Packrat: A Dependency Management System for R

J.J. Allaire — June 27, 2014

Reproducible Research

- Foundational as a basis for scientific claims

  "The goal of reproducible research is to tie specific instructions to data analysis and experimental data so that scholarship can be recreated, better understood and verified." *CRAN Task View on Reproducible Research*

- Crisis of confidence in results of data analysis due to lack of reproducibility.

- Across time (running the same analysis again years later)

- Across space (moving code from a desktop to a server, or between the systems of collaborators)

- In R we do better than in many environments, but we don't do well enough.

Tools for Reproducibility

- **Computation**
  - Can we execute again and get the same results?
  - Yes, because we preserve our analysis in R Scripts
- **Output**
  - Can we produce the same end-user output consistently?
  - Yes, because we have tools like Sweave and knitr
- **Configuraiton**
  - Can we run our computations and create our output with the same configuration across time and space?
  - No (or yes only with a lot of effort and bother)

Configuration Rot

- As packages evolve over the years they inevitably:
  - At best, change behavior in subtle ways
  - At worst, outright break previous code
- As a result, an analysis or report that works today against e.g. R 3.1 it unlikely to work without modification in 5 years time.
- This is already a widely observed problem with Sweave and knitr documents that users attempt to update with new data and assumptions (or even just re-run with code and data unchanged).

R-devel [RFC] A case for freezing CRAN

http://goo.gl/k77z6F

- Proposal to freeze CRAN along with R releases.
- Projects built against a given version of R/CRAN would be able to rely on stable package versions, and therefore be expected to continue to work in the future.
- Attractive notion because it's simple and requires no extra effort from users.

What if we could freeze CRAN?

That would solve part of the problem, but wouldn't account for:

- Packages obtained from other repositories
- Development versions of packages installed from R-forge and GitHub
- Internally developed packages
- Users (inevitably) needing one more feature or bugfix and requiring the very latest version of a package.

What would a frozen CRAN not have?

- Bug fixes delivered in a timely fashion.
- Vitality and dynamism associated with making work available immediately to the community.
- The ability to use older versions of R with newer versions of packages.

"To me it boils down to one simple question: is an update to a package on CRAN more likely to (1) fix a bug, (2) introduce a bug or downward incompatibility, or (3) add a new feature or fix a compatibility problem without introducing a bug? I think the probability of (1) | (3) is much greater than the probability of (2), hence the current approach maximizes user benefit." *Frank Harrell*

"People then will start finding ways around these limitations and then we're back to square one of having people use a set of R packages and R versions that could potentially be all over the place." *Gavin Simpson*

Freezing is the answer, but what to freeze?

- Freezing CRAN solve only a subset of the problem, and introduces it's own problems.
- The only complete answer to this problem is freezing projects.
- Individual projects should be able to freeze arbitrary combinations of R packages with a guarantee of being able to use them in the future.
- Note that even if we freeze CRAN we still need this as well, so why create the bother of freezing CRAN? Let's just do project freezing right!

How do other environments handle these concerns?

- Most have some variation of:
  - A per-project private library
  - The specification of explicit versions (or version ranges) of each dependency
  - The ability to programatically reconstruct the library based on the specifications
- Some examples:
  - Ruby Bundler (http://bundler.io/)
  - Node.js NPM (https://www.npmjs.org/)
  - Python Virtualenv (https://virtualenv.pypa.io/en/latest/)

How might a solution tailored to R users look?

- Fundamental difference at work: R users do not self-identify as software developers and therefore have little tolerance for additional workflow overhead.
- Any solution must therefore be highly automated, and work with both existing projects created without packrat as well as new projects.
- We want the same benefits (private library and capturing of dependencies), with none of the following required:
  - Hand editing of dependency declarations
  - Retrieval and management of package source code

Packrat as a Possible Solution

- Packrat is an R package that implements a dependency management system for R:
  - GitHub: https://github.com/rstudio/packrat
  - Will be submitted to CRAN later this year
- Creates a private package library for a given R project (i.e. working directory)
- `snapshot` function that records the package versions used by a project and downloads their source code for storage with the project.
- `restore` function that applies the snapshot to a directory (building packages from source as necessary)

Packrat Fundamentals

```
> packrat::init()
```

Create a packrat project within a directory, giving the project it's own private package library.

```
> packrat::snapshot()
```

Finds the packages in use in the project and stores a list of those packages, their current versions, and their *source code*.

```
> packrat::restore()
```

Restore the directory to the last snapshotted state (building packages from source as necessary).

Initializing a Project

```
> packrat::init()


Adding these packages to packrat:
            _
    packrat   0.2.0.130

Fetching sources for packrat (0.2.0.130) ... OK (GitHub)
Snapshot written to '~/projects/reshape/packrat/packrat.lock'
Installing packrat (0.2.0.130) ... OK (built source)
Bootstrap complete!
```

Snapshotting Installed Packages

```
> packrat::snapshot()


Adding these packages to packrat:
             _
    plyr       1.8.1
    Rcpp       0.11.2
    reshape2   1.4
    stringr    0.6.2

Fetching sources for plyr (1.8.1) ... OK (CRAN current)
Fetching sources for Rcpp (0.11.2) ... OK (CRAN current)
Fetching sources for reshape2 (1.4) ... OK (CRAN current)
Fetching sources for stringr (0.6.2) ... OK (CRAN current)
Snapshot written to '~/projects/reshape/packrat/packrat.lock'
```

Restoring the State of the Library

```
> packrat::restore()
```

```
Installing Rcpp (0.11.2) ... OK (downloaded binary)
Installing stringr (0.6.2) ... OK (downloaded binary)
Installing plyr (1.8.1) ... OK (downloaded binary)
Installing reshape2 (1.4) ... OK (downloaded binary)
```

Updating a Package from Github

```
> packrat::install_github("RcppCore/Rcpp")
```

```
> packrat::snapshot()
```

```
Upgrading these packages already present in packrat:
             from        to
    Rcpp   0.11.2   0.11.2.1

Snapshot written to '~/projects/reshape/packrat/packrat.lock'
```

```
> packrat::restore()
```

```
Installing Rcpp (0.11.2.1) ... OK (built source)
```

Bundling and Unbundling

```
> packrat::bundle()
```

```
The packrat project has been bundled at:
- "~/projects/reshape/packrat/bundles/reshape-2014-06-24.tar.gz"
```

```
> packrat::unbundle("reshape-2014-06-24.tar.gz", where = "~/Desktop")
```

```
- Untarring 'reshape-2014-06-24.tar.gz' in directory '~/Desktop'...
- Restoring project library...
Installing packrat (0.2.0.130) ... OK (built source)
Installing Rcpp (0.11.2.1) ... OK (built source)
Installing stringr (0.6.2) ... OK (downloaded binary)
Installing plyr (1.8.1) ... OK (downloaded binary)
Installing reshape2 (1.4) ... OK (downloaded binary)
Done! The project has been unbundled and restored at:
- "~/Desktop/reshape"
```

Anatomy of a Packrat Project

`.Rprofile`
Directs R to use the private package library (when it is started from the project directory).

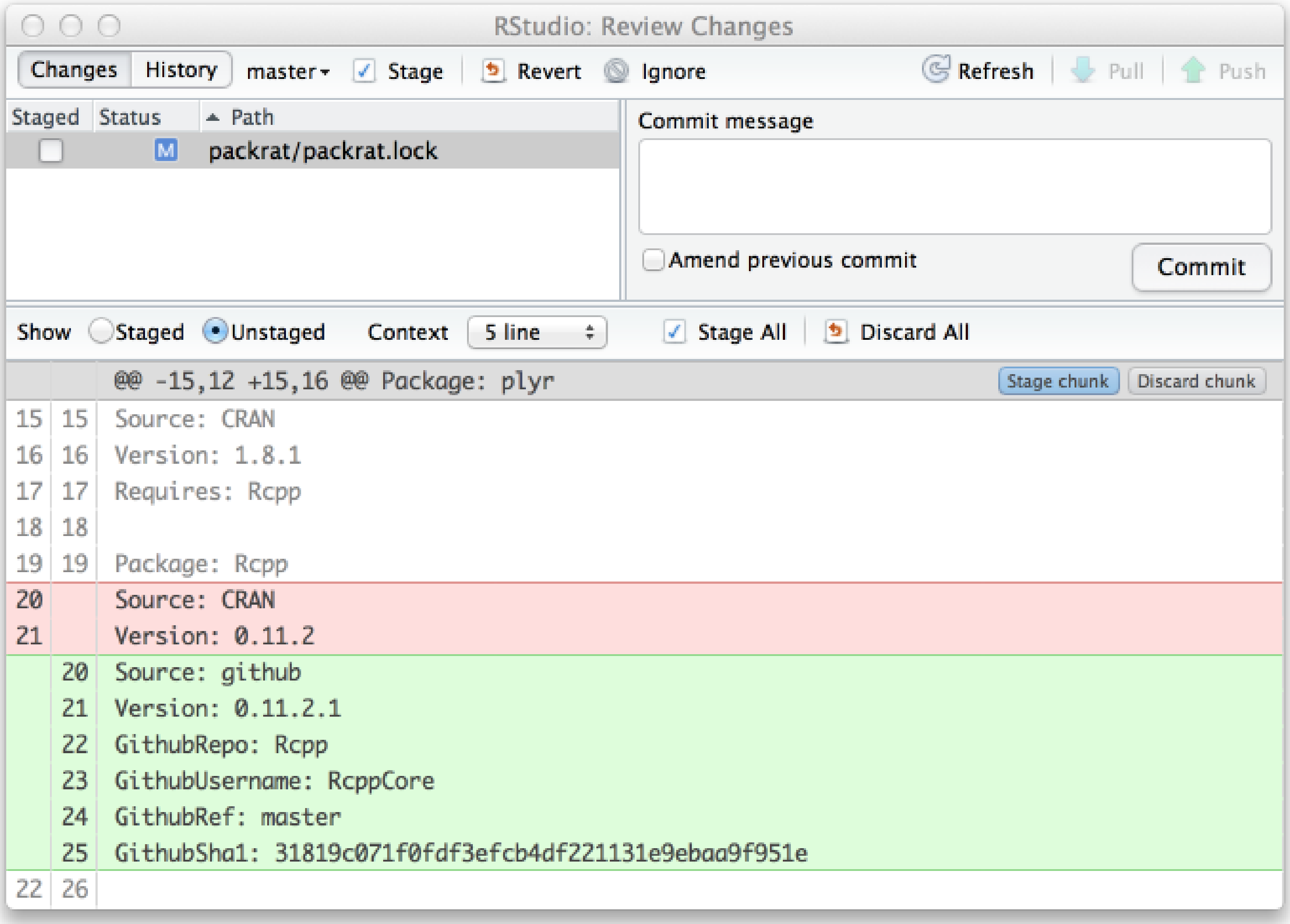
`packrat/lib/`
Private package library for this project.

`packrat/src/`
Source packages of all the dependencies that packrat has been made aware of.

`packrat/packrat.lock`
Lists the precise package versions that were used to satisfy dependencies, including dependencies of dependencies.

`packrat/packrat.opts`
Project-specific packrat options.

Packrat and Version Control



M packrat/packrat.lock

```
@@ -15,12 +15,16 @@ Package: plyr
 Source: CRAN
 Version: 1.8.1
 Requires: Rcpp

 Package: Rcpp
-Source: CRAN
-Version: 0.11.2
+Source: github
+Version: 0.11.2.1
+GithubRepo: Rcpp
+GithubUsername: RcppCore
+GithubRef: master
+GithubSha1: 31819c071f0fdf3efcb4df221131e9ebaa9f951e

```

Packrat Objectives

- Isolated, portable, and reproducible environment for R projects
- Capture all source code required to reproduce configurations
- Requires no changes to CRAN and capable of working with arbitrary other repositories
- Flexible and easy to use solution to the problem of reproducibility:
  - "One button" snapshot/restore
  - Simple and convenient archiving (bundle/unbunble)
  - Optional integration with version control

Questions?

Packrat website: http://rstudio.github.io/packrat

Packrat source: https://github.com/rstudio/packrat