Building Packages

Chao-Jen Wong, Nishant Gopalakrishnan, Marc Carson, and Patrick Aboyoun

Fred Hutchinson Cancer Research Center

20-21 May, 2010
R Packages
  Package Concept
  Creating R packages
  Package Tools

Package Dependencies and Namespaces

Unit Testing

Documentation
  Manual Pages
  Vignettes

Tools of the trade
  Version Control
  Efficient Editing

Resources
Outline

R Packages
  Package Concept
  Creating R packages
  Package Tools

Package Dependencies and Namespaces

Unit Testing

Documentation
  Manual Pages
  Vignettes

Tools of the trade
  Version Control
  Efficient Editing

Resources
Package Concept

R packages

A collection of source code allows the user to attach to R session when calling `library()` or `require()`.

Why write a package?

- Better way to organize your code.
- Ability to share software as R packages.
- Provide reliable access.
- Provide communication channel between the users and authors.
Creating a Package

Initialize a new package

> package.skeleton()

Source package

- Special files
  - Essential: DESCRIPTION and NAMESPACE.
  - Others: configure, LICENSE, COPYING, INDEX and NEWS.
- Subdirectories containing source code, documentation and other material.
Creating a Package

Subdirectories

<table>
<thead>
<tr>
<th>Directory</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>source files (.R)</td>
</tr>
<tr>
<td>data</td>
<td>files of data objects to be loaded by <code>data()</code></td>
</tr>
</tbody>
</table>
| inst      | content copied to the installed packages’ directory  
|           | `doc` – Sweave document (.Rnw)  
|           | `extdata` – misc. data objects (ASCII)  
|           | `unitTest` – unit testing functions |
| man       | Rd documentation |
| src       | source code in C, FORTRAN or C++ |
| tests     | test code in R |
Package Tools

R shell tools

➤ Used to manage packages (build, check and etc.).
➤ Can be accessed from a command shell.

Shell commands
Take the form: `R CMD operation`

```bash
$R CMD build package
$R CMD check package
$R CMD check --help
$R CMD INSTALL package
$R CMD REMOVE package
...```

Outline

R Packages
  Package Concept
  Creating R packages
  Package Tools

Package Dependencies and Namespaces

Unit Testing

Documentation
  Manual Pages
  Vignettes

Tools of the trade
  Version Control
  Efficient Editing

Resources
Package: affy
Version: 1.25.2
Title: Methods for Affymetrix Oligonucleotide Arrays
Author: Rafael A. Irizarry <rafa@jhu.edu>, Laurent Gautier

Maintainer: Rafael A. Irizarry <rafa@jhu.edu>
Depends: R (>= 2.8.0), Biobase (>= 2.5.5)
Imports: affyio (>= 1.13.3), Biobase (>= 2.5.5), graphics, grDevices, methods, preprocessCore, stats, utils
Suggests: tkWidgets (>= 1.19.0), affydata
Description: The package contains functions for exploratory
License: LGPL (>= 2.0)
Dependencies in DESCRIPTION

- Declare what packages are required to run your package.
- Clarify the relationship between your package and other packages.
- Give clear and reliable definition of the package’s behavior (namespaces).
Dependencies in DESCRIPTION

Depends
Packages expected to be attached

Imports
- Only a few functions or objects are used by this package.
- Not necessarily needed to be attached.
- Avoid the cost in time and space of accessing the unused functions.

Suggests
- Used in examples or vignettes.
- Introduce special functionality.
Namespace

Why give your package a namespace

- Avoid conflicts and confusion from multiple functions (from attached packages) with the same names.
- Control what are public (exported) and private.
Namespace

- Declare in the NAMESPACE file.
- Required being explicit about what is exported and imported.
- ‘import’ – entire package or specific objects, classes and methods.
  
  ```r
  import(Biobase)
  or
  importFrom(Biobase, openVignettes)
  ```

- ‘export’ – explicit list of objects, methods and classes.
  
  ```r
  exportPattern("^[^\\.\["]")
  export(...)
  exportClass(...)
  exportMethods(...)
  ```

- Sealed once the package is installed. Non-exported functions can be addressed by the `:::` operator.
Useful Tool: codetoolsBioC

> library(codetoolsBioC)
> ls(2)

[1] "findExternalDeps"
[2] "getRdFileNames"
[3] "writeNamespaceImports"
[4] "writeRUnitRunner"

writeNamespaceImports

Writes imports statements that can be included in a package’s NAMESPACE file.

> library(GenomicFeatures)
> writeNamespaceImports("GenomicFeatures")
Outline

R Packages
  Package Concept
  Creating R packages
  Package Tools

Package Dependencies and Namespaces

Unit Testing

Documentation
  Manual Pages
  Vignettes

Tools of the trade
  Version Control
  Efficient Editing

Resources
What is a unit test?

A function myFun

library(RUnit)

myFun <- function(a) {
  # input checking
  if(!is.numeric(a))
    stop("'a' should be of type 'numeric(1)'")
  if(length(a) != 1)
    stop("'a' should be of length 1")

  # calc factorial
  factorial(a)
}

Unit test for myFun

test_myFun <- function() {
  target <- 6
  current <- myFun(3)
  checkIdentical(target, current)
  checkException(myFun("A"))
  checkException(myFun(1:8))
}

Why Unit tests?

- Interface specification
- Ensures code correctness, e.g., when R changes
- Allows refactoring without breaking existing code
- Encourages writing simple, working code chunks that can be integrated into larger components
- Encourages collaboration – tests describe what is supposed to happen
- Helps describe bugs – ‘this test fails’
- Documentation for developer – what code is intended to do
The RUnit package

- Framework for test case execution
  - create a series of test functions
  - define a test suite (defineTestSuite)
  - run the tests (runTestSuite)
  - summarize results (printTextProtocol, printHTMLProtocol)

- Hint: use writeRUnitRunner from the codeToolsBioC package
Adding Unit tests to your package

- Create test functions
  - save in inst/unitTests folder of your package
- Function to create test suite, run tests, summarize results
  - use `writeRUnitRunner` to create the file containing the `.test` function
  - save in R folder of your package
- Function to call the `.test` function
  - save in the tests folder of your package
- Add RUnit to the Suggests field in DESCRIPTION
Running a unit tests

> library(nidemo)
> nidemo:::.test()
Outline

R Packages
  Package Concept
  Creating R packages
  Package Tools

Package Dependencies and Namespaces

Unit Testing

Documentation
  Manual Pages
  Vignettes

Tools of the trade
  Version Control
  Efficient Editing

Resources
Documentation Types

Manual pages

- Reference pages for R objects (functions, data sets, etc.)
- Written in “R documentation” (Rd) format
- Thoroughly checked during R CMD check
- Templates created by prompt* family of functions

Vignettes

- A task-oriented description of package functionality
- Contain simple ”HowTo”s that are built with the package
- Written in Sweave (.Rnw) format, which integrates R code into LaTeX documents
- Required component of a Bioconductor package

Details provided in Writing R Extensions manual.
Manual Pages (A Simple Example)

\name{name}
\alias{alternate name}
\title{name of manual page}
\description{Brief description of what this does.}
\usage{
  myfun(arg1, arg2 = FALSE)
}
\arguments{
  \item{arg1}{\code{arg1} is required}
  \item{arg2}{\code{arg2} is optional}
}
\details{Important details on how it does it.}
\value{Return type}
\seealso{\code{\link[pkg:pkgfun]{pkgfun}}}
\author{Your name here}
\examples{## R code to demo this}
\keyword{names from KEYWORDS file in R doc dir}
Manual Pages (Tips)

- Flip through “Writing R documentation files” chapter of *Writing R Extensions* manual
- Change manual page when underlying R object changes
- Make examples run fast and be robust to changes in annotations and web resources
- Run R CMD check on modified packages
Vignette (Skeleton)

\documentclass[11pt]{article}
\usepackage{Sweave}

\newcommand{\Rfunction}[1]{{\texttt{#1}}}
\newcommand{\Robject}[1]{{\texttt{#1}}}
\newcommand{\Rpackage}[1]{{\textit{#1}}}
\newcommand{\Rclass}[1]{{\textit{#1}}}

\title{Descriptive Title}
\author{your name}

\begin{document}
\maketitle
\end{document}
The \texttt{foo} package is great.

First load the \texttt{foo} package and then execute function \texttt{foo}.

\begin{verbatim}
library(foo)
foo()
\end{verbatim}
Vignette (Code Blocks)

```r
<<UnevaluatedCode, eval=FALSE>>=
longRunningFunction(bigDataObject)
@
<<UnseenCodeAndOutput, echo=FALSE>>=
options(width = 60)
@
<<UnseenMessages, results=hide>>=
library(Biobase)
@
<<IncludeGraphic, fig=TRUE>>=
plot(1:10)
@
<<KeepMyFormat, keep.source=TRUE>>=
loveMyFormat(arg1 = "first",
               arg2 = "second")
@
```
Sweave and Stangle Commands

Sweave – creates a post-(code block)-processed \LaTeX\ file
Stangle – creates an R script from code blocks

R commands

> library(tools)
> Sweave("foo.Rnw")
> texi2dvi("foo.tex", pdf=TRUE, clean=TRUE)
> Stangle("foo.Rnw")

Shell commands

R CMD Sweave foo.Rnw
R CMD texi2dvi --pdf --clean foo.tex
R CMD Stangle foo.Rnw
Outline

R Packages
  Package Concept
  Creating R packages
  Package Tools

Package Dependencies and Namespaces

Unit Testing

Documentation
  Manual Pages
  Vignettes

Tools of the trade
  Version Control
  Efficient Editing

Resources
Need for Version Control

Problems

- Projects consist of multiple files
- We add/remove/change content
- Multiple people editing same file -> merge changes
- Multiple machines/operating systems -> merge changes
- Go back to a previous snapshot

The wrong way

- User managed backups
Version control software

- svn
- Mercurial
- git
Bioconductor svn

- Devel Branch
  - https://hedgehog.fhcrc.org/bioconductor/trunk/madman/Rpacks

- 2.6 Release Branch
  - https://hedgehog.fhcrc.org/bioconductor/branches/RELEASE_2_6/madman/Rpacks
  - username:readonly password:readonly

Reference Book: *Version Control with Subversion*
http://svnbook.red-bean.com/
Useful svn commands: svn checkout

```
svn co
https://hedgehog.fhcrc.org/bioconductor/trunk/madman/Rpacks/BiocCaseStudies/ --username readonly --password readonly
```

```
ngopalak@compbio-00702lt:~/svn/Rpacks>
ngopalak@compbio-00702lt:~/svn/Rpacks> svn co https://hedgehog.fhcrc.org/bioconductor/trunk/madman/Rpacks/BiocCaseStudies/ --username readonly --password readonly
A BiocCaseStudies/R
A BiocCaseStudies/R/colors.R
A BiocCaseStudies/R/init.R
A BiocCaseStudies/R/mySessionInfo.R
A BiocCaseStudies/R/fixedWidthCat.R
A BiocCaseStudies/R/resample.R
A BiocCaseStudies/R/requiredLibs.R
A BiocCaseStudies/DESCRIPTION
A BiocCaseStudies/man
A BiocCaseStudies/man/markup.Rd
A BiocCaseStudies/man/requiredPackages.Rd
A BiocCaseStudies/man/parseLibVers.Rd
A BiocCaseStudies/man/mySessionInfo.Rd
A BiocCaseStudies/man/fixedWidthCat.Rd
A BiocCaseStudies/man/resample.Rd
A BiocCaseStudies/NAMESPACE
Checked out revision 46926.
```
Useful svn commands: svn log

svn log NAMESPACE | more

- Added functions to control output of integers in Sexprs
- removed allset from NAMESPACE exports
- added BiocCaseStudies package

Logs are useful only if useful commit messages are provided.
- Commit once conceptual change at a time.
Useful svn commands

- svn checkout
- svn add
- svn checkin
- svn update
- svn status
- svn log -v
- ...

Efficient Editing
Editing from the command line

- history()
- savehistory("foo")
- loadhistory("foo")
- reverse search on unix using Ctrl-R
Code editors

▶ Eclipse(StatET) http://www.walware.de/goto/statet
▶ Emacs(ESS) http://ess.r-project.org/
▶ Tinn-R
▶ Notepad++(NppToR) http://sourceforge.net/projects/npptor/

Advantages
▶ syntax highlighting
▶ auto indent code
▶ send code/functions to R console
Efficient work flows

Editing without building documentation or configure

```
R CMD check --no-vignettes --no-examples pkgs
R CMD INTSALL --no-docs pkgs
R CMD INSTALL --no-configure pkgs
R CMD INSTALL --help
```
.First <- function() {cat("\n Hello! \n\n")}

if (interactive()) {
  tryCatch({
    source("http://bioconductor.org/biocLite.R")
  }, error=function(e) invisible(NULL),
    warning=function(w) message("Not connected to the net"))
}

reload_pkg <- function(p) {
  detach(paste("package", p, sep = ":"), unload = TRUE,
      character.only = TRUE)
  library(p, character.only = TRUE)
}
Outline

R Packages
  Package Concept
  Creating R packages
  Package Tools

Package Dependencies and Namespaces

Unit Testing

Documentation
  Manual Pages
  Vignettes

Tools of the trade
  Version Control
  Efficient Editing

Resources
Resources

- *Version Control with Subversion*, http://svnbook.red-bean.com/
- *Sweave User Manual*, http://www.stat.uni-muenchen.de/~leisch/Sweave