Creación de paquetes de R

Alex Sanchez

UB

2013-07-15

Outline

Motivación

- ¿Porqué crear paquetes de R?
  - Nos obligará a
    - Pulir y estructurar el código, las funciones y los datos;
    - Documentar el trabajo y dar ejemplos claros.
  - Es la forma adecuada de compartir el trabajo.
  - Es la forma convenida de contribuir al crecimiento de R.
- ¿Porqué no habríamos de hacerlo?
  - Tedioso: El paquete tiene que verificarse y cumplir con los estándares.
  - Pesado: Puede ser algo más difícil adaptar un paquete que sus partes.

Objetivos del curso

- Aprender cual es la estructura de un paquete de R.
- Aprender a crear paquetes usando diversas aproximaciones
- Saber donde encontrar más información.

Materiales

- Manuales de referencia
  - Writing R Extensions
  - R Package Basics (Hadley Wickham)
  - Creating R Packages: A Tutorial (Friedrich Leisch)
  - Creación de paquetes con R studio
- Archivos de ejemplo
- Estas transparencias

Agradecimientos

- Francesc Carmona que creó nuestro primer "manualillo" y mantiene el blog de los erreros
- A los incansables redactores de tutoriales sobre este y tantos otros temas, especialmente
  - Oscar Perpiñan, que crea excelentes materiales y los comparte.
  - Hadley Wickam, que crea fantásticos paquetes y grandes presentaciones (127 en Slideshare)
- A los pacientes usuarios que esperan que se actualicen los paquetes y a los "pringados" que de verdad creen que no hace falta un paquete. Seguid así...

¿Qué herramientas se necesitan?

- Linux: Una instalación normal, preferiblemente con Latex.
- Windows
  A lo anterior añadir:
  - RTools (http://cran.r-project.org/bin/windows/Rtools/)
  - MikTex
- A través de Rstudio
  - Añadir el paquete `devtools()` a los anteriores

Antes de empezar-Algunos consejos genéricos

Extraído de Best Practices for Scientific Computing

- Write programs for people, not computers.
- Automate repetitive tasks
- Use the computer to record history
- Make incremental changes
- Use version control
- Don't repeat yourself (or others)
- Plan for mistakes
- Optimize software only after it works correctly
- Document design and purpose, not mechanics
- Collaborate

Outline

Pasos mínimos para la creación de un paquete

Los pasos necesarios para la creación de un paquete son

1. Creación de los objetos que contendrá el paquete (funciones y/o datos).
2. Creación del esqueleto del paquete
3. Redacción de la documentación. Retoques.
4. Compilación del paquete en Linux y creación de la versión para windows.
5. Instalación
6. Prueba y publicación.

Creación de los objetos (funciones, datos, . . .)

- Un paquete puede contener cualquier tipo de objetos de **R** : funciones, datos etc.
- Empezaremos programando las funciones y preparando los datos.
- El proceso de creación vigila que
  - no hayan errores sintácticos pero,
  - no controla si hay errores lógicos.
- El ejemplo que se presenta a continuación crea una función para resumir los datos de un data frame.

Ejemplo: función para una descriptiva básica.

```
resums <- function(dataFrame,fName){
  sink(paste(fName,"txt",sep="."))
  pdf(paste(fName,"pdf",sep="."))
  resumeixUna<-function(dataFr,colum){
   x<-dataFr[,colum]
   nomx<-names(dataFr)[colum]
   if (is.numeric(x)){
      hist(x,main=nomx); nomx; print(summary(x))
   }else{if (is.integer(x)){
      barplot(table(x),main=nomx);nomx;  print(summary(x))
    }else{if (is.factor(x)){
        plot(x,main=nomx); nomx; print(summary(x))}}}}
  for (i in 1:ncol(dataFrame))
    resumeixUna(dataFrame,i)
  dev.off()
  sink()
}
```

Estructura de directorios

- Todo paquete de R tiene que tener una estructura mínima
  - Un archivo `R/` con el código
  - Un archivo DESCRIPTION con información ("Metadatos") sobre el paquete
- Adicionalmente pueden existir muchos otros directorios con informaciones diversas
  - man/ directory where your function documentation.
  - the inst/CITATION file describes how to cite your package.
  - the inst/doc/ is used for larger scale documentation.
  - a NAMESPACE file describes which functions are part of the formal API of the package and are available for others to use.
  - tests/ and inst/tests/ contains unit tests.
  - the data/ contains .rdata files, used to include sample datasets.
  - y algunos más...

Creación de una estructura básica con `package.skeleton`

- Los directorios y archivos de información o documentación se pueden crear manualmente.
- R proporciona una función `package.skeleton` que permite automatizar el proceso.
  - La única información necesaria es el nombre del paquete y la lista de objetos que va a necesitar.
- Normalmente, para utilizar esta función procederemos...
  - Preparando el código que se desea incluir, ya sea desde la memoria o desde archivos
  - Invocando `package.skeleton` en la forma indicada a continuación.

package.skeleton en detalle

```
package.skeleton(name = "anRpackage", list,
        environment = .GlobalEnv,
        path = ".", force = FALSE,
        code_files = character())
```

- Los objetos se definen con *una de estas tres opciones*:
  - `list` character vector of names of objects
  - `environment` name of an environment e.g. .GlobalEnv
  - `code_files` character vector of names of source files
- Otros argumentos:
  - `name` required
  - `path` where to create the package directories
  - `force` whether to overwrite an existing set of directories.

Ejemplo

```
rm(list=ls())                              # Limpia la memoria
vas<- read.csv2 ("VASopenlap.csv") # Dataset para el paquete
source("resums.R")                         # Función principal
package.skeleton(name = "resums", list = "resums")
```

Ejercicio

- Arranca una sesión de R
- Inspecciona la ayuda de `package.skeleton`
- Carga tu función en memoria
- Crea el esqueleto de tu paquete

Estructura del esqueleto creado

El resultado de invocar la función `package.skeleton()` es una carpeta con una serie de directorios y archivos similares a los que se encuentran en cualquier paquete.

- Un archivo DESCRIPTION que debemos completar
- La carpeta `man` con archivos `.Rd` para la documentación.
- La carpeta `src` con el codigo que formará parte del paquete.
- La carpeta `R`, vacía, que contendrá el codigo compilado.
- Un archivo `Read-and-delete-me` *con instrucciones para completar el paquete*.

El archivo DESCRIPTION

- La información básica del paquete es fácil y directa de completar
- Otros componentes importantes
- `LazyData`: Si hay datos y se pone a `yes` no será preciso utilizar `data()` para cargralos.
- `Depends/Suggests/Imports`: Indica como se reutilzan otros paquetes o funciones de otros paquetes.

Depends/Suggests/Imports

Depends Comma-separated list of packages which are required to be attached in order to run your package. May include details of the version required: e.g.
Depends: R (>= 2.7.0), tcltk

Suggests Similar to Depends, but only necessary for examples and vignettes, so use of the package would be restricted but not impossible if these packages are unavailable.

Imports Lists packages whose name spaces are imported from using the import directive in the NAMESPACE file (or by using "::" or ":::" within the package), but which do not need to be attached. (":::" allows access to hidden objects.)

Documentación

- Las páginas de ayuda de los objetos `R` se escriben usando el formato "R documentation" (Rd), un lenguaje similar a LaTeX.
- Es aconsejable seguir estas orientaciones: Guidelines for Rd files
- Para generar el esqueleto de un fichero Rd es aconsejable usar:
  - `prompt`: genérica
  - `promptClass` y `promptMethods`: clases y métodos.
  - `promptPackage`: paquete
  - `promptData`: datos
- Todos los comandos disponibles están en el documento Parsing Rd files.

Retocando la documentación

- La carpeta `man` contiene varios archivos:
  - Uno con el nombre del paquete y
  - Uno para cada función que hayamos creado.
- Estos archivos, con extensión `.Rd` deben editarse para añadir documentación.
- Se trata de archivos *LateX-like*, pero apenas es preciso conocer este lenguaje porqué los lugares en donde debemos escribir estan indicados por comentarios.
- La documentación que redactemos será la que aparezca al invocar la ayuda del paquete.

Ejercicio

- Abrir la carpeta `man`
  - Observad los distintos tipos de ayuda
  - Completarlos cuidadosamente.
- Editar el archivo DESCRIPTION
  - Completadlo con los datos del paquete

Exportacion de objetos de los espacios de nombres

- R creará automáticamente un espacio de nombres (*NAMESPACE*) para el paquete.
- Cuando se carga un paquete con `library()`, sólo se adjuntan los objetos exportados, lo que los hace directamente visibles
  - *Exportemos únicamente los objetos que deseamos hacer visibles!*
- Conviene ser cuidadoso paa evitar tener que escribir ayudas innecesarias (debe de hacerse para cada objeto que se exporte).
- Para exportar dos objetos a, b utilizaremos:

```
export (a, b)
```

- Para exportar todos los elementos que se ajusten a un patrón utilizaremos:

```
exportPattern ("^ [^ \ \.]")
```

que lo exportará todo excepto si comienza con un punto.

Otros elementos del espacio de nombres (*NAMESPACE*)

import If you wish to use an item from a name space in another package, you can import it.
`import(pkg1, pkg2)` will import all items from packages pkg1 and pkg2.
`importFrom(pkg1, a, b)` will import just items a and b from package pkg1

Generic function If your package includes a function intended to be used as a generic function, this should be indicated in the NAMESPACE file using an S3method statement: e.g. `S3method(print, myclass)`

Outline

Construcción del paquete

- Una vez hemos creado un paquete (que llamamos *miPaquete*)
  - Hemos completado los archivos DESCRIPTION y NAMESPACE...
  - Hemos redactado las ayudas...
  - Hemos probado y depurado las funciones...
- Nos situamos en el directorio de donde cuelga *miPaquete* y...
  - Para comprobar que "todo está correcto" escribimos:

    ```
    R CMD check  miPaquete
    ```

  - Para compilarlo y crear un archivo instalable escribimos:

    ```
    R CMD build  miPaquete
    ```

  - Para instalarlo en nuestro sistema escribimos:

    ```
    R CMD INSTALL miPaquete
    ```

Comprobar

- Comprobar un directorio (desde línea de comandos):

```
R CMD check miPaquete/
```

- Comprobar un paquete ya construido (desde línea de comandos):

```
R CMD check miPaquete.tar.gz
```

- Esta comprobación incluye más de 20 puntos de prueba detallados en el manual Writing R extensions.

Construir

- Fuente o binario
  Se puede construir un fichero fuente en formato *tarball* (independiente de la plataforma, habitual en sistemas Unix) o en forma binaria (dependiente de la plataforma, habitual para Windows y Mac).
- Cómo hacerlo
  - Fuente en formato *tarball*
    - El resultado es un fichero *tarball* `miPaquete.tar.gz` que se puede distribuir a cualquier sistema.

```
R CMD build miPaquete/
```

  - Comprimido binario
    - El resultado es una copia comprimida de la versión **instalada** del paquete: depende del sistema operativo.

```
R CMD INSTALL --build miPaquete/
```

Comprobar y construir en sistemas Windows

- Para paquetes sin código compilado (C, Fortran), también se puede usar `R CMD check` y `R CMD build` en un sistema Windows.
- Para generar un binario hay que usar `R CMD INSTALL -build`.
  - Es posible que haya que modificar la variables de entorno `TEMP` y `TMP` de forma que **sólo** contengan caracteres ASCII.
- Para paquetes con código compilado, o en caso de problemas con los comandos anteriores, hay que usar Rtools.
- Se pueden instalar fuentes *tarball* con (ver R installation and administration):

```
install.packages(miPaquete.tar.gz, type='source')
```

Repositorios

- El principal repositorio de paquetes estables es CRAN.
  - Publicar en este repositorio conlleva la aceptación de unas condiciones.
  - Para publicar en CRAN hay que subir el fichero fuente *tarball* resultado de `R CMD build` via FTP anónimo a la dirección ftp://CRAN.R-project.org/incoming/ y enviar un correo en texto plano a CRAN@R-project.org.
  - Es imprescindible haber comprobado el fichero con `R CMD check -as-cran` antes de subirlo al servidor FTP. El resultado de esta comprobación **no** debe contener errores ni advertencias (`warnings`).
  - Más detalle en el apartado Submission de las condiciones de CRAN y en el manual R Extensions.

Repositorios

- Otros repositorios destacables son:
  - R-Forge (versiones de desarrollo)
  - Bioconductor (paquetes de bioinformática)
  - Proyectos R en Github (versiones de desarrollo)
  - RForge (versiones de desarrollo)