Anonymized e-mail interviews with R package maintainers active on *CRAN* and *GitHub*

Tom Mens
Software Engineering Lab, COMPLEXYS Research Institute
University of Mons, Belgium

***Abstract*—This document reproduces an anonymised and sanitised version of interviews that have been carried out with R package maintainers over e-mail. The goal was to get a better understanding in how R package maintainers develop and distribute their packages through *GitHub* and *CRAN*.**

All four interviewed developers were actively maintaining packages on *GitHub*, some were also active on *CRAN*. In order to have a representative and unbiased selection of R package maintainers, they were selected based on their profile (the number of R packages they maintain on *GitHub* and/or *CRAN*) as well as their gender (two interviewees were male, two were female).

I. Interview of first R package maintainer

At the time of the interview, this package maintainer had all his or her R packages only available on *GitHub*, and none of them on *CRAN*.

- *Why did you choose for* GitHub *as a development platform for your R packages?*
  Having been turned off by the inflexibilities and shortcomings of other version control systems like SVN, TortoiseSVN and the like, I checked out *Git* about 3-4 years ago and really came to love it. Then two things happened that motivated moving over to *GitHub*:
  1) I came across some work of [***ANONYMISED***], we started "talking" and basically he said: "why don't you move your blog-post stuff over to *GitHub*, that way everyone can contribute and has an easy to use single point of reference. More and more people in the R universe are doing it, so come along".
  2) The more I used *Git*, the more I felt I needed better support for managing my TODO list of things I would like to do in my code. So I checked out *GitHub*'s issue mechanism and since then been totally hooked.
- *Are there any specific reasons (technical or others) why your packages are only made available through* GitHub *and not through* CRAN*?*
  These two aspects are probably the most compelling ones for me:
  1) *GitHub* is **easy** to understand and use, *CRAN* is not. I'd call myself a more or less experienced R developer, still all those R CMD things and the entire mechanism of publishing of *CRAN* is somewhat intimidating for me. To me *CRAN* represents the "old" way/style of doing stuff on the internet and in coding, while *GitHub*, travisCI, CodeCov, StackOverflow etc. are **superb** examples of how infrastructure technology can actually enable people to do the things they actually want to do (produce awesome code/packages) instead of spending weeks configuring things in order to get them to work. What adds to this intimidation is the fact that you (or at least I) think "oh man, my package has to be REALLY good in order to be on *CRAN*; there are so many expert packages out there, you're gonna be laughed at if it's not perfect". That might just be me, but I would guess that other developers feel a similar purely psychological or social barrier to using anything related to *CRAN* or the people behind it. I'd even argue that most of these perceived barriers are causef by the general tone on the mailing list (some examples: http://bit.ly/1koPQax, http://bit.ly/1MEum6l, http://bit.ly/1Wwmq6i). Compare that to the tone on http://stackoverflow.com/questions/tagged/r, it's like day and night.
  2) *GitHub* makes it so easy to reach out to other programmers for ideas, feature requests, bugs in a **structured** yet very intuitive and fun way. On *CRAN*, you're bound to stick to e-mailing package authors and/or raising attention through those tried-and-true, but IMO somewhat outdated mailing lists in the R universe. Writing code in e-mails isn't fun, it's easy to lose track of things, it's simply not the right mechanism of collaborating on code and ideas involving code.
     By structured I mean things like:
     - the issue tracking mechanism with all its people- and code-linking capabilities and its flexible tagging system
     - being able to fork an entire repo to play around with it and then issuing a pull request to the author if you think you improved something
     - being able to reference specific "states of the world" (commits, releases etc.)
- *Which of* GitHub*'s software development facilities do you rely on (e.g. version control, merging, continuous*

integration, issue tracking, and so on)?
  - version control
  - branching (I like this model very much: http://nvie.com/posts/a-successful-git-branching-model/)
  - continuous integration via travisCI
  - issue tracking
  - forking
  - pull requests
- *Do you also use* GitHub *as a platform for distributing your R packages?*
  Yes.
- *Why and how?*
  As I could never really bring up the courage and time to check out how things actually need to work on a technical level in order to publish on *CRAN*, I simply use *GitHub* and having a lot of fun with it. Currently, almost only through the use of `devtools::install_github` as, for example, outlined in [***ANONYMIZED***]. But recently I've also experimented with drat (https://github.com/eddelbuettel/drat) and that looks promising. I'd probably always keep the `devtools::install_github` mechanism, but bring things more in line with R's repository philosophy by using something like drat. I also plan to look into https://github.com/RevolutionAnalytics/checkpoint and try to bring everything together in a way that fits me.
- *What are the **advantages** that you have perceived in using* GitHub *for developing, distributing and installing R packages?*
  - social interaction with coders
  - conversations are "spot-on" as it's easy to illustrate ideas with actual code (including syntax highlighting, a HUGE enabler IMO) or link to actual code
  - keeping track of things to do via the issue management system
  - quick and easy forking an re-integration via pull requests
  - in combination with devtools, things simply work, it's even fun
- *What are the **disadvantages** that you have perceived in using* GitHub *for developing, distributing and installing R packages?*
  Sometimes, it's a bit hard to mentally keep track of who's the originator of a repository and who is just forking a repository. E.g., it always takes me about 5 minutes to recall who is doing what with regard to roxygen/roxygen2: if you search for "roxygen github" you end up here https://github.com/klutometis/roxygen. If you search for "roxygen2 github", you end up here https://github.com/yihui/roxygen2
  It'd be nice to be able to sort repositories (manually or alphabetically) and/or to tag one's own "favorite repos". At times it seems that the main user page is mainly designed for **others**, not for yourself (which is okay, though). For example, [***ANONYMIZED***].
  I totally agree with Dirk's main point in http://dirk.eddelbuettel.com/code/drat.html (Section "Drat Use Case 1: GitHub"): `devtools::install_github` is good, but we should stick to R's repo logic. I'd be cool if something like this would already be supported out-of-the box by *GitHub*.
  I never had that experience, but I would imagine that as much as I'm turned off by *CRAN*, some people might be by having to use *GitHub* in order to try out stuff I wrote.
- *When developing R packages, have you encountered any specific problems related to managing package dependencies?*
  Yes. Whenever I write code, I entertain the idea that this code should be as fit for productive usage as possible. Especially with respect to package dependencies, the risk of things breaking at some point due to the fact that a version of a dependency has changed without you knowing about it is immense. That actually cost us weeks and months in a couple of professional projects I was part of. While it's rather a philosophical than a technical question how package dependencies are/should be handled in R, I personally think it's REALLY relevant to at least be ABLE to be very specific and rigid with regard to your dependencies. And I think the R universe could provide better tools to fit the needs of developers and professionals out there in a better way. But in that regard I like efforts such as https://github.com/rstudio/packrat and https://github.com/RevolutionAnalytics/checkpoint very very much. And I very much like the fact that more and more people seem to be aware that this is something one should manage systematically. Back in the days, I tried to raise attention for it on R-devel mailing list without any success [***ANONYMIZED***]. I guess that was actually the point where I decided that I liked StackOverflow much better and would stop using the mailing lists ;-))
  A better systematic for dependency management together with making your codebase more robust against changes in dependencies is the thing that I actually spend most of my time cracking my brain about as I feel the R universe could do a much better job in that respect. And I think it's still one of the number one reasons why people are hesitant to use R in productive rather than academic contexts while there is clearly a demand for using it productively (e.g. http://www.earl-conference.com/, http://www.r-bloggers.com/demo-r-in-sql-server-2016/, etc.).
  So I while there's clearly a purely technical aspect to the assessment of R's dependency management system, I also feel that, on R's side, there is great potential in learning from tried-and-true paradigms from the software engineering world. A lot of R developers do not consider themselves programmers as they usually picked up R at university studying something other than software engineering/informatics etc. (as is also true for myself: business administration and then applied statistics). So they typically never heard of things like the SOLID principles of OOP, design patterns, interfaces, base classes,

class inheritance etc. While that's totally fine, it does in fact really materialize when it comes to how well things fit together in such a highly distributed system as R's add-on package universe. And to that end, *GitHub* is also doing a superb job of "educating people without them even knowing it". Or put differently, by going out of the way of giving developers the opportunity to check out how **other** developers are approaching certain tasks and "how they are doing things" people learn from each other in a natural/non-intrusive way. And that's simply terrific! I personally learned **so much** by peeking into the *GitHub* projects of https://github.com/hadley, https://github.com/yihui, https://github.com/rstudio (https://github.com/jcheng5 and https://github.com/wch in particular). They inspired my to do a better job with regard to applying a better software design/architecture [***ANONYMIZED***]. Another milestone for my personal R dev skills was me being able to check out HOW exactly Joe Cheng implemented that awesome reactivity stuff in http://shiny.rstudio.com/ that he borrowed from https://www.meteor.com/. I vividly remember watching his interview from useR! 2014 (https://www.youtube.com/watch?v=uJm-its3ZWM) and not being able to hold back any longer with the desire to understand how this actually works. So I checked out the *GitHub* repo of shiny, learned from it and created my own little thing [***ANONYMIZED***]. While still being far from a full-fledged package that I'd comfortably point someone to for actual use in a project, it DEFINITELY brought my R game to higher level.
So there's this clear learning aspect that *GitHub* offers, as well. While source code is technically also available on *CRAN*, it takes much more effort to check out the actual code of another programmer - so no one is doing it. While there's arguably a higher code/package quality on *CRAN* than on *GitHub*, the platform/design keeps other developers from learning to become as good as the top stars on *CRAN*.

II. Interview of second R package maintainer

At the time of the interview, this package maintainer had all his or her R packages only available on *GitHub*, and none of them on *CRAN*.

- *Why did you choose for* GitHub *(instead of* CRAN*) for* ***developing*** *your R package(s)?*
  We chose *GitHub* for development in order to take advantage of the version control/branches/etc software development tools offered.
- *Do you also use* GitHub *as a platform for* ***distributing*** *your R packages?*
  We are also using *GitHub* as a platform for distribution of the package. For development of packages, *GitHub* is very useful, and *CRAN* is really not appropriate for development. For distribution, of course it is more of a pain for end users to have to get packages from anywhere other than *CRAN*.
- *What are the* ***advantages*** *and* ***disadvantages*** *of using* GitHub *instead of* CRAN *for developing, distributing and installing R packages?*
  There are a few reasons we [***ANONYMIZED***] have only distributed very few of our packages on *CRAN*:
  1) *CRAN* checks are not easy to pass. While we are in agreement with their requirements for vignettes and working examples, and we enforce these rules ourselves, there are many other small things that make it difficult to get a package posted to *CRAN*.
  2) Many of our packages are quite specialized and have a potentially very small group of users. The work of getting onto *CRAN* is not worth the reward in these cases.
  3) Every time you need to update the package on *CRAN* you have to deal with all their hoops again. We believe in frequent updates, so this is time-consuming.
  4) A package up on *CRAN* must be maintained to a high standard. This is fine, but we have many packages with little to no maintenance budget. It is not worth it for us to have to deal with updates for each small revision of R if the package is still working fine for our core users.

  That's the gist. I think that *CRAN* is a great resource, and it is certainly great to have a central clearing house of packages that have been curated to some degree. The curation, however, is not without cost. I would note that Hadley Wickham (ggplot, etc) now has packages that he has chosen not to upload to *CRAN*, for reasons similar to those I have cited above.
- *When developing/maintaining R packages, have you encountered any specific problems related to managing package dependencies?*
  Yes, it definitely is a problem sometimes. There have actually been a few times when I have rewritten a function in my own package because of that difficulty, especially with packages that themselves have many dependencies. Package creators, however, are getting more disciplined about only requiring packages that are essential, and using the ability to require a function rather than a package when possible. The biggest issue we have is multiple layers of dependencies, some of which are on *CRAN* and some of which are not. That can be difficult to keep in sync, but usually, if your package is not on *CRAN*, you can just keep it using the older dependency for a while until you have time to sort that issue.

III. Interview of third R package maintainer

At the time of the interview, this package maintainer had all his or her R packages available on *CRAN*, and several of these were also available on *GitHub*.

- *Do you use Github as a platform for* ***developing*** *R package(s)?*
  Yes.

- *Which of GitHub's software development facilities do you rely on (e.g. version control, merging, continuous integration, issue tracking, and so on)?*
  I've been using all of your examples, i.e. version control, branching, merging, Travis CI, issue tracking, also development in teams under an organizational account.
- *What are the reasons (technical or others) why some of your packages are only available through* GitHub *and not through* CRAN*?*
  I don't have any R packages that are on *GitHub* but not on *CRAN*. I do have some on *CRAN* that are not on *GitHub* but that's because I did not get to it yet. I plan to have all of my *CRAN* packages in *GitHub*.
- *Do you also use* GitHub *as a platform for* ***distributing*** *your R packages? Why and how?*
  I only do it if there is a an update of the package that users might need immediately, e.g. a bug fix. I just tell the users to get the updated version from *GitHub* and provide the link.
- *For those of your packages that are available on* CRAN *and* GitHub*, is there a difference between both (e.g. stable version versus development version)?*
  Yes. I usually have a "cran" branch which matches the *CRAN* version. A "master" branch (or one that I set as default) would be the current development version. I might have some experimental branches as well.
- *What are the* ***advantages*** *that you have perceived in using* GitHub *for developing, distributing and installing R packages?*
  - Version control.
  - Super easy branching (in comparison with e.g. Subversion). [The two above are more about Git than *GitHub*.]
  - Having the code available from anywhere (since I work at different days on different computers).
  - Sharing the code with co-workers and users.
  - Travis CI allows to check the code against different versions of R, so that I don't need to re-install R every time a new development version is released.
  - Issue tracker helps me not to forget things that I wanted to fix or enhance.
  - I'm sure I'm forgetting something, but overall I really love *GitHub*!
- *What are the* ***disadvantages*** *that you have perceived in using Github for developing, distributing and installing R packages?*
  I can't think of any that are specific to R packages. The only feature I was thinking would be nice is to be able to set specific branches as private/public, instead of having this setting to apply to the whole repository.
- *When maintaining R packages, have you encountered any specific problems related to managing package dependencies?*
  Nothing that would be related to *GitHub*. I had one case where my package heavily depended on another package and after a while that package was removed from *CRAN* and stopped being maintained. So I had to remove one of the main features of my package. Now I try to minimize dependencies on packages that are not maintained by "established" maintainers or by me ;-) Nowadays it has become a standard that PhD students in statistics develop an R package as a byproduct of their PhD thesis. But for students who leave academia or move to other projects it's often a low priority if at all to maintain their package.

IV. Interview of fourth R package maintainer

At the time of the interview, this package maintainer had all his or her R packages available on *GitHub*, and most of them (except for the experimental or immature ones) also on *CRAN*.

- *Which of* GitHub*'s software development facilities do you rely on for* ***developing*** *R packages (e.g., collaborative software development, software versioning, branching, merging, continuous integration, issue tracking, bug tracking, developer communication, automated testing, and so on)?*
  All of them.
- *What are the reasons (technical or others) why some of your packages are only available through* GitHub *and not through* CRAN*?*
  Generally, all my packages go to *CRAN* unless they are either experimental or still young.
- *Do you also use* GitHub *as a platform for* ***distributing*** *your R packages? Why and how?*
  Yes, but just for development versions and experimental packages. I tell people to use devtools::install_github().
- *For those of your packages that are available both on* CRAN *and* GitHub*, is there a difference between both (e.g. stable version versus development version)?*
  CRAN = release
  GitHub = development
- *What are the* ***advantages*** *and* ***disadvantages*** *that you have perceived in using* GitHub *for developing, distributing and installing R packages?*
  CRAN is used by far more people, but the process to get a package on to *CRAN* is more onerous. Some of the challenge is incidental (i.e. dealing with *CRAN* maintainers can be frustrating), but some of it is fundamental, as you must check that your package will work in a wide variety of situations (and that it doesn't break other packages without warning).
  GitHub is lighter weight so it's easier to distribute development versions, but code form *GitHub* also has less quality control.
- *When maintaining R packages, have you encountered any specific problems related to managing package dependencies?*
  It's a bit of a hassle when your package depends on other development versions, but there are changes in the latest version of devtools to make this easier.