

# Costruire un package R con RStudio



Data Analytics & Cognitive Solutions

08 – 05 – 2019

Nicola Procopio

# Obiettivi e pubblico di riferimento

La guida si pone come obiettivo quello di essere un memorandum molto snello della procedura per creare un package in R, conscio che esistono guide ottime e approfondite oltre che tutorial video.

Il lettore di riferimento è lo sviluppatore R, ovvero chi già conosce il linguaggio e lo utilizza abitualmente per creare delle funzioni.

Il package può servire per consegnare un prodotto al cliente che non sia uno script, ma una libreria completa di esempi e documentazione, oppure per rendere più ordinato il lavoro, ...

Nelle slide seguenti verrà mostrato come creare un package in 9 semplici step.

# Step 1: installare i package

Per poter costruire dei package la prima cosa è installare i package:

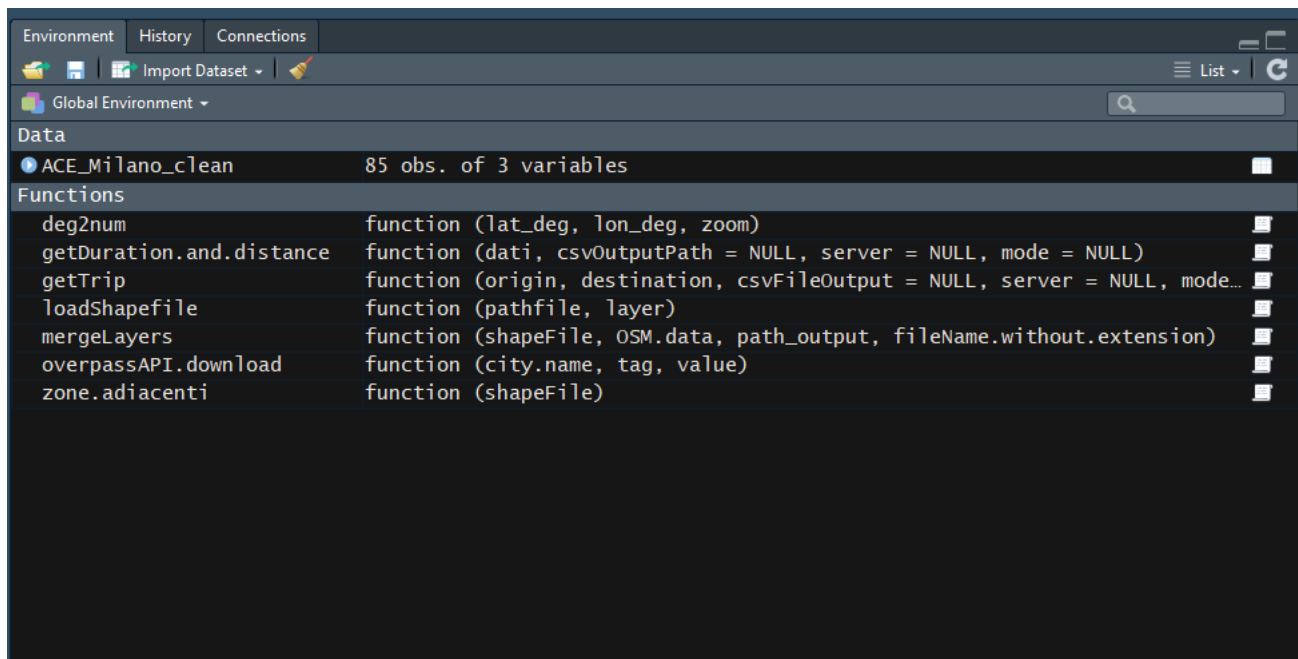
- devtools
- roxygen2
- latexpdf

Inoltre bisogna avere installato LaTeX [<http://www.lorenzopantieri.net/LaTeX.html>] sul proprio PC, consiglio di installare direttamente MikTeX [<https://miktex.org/>]

## Step 2: caricare la workspace

Una volta implementate le funzioni e i dati che vogliamo inserire nella nostra libreria li carichiamo sulla workspace di RStudio.

Il mio consiglio è di creare due versioni degli script, una commentata che conserveremo (per «riprendere» il codice in caso di eventuali aggiornamenti in futuro) e una pulita dai commenti da caricare nella workspace, così di ridurre al minimo gli errori dovuti ai caratteri non ASCII in fase di check del package

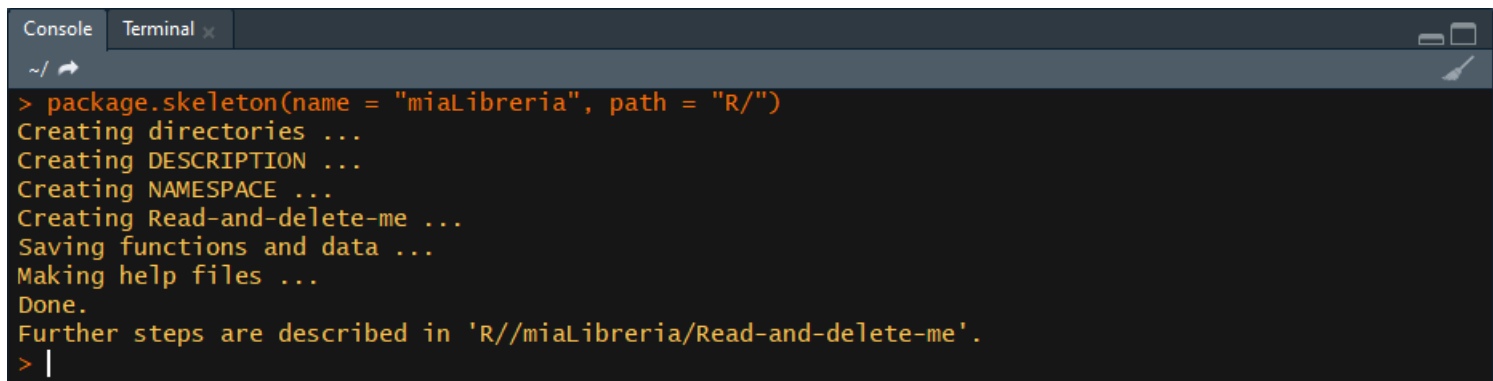


## Step 3: package.skeleton

Il comando `package.skeleton()` prende tutto quello che sta nella workspace e crea una cartella con la struttura di una libreria R.


I parametri (minimi) da dare sono il nome della libreria, la cartella dove inserire quanto creato:


- **`package.skeleton(name = "miaLibreria", path = "R/")`**

A screenshot of a terminal window with a dark background. The title bar shows 'Console' and 'Terminal x'. The prompt is '~/'. The user enters the command: `> package.skeleton(name = "miaLibreria", path = "R/")`. The terminal outputs the following steps: `Creating directories ...`, `Creating DESCRIPTION ...`, `Creating NAMESPACE ...`, `Creating Read-and-delete-me ...`, `Saving functions and data ...`, `Making help files ...`, and `Done.`. It then displays the message: `Further steps are described in 'R//miaLibreria/Read-and-delete-me'.` The prompt `> |` is visible at the bottom.


## Step 4: compilare la documentazione


Il passo più noioso nel creare il package ma fondamentale, nella cartella creata troverete i seguenti file e sottocartelle.


 data

 man

 R

 DESCRIPTION

 NAMESPACE

 Read-and-delete-me

Data: contiene il file dati che avevamo caricato

man: la documentazione di dati e funzioni

R: contiene gli script delle funzioni

DESCRIPTION: il file descrittivo del pacchetto (nome, descrizione, autore, dipendenze, ...)

NAMESPACE: cosa bisogna importare ed esportare

Read-and-delete-me: i passi da seguire per la costruzione



## Step 4.1: DESCRIPTION

Bisogna compilare:

- Title
- Author
- Maintainer: inserite la vostra e-mail tra < >
- Description
- License: per ora ho sempre usato GPL-3  
[<https://www.gnu.org/licenses/gpl-3.0.en.html>]

```
Package: miaLibreria
Type: Package
Title: What the package does (short line)
Version: 1.0
Date: 2018-03-29
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?
```

Se il vostro package utilizza altri package inserire tra Description e License:

- Imports
- Depends

Es. Imports: dplyr

## Step 4.2: NAMESPACE

NAMESPACE è un file testuale che si presenta con una sola riga:  
`exportPattern("^[:alpha:]]+")`

Qui bisogna inserire cosa vogliamo esportare dal package e cosa importare per farlo funzionare.

Le stringhe da inserire sono le seguenti:

- **export()** per le funzioni che abbiamo scritto.  
*Es. `export(deg2num)`*
- **import()** se ci serve importare un intero package.  
*Es. `import(dplyr)`*
- **importFrom()** se vogliamo una specifica funzione.  
*Es. `importFrom(dplyr, left_join)`*  
se ci servono più funzioni verrà scritta una riga per ogni funzione.
- **useDynLib()** per importare funzioni da C.  
*Es. `useDynLib(testthat, reassign_function)`*

```
import(magrittr)
importFrom(graphics, points)
importFrom(rgdal, readOGR)
importFrom(raster, area)
importFrom(raster, plot)
importFrom(osmdata, getbb)
importFrom(osmdata, opq)
importFrom(osmdata, add_osm_feature)
importFrom(osmdata, osmdata_sp)
importFrom(spatialEco, point.in.poly)
importFrom(utils, write.csv2)
importFrom(osrm, osrmRoute)
importFrom(rgeos, gTouches)
export(loadShapefile)
export(overpassAPI.download)
```



## Step 4.3: documentazione delle funzioni

Per ogni funzione esportata e dati bisogna compilare la documentazione.

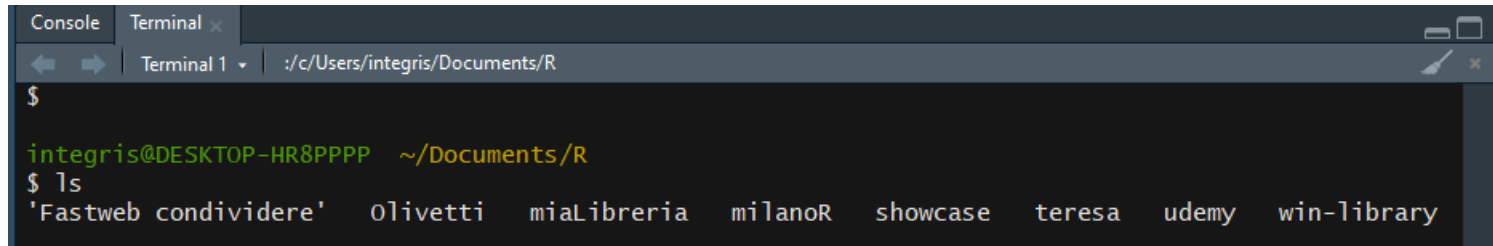
Fare attenzione sempre ai caratteri non ASCII, tipo lettere accentate e alla sezione example.

Inserite il codice di esempio preceduto da # così R lo vedrà come commento e non cercherà di compilarlo.

```
\name{overpassAPI.download}
\alias{overpassAPI.download}
\title{
overpassAPI.download
}
\description{
Consente di scaricare dati da Openstreetmap utilizzando le Overpass API
}
\usage{
overpassAPI.download(city.name, tag, value)
}
\arguments{
  \item{city.name}{
una stringa. La città sulla quale si vuole effettuare il download.
}
  \item{tag}{
una stringa. Il tag di Openstreetmap (es. "railway", "amenity", "tourism", ...)
}
  \item{value}{
una stringa. Il valore del tag (es. "station", "museum", ...)
}
}
\value{
una large list di 8 elementi restituiti da OSM. 5 di 8 elementi sono Spatial DataFrame
}
\author{
Nicola Procopio
}
\examples{
## require(OGM)
## ATM <- overpassAPI.download(city.name = "Milano", tag = "railway", value = "station")
}
```

## Step 5: R CMD build

Fatti i passi precedenti possiamo eliminare il file Read-and-delete-me. Su RStudio andare sul terminale e posizionarsi nella cartella contenente lo scheletro del pacchetto (la cartella R che contiene la sotto-cartella miaLibreria nell'esempio).



```
Console Terminal x
Terminal 1 ▾ :/c/Users/integris/Documents/R
$
integris@DESKTOP-HR8PPPP ~/Documents/R
$ ls
'Fastweb condividere' Olivetti miaLibreria milanoR showcase teresa udemy win-library
```

Per costruire il package (file tar.gz), si lanci il comando **build** seguito dal nome della libreria (miaLibreria nell'esempio)

```
integris@DESKTOP-HR8PPPP ~/Documents/R
$ R CMD build miaLibreria
```

## Step 6: R CMD check

Una volta creata miaLibreria\_1.0.tar.gz non resta che posizionarci nella cartella che la contiene e fare check del file.

```
integrism@DESKTOP-HR8PPPP ~/Documents/R
$ R CMD check miaLibreria_1.0.tar.gz
```

La procedura crea una cartella Rcheck:

- Se la procedura non va a buon fine possiamo rileggere gli errori nel log 00check
- Se la procedura va a buon fine troveremo all'interno la versione pdf del manuale

## Step 7: installazione del pacchetto

Per installare e usare il pacchetto, da console di RStudio dare i comandi:

- **`install.packages("R/miaLibreria_1.0.tar.gz", type = "source", repos = NULL, dependencies = TRUE)`**
- **`library(miaLibreria)`**
- **`help(miaLibreria)`**

Per maggiori approfondimenti sulla costruzione dei package R vi consiglio:

- Il testo «**R packages**» [<http://r-pkgs.had.co.nz/>]
- Il tutorial youtube «**Create an R package with RStudio**» [<https://www.youtube.com/watch?v=9PyQlbAEujY&t=685s>]



## Milan

*Corso Sempione, 62  
20149 Milano (MI)*

## Pisa

*Via Forti Umberto, 1  
56121 Pisa (PI)  
Tel: +39 050 9655012*

## Rome

*Via Giovanni Squarcina, 3  
00143 Roma, Italy  
Tel: +39.06.58230391  
Fax: +39.06.58230391*

## Rende

*Via Pedro Alvares Cabral, 6  
87036 Rende (CS)  
Tel: +39 0984 395257*

Where we are