Packages and unit testing in R

Adrienn Szabó

DMS Group, MTA SZTAKI

October 2, 2014

Where to find ready-to-use packages

CRAN (The Comprehensive R Archive Network) is a network of ftp and web servers around the world that store identical & up-to-date versions of code and documentation for R.

There are many mirrors around the world.

CRAN

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2014-07-10, Sock it to Me) R-3.1.1.tar.gz, read what's new in the latest

Where are my installed packages?

When you try to load a package, R searches for a match by name in some places in order. . .

Code: listing library paths

```
> .libPaths()
[1] "/home/adri/R/x86_64-pc-linux-gnu-library/3.0"
[2] "/usr/local/lib/R/site-library"
[3] "/usr/lib/R/site-library"
[4] "/usr/lib/R/library"
```

DMS

How to install new packages?

In your R script or R console:

Code: to install a new library

```
# installing into an existing dir as local library
install.packages("plyr", lib="~/R-packages")

# default path will be ~/R/x86_64-pc-linux.../3.1
install.packages("ggplot2") # will create a dir
```

Note: you can use a file named "`~/.Renviron`" to install everything to one place without specifying the local library each time:

```
R_LIBS=/home/yoda/my-R-lib/
```

How to use packages?

In your R script or console, you can "source" any installed package with the `library()` function:

Code: to load an installed library

```
library(ggplot2)
library(plyr, lib.loc = "~/R-packages")
```

Note: some packages will print something about themselves during loading, others don't print anything.

If there's an error, you'll notice.

Some useful packages

plyr	For doing "groupwise" operations with your data (summarizing, rearranging, etc.).
stringr	Tools for regular expressions and character strings.
ggplot2	Library for making beautiful, layered, customizable plots.
httr	A set of useful tools for working with http connections.
jsonlite	Read and create JSON data tables.

Some more useful packages

devtools	An essential suite of tools for turning your code into an R package.
testthat	Provides an easy way to write unit tests for your code projects.
parallel	Use parallel processing in R to speed up your code or to crunch large data sets.
Rccp	Write R functions that call C++ code for lightning fast speed.
lubridate	Makes it easier to work with dates and times.

Package development

Going towards object oriented from an interactive scripting environment.

Packages are the fundamental units of reproducible R code. They include (in a directory named after the package):

- reusable R functions (`R/` directory),
- documentation (`man/` directory),
- sample data (`data/` dir),
- and of course tests! (usually in `inst/tests/`).

Package development - terminology

Package	An extension of the R base system with code, data and documentation in standardized format.
Library	A directory containing installed packages.
Repository	A website providing packages for installation.
Source	The original version of a package with human-readable text and code.
Binary	A compiled version of a package with computer-readable text and code, may work only on a special platform.

Package development - terminology

Base packages Part of the R source tree, maintained by R Core.

Recommended packages Part of every R installation, but not necessarily maintained by R Core.

Contributed packages All the rest.

Package development - why?

If you are performing raw coding in R, one of the following is true:

- You are ignoring existing public functions
- The method is too user-specific to have a general function
- This may be a place for a new package

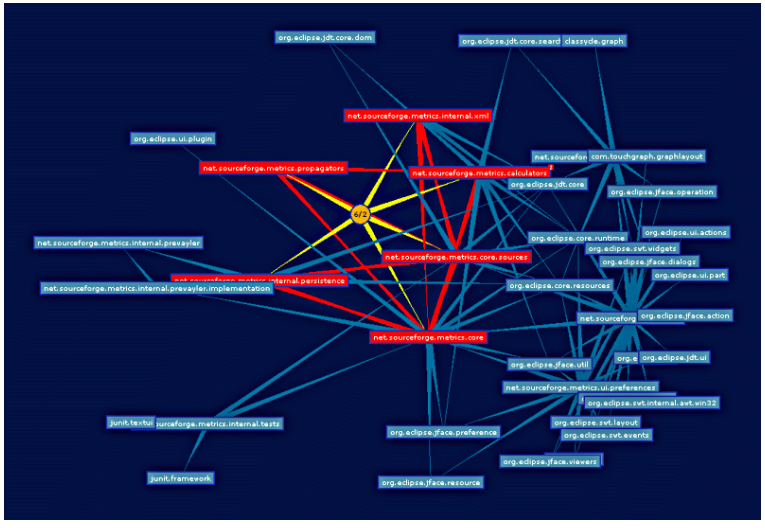
Package development - when not?

- It would be regrettable to spend 100 hours building something that already exists...
- Search CRAN for packages related to your idea (`cran.r-project.org`), check if overlapping packages are adequate
- Other repositories to check:
  - R Forge: `rforge.net`
  - Bioconductor: `bioconductor.org`
  - GitHub: `github.com`,
  - BitBucket: `bitbucket.org`,
  - etc.

Package development - problems

Just like everywhere else, where different versions of software packages build upon each other, problems may arise. . .

„R package dependencies can be frustrating.”



A solution: PackRat (rstudio.github.io/packrat/)

Inside a package

You need or should have these files and subdirectories inside your dir (that has the package's name):

- **`DESCRIPTION`**: metadata about the package
- **`R/`**: where your R code lives in `xy.R` files
- **`man/`**: function documentation
- **`data/`**: sample datasets, necessary data files (or other R objects)
- **`inst/tests/`**: `testthat` unit tests
- **`NAMESPACE`**: ensures that your package plays nicely with others
- **`src/`**: compiled C, C++ and fortran source code
- some more, less commonly used subdirs/files: `vignettes/`, `exec/`, `configure`

DMS

Why should you write unit tests?

- Fewer bugs
- Better code structure
- Easier to pick up where you left off
- Increased confidence when making changes

Note: there are some fundamental differences from unit testing in OO languages because R is, at heart, a functional programming language.

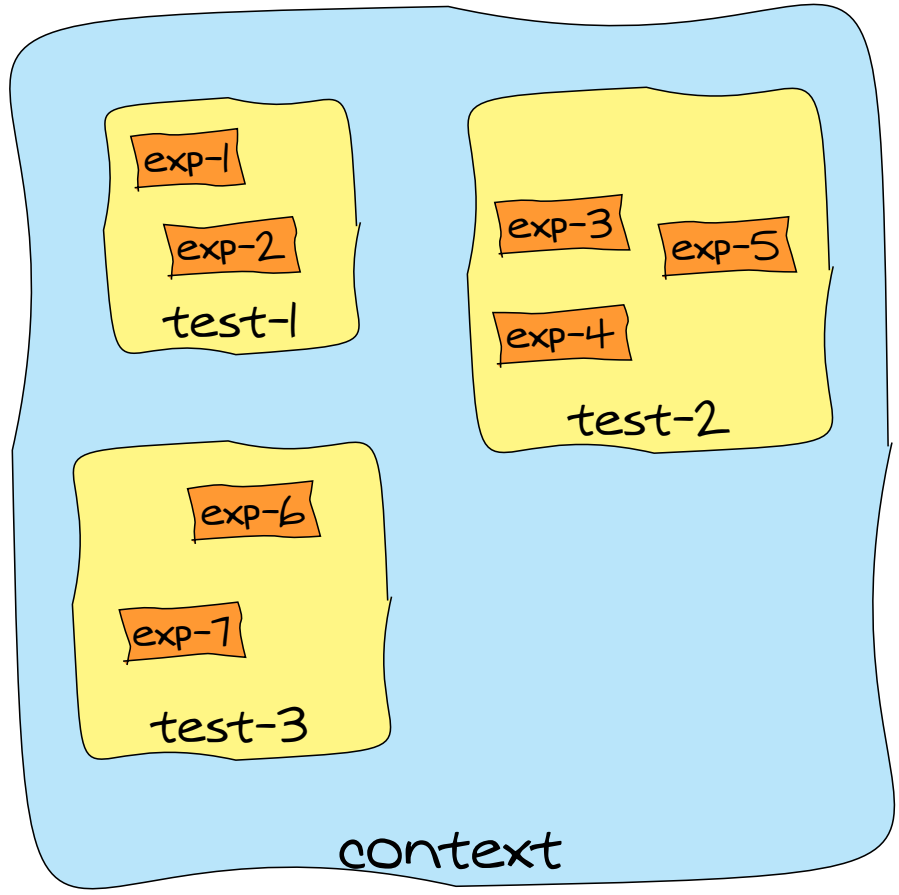
Why testthat and not svUnit or RUnit?

- simpler, cleaner usage
- easy way to test for warnings
- testthat caches tests, runs faster
- quick stats from github: out of 20 R projects only 5 had any kind of tests, 4 used testthat, 1 used RUnit

Structuring our tests

`testthat` has a hierarchical structure made up of

- expectations
- tests
- context



Expectations

An expectation describes the expected result of a computation:

- Does it have the right value and right class?
- Does it produce error messages when it should?

There are 11 types of built in expectations.

Note: This is the most basic building block of tests, the level of „assert” in other unit test frameworks.

Tests

A test groups together multiple expectations to test one function, or tightly related functionality across multiple functions.
A test is created with the `test_that()` function.

Code: a test

```
test_that("str_length is number of characters", {
  expect_equal(str_length("abc"), 3)
  string <- "Testing is fun!"
  expect_match(string, "Testing")
  expect_that(log(-1),
              gives_warning("NaNs produced"))
})
```

Contexts

A context groups together multiple tests that test related functionality.

Note: Normally there is one context per file.

Code: a context

```
context("String length")
test_that("str_length is number of characters", {
  expect_equal(str_length("a"), 1)
  expect_equal(str_length("abcd"), 4)
})
test_that("str_length of missing is missing", {
  expect_that(str_length(NA), equals(NA_integer_))
  expect_that(str_length("NA"), equals(2))
})
```

Workflow

You have 3 options for automated testing:

- Run all tests in a file or directory test_file() or test_dir()
- Have R CMD check run your tests
- Automatically run tests whenever something changes with autotest

Sources I

- cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf
- r-pkgs.had.co.nz/tests.html
- journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf
- www.r-bloggers.com/automatically-convert-runit-tests-to-testthat-tests/
- www.johnmyleswhite.com/notebook/2010/08/17/unit-testing-in-r-the-bare-minimum/
- www.hsph.harvard.edu/statinformatics/soft/files/buildingrpackages.pdf

Sources II

- rstudio.github.io/packrat/
- metrics.sourceforge.net/
- blog.yhathq.com/posts/10-R-packages-I-wish-I-knew-about-earlier.html
- https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages
- gastonsanchez.com/Handling_and_Processing_Strings_in_R.pdf
- www.news-sap.com/business-one-starter-package-sme/ (image)