Journal of open research software

Boettiger, C et al 2015 Building Software, Building Community: Lessons from the rOpenSci Project. *Journal of Open Research Software,* 3: e8, DOI: http://dx.doi.org/10.5334/jors.bu

ISSUES IN RESEARCH SOFTWARE

Building Software, Building Community: Lessons from the rOpenSci Project

Carl Boettiger[1], Scott Chamberlain[2], Edmund Hart[3] and Karthik Ram[2]

[1] Department of Environmental Science, Policy, and Management; University of California, Berkeley, CA, USA
cboettig@gmail.com

[2] The rOpenSci project, Berkeley Institute for Data Science. University of California, Berkeley, Berkeley CA 94720
scott@ropensci.org, karthik.ram@berkeley.edu

[3] National Ecological Observatory Network, Boulder CO & Dept of Biology, University of Vermont, Burlington, VT
edmund.m.hart@gmail.com

rOpenSci is a developer collective originally formed in 2011 by graduate students and post-docs from ecology and evolutionary biology to collaborate on building software tools to facilitate a more open and synthetic approach in the face of transformative rise of large and heterogeneous data. Born on the internet (the collective only began through chance discussions over social media), we have grown into a widely recognized effort that supports an ecosystem of some 45 software packages, engages scores of collaborators, has taught dozens of workshops around the world, and has secured over $480,000 in grant support. As young scientists working in an academic context largely without direct support for our efforts, we have first hand experience with most of the the technical and social challenges WSSSPE seeks to address. In this paper we provide an experience report which describes our approach and success in building an effective and diverse community.

Keywords: R; open science; ropensci; data science

Introduction

The rOpenSci project has grown from unassuming beginnings into a diverse and well recognized effort to develop, maintain, and promote sustainable software tools for interacting with large and heterogeneous open data resources across the web. The project has focused on four broad mechanisms: (1) the development and maintenance of new software tools, (2) building a user and developer community around these tools and approaches, (3) educating domain scientists to be more active consumers of software tools and emerging open data resources and also producers of more reproducible, extensible, and transparent research, and (4) pursuing the long term sustainability of the rOpenSci project itself. **Figure 1** shows a conceptual diagram illustrating how the rOpenSci project serves as an interface connecting diverse elements of open science research.

Over the past three years, rOpenSci has helped create, grow, and support a community and practice of software development among full-time researchers in various domains primarily centered around the earth and environmental sciences. Our success in this area has been notable in that most practicing researchers either lack the necessary programming skills or the appropriate incentives to engage in software development. We attribute our success in this area to 1) widespread familiarity with R, 2) our intense and sustained community building efforts, and 3) strong ties to the domains we support.

In this paper, we share our approaches and experiences in building this community. We hope and believe that through collections of enough experiences such as this, that the WSSSPE (Working towards Sustainable Software for Science: Practice and Experiences) community [1] can better identify common themes in successful efforts as well as areas of opportunity for our own organization to improve.

Building Community

Why R?

Academic researchers use a variety of programming languages with different strengths and weaknesses, and many languages provide much of the same functionality. Why then, has the rOpenSci project chosen to focus exclusively on R? The first reason is that R has already been widely adopted among the domain sciences in which rOpenSci began: ecology, evolution, and statistics. This allowed the rOpenSci project to focus on empowering users by providing new software tools and approaches directly, rather than merely focus on introductory programming using existing tools. Working in a language that has already gained prominence within a particular scientific domain provides a two-fold boost to our community: researchers already familiar with R face a lower barrier of entry, while those not yet familiar are more likely to appreciate the value of learning the language. More than a decade of exposure in required statistics classes has made the language and its vocabulary

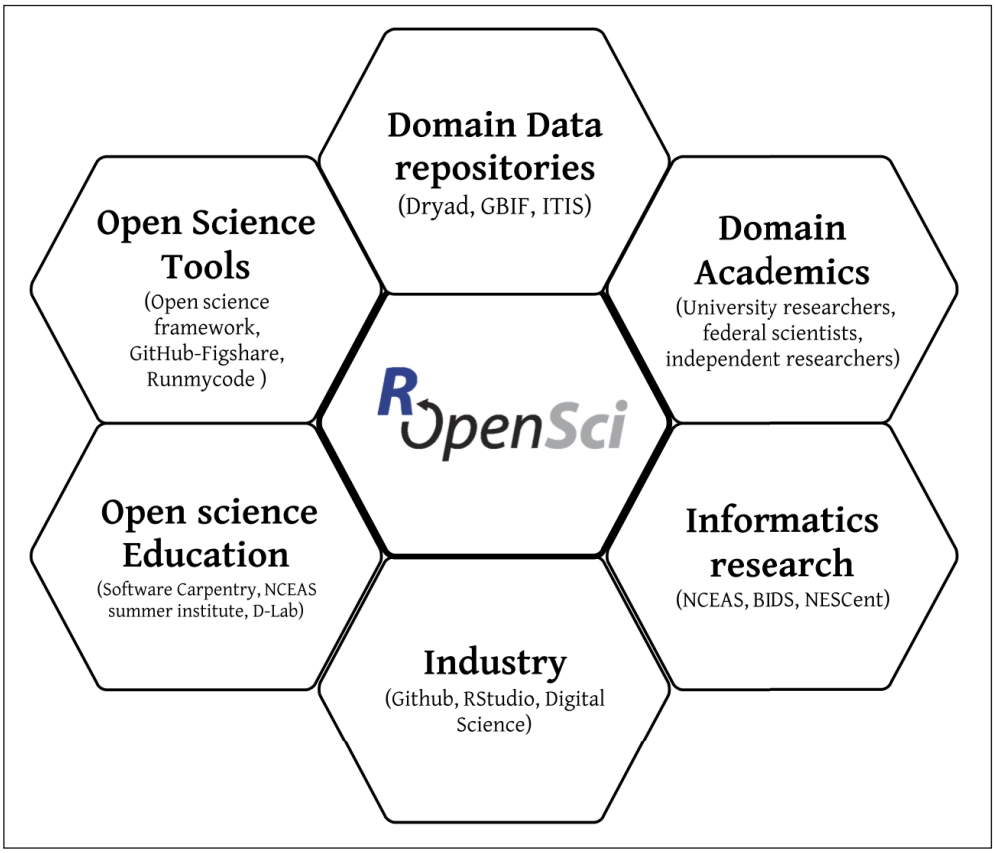

Figure 1: rOpenSci's role and connections in the open science sphere. Projects listed in each category are examples and not meant to be exhaustive.

familiar to a large group of researchers. This has made it easier for us to onboard researchers without spending considerable resources and time on more basic training. Related projects such as Software Carpentry have played a key role in directing more researchers to our efforts.

Second, focusing on a single language promotes a sense of community. Despite the wide range of dialects introduced by different styles, skills, and language extensions provided by packages, we have found that users across scientific domains identify with other users of the same language. This has brought recognition of the project from the larger R community, as well as facilitated the project reaching out into other domains where R has been widely adopted (including archeology and social sciences).

This same community identity can sometimes make it difficult to discuss the objective strengths of the language to users that already identify with a different programming language. Keeping that in mind, we briefly identify some of these strengths.

- R is the most popular language for statistical programming. It is used by the many of the world's largest companies, creating a high demand for these skills. In 2014, DICE Media's survey of salaries in tech showed R programmers topping the charts in a list of over 200 technical languages and skills.
- R was designed as a statistical research language from the start. No other major language has key statistical concepts like 'missing data' and support for heterogeneous data frames baked in at the lowest level.
- R blurs the distinction between developer and user. In most computer languages, only the developer needs to understand the language in which the software is written. Such separation between users and developers does not work well in a data-driven research context, which cannot be restricted to a small number of tasks with predictable input and output, and frequently requires that the user can extend or modify the functionality. Consequently, R is designed for precisely this context [2].
- R has a rich tradition of literate programming and reproducible research, through tools such as Sweave and knitr [4, 5] that have long been tightly integrated with LaTeX and scientific publishing.
- R is easy to install. The R project provides platform-specific binaries of both the base software and the essential and rapidly growing library of packages without relying on third party distributors.

Deliberate community-building efforts

Several of our core activities (outside of software development) have been entirely focused on building community. Since the core team for the project has a substantial and vocal social media (primarily twitter) presence, we regularly engage (and are often engaged) in various discussions with researchers from all career stages on topics ranging from computational efficiency and data publication, to reproducible research and literate programming. Such interactions have allowed us to identify opportunities where researchers with considerable domain expertise

would benefit from our software development experience to turn an idea into a piece of reusable code. Many such successful collaborations have involved early career researchers including new and newly appointed faculty. We believe that targeting this particular demographic has the highest potential to transform the culture of science.

In addition to our social media presence, we have spent countless hours over the past 18 months running workshops around the world (6 countries, 500+ participants). Our workshops have typically consisted of 1–4 hour sessions where we provide a rationale for open science and open software, and then walk participants through powerful use-cases closely tailored to their sub-domain. Using a cloud instance of R/RStudio for our workshops, we have successfully eliminated time wasted on installation and trouble-shooting allowing us to focus on demos and discussion.

Besides such workshops, we also modeled a highly successful coding un-conference, the first of which we held in March 2014 in San Francisco (http://ropensci.github.io/hackathon/). Our model differed from traditional hackathons in various ways. To combat the problem of gender imbalance at these events, we invested considerable time to locate and invite women participants at various levels of expertise to serve both as mentors but also attend as beginners. Our efforts results in 40% women attendees. We then invited several experts in R to attend, providing participants with an opportunity to closely interact with many of the language's well-known developers over two full days. Finally, we provided full financial support to everyone and held the event at an ideal location provided by our industry partner (GitHub). The event was a success for building a more inclusive community of research software developers in R, as addressed in a short video documentary about the event.

Building community by building standards

A commitment to sustainable software and reproducible research has always been a core part of rOpenSci development. As the project has gained recognition, our efforts to practice, teach, and facilitate the adoption of software development best practices have further contributed to the growth and strength of our community. Other developers contribute software they have already written to the rOpenSci project because of the project's reputation for best practices in writing sustainable software acts as a brand of approval for their work. Many rely on the practices in place at rOpenSci as a model of best practices in their own work. Our practices continue to evolve through a dialog with our community, and we will not attempt to summarize them here. An overview can be found on our organizational style guide.

rOpenSci also recognizes the challenges in adopting these practices, particularly in the context of domain-specific research. A recent post on our blog discusses these issues at greater length: ropensci.org/blog/2014/06/09/reproducibility.

The challenges for sustainable software in science go beyond best-practices of software development to affect the research workflow itself. As such, rOpenSci has sought to develop, test, model and promote scientific workflow practices to help ensure at least computational reproducibility [3] in scientific research. **Figure 2** provides a conceptual illustration of this workflow, indicating each of the points at which rOpenSci-developed software plays a role in facilitating the pipeline.

A steady dialog with our research community and other scientific software development communities has always been a key part of our development practices. For example, rOpenSci participated in a community effort to enumerate the challenges to sustainable software across the software lifecycle as part of a two-day workshop that brought members of the ecological and environmental sciences community from across industry, government, and academic positions to Santa Barbara. The products of this meeting were used to inform an SI2 proposal to NSF for an Institute for Sustainable Earth and Environmental Software (ISEES).

Perhaps the biggest key to our community building efforts have been the fact that we are still strongly embedded in the research community itself. Through both our own research and publications and/or close ties with practicing domain scientists in ecology and evolution, our team has always had a strong sense of the skills, needs and motivation of our audience. While some of the core team now charts career trajectories with more exclusive emphasis on academic software development and training, we could all draw on established reputations as practicing researchers before introducing ourselves as software providers to that community.

Results

As scientists, we must hesitate to generalize too broadly from our experiences with a single project. Ultimately, questions such as what factors contribute to that success and what factors create barriers or have little impact can only be answered with data that can compare across the experiences of many projects, successful and unsuccessful. Only then can we tease out the roles of personalities, timing, and other events beyond our control from the role of our delibrate choices. We can at best document our experiences and advance as hypotheses those choices which we have made as being causal factors in that success.

Nevertheless, we can quantify what we mean by the success of the rOpenSci project in a more empirical way.

We have come to see the success in the rOpenSci project primarily through the growth of our community of contributors – individuals who often had little or no previous experience with developing software or encountering best practices for sustainable software, who have become interested in the project and sought to make their own direct contributions to our growing software ecosystem. The growth in these community contributors, far more than the growth in downloads or other metrics of use of our software is the essential data behind our claim of the success of the project so far.

Figures 3 and **4** illustrate the growth of the rOpenSci community as measured in both number of contributors and the size of contributions (in lines of code committed across all rOpenSci software projects, excluding those contributions from the core members who are also the authors of this paper), respectively. Communities are

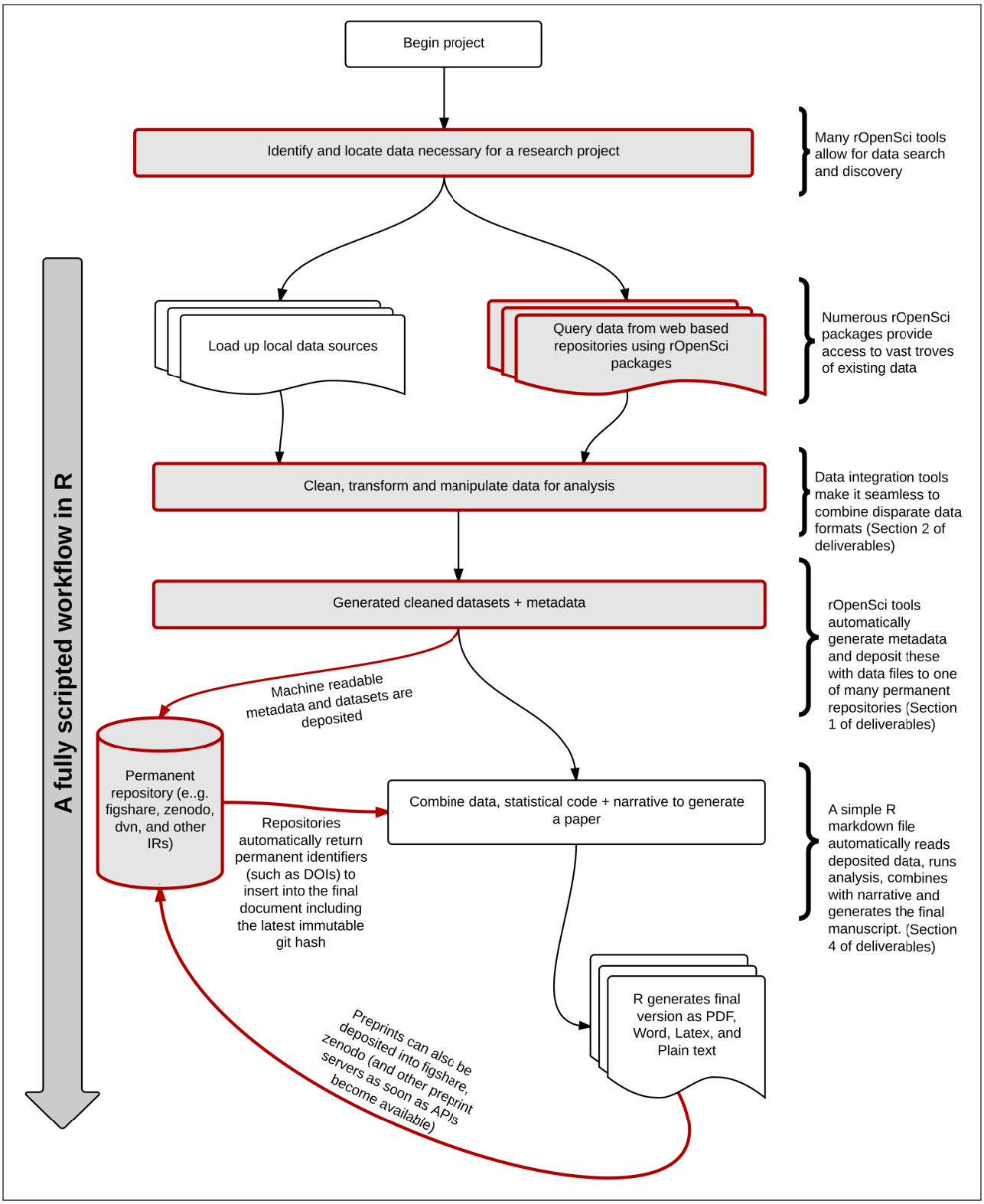

Figure 2: A conceptual data flow using our tool chain. Darker borders and shaded boxes indicate areas developed by rOpenSci tools. Remaining boxes are features that already exist in R.

diverse and multifaceted and thus not easily captured in a few quantitative summary statistics. Inferring the reasons or mechanisms responsible is a task for a larger study across data from multiple communities and thus beyond the scope of this case study.

Recommendations

Based on our own experiences and successes described above, we summarize the following three lessons which we feel may be applicable to other open source scientific communities.

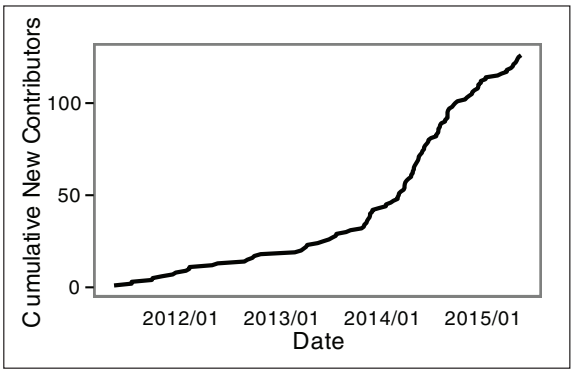

Figure 3: Growth of the rOpenSci community: cumulative number of contributors, 2011–2015.

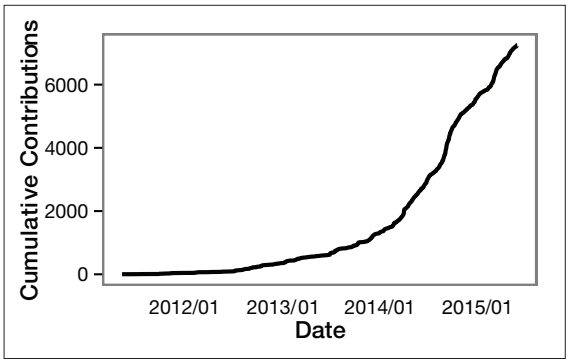

Figure 4: Growth of the rOpenSci community: cumulative number of lines of code committed by volunteer contributors, 2011–2015.

1. Create dynamic & community driven policies

rOpenSci practices and policies evolve as the result of a community driven process. Our style guide, sustainability practices, software platforms, and other norms evolve with input and suggestions from our community, in the same open-source spirit as our software itself. We recommend that other open-source communities should likewise consider treating policies as subject to the same open source and community driven practices as their software.

2. Active & postive engagement

rOpenSci has always pursued positive engagement with its broader community, both through an active presence on social media including blogs and twitter, as well as in-person workshops, conferences, and other events. rOpenSci has actively prioritized diversity and encouraged participation regardless of prior experience. We suggest other open source communities seek to actively and postively respond to contributors, support their efforts and the efforts of related projects.

3. Learn from other communities

This kind of active engagement is also a valuable learning opportunity for the organization itself. An active dialog with researchers has informed our focus on facilitating reproducible research. Close engagement with the R community and other other open source initiatives such as the Jypter project has often leads us to adopt new approaches, packages, and practices. We recommend any other open source community similarly see engagement not only as means to grow, but an opportunity to learn and adapt.

Conclusions

Through focused community building efforts the rOpenSci project has built a community of users and contributors. Key to these efforts have been active social media engagement, travel to conferences and the running of a hackathon that brought together novice developers and language experts to promote learning. Also by promoting reproducibility we help encourage sustainability. Reproducible documents are inherently more sustainable because they are based on annotated code. By investing in community outreach efforts, building off existing best practices, and promoting reproducible workflows, we hope to create a sustainable ecosystem of users and contributors allowing us to have a greater impact than we would alone.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This paper was originally presented at WSSSPE 2014, and the authors would like to acknowledge valuable feedback of the four reviewers WSSSPE reviewers and many workshop participants for engaging discussions and input that have helped shape this paper. The authors also acknowledge the support of a grant from the Alfred P Sloan Foundation to rOpenSci.

References

1. **Katz, D S, Choi, S-C T, Lapp, H, Maheshwari, K, Löffler, F, Turk, M, Hanwell, M D, Wilkins-Diehr, N, Hetherington, J, Howison, J, Swenson, S, Allen, G D, Elster, A C, Berriman, B** and **Venters, C** 2014 Summary of the First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1). *Journal of Open Research Software*, 2: 1–32.
2. **Ooms, J** 2014 The OpenCPU System: Towards a Universal Interface for Scientific Computing through Separation of Concerns, 1–23.
3. **Peng, R D** 2011 Reproducible research in computational science. *Science*, 334: 1226–1227. DOI: http://dx.doi.org/10.1126/science.1213847. PMid: 22144613; PMCid: PMC3383002.
4. **Xie, Y** 2013 Dynamic documents with R and knitr. Chapman; Hall/CRC, Boca Raton, Florida.
5. **Xie, Y** 2014 Knitr: A comprehensive tool for reproducible research in R. *in* V. Stodden, F. Leisch, and R. D. Peng, editors. *Implementing reproducible computational research*. Chapman; Hall/CRC.

How to cite this article: Boettiger, C, Chamberlain, S, Hart, E and Ram, K 2015 Building Software, Building Community: Lessons from the rOpenSci Project. *Journal of Open Research Software* 3: e8, DOI: http://dx.doi.org/10.5334/jors.bu

Published: 16 November 2015

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS