The Journal of Open Source Software

outsider: Install and run programs, outside of R, inside of R

Dominic J. Bennett[1, 2], **Hannes Hettling**[3], **Daniele Silvestro**[1, 2], **Rutger Vos**[3], **and Alexandre Antonelli**[1, 2, 4]

1 Gothenburg Global Biodiversity Centre, Box 461, SE-405 30 Gothenburg, Sweden **2** Department of Biological and Environmental Sciences, University of Gothenburg, Box 461, SE-405 30 Gothenburg, Sweden **3** Naturalis Biodiversity Center, P.O. Box 9517, 2300 RA Leiden, The Netherlands **4** Royal Botanic Gardens, Kew, TW9 3AE, Richmond, Surrey, UK

DOI: 10.21105/joss.02038

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @arfon

Submitted: 21 January 2020
Published: 23 January 2020

Statement of need

Enable integration of R and non-R code and programs to facilitate reproducible workflows.

Summary

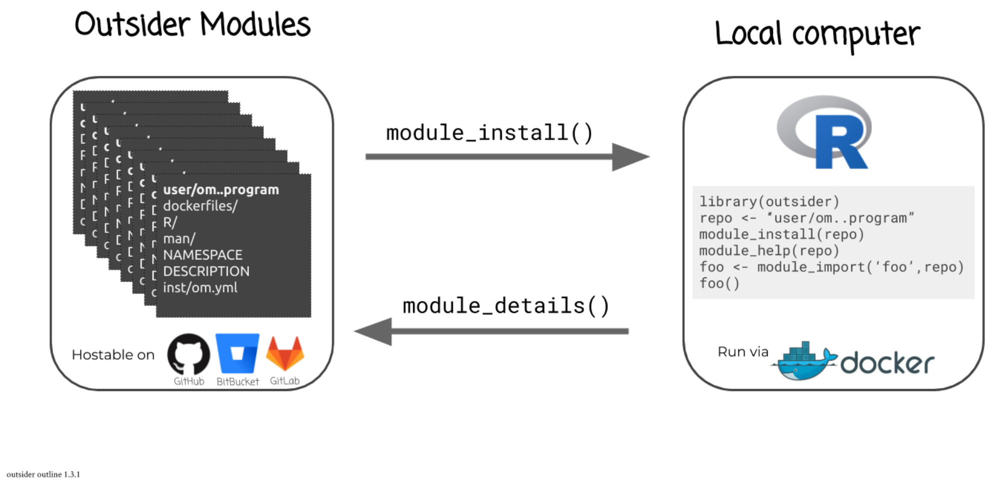
In many areas of research, product development and software engineering, analytical pipelines – workflows connecting output from multiple software – are key for processing and running tests on data. They can provide results in a consistent, modular and transparent manner. Pipelines also make it easier to demonstrate the reproducibility of one's research as well as enabling analyses that update as new data are added. Not all analyses, however, can necessarily be run or coded in one's favoured programming language as different parts of an analysis may require external software or packages. Integrating a variety of programs and software can lead to issues of portability (additional software may not run across all operating systems) and versioning errors (differing arguments across additional software versions). For the ideal pipeline, it should be possible to install and run any command-line software, within the main programming language of the pipeline, without concern for software versions or operating system. R (CRAN, 2019) is one of the most popular computer languages amongst researchers, and many packages exist for calling programs and code from non-R sources (e.g. `sys` (Ooms, 2019) for shell commands, `reticulate` (RStudio, 2019) for `python` and `rJava` (Urbanek, 2019) for `Java`). To our knowledge, however, no R package exists with the ability to launch external programs originating from *any* UNIX command-line source.

The `outsider` packages work through `docker` (Docker Inc., 2020a) – a service that, through OS-level virtualization, enables deployment of isolated software "containers" – and a code-sharing service, e.g. GitHub (GitHub, 2019), to allow a user to install and run, in theory, any external, command-line program or package, on any of the major operating systems (Windows, Linux, OSX).

How it works

`outsider` packages provide an interface to install and run *outsider modules*. These modules are hostable on GitHub (GitHub, 2019), GitLab (GitLab, 2019) and/or BitBucket (BitBucket, 2019) and consist of two parts: a (barebones) R package and a Dockerfile. The Dockerfile details the installation process for an external program contained within a Docker image, while the R package comprises functions and documentation for interacting with the external program via a Docker container. For many programs, Dockerfiles are readily available online and require minor changes to adapt for `outsider`. By default, a module's R code simply passes command-line arguments through Docker. After installation, a module's functions can then be imported and launched using `outsider` functions. Upon running a module's code, `outsider` code will first launch a Docker container of the image as described by the module's Dockerfile. `outsider` then facilitates the communication between the module's R code and the Docker container that hosts the external program (developers of modules have the choice of determining default behaviours for handling generated files). `outsider` modules thus wrap external command-line programs into R functions in a convenient manner. `outsider` functions allow users to look up available modules and determine build statuses (i.e. whether the package is passing its online tests) before installing.

At time of writing, `outsider` modules for some of the most popular bioinformatics tools have been developed: BLAST (Altschul, Gish, Miller, Myers, & Lipman, 1990), MAFFT (Katoh, Kuma, Toh, & Miyata, 2005), *BEAST (Bouckaert, 2019), RAxML (Stamatakis, 2006), bamm (Rabosky, n.d.), PyRate (Silvestro, Salamin, & Schnitzler, 2014). (See the `outsider` website for an up-to-date and complete list). All that is required to run these modules is R and Docker. Docker Desktop (Docker Inc., 2020b) can be installed for all operating systems but for older versions of OSX and Windows the legacy "Docker Toolbox" (Docker Inc., 2020c) may instead need to be installed. (Note, users may need to create an account with Docker-Hub to install Docker.)



```
library(outsider)
repo <- "user/om..program"
module_install(repo)
module_help(repo)
foo <- module_import('foo',repo)
foo()
```

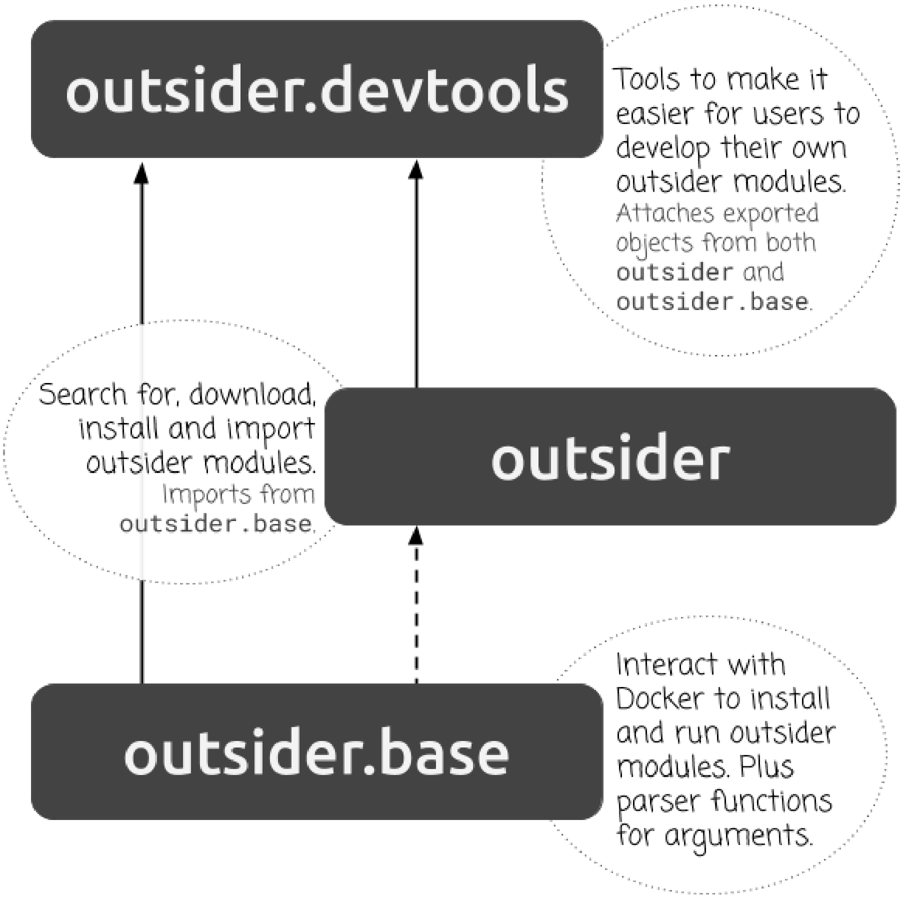
Figure 1: An outline of the outsider module ecosystem.

Code structure

The code-base that allows for the installation, execution and development of `outsider` modules is held across three different R packages. For end-users of modules, however, only the `outsider` module is required. For those who wish to develop their own modules, the `outsider.devtools` package provides helper functions for doing so. In addition, there is a test suites repository that hosts mock analysis pipelines that initiate several modules in sequence to test the interaction of all the packages.

- `outsider`: The main package for installing, importing and running **outsider modules** (Bennett, 2020a).
- `outsider.base`: The package for low-level interaction between module R code and Docker containers (not user-facing) (Bennett, 2020b).

- `outsider.devtools`: The development tools package for facilitating the creation of new modules (Bennett, 2020c).
- "outsider-testuites": A repository hosting a series of "test" pipelines for ensuring modules can be successfully strung together to form R/non-R workflows (Bennett, 2020d).


outsider.devtools
Tools to make it easier for users to develop their own outsider modules.
Attaches exported objects from both outsider and outsider.base.
Search for, download, install and import outsider modules.
Imports from outsider.base.
outsider
outsider.base
Interact with Docker to install and run outsider modules. Plus parser functions for arguments.

outsider packages outline 1.0.0

Figure 2: How the outsider packages interact

Examples

Saying hello from Ubuntu

By hosting a Docker container, `outsider` can run any UNIX-based, external command-line program. To demonstrate this process we can say "hello world" via a container hosting the Ubuntu operating system. In this short example, we will install a small `outsider module` – `om..hello.world` – that installs a local copy of the latest version of Ubuntu and contains a function for saying hello using the command `echo`.

```
library(outsider)
# outsider modules are hosted on GitHub
# this repo is a demonstration of an outsider module
# it contains a function for printing 'Hello World!'
repo <- 'dombennett/om..hello.world'
module_install(repo = repo)

# look up the help files for the module
module_help(repo = repo)

# import the 'hello_world' function
hello_world <- module_import(fname = 'hello_world', repo = repo)

# run the imported function
hello_world()
#> Hello world!
#> ------------
#> DISTRIB_ID=Ubuntu
#> DISTRIB_RELEASE=18.04
#> DISTRIB_CODENAME=bionic
#> DISTRIB_DESCRIPTION="Ubuntu 18.04.1 LTS"
```

A basic bioinformatic pipeline

To better demonstrate the power of the `outsider` package, we will run a simple bioinformatic pipeline that downloads a file of biological sequence data (borrowed from (Hesselberth, 2017)) and aligns the separate strands of DNA using the multiple sequence alignment program MAFFT (Katoh et al., 2005). Note that we can pass arguments to an `outsider` module, such as `mafft` in the example below, using separate R arguments for each command-line argument.

```
library(outsider)
repo <- 'dombennett/om..mafft'
module_install(repo = repo)
mafft <- module_import(fname = 'mafft', repo = repo)

# some example file
download.file('https://molb7621.github.io/workshop/_downloads/sample.fa',
              'sample.fa')

# run maft with --auto and write results to alignment.fa
mafft(arglist = c('--auto', 'sample.fa', '>', 'alignment.fa'))

# view alignment
cat(readLines('alignment.fa'), sep = '\n')
#> >derice
#> -actgactagctagctaactg
#> >sanka
#> -gcatcgtagctagctacgat
#> >junior
#> catcgatcgtacgtacg-tag
#> >yul
#> -atcgatcgatcgtacgatcg
```

For more detailed and up-to-date examples and tutorials, see the `outsider` GitHub page (Bennett, 2020e).

Availability

`outsider` (and its sister packages) are open-source software made available under the MIT licence allowing reuse of the software with limited constraints. It is aimed that all packages will be made available through CRAN (CRAN, 2019), e.g. `install.package("outsider")`. Currently, all are available from GitHub source code repositories using the `remotes` package, e.g. `remotes::install_github("ropensci/outsider")`

Funding

This package has been developed as part of the supersmartR project (Bennett, 2018) which has received funding through A.A. (from the Swedish Research Council [B0569601], the Swedish Foundation for Strategic Research and a Wallenberg Academy Fellowship) and through D.S. (from the Swedish Research Council [2015-04748]).

Acknowledgements

The authors wish to thank Daniel Nüst and Bob Rudis for taking the time to review the package and providing valuable ideas and suggestions. Additionally, we would like to thank Anna Krystalli for overseeing the review process and providing useful advice in turn. Finally, we would like to thank all the people behind ROpenSci and JOSS for making this project and its review possible.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, *215*(3), 403–410. doi:10.1016/S0022-2836(05)80360-2

Bennett, D. J. (2018). SupersmartR. Retrieved June 27, 2018, from https://github.com/AntonelliLab/supersmartR

Bennett, D. J. (2020a). Outsider: Install and run programs, outside of r, inside of r. doi:10.5281/zenodo.3615177

Bennett, D. J. (2020b). Outsider: Base package. doi:10.5281/zenodo.3615056

Bennett, D. J. (2020c). Outsider: Development tools package. doi:10.5281/zenodo.3615074

Bennett, D. J. (2020d). Outsider: Test suites. doi:10.5281/zenodo.3614982

Bennett, D. J. (2020e). Outsider: Install and run programs, outside of r, inside of r. Retrieved January 20, 2020, from https://github.com/ropensci/outsider

BitBucket. (2019). Bitbucket. Retrieved October 29, 2019, from https://bitbucket.org/

Bouckaert, T. G. A. B.-S., Remco AND Vaughan. (2019). BEAST 2.5: An advanced software platform for bayesian evolutionary analysis. *PLOS Computational Biology*, *15*(4), 1–28. doi:10.1371/journal.pcbi.1006650

CRAN. (2019). The comprehensive r archive network. Retrieved January 10, 2019, from https://cran.r-project.org

Docker Inc. (2020a). Docker. Retrieved January 23, 2020, from https://www.docker.com/

Docker Inc. (2020b). Docker desktop. Retrieved January 23, 2020, from https://www.docker.com/products/docker-desktop

Docker Inc. (2020c). Docker toolbox. Retrieved January 23, 2020, from https://docs.docker.com/toolbox/

GitHub. (2019). GitHub. Retrieved February 11, 2019, from https://www.github.com/

GitLab. (2019). GitLab. Retrieved October 29, 2019, from https://gitlab.com/

Hesselberth, J. (2017). Genome analysis workshop. Retrieved June 27, 2018, from https://molb7621.github.io/workshop/

Katoh, K., Kuma, K.-i., Toh, H., & Miyata, T. (2005). MAFFT version 5: Improvement in accuracy of multiple sequence alignment. *Nucleic acids research*, *33*(2), 511–8. doi:10.1093/nar/gki198

Ooms, J. (2019). Sys: Powerful replacements for base::System2. Retrieved January 10, 2019, from https://github.com/jeroen/sys

Rabosky, D. L. (n.d.). Automatic Detection of Key Innovations, Rate Shifts, and Diversity-Dependence on Phylogenetic Trees. *PLOS ONE*, *9*(2), e89543. doi:10.1371/journal.pone.0089543

RStudio. (2019). Reticulate: R interface to python. Retrieved January 10, 2019, from https://github.com/rstudio/reticulate

Silvestro, D., Salamin, N., & Schnitzler, J. (2014). PyRate: A new program to estimate speciation and extinction rates from incomplete fossil data. *Methods in Ecology and Evolution*, *5*(10), 1126–1131. doi:10.1111/2041-210X.12263

Stamatakis, A. (2006). RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics (Oxford, England)*, *22*(21), 2688–90. doi:10.1093/bioinformatics/btl446

Urbanek, S. (2019). RJava: R to java interface. Retrieved January 10, 2019, from https://github.com/s-u/rJava