Creare packages per  sotto Windows XP

Fabio Frascati[1]

30 settembre 2006

[1]Fabio Frascati, Laurea in Statistica e Scienze Economiche conseguita presso l'Università degli Studi di Firenze, fabiofrascati@yahoo.it

Indice

1 Introduzione

Questo breve articolo vuol essere una guida schematica alla creazione di *packages* per il *software* **R**. Il sistema operativo considerato è *Windows XP*. Ci sono almeno tre buone ragioni per scrivere un *package*:

1. Essere obbligati a documentare il proprio codice sorgente significa assicurarsi che questo funzioni correttamente perché testato certamente più volte. Una volta creata la documentazione necessaria (e caricato il *package*), il semplice comando `help(topic="nomecomando")` o `?"nomecomando"` dalla *console* di **R** permetterà di visionarla immediatamente

2. E' un modo per poter rendere disponibile a tutti il proprio lavoro e questo incrementerà la probabilità che sia realmente privo di errori. Inoltre sarà più facile maturare nuove idee di ricerca a partire dai commenti e dalle osservazioni degli altri

3. Riunire più funzioni sotto un'unica libreria consente di poter gestire il proprio lavoro in maniera pratica ed efficiente

Chiunque avesse suggerimenti o volesse segnalare errori ed imprecisioni è pregato di farlo al mio indirizzo email: fabiofrascati@yahoo.it.

2 Cosa è necessario

Munirsi dei seguenti componenti:

1. **tools.zip** Questo *file* racchiude una serie di *Unix utilities*. Esso è frutto del lavoro di *Brian Ripley* e *Duncan Murdoch* ed è disponibile all'indirizzo *Web* http://www.murdoch-sutherland.com/Rtools/tools.zip oppure http://www.stats.ox.ac.uk/pub/Rtools/tools.zip

2. **Perl** Il *Perl* è un linguaggio di *scripting*. Selezionare la versione per *Windows XP* all'indirizzo *Web* http://www.activestate.com/Products/ActivePerl/Download.html

3. **Microsoft html help compiler** Il compilatore di *files* con estensione *.chm* si trova all'indirizzo *Web* http://www.microsoft.com/office/ork/xp/appndx/appa06.htm oppure http://www.msdn.microsoft.com/library/en-us/htmlhelp/html/hwmicrosofthtmlhelpdownloads.asp

4. **LaTeX 2ε** La versione più recente di LaTeX 2_ε si trova all'indirizzo *Web* http://www.miktex.org/

5. **Rd.sty** Questo è il *file* di stile (*.sty*) che rappresenta la standard per il manuale in formato *PDF* che accompagna il *package*. E' disponibile all'indirizzo *Web* http://stuff.mit.edu/afs/sipb/project/r-project/lib/R/share/texmf/Rd.sty

6. **WinZip** Programma *shareware* di compressione (http://www.winzip.com/). Un'alternativa completamente *freeware* e disponibile in italiano è **7-ZIP** (http://www.7-zip.org)

Si assume l'installazione di **R** (http://www.cran.r-project.org/) nella sua ultima versione (2.3.1). L'intero articolo considera `C:\Programmi\R\R-2.3.1` come *home directory*. Per conoscere la propria *home directory* digitare da *console* il comando `R.home()`.

3 Primi passi

Queste istruzioni sono presenti sull'*Home Page* di *Duncan Murdoch* (visionabile all'indirizzo *Web* http://www.murdoch-sutherland.com/Rtools/) ma vengono riportate qui di seguito per motivi di praticità.

1. *tools.zip*. Salvare questo *file* in una *directory* che supponiamo essere *C:\Rtools*. Cliccare con il tasto destro del mouse su *tools.zip* e scegliere la voce *WinZip - Extract to here*

2. *Perl*. Salvare il *file* con estensione *.zip* in una *directory* che supponiamo essere *C:\Perl*. Cliccare con il tasto destro del mouse su *.zip* e scegliere la voce *WinZip - Extract to here*. Il *file* con estensione *.zip* a cui si riferisce il presente manuale ha nome *ActivePerl-5.8.8.817-MSWin32-x86-257965.zip*

3. *Microsoft html compiler*. Questo sarà adoperato per creare i *files* di *help* in *html* compilato. Dopo averne effettuato il *download*, seguire le istruzioni di default per l'installazione

4. LaTeX 2_ε. Dopo averne effettuato il *download*, seguire le istruzioni di default per l'installazione

5. *Rd.sty*. Copiare il *file* di stile *Rd.sty* nella *directory* `C:\texmf\tex\latex\base` ed effettuare il *refresh* di *MiKTeX* (*Start - Programmi - MiKTeX - MiKTeX Options - Refresh now*)

6. *Variabile d'ambiente Path*. Selezionare il percorso *Start - Impostazioni - Pannello di controllo*. Cliccare due volte su *Sistema* e scegliere la scheda *Avanzate*. Cliccare su *Variabili d'ambiente*. E' necessario modificare la variabile di sistema denominata *Path*. Evidenziare la variabile e selezionare il pulsante *Modifica*. Portarsi con la freccia di sinistra all'inizio della stringa in corrispondenza della casella di testo *Valore variabile*. Scrivere (facendo attenzione di essere posizionati all'inizio della stringa) non preoccupandosi di distringuere tra lettere minuscole e maiuscole. Digitare il seguente codice sulla linea senza interruzione:

   ```
   C:\Rtools\tools\bin;C:\Perl\ActivePerl-5.8.8.817-MSWin32-x86-257965\perl\bin;
   C:\Programmi\R\R-2.3.1\bin;C:\Programmi\HTML Help Workshop;
   ```

Confermiamo digitando per tre volte il tasto *OK* e verifichiamo che tutto sia stato installato correttamente. Apriamo la finestra del *DOS* (*Start - Programmi - Accessori - Prompt dei comandi* oppure *Start - Esegui - cmd - OK*) e digitiamo il comando *path* sul *prompt*. Controllare che i percorsi visualizzati corrispondano a quelli effettivi dei componenti appena installati. Se questo non risulta, chiudere la finestra del *DOS*, ritornare al precedente punto *Variabile d'ambiente Path* e riprovare fino ad esito raggiunto.

4 La struttura base (package framework)

Il comando `package.skeleton()` crea una *directory* contenente la struttura base (*package framework*) di un *package* per **R**. Questa cartella ha lo stesso nome del *package* che si intende realizzare (ad esempio **prova**) ed è costituita dai seguenti elementi:

- **Read-and-delete-me** breve lista dei passi necessari per costruire un *package*
- **DESCRIPTION** descrizione del *package*
- **man** *subdirectory* che contiene i *files* di *help* per ogni oggetto di **R** che si è deciso di includere nel *package*. Questi *files* sono in formato *.Rd* (*R documentation*)
- **R** *subdirectory* che contiene i *files* di *source code* (*.R*) per ogni oggetto di tipo funzione
- **data** *subdirectory* che contiene i *files* di *source code* (*.rda*) per ogni oggetto di tipo data frame
- **src** *subdirectory* che contiene i *files* di *source code* per linguaggi *Fortran, C, C++, ...*

La funzione `package.skeleton()` ha i seguenti argomenti:

- `name` stringa con il nome del *package* e della cartella che contiene la struttura base di questo
- `path` percorso in cui inserire la cartella contenente la struttura base del *package*. Si consiglia di adoperare `C:\Programmi\R\R-2.3.1\src\library` e dunque scrivere:

  ```
  path=c("C:\\Programmi\\R\\R-2.3.1\\src\\library")
  ```

- `list` vettore di stringhe ognuna corrispondente al nome di un oggetto di **R** da inserire nel *package*

- `force` valore logico che indica se è consentito sovrascrivere (`TRUE`) oppure no (`FALSE`) una *directory* già esistente. Per default vale `force = FALSE`

Se si omette il parametro `list`, tutti gli oggetti contenuti nel *workspace* entreranno a far parte del *package*. Per comodità si può allora iniziare a definire tutti gli oggetti da inserire nel *package* solo dopo aver ripulito il *workspace* (con il comando `rm(list=ls())`). Così facendo possiamo omettere `list` come parametro del comando `package.skeleton()`. La struttura base per la costruzione del *package* è terminata. Adesso occorre proseguire con la descrizione del *package* e la documentazione dei singoli oggetti (funzioni e data frame). I *files* con estensione *.Rd* possono essere editati attraverso *Blocco note* o *WordPad*.
Un esempio di utilizzo del comando `package.skeleton()` è dato dal seguente codice **R**:

```
> pippo<-function(x,y) x+y
> pluto<-function(x,y) x-y
> topolino<-function(x,y) x*y
> ls()
[1] "pippo"    "pluto"    "topolino"
> percorso<-c("C:\\Programmi\\R\\R-2.3.1\\src\\library")
> package.skeleton(name="prova",path=percorso,list=c("pippo","pluto"),force=FALSE)
Creating directories ...
Creating DESCRIPTION ...
Creating Read-and-delete-mes ...
Saving functions and data ...
Making help files ...
Created file named 'C:\PROGRAMMI\R\R-2.3.1\src\library/prova/man/prova-package.Rd'.
Edit the file and move it to the appropriate directory.
Created file named 'C:\PROGRAMMI\R\R-2.3.1\src\library/prova/man/pippo.Rd'.
Edit the file and move it to the appropriate directory.
Created file named 'C:\PROGRAMMI\R\R-2.3.1\src\library/prova/man/pluto.Rd'.
Edit the file and move it to the appropriate directory.
Done.
Further steps are described in C:\PROGRAMMI\R\R-2.3.1\src\library/prova/Read-and-delete-me
```

E' equivalente scrivere `percorso<-c("C:/Programmi/R/R-2.3.1/src/library")` nel codice sopra. Adesso è possibile eliminare tutti i *files* nominati *Read-and-delete-me*. Uno di questi è contenuto in **prova** e gli altri due nelle sottocartelle **man** e **src**. E' consigliabile eliminare anche il *file* di nome *prova-package.Rd* nella sottocartella **man** perchè il suo contenuto è identico a quello del *file* di *DESCRIPTION* presentato nella successiva sezione. La cartella **data** viene creata solo se il *package* contiene almeno un oggetto di tipo data frame. La sottocartella **src** può essere direttamente eliminata se il *package* non adopera linguaggi del tipo *Fortran, C, C++, ...*

5 Modifica del file di DESCRIPTION

Il *file* di *DESCRIPTION* contiene le informazioni di base che caratterizzano il *package*. Questo esempio appartiene proprio al *package* **prova**:

```
Package: prova
Type: Package
Title: What the package does (short line)
Version: 1.0
Date: 2006-08-27
Author: Who wrote it
Maintainer: Who to complain to <yourfault@somewhere.net>
Description: More about what it does (maybe more than one line)
License: What license is it under?
```

Alcune osservazioni:

1. Il *file* può essere editato con *Blocco note* o *WordPad*
2. Le voci facoltative sono solo `Type:` e `Date:`
3. Se una frase continua sulla riga successiva occorre usare *TAB* o *SPACE*
4. La versione del *package* è una serie di interi separati da un punto (.) o da un segno meno (-). Un esempio è dato da `Version: 1.1.2` oppure `Version: 1.1-2`
5. E' possibile inserire la voce `Depends:`. Questa serve a specificare la versione di **R** a partire dalla quale è lecito adoperare il *package*. Attraverso `Depends:` si può anche dichiarare la dipendenza da altri *packages*. Un esempio è dato da `Depends: R (>= 2.0.1), coda (>= 0.9-2), MASS` oppure `Depends: R (>= 2.0.1)`
6. E' possibile inserire la voce `URL:`. Questa serve a specificare un indirizzo *Web* in cui reperire maggior documentazione riguardo il *package*. Un esempio è dato da `URL: http://www.packageprova.it` oppure `URL: http://www.address1.it, http://www.address2.com`
7. Di solito la voce `License:` viene definita in uno dei seguenti modi:
   - `License: GPL`
   - `License: GPL (version 2 or later)`
   - `License: GPL Version 2 or newer`

Maggiori informazioni su come editare il *file* di *DESCRIPTION* si trovano nella guida ufficiale "Writing R Extensions" sul sito del *CRAN* [8]. Una volta installato il *package* `prova`, è possibile richiamare le informazioni contenute nel *file* di *DESCRIPTION* direttamente dalla *console* di **R**. Il comando da utilizzare in questo caso è `packageDescription(pkg="prova")`. Se si desidera visualizzare anche la lista delle funzioni contenute nel *package*, utilizzare `library(help=prova)` oppure `help(package=prova)`.

6 Modifica dei files .Rd

La sottocartella **man** contiene un *file* di documentazione (*.Rd)* per ogni oggetto (funzione o data frame) che si è deciso includere nel *package*. Il seguente esempio si riferisce alla funzione **pippo**:

```
\name{pippo}
\alias{pippo}
%- Also NEED an '\alias' for EACH other topic documented here.
\title{ ~~function to do ... ~~ }
\description{
  ~~ A concise (1-5 lines) description of what the function does. ~~
}
\usage{
pippo(x, y)
}
%- maybe also 'usage' for other objects documented here.
\arguments{
  \item{x}{ ~~Describe \code{x} here~~ }
  \item{y}{ ~~Describe \code{y} here~~ }
}
\details{
  ~~ If necessary, more details than the description above ~~
}
\value{
```

```
  ~Describe the value returned
  If it is a LIST, use
  \item{comp1 }{Description of 'comp1'}
  \item{comp2 }{Description of 'comp2'}
  ...
}
\references{ ~put references to the literature/web site here ~ }
\author{ ~~who you are~~ }
\note{ ~~further notes~~

 ~Make other sections like Warning with \section{Warning }{....} ~
}
\seealso{ ~~objects to See Also as \code{\link{help}}, ~~~ }
\examples{
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function(x,y) x+y
}
\keyword{ ~kwd1 }% at least one, from doc/KEYWORDS
\keyword{ ~kwd2 }% __ONLY ONE__ keyword per line
```

Uno qualunque tra questi *files* di *help* nella cartella **man** contiene quindi varie voci che devono essere editate. Le prime due (per un oggetto funzione) sono:

- \name{} nome del comando che appare in alto a sinistra nella pagina di *help* in formato *html*.
- \alias{} consente di inserire i nomi di tutti gli oggetti che si desidera siano documentati nel *file* attuale. E' quindi possibile creare un solo *file* di *help* per documentare più oggetti. Ad esempio:

  \alias{pippo}
  \alias{pippo.default}
  \alias{pippo.formula}

- \title{} titolo informativo sul comando
- \description{} breve descrizione sull'utilità del comando
- \usage{} sintassi del comando
- \arguments{} lista degli argomenti del comando
- \details{} descrizione dettagliata del comando che integra quanto riportato nella voce \description{}
- \value{} lista dei valori di ritorno del comando
- \references{} voce per i riferimenti bibliografici. Adoperare \url{} per materiale disponibile sul *Web*
- \note{} sezione dedicata a note ed osservazioni
- \author{} informazione riferita all'autore del *file .Rd*. Adoperare \email{} senza i delimitatori (<> o ()) oppure \url{} per riferimenti sul *Web*
- \seealso{} puntatori ad altri comandi di **R**. Adoperare \link{\code{...}} per riferirsi a questi.

- \examples{} esempi direttamente richiamabili dalla *console* di **R**. Alcune utili funzioni da adoperare all'interno della voce \examples{}:

  \dontrun{} comandi da mostrare ma non eseguire (esempio: \dontrun{plot(x)})

  \dontshow{} comandi da eseguire ma non mostrare (esempio: \dontshow{log(x)})

- \keyword{} contiene una tra le parole chiave standard. Queste sono indicate nel *file KEYWORDS.db* nella cartella C:\Programmi\R\R-2.3.1\doc\KEYWORDS.db. La sezione denominata "Statistics" contiene le seguenti parole chiave con il relativo argomento:

datagen	Functions for generating data sets
distribution	Probability Distributions and Random Numbers
univar	simple univariate statistics # != S
htest	Statistical Inference
models	Statistical Models
models|regression	Regression
models|regression|nonlinear	Non-linear Regression # only?
robust	Robust/Resistant Techniques
design	Designed Experiments
multivariate	Multivariate Techniques
ts	Time Series
survival	Survival Analysis
nonparametric	Nonparametric Statistics # w/o 'smooth'
smooth	Curve (and Surface) Smoothing
smooth|loess	Loess Objects
cluster	Clustering
survey	Complex survey samples
tree	Regression and Classification Trees

  Inserire nella documentazione una parola chiave consente alla funzione associata di poter essere rintracciata. Supponiamo che la funzione **pippo** contenga \keyword{univar} nel proprio *file* di *help*. Se dalla *console* di **R** digitiamo il comando help.search(keyword="univar"), la lista di funzioni restituita a video comprenderà la funzione **pippo**. Da ricordare che solo una parola chiave per linea è ammessa:

```
\keyword{univar}
\keyword{models|regression}
\keyword{ts}
```

In aggiunta alle sezioni predefinite (\name{}, \alias{}, \title{}, ...), è possibile inserire anche voci personalizzate del tipo \section{titolo sezione}{...}. Per esempio:

```
\section{Warning}{Occorre prestare attenzione al fatto che ...}
```

Inserire la sezione addizionale prima della voce \note{} oppure \seealso{}. Alcuni utili comandi di formattazione per i *file .Rd* sono di seguito elencati:

- \cr consente di andare a capo (comando di nuova linea)
- \R consente di evidenziare il nome del *software*
- \code{} consente di applicare il *font* di tipo *Typewriter* all'argomento in parentesi graffe
- \bold{} consente di applicare il *font* di tipo *Bold* all'argomento in parentesi graffe
- \emph{} consente di applicare il *font* di tipo *Italic* all'argomento in parentesi graffe

- `\sQuote{}` consente di mettere tra apici l'argomento in parentesi graffe
- `\dQuote{}` consente di mettere tra doppi apici l'argomento in parentesi graffe
- `\link[stats]{cor}` consente un *link* a *cor.html* (comando `cor()` del *package* `stats`)
- `\link[stats:cor]{var}` consente un *link* a *cor.html* (comando `var()` del *package* `stats`)
- `\url{}` consente di accedere alla URL inserita in parentesi graffe
- `\file{}` consente di scrivere il nome di un *file*
- `\acronym{}` consente di scrivere un acronimo (esempio GNU)
- `\eqn{}` scrivere le formule matematiche in linea
- `\deqn{}` scrivere le formule matematiche in centro pagina

Si può inserire un commento nella pagina di *help* dopo il simbolo di percentuale (`%`). Il resto della linea verrà in questo modo completamente ignorato. Il comando `\concept{}` permette di inserire un riferimento per un indice concettuale. Se aggiungiamo nel *file pippo.Rd* la voce:

```
\concept{concetto}
```

possiamo rintracciare la funzione documentata in maniera pratica e veloce. Digitiamo da *console* il comando:

```
> help.search(pattern="concetto")
```

ed avremo a video una lista di funzioni comprensiva di **pippo**. Le pagine di documentazione per i data frame seguono una struttura simile a quella vista per gli oggetti funzione. Maggiori dettagli su come editare *files* di documentazione si possono trovare facilmente in letteratura [4, 5, 7, 8].

7 Verifica dei files .Rd

Scrivere i *files* di documentazione richiede pazienza ed esercizio. Due sono i consigli da seguire:

1. Eseguire la documentazione di una funzione alla volta in modo da concentrarsi su di un solo oggetto
2. Creare il *file .Rd* in maniera incrementale e non ambire a pagine di documentazione molto elaborate. Ogni incremento apportato deve essere verificato (anche più di una volta)

Aprire la finestra del *DOS*. Seguendo il nostro esempio scriviamo:

```
cd C:\Programmi\R\R-2.3.1\src\library\prova\man
```

Supponiamo di voler verificare il *file* con nome *pippo.Rd*. Scriviamo sul *prompt* del *DOS*:

```
R CMD Rdconv -t=html -o=pippo.html pippo.Rd
```

Questo produrrà il *file pippo.html* nella cartella `C:\Programmi\R\R-2.3.1\src\library\prova\man`. Usare un *browser* per visualizzarlo e controllare che tutto funzioni correttamente. In caso affermativo ripetere la procedura per ogni altro oggetto documentato.

8 Aggiunta di una singola funzione al package framework

Se una singola funzione deve essere aggiunta al *package framework*, occorre posizionare i relativi *files .R* e *.Rd* nelle appropriate *subdirectories*. Supponiamo di voler aggiungere la funzione **minni** al *package* **prova** direttamente dalla *console* di **R**. Definiamo innanzitutto la funzione **minni**:

```
> minni<-function(x,y) x**2+y**2
```

Il *file minni.R* viene creato e salvato nella sottocartella **R** attraverso il comando `dump()`:

```
> dump(list="minni",file="C:/Programmi/R/R-2.3.1/src/library/prova/R/minni.R")
```

Per documentare la funzione **minni** occorre il comando `prompt()`. Con questo possiamo creare il *file minni.Rd* e salvarlo nella sottocartella **man**:

```
> prompt(object=minni,filename="C:/Programmi/R/R-2.3.1/src/library/prova/man/minni.Rd")
```

9 Documenti aggiuntivi al package framework

In aggiunta ai *files .Rd* di *help*, è possibile includere documenti in altri formati. Il consiglio è quello di adoperare il più possibile il formato *PDF*. Questo consente di poter leggere il documento su tutte le piattaforme e in maniera assai veloce.
La locazione per i *files* aggiuntivi di documentazione è la nuova sottocartella **doc** della cartella **inst**. Quest'ultima deve essere creata e posizionata nel percorso `C:\Programmi\R\R-2.3.1\src\library\prova`. Una volta installato il *package* **prova**, è possibile accedere alla documentazione aggiuntiva attraverso **Aiuto - Guida Html** - *Packages* - *prova* - *directory*.

10 Creazione e verifica del package

Dalla finestra del *DOS* digitare il seguente codice:

```
cd C:\Programmi\R\R-2.3.1\src\library
```

Proseguire con il comando:

```
R CMD check prova
```

`R CMD check` creerà il *package* **prova** in una nuova cartella chiamata **prova.Rcheck** che si trova nel percorso `C:\Programmi\R\R-2.3.1\src\library\prova.Rcheck`. La cartella **prova.Rcheck** contiene anche altri *files* che sono il resoconto di numerose verifiche. `R CMD check` testa la corretta installazione del *package* e che nel *file* di *DESCRIPTION* non sia stato omesso niente. Altre verifiche riguardano il fatto che ad ogni oggetto corrisponda un *file* di *help* e che quest'ultimo sia stato editato senza errori. Per ulteriori informazioni riguardo tutti controlli effettuati, consultare il *file 00check.log* presente nella stessa cartella **prova.Rcheck**. In quest'ultima è presente anche una copia del manuale del *package* (in formato LaTeX 2ε) dal nome *prova-manual.tex*.
Per trasferire il *package* ad altri *computers* possiamo convertirlo in formato *.zip*. Cliccare con il tasto destro del mouse su **prova** e scegliere la voce *WinZip - Add to prova.zip*.

11 Creazione immediata del package in formato compresso (zip)

Dalla finestra del *DOS* digitare il seguente codice:

```
cd C:\Programmi\R\R-2.3.1\src\library
```

Proseguire con il comando:

```
R CMD build --binary --use-zip prova
```

`R CMD build --binary --use-zip` crea un *file* con estensione *.zip* che costituisce direttamente il *package* in formato compresso. Il *file prova_1.0.zip* (*1.0* è la versione del *package* indicata nel *file* di *DESCRIPTION*) si trova nella *directory* `C:\Programmi\R\R-2.3.1\src\library`.

12 Installazione del package

Possiamo installare *prova.zip* in **R** tramite la voce di menù: **Pacchetti - installa pacchetti da file zip locali...**. Posizionarsi sulla *directory* dove è memorizzato *prova.zip* e confermare con un doppio click. Un metodo alternativo prevede l'utilizzo della finestra *DOS*. Digitare il seguente codice:

```
cd C:\Programmi\R\R-2.3.1\src\library
```

Proseguire con il comando:

```
R CMD INSTALL prova
```

`R CMD INSTALL` creerà la cartella **prova** nel percorso `C:\Programmi\R\R-2.3.1\library`. Un metodo alternativo consiste nel digitare il comando:

```
> install.packages(pkg="C:/Programmi/R/R-2.3.1/src/library/prova_1.0.zip",repos=NULL)
```

dalla *console* di **R**.

13 Caricamento del package

Per caricare il *package* in **R** utilizzare il comando `library(package=prova)` oppure `require(package=prova)` dalla *console*. Un metodo alternativo consiste nello scegliere la voce **Pacchetti - Carica pacchetto...** dal menù a tendina.

14 Manuale del package in formato PDF

Dalla finestra del *DOS* digitare il seguente codice:

```
cd C:\Programmi\R\R-2.3.1\src\library
```

Proseguire con il comando:

```
R CMD Rd2dvi --pdf prova
```

`R CMD Rd2dvi --pdf` crea un *file* con estensione *.pdf* che costituisce il manuale di riferimento al *package* `prova`. Il manuale *prova.pdf* si trova nella *directory* `C:\Programmi\R\R-2.3.1\src\library`. Se si desidera visionare il *file latex* che genera il manuale in formato *pdf*, basta scrivere:

```
R CMD Rd2dvi --pdf --no-clean prova
```

Ciò che cerchiamo è contenuto nella cartella **.Rd2dvi**. E' possibile anche modificare il titolo del documento (per default `Package 'prova'`) attraverso il comando:

```
R CMD Rd2dvi --pdf --title=MioTitolo prova
```

15 Rimozione di un package installato

Dalla finestra del *DOS* digitare il seguente codice:

```
cd C:\Programmi\R\R-2.3.1\src\library
```

Proseguire con il comando:

```
R CMD REMOVE prova
```

`R CMD REMOVE` rimuoverà la cartella **prova** dal percorso `C:\Programmi\R\R-2.3.1\library`. Un metodo alternativo consiste nel digitare il comando:

```
> remove.packages(pkgs="prova")
```

dalla *console*. Riavviare **R** e digitare il comando `library()`. Il *package* **prova** non figura più nella lista dei *packages* da installare.

16 Sottoporre un package al CRAN

Dalla finestra del *DOS* digitare il seguente codice:

```
cd C:\Programmi\R\R-2.3.1\src\library
```

Proseguire con il comando:

```
R CMD build prova
```

`R CMD build` crea un *file .tar.gz* nel percorso `C:\Programmi\R\R-2.3.1\src\library`. E' possibile sottoporlo al *CRAN* all'indirizzo `ftp://cran.r-project/org/incoming/` (meglio se accompagnato da un avviso alla *mail* cran@r-project.org).

Riferimenti bibliografici

[1] V. Carey. An Introduction to the R package mechanism. Published on the URL: http://www.biostat.harvard.edu/courses/individual/bio271/lectures/L6/Rpkg.pdf, 2002.

[2] R. Irizarry. R Packages. Published on the URL: http://www.biostat.jhsph.edu/~bcaffo/statcomp/files/rpacks.pdf, 2003.

[3] Angelo M. Mineo. Una guida all'utilizzo dell'ambiente statistico R. Published on the URL: http://www.cran.r-project.org/doc/contrib/Mineo-dispensaR.pdf, 2003.

[4] D. Murdoch. Bulding R for Windows. Published on the URL: http://www.murdoch-sutherland.com/Rtools/, 2005.

[5] P. Rossi. Making R Packages Under Windows. Published on the URL: http://gsbwww.uchicago.edu/fac/peter.rossi/research/bayes%20book/bayesm/Making%20R%20Packages%20Under%20Windows.pdf, 2006.

[6] F. Schaarschmidt. Simple Creation of R packages under Windows. Published on the URL: http://www.bioinf.uni-hannover.de/teaching/fallstudien/schaarschmidt2.pdf, 2004.

[7] R Development Core Team. Guidelines for Rd files. Published on the URL: http://developer.r-project.org/Rds.html, 2005.

[8] R Development Core Team. Writing R Extensions. Published on the URL: http://www.cran.r-project.org/doc/manuals/R-exts.pdf, 2006.