On the Development and Distribution of R Packages: An Empirical Analysis of the R Ecosystem

Alexandre Decan Tom Mens Maelick Claes
Philippe Grosjean
COMPLEXYS Research Institute*
University of Mons, 7000 Mons, Belgium
{ firstname.lastname }@umons.ac.be

ABSTRACT

This paper explores the ecosystem of software packages for R, one of the most popular environments for statistical computing today. We empirically study how R packages are developed and distributed on different repositories: *CRAN*, *BioConductor*, *R-Forge* and *GitHub*. We also explore the role and size of each repository, the inter-repository dependencies, and how these repositories grow over time. With this analysis, we provide a deeper insight into the extent and the evolution of the R package ecosystem.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance and Enhancement—*Version Control*; D.3.3 [**Software Engineering**]: Programming Languages

Keywords

software ecosystem, software distribution, software development, package, R, software repository mining

1. INTRODUCTION

A wide range of environments for statistical computing have been proposed and used over the years. The most important ones are SAS, SPSS, R, Stata, Statistica, Minitab, Systat and JMP. While SAS and SPSS have been the market leaders in the past, R is catching up rapidly. According to Hornik [10], the number of R packages available in *CRAN*, the main and official distribution of R packages, is witnessing a slightly "sub-exponential" growth. Muenchen [12] counted the number of CRAN packages against each major release of R (until version 3.0.2), and observed a quadratic growth. He states: "To put this growth in perspective [...] During 2013 alone, R added more functions/procs than SAS Institute has written in its entire history!"

The R programming language also continues to increase in popularity. The Tiobe index for June 2015[1] ranked R as the 13th most popular programming language. The "Transparent Language Popularity Index"[2] ranked R as the 14th most popular programming language, all language categories confounded.

Because of this popularity of the R language and its software package ecosystem, it is important to study this ecosystem in more detail. In particular, we wish to get insight in the effect of other package repositories on *CRAN*.

Taking the point of view of a R package user, we focus on the following questions in this paper. Where and how are packages developed and distributed? How do packages depend on one another? Do we observe change over time related to these questions?

The R community has raised concerns about the way R packages are currently distributed, what problems current package management systems suffer from, and how they could be solved [13]. Among others, we wish to know whether the increasing popularity of *GitHub* as a host for developing many R packages has become a "game changer". Indeed, it is quite straightforward to develop and install packages directly from *GitHub*, possibly avoiding the need for having one's package distributed on *CRAN* or *BioConductor*.

The remainder of this paper is structured as follows. Section 2 discusses the R package repositories that will be considered as part of our analysis. Section 3 presents related work. Section 4 explains how we have extracted all data necessary for our analysis. Section 5 presents the results of our exploratory empirical analysis. Section 6 discusses the threats to validity of our research, Section 7 presents future work, and Section 8 concludes.

2. CONSIDERED R REPOSITORIES

The R ecosystem is based on software packages. The main R distribution installs by default a dozen or so base packages, and another dozen or so recommended packages (the exact number depends on the installed version of R). These packages will be excluded from the analysis in this paper. In addition to these core R packages, many additional pack-

*This research was carried out in the context of ARC research project AUWB-12/17-UMONS- 3 financed by the Ministère de la Communauté française - Direction générale de l'Enseignement non obligatoire et de la Recherche scientifique, Belgium.

ECSAW '15, September 07 - 11, 2015, Dubrovnik/Cavtat, Croatia

DOI: http://dx.doi.org/10.1145/2797433.2797476

[1] www.tiobe.com/index.php/content/paperinfo/tpci
[2] lang-index.sourceforge.net, July 2013 update.

Table 1: Characteristics of considered R package repositories

Repository name	URL	Number of packages [Date]	Role	Development status
CRAN	cran.r-project.org	6411 [19-03-2015]	Distribution only	Stable package releases only
BioConductor	bioconductor.org	997 [19-03-2015]	Distribution only	Stable package releases only
GitHub	github.com	5150 [17-02-2015]	Mainly development, rolling release distro (e.g., with devtools)	*Git* version control
R-Forge	r-forge.r-project.org	1883 [18-03-2015]	Mainly development, rolling release distro (e.g., with devtools)	*SVN* version control

ages are developed and distributed by different contributors through different repositories.

2.1 R package distributions

CRAN, the Comprehensive R Archive Network, can be considered as the official repository that contains the broadest collection of R packages. It aims at providing stable and high-quality packages compatible with the latest version of R. Quality is ensured by forcing package maintainers to follow a strict policy. Packages are tested daily using the command-line tool R CMD check. When these tests fail, package maintainers have to resolve the problems before the next major R release. Problematic packages are archived, making it impossible to install them automatically[3], as they will no longer be included in *CRAN* until a new package version is released that resolves the problems.

Another important distribution of R packages is *BioConductor* (abbreviated to *BioC* hereafter), focusing on software packages and datasets dedicated to bioinformatics. The datasets are subdivided into experiment data and annotation data. Together, they make up more than half of all packages available on *BioC*. Since the focus of this paper is on R's *software* ecosystem, these datasets will be treated separately during our analysis.

While fully compatible with the format of *CRAN* packages, *BioC* packages are not installed by default: users must configure their R installation with a *BioC* mirror. As for *CRAN*, packages that fail the daily check will be dropped from the next release of *BioC*.[4]

2.2 R package development forges

While *CRAN* and *BioC* are the largest and most well-known package repositories for the *distribution* of R packages, other repositories are frequently used for the *development* of R packages. Thanks to R packages like devtools (available since 2011), development of R packages is facilitated, and direct installation of packages from a source code repository like Git or Subversion becomes straightforward. As such, there is no longer a strict need to rely on package distributions. Therefore, R development forges have become an important and integral part of the R package ecosystem.

The two R package development forges that we consider in our study are *R-Forge* and *GitHub*. *R-Forge* is specialized at hosting R code. Its main target is to provide a central platform for the development of R packages, offering SVN repositories, daily built and checked packages, bug tracking, and so on. *GitHub*, a web platform for Git's distributed version control system, is also becoming increasingly popular for R package development.

Table 1 provides a brief comparison of the four R package repositories considered in our study. It also provides an idea of the size of each repository, in terms of the number of provided R packages. When installing packages from *CRAN* or *BioC*, only the most recent version is guaranteed to work with the latest version of R.

Support for multiple repositories is built deeply into R. For example, the R function install.packages can take the source repository as an optional argument, or can be used to install older versions of a given package. While *R-Forge* and *GitHub* are focused on development, packages can be installed directly from these forges using functions of the devtools package. For example, the function install_github allows the installation of R packages directly from *GitHub*, while the function install-svn allows the installation from a Subversion repository (such as the one used by *R-Forge*). By default, these functions will install the latest package version, but optional parameters can be used to install a specific commit.

3. RELATED WORK

Our research aims at studying the characteristics of R packages in function of the repository in which they are developed or distributed. A similar question, though not restricted to R packages, was studied by Beecher et al. in [3, 1]. They explored how the structure, complexity and decay of open source projects may be influenced by the repository in which they are retained (e.g., SourceForge, Savannah, Debian, RubyForge, GNOME, KDE). They concluded that membership of a particular repository may depend on the maturity and quality of the project. For example, SourceForge tends to host more early inceptors and immature projects, while Debian tends to hosts high-quality mature projects.

A number of researchers have studied the evolution of R packages. We are not aware of any studies taking into account other R package distributions (such as *BioC*) or development forges (such as *R-Forge* or *GitHub*). All related research seems to be restricted to the official package distribution *CRAN*.

German et al. [7] studied the evolution of *CRAN* from an ecosystemic viewpoint. They compared the characteristics, growth, dependencies and community structure of core packages and user-contributed packages. They also analysed the user and develop communities by studying mailing list

[3]It is still possible to install them manually.

[4]The check results and available packages for current and past releases of *BioC* are available on www.bioconductor.org/checkResults

traffic. In earlier work [5], we have studied the maintainability of *CRAN* packages in terms of errors discovered by the R CMD check tool, and how this relates to package dependencies and package updates. Based on these insights, in [4] we presented a web-based dashboard for helping R package maintainers deal with such issues.

There has been quite some empirical research on the software ecosystem of *GitHub* repositories, though not specifically related to R packages. Dabbish et al. [6] focused on the social and community aspects of *GitHub*. Vasilescu et al. [14] compared the involvement on *GitHub* with the activity on Stack Overflow, a Q&A website for software developers. Vasilescu et al. [15] studied a large sample of *GitHub* projects developed in Java, Python and Ruby. They compared direct code modifications (commits) with indirect ones (pull requests) and related this to success or failure of continuous integration with TRAVIS-CI. Gousios et al. [8, 9] provide *GHTorrent*, a scalable way to analyse *GitHub* data.

4. DATA EXTRACTION

An R package is a tar.gz archive that can contain code, examples, tests, data sets, and so on. It also includes a mandatory *DESCRIPTION* file that contains metadata describing the package's characteristics such as name, purpose, maintainer, version, imports and dependencies.

In order to carry out our empirical analysis, we need to extract data from each of the targeted R package repositories. We wrote a set of R and Python scripts to retrieve and process the data. A replication package is available on github.com/ecos-umons/iwseco2015.

CRAN – The metadata of R packages on *CRAN* was retrieved using *extractoR*[5], an R package that we already developed and used in earlier work [5]. It downloads the *CRAN* package sources, extracts their contents and stores the *DESCRIPTION* file metadata into R *data.frame* objects. Using this tool we collected, since September 2013, the metadata of 7,654 packages with their associated versions and dependencies. We explored in detail the state of *CRAN* on 19 March 2015, containing 6,411 non-archived packages.

BioC – On the same date, we extracted the metadata from the *DESCRIPTION* files of all 997 R software packages corresponding to release 3.0 of *BioC*. A list of candidate packages was retrieved from the SVN[6] and the *DESCRIPTION* files were retrieved. We separately considered *annotation data* packages and *experiment data* packages for our analysis.

R-Forge – We first identified 1,883 repositories on *R-Forge* on 18 March 2015 using the full repository list[7]. Some of those repositories contained no package, while others contained more than one package, since we identified several *DESCRIPTION* files corresponding to distinct package names. An automatic script extracted the metadata of 2,217 packages based on the *DESCRIPTION* files contained in those 1,883 repositories.

GitHub – *GitHub Archive*[8] is a tool that collects and stores data from *GitHub* that usually are not available anymore on *GitHub* after some time. We relied on the events stream collected by *GitHub Archive* to identify potential active R repositories on *GitHub*. First, we collected from *GitHub Archive* all events that occurred for two years until 31 December 2014, when GitHub released its new (partially backward incompatible) API. Then, we filtered out all events of type *PushEvent* that were related to repositories whose language was flagged 'R' by their owner. This resulted in a list of 121,385 candidate repositories that could contain R packages. We checked for the presence of a *DESCRIPTION* file at the root of those repositories, and fetched this file on 17 February 2015. As we are interested in comparing *GitHub* with other package sources like *CRAN* and *BioC*, if packages with the same name were found in two or more distinct repositories (including forks), we assumed the package belonging to the oldest repository to be the main one. Importantly, we also filtered out the repositories belonging to the *GitHub* accounts `cran` and `rpkg`, as these two accounts were used to mirror (part of) the contents of *CRAN*. This resulted in 5,150 distinct R packages from *GitHub*.

[5]github.com/ecos-umons/extractoR
[6]hedgehog.fhcrc.org/bioconductor/branches/RELEASE_3_0/madman/Rpacks/ (username/password: readonly)
[7]r-forge.r-project.org/softwaremap/full_list.php
[8]www.githubarchive.org

5. EMPIRICAL RESULTS

5.1 Where and how are R packages developed and distributed?

Figure 1 shows the overlap of R packages on different distributions and development forges. **The overlap between *CRAN* and *BioC* is very limited**, as expected. Both package distributions only have 4 packages in common, corresponding to 0.4% of all considered *BioC* packages and only 0.06% of all considered *CRAN* packages.

Many packages on *CRAN* also appear on *GitHub*, since both repositories serve different purposes (distribution and development, respectively). We also observe in Figure 1 that the intersection of packages that can be found on *CRAN* and *GitHub* is non negligible. 18.1% of all considered *CRAN* packages can also be found on *GitHub*. 22.5% of all R packages on *GitHub* are also present on *CRAN*. This relatively large proportion of overlap can be explained by the fact that *CRAN* is only a distribution platform, and cannot be used for collaborative development. Hence, ***many* R packages are developed on *GitHub*, while stable releases of these packages are published on *CRAN***. We observe something similar when comparing *BioC* with *GitHub*, and for the same reasons. Indeed, 20.6% of all packages on *BioC* have a counterpart on *GitHub*.

R-Forge has 12.3% of its packages in common with *GitHub*, while as much as 45.2% of its packages are in common with *CRAN*. This shows that ***R-Forge* serves as a development platform for *some* of the packages that get distributed through *CRAN***. This is not true for *BioC*: only 1.1% of all *R-Forge* packages are also available on *BioC*.

One of our goals is to study whether development forges like *GitHub* are overtaking *CRAN* and *BioC* as a primary source of R packages. To do so, we consider the set of all R packages that are available on *GitHub* on 17 February 2015, and study since when they were created and had a counterpart in other R package repositories.

Figure 2 suggests that the monthly number of newly created repositories for *CRAN* and *BioC* packages on *GitHub* is slightly increasing over time. This seems to imply that, over time, **developers of packages that are distributed on *CRAN* and *BioC* decide to use *GitHub* as a host**

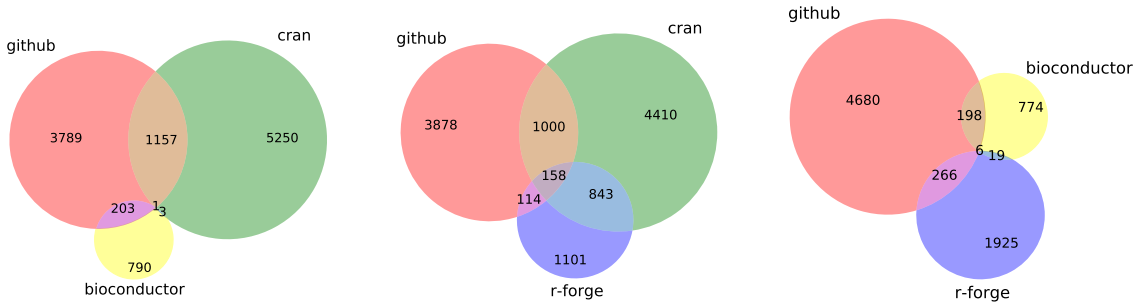

Figure 1: Intersections of R packages belonging to *GitHub*, *CRAN*, *BioC* and *R-Forge*

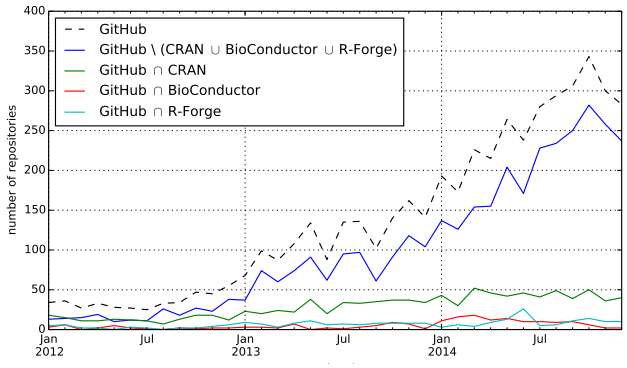

Figure 2: Monthly number of newly created repositories on *GitHub* containing R packages.

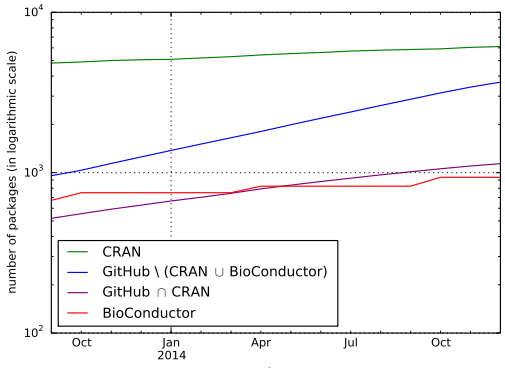

Figure 3: Evolution of the number of R packages in *CRAN*, *GitHub* and *BioC*.

for developing their packages. This does not seem to affect the growth of the packages in the *CRAN* and *BioC* distributions. Figure 3 shows that the number of packages in *GitHub* ∩ *CRAN* grows faster than the number of packages distributed on *CRAN*.

We did not find evidence of packages disappearing from *CRAN* or *BioC* due to their migration to *GitHub*. As shown in Figure 1, in March 2015 *CRAN* and *BioC* still remain the primary sources for the *distribution* of stable R packages. As such, **development of R packages through *GitHub* seems to complement distribution of packages through *CRAN* and *BioC***, and perhaps even has a catalyst effect.

Figures 2 and 3 also reveal that ***GitHub* is increasingly hosting R packages that do not have a counterpart in *CRAN* or *BioC***. Many of these packages are no longer actively maintained today. Those that do, may be developed for personal use only, or could still be unstable but at some point in the future may turn into stable packages that could become distributed in *CRAN* or *BioC*.

5.2 How do packages depend on one another?

Every R package needs to specify in its *DESCRIPTION* file the packages it depends upon. In our analysis we consider as dependencies those packages that are listed in the Depends: and Imports: fields of the *DESCRIPTION* file, as these are the ones that are required to install and load a package.[9]

[9]R CMD check also verifies Suggests: dependencies.

If an R package depends on another, do these packages belong to the same repositories, or do we observe many inter-repository dependencies? We expect most of such inter-repository dependencies to go towards *CRAN* since it is the official R package distribution. We also expect many dependencies from other repositories towards *BioC* since it is an active package distribution that offers the same quality checks as *CRAN*, and also because it contains many useful datasets. Therefore, when considering dependencies to packages belonging to *BioC*, we also count dependencies to dataset packages belonging to *BioC*.

As can be seen in Figure 1, the same package may belong to different repositories (for example, *GitHub* may store the development version while *CRAN* may contain the stable release version of the package). To cope with this, we define a total order $>$ on the set $\mathcal{S} = \{$*CRAN*, *BioC* software, *BioC* datasets, *GitHub*, *R-Forge*, Unknown$\}$ such that *CRAN* $>$ *BioC* software $>$ *BioC* datasets $>$ *GitHub* $>$ *R-Forge* $>$ Unknown. This total order privileges the distributed version of a package over its development version. For example, if a package p_1 on *GitHub* depends on a package p_2 that belongs to both *CRAN* and *GitHub*, it will be counted as a dependency from *GitHub* to *CRAN*.

Using this total order we can introduce some dependency metrics. Let $\alpha, \beta \in \mathcal{S}$ be two R package repositories (i.e., sets of packages). We define:

$$\text{deps}(\alpha,\beta) = \{(p_1,p_2) \in \alpha \times \beta \mid p_1 \text{ depends on } p_2 \ \wedge \nexists \gamma \in \mathcal{S} : (p_2 \in \gamma \wedge \gamma > \beta)\}$$

Table 2: Number of packages primarily belonging to repository α that depend on or are needed by at least one package primarily belonging to repository β.

Metrics	α	β					
		CRAN	*BioC* software	*BioC* datasets	*GitHub*	*R-Forge*	Unknown
$\mathsf{dependsOn}(\alpha,\beta)$	*CRAN*	**61.0%**	2.1%	0.1%	0%	0%	0%
$\mathsf{requiredBy}(\alpha,\beta)$		**24.9%**	5.8%	0.1%	15.7%	8.6%	–
$\lvert\mathsf{deps}(\alpha,\beta)\rvert$		**10,560**	212	4	1	1	0
$\mathsf{dependsOn}(\alpha,\beta)$	*BioC* software	**58.8%**	**77.1%**	9.3%	0.2%	0.1%	1.1%
$\mathsf{requiredBy}(\alpha,\beta)$		6.5%	**26.5%**	2.8%	12.6%	4.8%	–
$\lvert\mathsf{deps}(\alpha,\beta)\rvert$		1,615	**2,748**	133	2	1	13
$\mathsf{dependsOn}(\alpha,\beta)$	*BioC* datasets (1,115 packages)	15.2%	**77.1%**	17.5%	0%	0%	0%
$\mathsf{requiredBy}(\alpha,\beta)$		0.2%	6.2%	4.3%	1.9%	0.3%	–
$\lvert\mathsf{deps}(\alpha,\beta)\rvert$		333	**1,567**	204	0	0	0
$\mathsf{dependsOn}(\alpha,\beta)$	*GitHub*	**48.9%**	5.2%	7.4%	5.7%	0.3%	2.7%
$\mathsf{requiredBy}(\alpha,\beta)$		0%	0%	0%	4.9%	0.1%	–
$\lvert\mathsf{deps}(\alpha,\beta)\rvert$		**8,614**	684	37	386	15	156
$\mathsf{dependsOn}(\alpha,\beta)$	*R-Forge*	**37.2%**	2.3%	0.1%	0.7%	5.8%	1.5%
$\mathsf{requiredBy}(\alpha,\beta)$		0.1%	0.1%	0%	0.6%	5.0%	–
$\lvert\mathsf{deps}(\alpha,\beta)\rvert$		**1,830**	93	4	19	136	36

$$\mathsf{dependsOn}(\alpha,\beta) = \frac{|\{p_1 \in \alpha \mid \exists p_2 \in \beta : (p_1,p_2) \in \mathsf{deps}(\alpha,\beta)\}|}{|\alpha|}$$

$$\mathsf{requiredBy}(\alpha,\beta) = \frac{|\{p_1 \in \alpha \mid \exists p_2 \in \beta : (p_2,p_1) \in \mathsf{deps}(\beta,\alpha)\}|}{|\alpha|}$$

$|\mathsf{deps}(\alpha,\beta)|$ counts all dependency relationships from packages in α to packages in β, $\mathsf{dependsOn}(\alpha,\beta)$ gives the ratio of distinct packages in α depending on at least one package in β, and $\mathsf{requiredBy}(\alpha,\beta)$ the ratio of distinct packages in α on which at least one package in β depends. Table 2 presents these metrics for all pairs of considered R package repositories. Unknown represents those dependencies for which we did not find a matching package name in any of the considered repositories. This value was especially high for *GitHub* (140 packages with 156 dependencies to 89 unknown packages).

***CRAN* is self-contained**: the majority of dependencies of its packages stay within *CRAN*: 61% of all *CRAN* packages depend on another *CRAN* package. This is expected, since otherwise the packages would not pass the R CMD check. Note that only 24.9% of all *CRAN* packages are required by other *CRAN* packages.

***BioC* depends primarily on *CRAN* and on itself**: 58.8% of all *BioC* packages depend on *CRAN* packages, while 77.1% of all *BioC* packages depend on other *BioC* packages. Similar to *CRAN*, 26.5% of all *BioC* packages are required by other *CRAN* packages. We also observe that 9.3% of *BioC* software packages depend on *BioC* datasets.

GitHub* and *R-Forge* depend primarily on *CRAN: 87.1% of *GitHub* dependencies and 86.4% of *R-Forge* dependencies go to *CRAN* packages.

This shows that **CRAN is still at the center of the ecosystem** and that it has a minority of packages forming a core required by other packages both from CRAN and other sources.

6. THREATS TO VALIDITY

We only considered a subset of the R ecosystem, consisting of only 4 package repositories, but covering more than 12,000 distinct R software packages. While other R package repositories exist, given their small size we have not included them in our analysis. The Omega Project for Statistical Computing" (www.omegehat.org) hosts around one hundred R packages. *RForge* (rforge.net), not to be confused with *R-Forge*, provides a collaborative environment for R package developers based on SVN repositories, and contains less than two hundred packages, many of which are no longer active. *GitHub* competitors like *BitBucket* (bitbucket.org), *Gitorious* (www.gitorious.com) and *Gitlab* (gitlab.com) are considerably less frequently used for hosting R package development.

While for *BioC* we explicitly excluded (or treated differently) the packages containing datasets, we were not able to do the same for the other repositories, since we found no automated way to distinguish "ordinary" software packages from datasets. If an R package contains both data and functions, it is hard to decide whether it should be regarded as a software package or a dataset.

For part of our analysis, we relied on information extracted from SVN or Git, or from hosting services like *GitHub*. There are many potential perils and pitfalls that should be taken into consideration when doing so [2, 11]. Some of them can be avoided, others or inherent to the limitations of the considered version control systems or hosting services. For example, how should forking be taken into account? In our analysis, we excluded all forks. We also relied on *GithubArchive* as a proxy of *GitHub* data to extract events, but we cannot guarantee that this data is fully consistent and complete.

For *GitHub*, we assumed that the R package *DESCRIPTION* file always resides in the root directory of each Git repository. While this may have lead to the exclusion of some packages in our analysis, it avoids the inclusion of many "false positive" packages that are identical clones of existing *CRAN* package sources used by developers to avoid external dependency problems.

We also found that some *GitHub* accounts containing R packages (in particular, accounts cran and rpkg) actually served as partial mirrors of *CRAN*. These accounts were excluded from our analysis, but we have no guarantee that other accounts may also be mirrors of R packages developed or distributed elsewhere.

The chosen date of the R package ecosystem snapshot, and the chosen duration for the historical analysis may influence our results. Repeating the same analysis for other dates would allow us to confirm the observed results. For the historical analysis of the *GitHub* data, we based ourselves on the packages still existing in *GitHub* in February 2015. We were not able to extract historical information from *GitHub* repositories that have been removed before that date.

7. FUTURE WORK

Our analysis of the R software package ecosystem can be extended in many ways. An important avenue of future research is to study the evolution over time: do we observe important changes in the *diversity* of the ecosystem over time? Is there a relationship between the *quality* and *survival* of the packages contained in each considered repository? How does the *popularity* of packages (e.g., expressed in number of downloads or number of incoming dependencies) relate to these characteristics? How easy is it to *install* R packages depending on their provenance? What happens to R packages (especially in *CRAN* and *BioC*) after they have become *archived*?

Another future research track concerns a *socio-technical analysis* of the R package ecosystem: how do package developers collaborate? Do we observe a different collaboration behaviour depending on the considered repository? Does this behaviour correlate with desirable properties of packages?

Finally, we wish to study the impact of ongoing initiatives to facilitate R package management and producing reproducible results. Examples are the *Drat R Archive Template*[10], the *Managed R Archive Network* [11], the *Reproducible R Toolkit*[12], and packrat, a portable and reproducible dependency management system for R projects.

8. CONCLUSION

In this paper, we studied the ecosystem of R packages beyond the official *CRAN* repository. We also considered the *BioConductor* package distributions, and we explored two R package development forges: *GitHub* and *R-Forge*. In total, we analysed the origin and the dependencies of more than 12,000 packages that were still available in March 2015.

We observed that *CRAN* remains the center of the R package ecosystem, since its packages do not depend on external packages, while *BioConductor*, *R-Forge* and *GitHub* strongly depend on *CRAN* packages. *BioConductor* also contains many packages required by the others, but with an order of magnitude difference compared to *CRAN*.

We also observed that *GitHub* is becoming increasingly used as a collaborative development platform for R packages, both for packages already distributed on *CRAN* and *BioConductor*, as well as for new packages that do not have any counterpart in the considered distributions or forges. We did not observe any positive or negative effect of this increased use of *GitHub* on the growth of the number of *CRAN* or *BioConductor* packages.

In the future we intend to explore "the *GitHub* effect" in more detail, taking into account other factors such as the quality and longevity of packages, as well as a possible effect on the collaborative behaviour of package developers.

[10] github.com/eddelbuettel/drat

[11] mran.revolutionanalytics.com/packages

[12] projects.revolutionanalytics.com/rrt

9. REFERENCES

[1] K. Beecher, A. Capiluppi, and C. Boldyreff. Identifying exogenous drivers and evolutionary stages in FLOSS projects. *Journal of Systems and Software*, 82(5):739–750, 2009.

[2] C. Bird, P. C. Rigby, E. T. Barr, D. J. Hamilton, D. M. Germán, and P. T. Devanbu. The promises and perils of mining Git. In *Working Conf. Mining Software Repositories*, pages 1–10. IEEE Computer Society, 2009.

[3] A. Capiluppi and K. Beecher. Structural complexity and decay in FLOSS systems: An inter-repository study. In *European Conf. Software Maintenance and Reengineering*, pages 169–178, March.

[4] M. Claes, T. Mens, and P. Grosjean. maintaineR: A web-based dashboard for maintainers of CRAN packages. In *Int'l Conf. Software Maintenance and Evolution*, 2014.

[5] M. Claes, T. Mens, and P. Grosjean. On the maintainability of CRAN packages. In *Int'l Conf. on Software Maintenance, Reengineering, and Reverse Engineering*, pages 308–312, 2014.

[6] L. A. Dabbish, H. C. Stuart, J. Tsay, and J. D. Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *Int'l Conf. Computer Supported Cooperative Work*, pages 1277–1286, 2012.

[7] D. M. Germán, B. Adams, and A. E. Hassan. The evolution of the R software ecosystem. In *European Conf. Software Maintenance and Reengineering*, pages 243–252, 2013.

[8] G. Gousios and D. Spinellis. GHTorrent: Github's data from a firehose. In *Working Conf. Mining Software Repositories*, pages 12–21, 2012.

[9] G. Gousios, B. Vasilescu, A. Serebrenik, and A. Zaidman. Lean GHTorrent: GitHub data on demand. In *Working Conf. Mining Software Repositories*, pages 384–387, 2014.

[10] K. Hornik. Are there too many R packages? *Austrian Journal of Statistics*, 41(1):59–66, 2012.

[11] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. Germán, and D. Damian. The promises and perils of mining GitHub. In *Working Conf. Mining Software Repositories*, pages 92–101, 2014.

[12] R. A. Muenchen. The popularity of data analysis software. Technical report, r4stats.com, 2015. Last consulted on 8 April 2015.

[13] J. Ooms. Possible directions for improving dependency versioning in R. *R Journal*, 5(1):197–206, June 2013.

[14] B. Vasilescu, V. Filkov, and A. Serebrenik. StackOverflow and GitHub: Associations between software development and crowdsourced knowledge. In *SocialCom*, pages 188–195. IEEE, 2013.

[15] B. Vasilescu, S. Van Schuylenburg, J. Wulms, A. Serebrenik, and M. van den Brand. Continuous integration in a social-coding world: Empirical evidence from GitHub. In *Int'l Conf. Software Maintenance and Evolution*, pages 401–405, Sept 2014.