Authoring R Packages

Florian Hahne

Fred Hutchinson Cancer Research Center

Feb 13 2008 / Advanced R for Bioinformatics
Outline

1 Introduction
   The Package Concept
   Package Management

2 Package Authoring
   Package Structure
   Documentation
   Name Space

3 My First Package
Source of information

The complete documentation for managing and authoring of R packages is the *Writing R Extensions* manual.
http://cran.r-project.org/doc/manuals/R-exts.html

Guidelines for authoring Bioconductor packages can be found here:
http://wiki.fhcrc.org/bioc/Package_Guidelines
What is a Package?

• collection of functions and data structures addressing a particular problem
• structure to organize code and its documentation
• vehicle to distribute and share software
Purpose of a Package

• keep all pieces of the software together
• common installation process
• allow for shared development (versions)
• unified way to access documentation
• allow for quality control and testing
R Functions for Package Management

- **search()**
  ```
  [1] ".GlobalEnv"          "package:stats"
  [7] "package:methods"     "Autoloads"
  [9] "package:base"
  ```

- **.packages()**
  ```
  [1] "stats"    "graphics"  "grDevices" "utils"
  [5] "datasets" "methods"  "base"
  ```

- **system.file(package="tools")**
  ```
  [1] "/home/fhahne/R/bin/R-2-7-0/lib/R/library/tools"
  ```
R Functions for Package Management

- `packageDescription("base")`
  
  Package: base
  Version: 2.7.0
  Priority: base
  Title: The R Base Package
  Author: R Development Core Team and contributors worldwide
  Maintainer: R Core Team <R-core@r-project.org>
  Description: Base R functions
  License: GPL (>= 2)
  Built: R 2.7.0; ; Mi 6. Feb 18:04:55 PST 2008; unix

  -- File: /home/fhahne/R/bin/R-2-7-0/lib/R/library/base
R Functions for Package Management

- `download.packages()`
- `install.packages()`
- `update.packages()`
- `remove.packages()`
- `available.packages()`
- `old.packages()`
- `new.packages()`
- `installed.packages()`
A library stores a **collection of packages**

- **multiple libraries** \((\text{R\_LIBS})\)
  
  ```
  > .Library
  [1] "/home/fhahne/R/bin/R-2-7-0/lib/R/library"
  > .libPaths()
  [1] "/home/fhahne/R/source/R-2-7-0/localPackages"
  [2] "/home/fhahne/R/bin/R-2-7-0/lib/R/library"
  ```

- **version specific libraries** \((\text{R\_LIBS\_USER})\)

- **site specific libraries** \((\text{R\_LIBS\_SITE})\)
Folder Structure

On the system level a package is a collection of folders

- "./R": all native R code for the package
- "./man": the documentation for all (exported) code items
- "./src": foreign code
- "./data": data sets that can be loaded using `data()`
- "./inst/doc": additional documentation (Vignettes)
- (distribution specific subdirectories windows and unix for R and man)
DESCRIPTION File

The DESCRIPTION file contains the basic information about a package and settings for package installation and loading

- **mandatory fields** Package, Version, License, Description, Title, Author, Maintainer
- **fields stating dependencies**: Depends, Suggests, Imports
- **installation and loading settings**: Collate, LazyLoad, LazyData
- **biocViews**: terms in a controlled vocabulary to characterize the package
Package: Biobase
Title: Biobase: Base functions for Bioconductor
Version: 1.17.8
Author: R. Gentleman, V. Carey, M. Morgan, S. Falcon
Description: Functions that are needed by many other packages or which replace R functions.
Suggests: tkWidgets, ALL
Depends: R (>= 2.6.0), tools, methods, utils
Maintainer: Biocore Team c/o BioC user list <bioconductor@stat.math.ethz.ch>
License: The Artistic License, Version 2.0
          AllGeneric.R
          VersionsClass.R
          VersionedClasses.R methods-VersionsNull.R methods-VersionedClass.R
          DataClasses.R methods-aggregator.R methods-ANY.R
          methods-MIAME.R methods-annotatedDataset.R
          methods-eSet.R methods-ExpressionSet.R methods-MultiSet.R
          methods-SnpSet.R methods-NChannelSet.R
          anyMissing.R
          methods-ScalarObject.R
          zzz.R
LazyLoad: yes
biocViews: Infrastructure, Statistics
Tools to Create Package Structure

• `package.skeleton()` helps to build a basic package structure and templates for the documentation
• all objects in the working directory are added to the package
• this is just a template; there remains a lot for the user to do...
Code Organisation

Organisation of files in the \texttt{R} directory is up to the user, there are some \textbf{guidelines}:

- all class definitions in one file
- all generic function definitions in one file
- functions including all necessary helper functions into separate files or organize according to conceptual entities
Building Packages

There are a number of command line tools to build R packages:

- **R CMD build**: create `.tar.gz` from package source
- **R CMD check**: check documentation, code and run all examples and Vignettes
- **R CMD INSTALL**: install a package `.tar.gz`
Help Pages

documentation of functions in the **help pages**, of concepts and work flows in **Vignettes**.

help pages:

- show capabilities, inputs and outputs
- quality control: examples are run during `R CMD check`
functions of the `prompt` family can help create template documentation files:

- `prompt()`: functions, objects
- `promptPackage()`: package-related documentation
- `promptClass()`: S4 class documentation
- `promptMethods()`: S4 method documentation
.Rd format

Syntax similar to that of \LaTeX is used for the documentation files

- **special markup:** \kbd, \code, ...
- **links:**
  \begin{verbatim}
  \code{\link{foo}}
  \code{\link[pkg]{foo}}
  \code{\link[pkg:bar]{foo}}
  \end{verbatim}
Create a new ExpressionSet instance by selecting a specific channel

This generic function extracts a specific element from an object, returning a instance of ExpressionSet.

Usage

channel(object, name, ...)

Arguments

- object: An S4 object, typically derived from class
- name: A (length one) character vector channel names.
- ...: Additional arguments.

Value

Instance of class ExpressionSet.

Author

Biocore

Examples

```r
obj <- new("NChannelSet",
             R=matrix(runif(100), 20, 5),
             G=matrix(runif(100), 20, 5))
```

# G channel as ExpressionSet

```r
channel(obj, "G")
```
Vignettes

document integrating code and text that describes how to perform a specific task.

• created using Sweave()
• \LaTeX{} document
• special markup for code chunks

    <<chunkName, some options...>>=  
    R code...  
@

• and for inline notation \Sexpr{R expression}
Concept of Name Spaces

- control which symbols are visible and which values are used during evaluation
- control which functionality is exported
- prevent undeliberate shadowing of symbols in the search path
- reduce overhead of symbol lookup by distinguishing between package attachment and package loading
import("methods")
import("AnnotationDbi")
importFrom("annotate", "getAnnMap")
import("Biobase")
import("survival")
importFrom("graphics", "plot")

useDynLib("genefilter")

export("Anova", "allNA", "anyNA", "coxfilter",
     "cv", "eSetFilter", "varFilter", "featureFilter",
     "fastT", "ttest", "shorth", "half.range.mode",
     "rowttests", "colttests", "rowFtests", "colFtests",
     "rowSds", "rowVars", "dist2",
     "filterfun", "findLargest", "gapFilter",
     "genefilter", "genescale", "getFilterNames",
     "getFuncDesc", "getRdAsText", "isESet", "kOverA", "maxA", "pOverA",
     "parseArgs", "parseDesc", "setESetArgs", "showESet")

exportClasses("rowROC")
exportMethods("genefinder", "show", "plot", "[", "sens", "spec",
          "area", "pAUC", "AUC", "rowpAUCs", "nsFilter")
Let’s try this out and build our own first package...