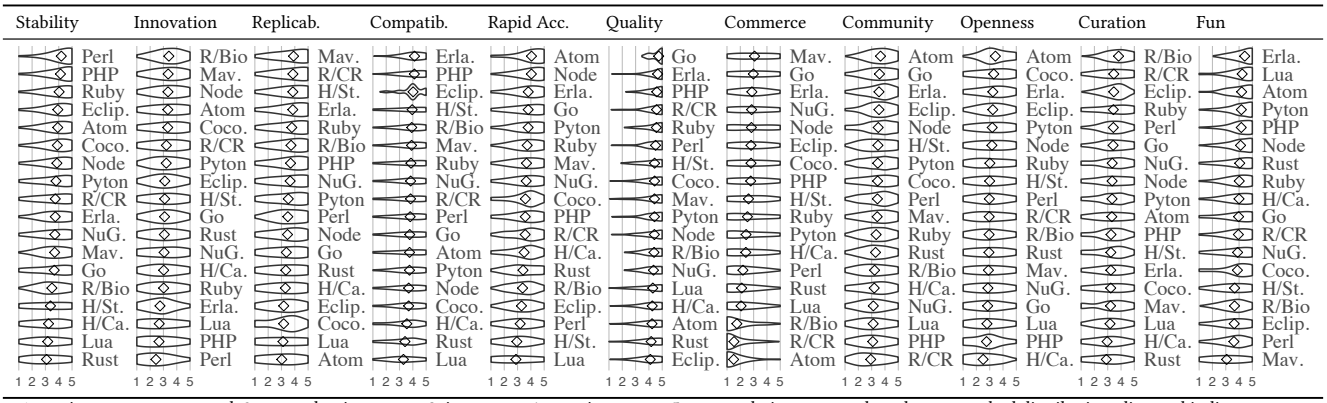
- When adding a dependency to <package>, I usually do significant research to assess the quality of the package or its maintainers, before relying on a package that seems to provide the functionality I need. — [agreement scale, as above] **results shown in Figure 3d**
- It's only worth adding a dependency if it adds a substantial amount of value. — [agreement scale, as above] **results shown in Figure 3c**
- I often choose NOT to update <package> to use the latest version of its dependencies. — [agreement scale, as above] **results shown in Figure 3h**
- When adding a dependency, I usually create an abstraction layer (i.e., facade, wrapper, shim) to protect internals of my code from changes. — [agreement scale, as above]
- When working on <package>, I often copy or rewrite segments of code from other packages into my package, to

avoid creating a new dependency. — [agreement scale, as above]
- When working on <package>, I must expend substantial effort to find versions of all my dependencies that will work together. — [agreement scale, as above]
- (OPTIONAL) Compare <ecosystem> with other ecosystems you've used or heard about – does one have some features that the other should adopt? If so, name the other ecosystem(s) and describe the feature(s). — [text field]
- (OPTIONAL) Why do you think people chose to design these other ecosystem(s) differently from <ecosystem>? — [text field]

Part V: Demographics and motivations.
- Age
  - 18-24
  - 25-34
  - 35-44
  - 45-54
  - 55-64
  - 65+
- Gender — [male/female/other]
- Formal computer science education/training
  - None
  - Coursework
  - Degree
- How many years have you been contributing to open source? (in any way, including writing code, documentation, engaging in discussions, etc) — [same time scale as "years used ecosystem" above]
- How many years have you been developing or maintaining software? — [same as previous]
- (OPTIONAL) Is there anything else we should have asked, that would help us better understand your experience with community values and breaking changes in <ecosystem> If so, tell us about it: — [text field]

B PLOTS OF ADDITIONAL RESULTS

Figure 4 plots personal values for each ecosystem.



1: not important or opposed, 2: somewhat important, 3: important, 4: very important, 5: extremely important; plots show smoothed distribution; diamond indicates mean

Figure 4: Personal values.