Building R Packages
An Introduction

Thilo Klein

Cambridge Judge Business School

26 January 2015

Outline

Why build an R package?

Accessible

- Functions and objects contained in a package and installed on a machine can be easily loaded:
  > library(myPackage)
- Many R users develop their own functions that they use regularly
- Putting code into a package can be worthwhile, even for a sole user

Reliable

- Documentation structure is familiar, and it is easy to edit
- Basic checks and tests can be automated

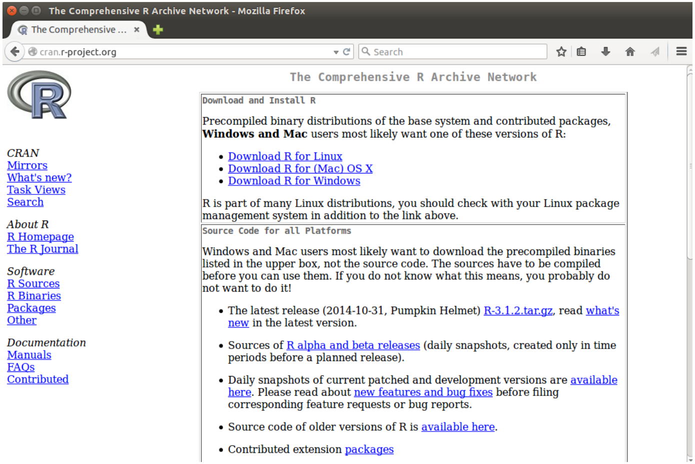
Clarity

- The process of organizing code and data into a package requires a project to become organized and set specific goals

Sharing data, functions, and an analysis online

CRAN features 6,221 packages, as of 26/1/2015

(up from 3,646 in 2012 and 2,564 in 2010).



The Comprehensive R Archive Network

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

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 (2014-10-31, Pumpkin Helmet) R-3.1.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

What are all these packages?

Methods

- Facilitate the use of a new or existing statistical technique
- Provide tools for graphics, data exploration, complex numerical techniques, making it easier to work with big data sets, etc.

Open research

- Researchers publish packages that implement new methods or release data, which supports reproducibility

Data

- Sharing old, new, simulated, or research data sets
- Many of the best packages have both methods and data

Keep an eye out

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

Ultimate goal

- Build a package to fulfill a need

Considerations

- The span of R users is wide: applied, software development, visualization, teaching, etc.
- Even if a method is already available, it doesn't mean it was written efficiently, is accurate, or reaches all audiences
- May be preferable to help improve an existing package than to build a new one from the ground-up

Outline

So you want to build a package...

It would be regrettable to spend 100 hours building something that already exist

Review CRAN packages for packages related to your idea

- cran.r-project.org
- Look for similar topics
- Identify the audience of other packages
- Check if overlapping packages are adequate

Other repositories to check/consider

- R-Forge: r-forge.r-project.org
- GitHub: github.com
- This list is not exhaustive!

So you are going to build a package...

Mission and goals

- Establish clear aims for the software before starting and choose a clear point at which you will publish your work

Achieve the basics

- Make software that runs, is efficient, and does what it claims
- The software should be intuitive for the target audience

Good coding practices

- Implement clean coding practices so others can review and verify your work

Document your work

- Provide helpful documentation with many examples

Example package: matchingMarkets

matchingMarkets: contains R code for matching algorithms such as the deferred-acceptance algorithm for college admissions [...]

s.prefs matrix of dimension nColleges x nStudents with the ith column containing student i's ranking over colleges in decreasing order

```
## 2 students, 2 colleges with one place each, random preferences:
R> daa(nStudents=2, nSlots=c(1,1))
$s.prefs
     [,1] [,2]
[1,]    1    1
[2,]    2    2
$c.prefs
     [,1] [,2]
[1,]    2    2
[2,]    1    1

$matches[[1]]
[1] 2
$matches[[2]]
[1] 1
```

Outline

Overview

Step 1: Create the package files

- The basic package files are created automatically for RStudio's project type 'R Package'

Step 2: Edit the package files

- Fill in the DESCRIPTION and help files (man > .Rd)
- Edit or add a NAMESPACE file
- Function or data updates should be done within the package files

Step 3: Build, check, and install the package

- Use RStudio's 'Build' tab to build, check, and install the package
- Usually errors arise when checking the package, so return to step 2 as needed

Step 1: Create the package files

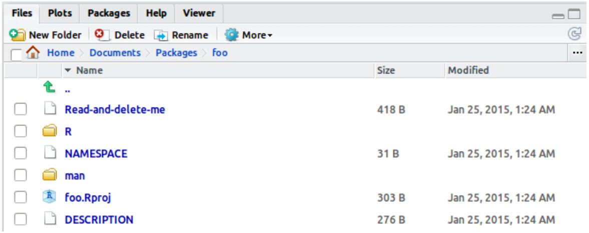
In RStudio, follow the steps below

Create package source folder

1. File > New project
2. Create project from: New Directory
3. Project type: R Package
4. Package name: foo
5. choose a file path for the project and click on 'Create Project'

Step 1: Create the package files

The package source folder, which has the same name as the package, contains several files and folders that were automatically generated



Name	Size	Modified
..		
Read-and-delete-me	418 B	Jan 25, 2015, 1:24 AM
R		
NAMESPACE	31 B	Jan 25, 2015, 1:24 AM
man		
foo.Rproj	303 B	Jan 25, 2015, 1:24 AM
DESCRIPTION	276 B	Jan 25, 2015, 1:24 AM

Read-and-delete-me	file to be deleted
R (folder)	contains .R files for each function
NAMESPACE	manages function, method, and dependency info
man (folder)	optional help files
foo.Rproj	R project file
DESCRIPTION	general package information

Step 2: Edit the package files

Add a custom function foo.R und documentation foo.Rd to the package.

Edit R > foo.R

```
## my custom multiply function
foo <- function(x,y){
  x*y
}
```

Create man > foo.Rd

```
\name{foo}
\alias{foo}
\title{My custom multiply function.}
\description{Multiplies two numbers.}
\usage{foo(x,y)}
\arguments{
  \item{x}{numeric vector}
  \item{y}{numeric vector}
}
```

Step 3: Build, check, and install the package

Package build

- From the 'Build' tab, click 'Build & Reload'
- Congrats – you just built the package foo!

Package check

- From the 'Build' tab, click 'Check'
- If a package is being submitted to CRAN, it must pass check

Package install

- Click 'More' from the 'Build' tab
- Click 'Build source package'
- Now you have foo.tar.gz which you can share (email or put on the web somewhere to download)

Outline

Next steps and useful resources

Sharing your package

- Learn about subversion and/or git revision control and
- Have a look at R-Forge (r-forge.r-project.org) and/or GitHub (github.com) repositories

Helpful references

- R packages
  r-pkgs.had.co.nz
  by Hadley Wickham
- Advanced R
  adv-r.had.co.nz
  by Hadley Wickham

Remarks

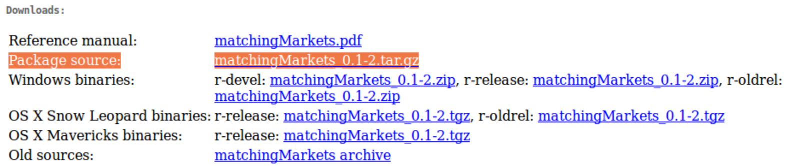
Packages can lead to papers

- Initially a package may provide support for an applied and methodological paper in the name of open research
- A robust package can have its own paper

Two journals to consider, both with free access

- Journal of Statistical Software – www.jstatsoft.org
- R Journal – journal.r-project.org

Find the source of packages on their CRAN pages



Downloads:

Reference manual:	matchingMarkets.pdf
Package source:	matchingMarkets_0.1-2.tar.gz
Windows binaries:	r-devel: matchingMarkets_0.1-2.zip, r-release: matchingMarkets_0.1-2.zip, r-oldrel: matchingMarkets_0.1-2.zip
OS X Snow Leopard binaries:	r-release: matchingMarkets_0.1-2.tgz, r-oldrel: matchingMarkets_0.1-2.tgz
OS X Mavericks binaries:	r-release: matchingMarkets_0.1-2.tgz
Old sources:	matchingMarkets archive

Thilo Klein

klein.uk
github.com/thiloklein/matchingMarkets