R and Reproducibility
A Proposal

David Smith

Revolution Analytics

DSC 2014

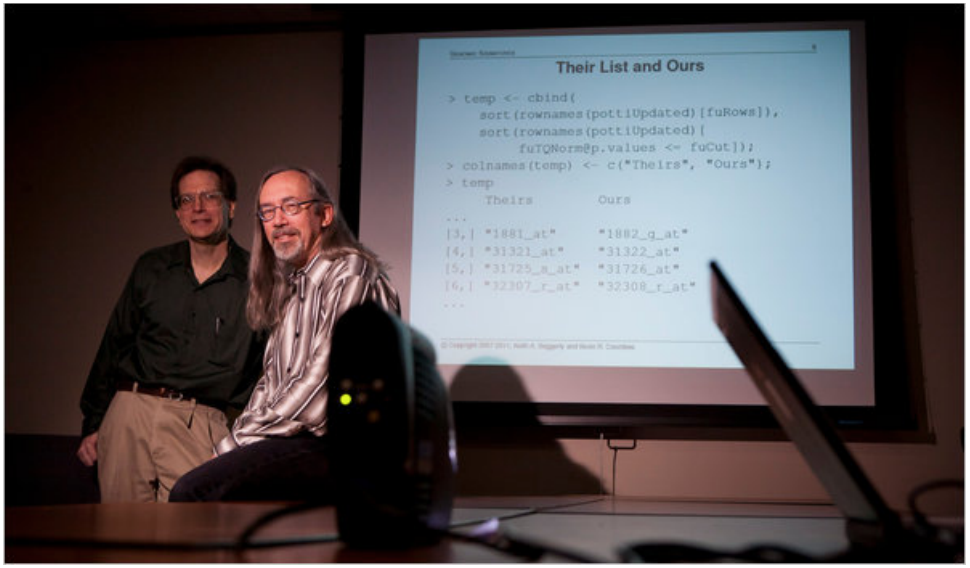
Why care about reproducibility?

Academic / Clinical Research

- Verify results
- Advance Research

Business

- Production code
- Reliability
- Firewalls
- Reusability
- Regulation

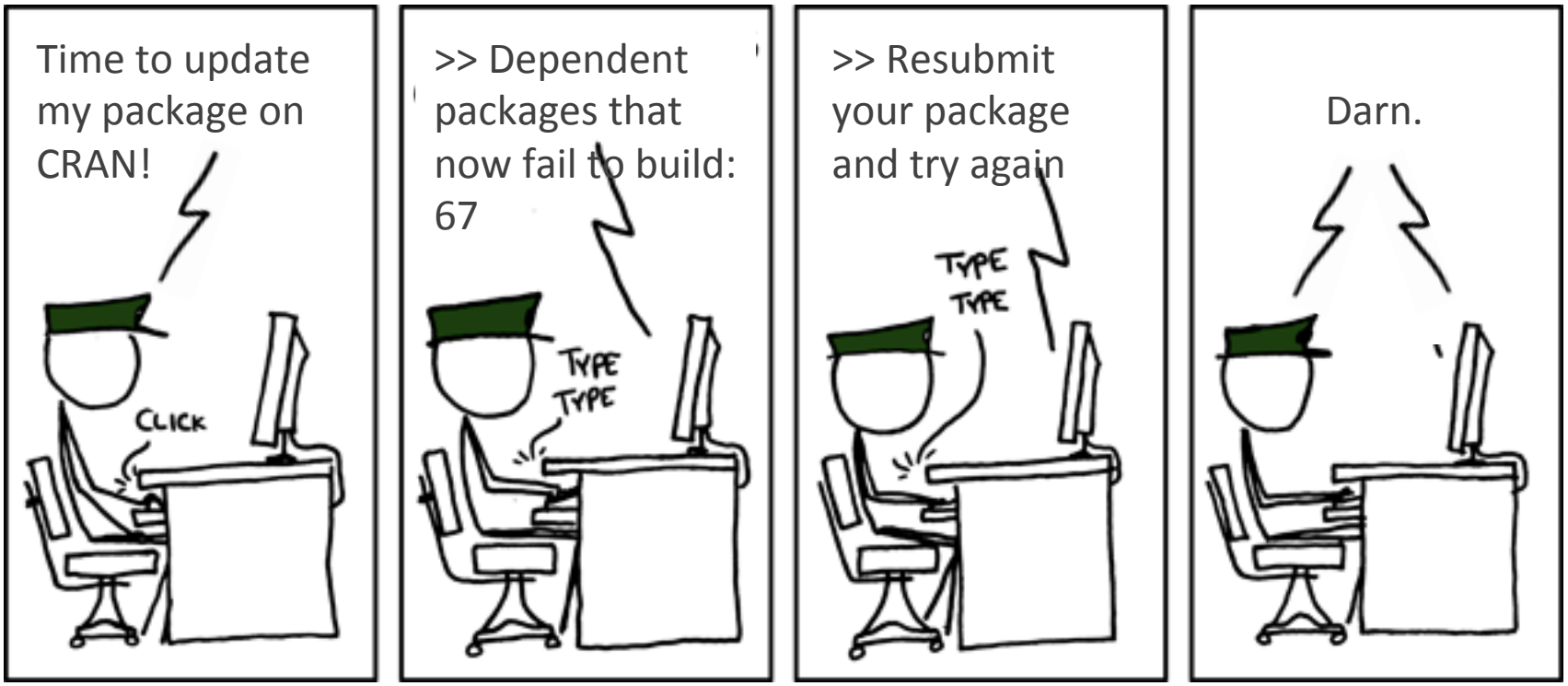


www.nytimes.com/2011/07/08/health/research/08genes.html
http://arxiv.org/pdf/1010.1092.pdf

Package Problem #1 : The User

http://xkcd.com/234/

Package Problem #2: The Author



http://xkcd.com/970/

CRAN is a moving target

- R itself is quite stable
- The big problem is with packages
  - Packages update in near real time
  - Difficult (possible, but difficult for regular users) to use specific package versions

- Users see “R” as “R + packages”

A Downstream "Stable Branch"

"Current Branch" **"Stable Branch"**

CRAN → DOWNSTREAM → MRAN

 → DOWNSTREAM → Revolution R

Making R Reproducible

- Change the default way R handles packages
- "Snapshot" CRAN package ecosystem with R releases
  - By default, users grab older versions of packages
- Tag scripts with an identifier to match with packages
  - "Reproducible R" version number?
  - DocID?

Upstream: R unchanged

"I don't see why CRAN needs to be involved in this effort at all. A third party could take snapshots of CRAN at R release dates, and make those available to package users in a separate repository. It is not hard to set a different repository than CRAN as the default location from which to obtain packages."

-- Duncan Murdoch, r-devel, March 2014

Not a new idea

- Ooms, “Possible Directions for Improving Dependency Versioning in R”, R Journal 5/1
- BioConductor Project
- Revolution R Enterprise
- packrat / gRAN
- Linux distros

Default behavior is critical

- Packrat solves this very well
  - Project + package dependencies stored in Github
- gRAN is also very promising
  - Pushing solution to gRAN server helps
- But:
  - Fragmentation: No CRAN “repository of record”
  - Not default behaviour
  - Not easy to share reproducibly for “normal” users

MRAN repository : requirements

- Bandwidth
- Storage
- Latency (alternatives to mirroring)
- Availability & monitoring
- Security
- Binary package archives
- Ability for package developers to “fall forward” to “development branch” packages
- Coordination with package authors with “Reproducible R” version updates
  - Goal: a consistent set of mutually-compatible packages every 6 months

MRAN - Implementation

- Use rsync to mirror CRAN at regular intervals
  - Only downloads changed packages
- Use zfs to store incremental snapshots
  - Storage only required for new packages
- Organize snapshots into a labelled hierarchy
  - Current and previous versions in same tree
- CRAN snapshot server hosted by cloud provider
  - Availability and latency
- Open-source process

RRT: The R Reproducibility Toolkit

- Open Source R Package (GPLv2)
- From an R project folder:
  - Detect packages used by scripts
    - Including dependencies
  - Download and install from MRAN
- github.com/RevolutionAnalytics/RRT
  - Pre-alpha!

Example

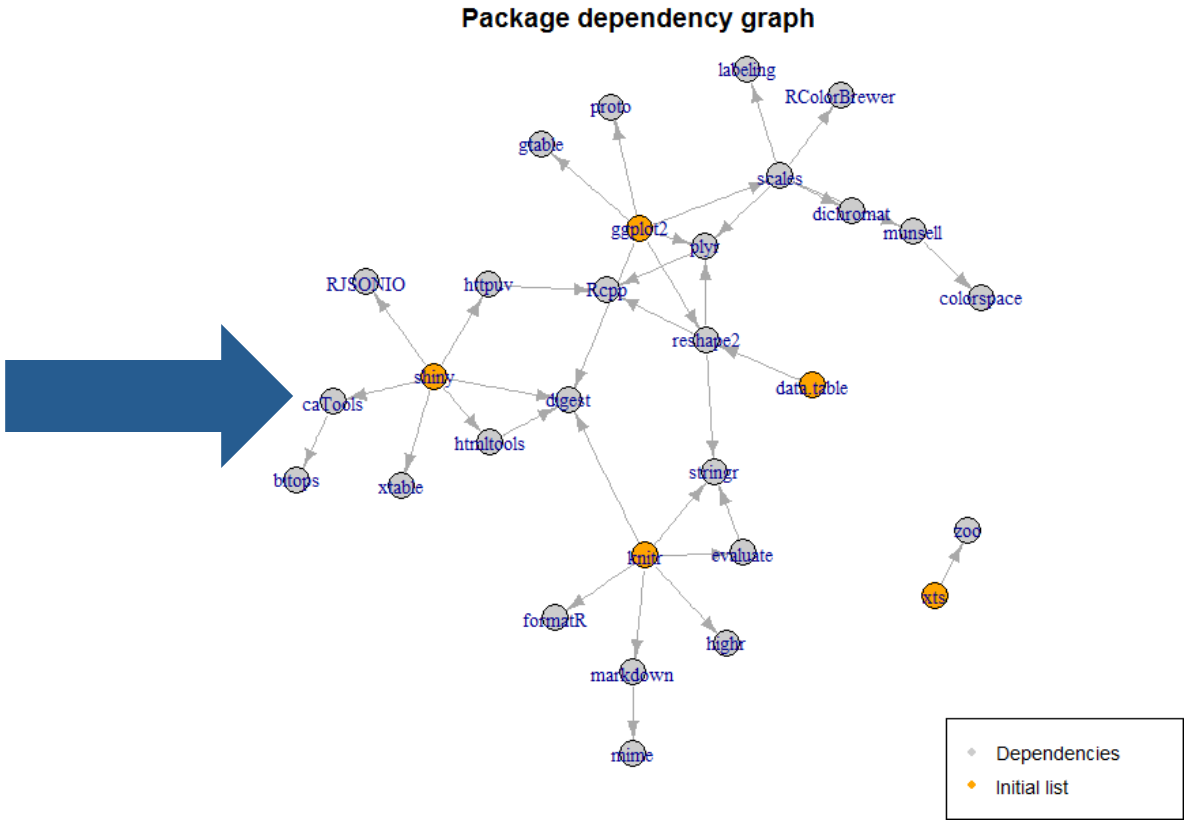
- R script file using 6 most popular packages

myScript.R

Source

```
1  ## Example script using packages
2  require(ggplot2)
3  require(data.table)
4  require(knitr)
5  require(xts)
6  require(shiny)
7
8  print(sessionInfo())
```

5:12 (Top Level) R Script



A lot yet to be done…

- MRAN server
  - Provisioning, automation, testing, maintenance
  - Naming
- R user default experience
  - Client-side tools
  - Downstream distribution
- Handling foreign packages (local, GitHub, etc)
- User testing
- Developer tools (R-Forge)

Thank You

David Smith
david@revolutionanalytics.com
blog.revolutionanalytics.com