نوشتن تابع‌ها، ساختن بسته در R و انتشار آن

پیمان شریفی

۱- ساختن بسته (Packages)

تهیه بسته‌ای در R، شامل توابعی که از آنها زیاد استفاده می‌شود، می‌تواند به صرفه‌جویی در زمان برای تجزیه داده‌ها کمک کند. در این نوشته، کوشیده شده است که به‌طور خلاصه به بررسی اصول اولیه گسترش بسته در R پرداخته شود. برای کسب اطلاعات بیشتر در این باره کتاب R Packages (بصورت آنلاین و رایگان) پیشنهاد می‌شود که توسط گسترش‌دهنده بسته devtools نوشته شده است (Wickham, 2015). این کتاب به گونه‌ای نوشته شده است که بیشتر کاربران تازه‌کار R (یعنی افرادی که آشنایی کمی با R دارند) خواهند توانست آنچه را که توضیح داده می‌شود، درک کنند.

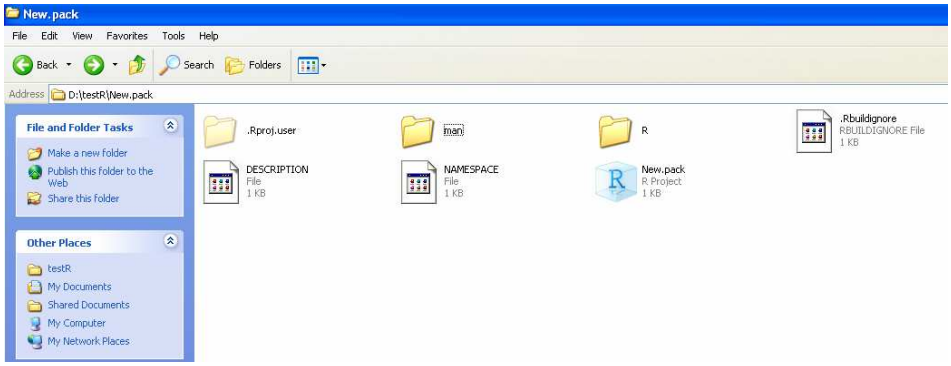
ایجاد بسته جدید در R می‌تواند به‌طورمستقیم با اجرای دستور زیر انجام شود (Johnston, 2019). توزیع اینکه نیاز است بسته devtools از قبل دریافت و پیاده‌سازی شود.

```
devtools::create('pkg_name')
```

یا با بکار بستن گزینه‌های زیر، یک بسته جدید R در RStudio ساخته می‌شود:

File -> New Project -> New Directory -> R Package

با این کار، یک دایرکتوری (directory) نو به‌نام New.pack (یا هر نام دیگر) به‌همراه فایل و فولدرهای زیر ساخته می‌شود (شکل ۱):



شکل ۱) نمایی از فایل‌ها و فولدرها پس از ساخته شدنبسته جدید

•یک زیرفولدر با نام R/ که دربرگیرنده تمام کدهای R با پسوند R. است که بسته R جدید آنها را خواهند داشت؛

•یک زیرفولدر با نام Man/ که دربرگیرنده فایل‌هایی به‌عنوان برچسب با پسوند Rd. برای هر کدام از تابع‌های بالا است. چگونگی ذخیره این فایل‌ها در بخش ساختن فایل‌های با پسوند Rd. گفته شده است؛

•یک فایل DESCRIPTION: شامل اطلاعات درباره بسته (مانند شماره نسخه، عنوان، نویسنده و غیره)؛

•یک فایل NAMESPACE: شامل توابع R یا کدی است که R هنگام نصب "بسته رسمی" به دنبال آن خواهد بود؛

•یک فایل .Rproj: به RStudio می‌گوید که ساختار پوشه برای (توسعه بسته) برای چه مواردی استفاده می‌شود و گزینه‌های RStudio را تنظیم می‌کند.

۲- ایجاد توابع

توابع (Functions) جدید R در پوشه R / قرار می‌گیرند. برای آموزش چگونگی ایجاد یک تابع، یک تابع ساده به شکل زیر ساخته می‌شود. برای این کار، تابع function بکار گرفته می‌شود. ساختار عمومی یک تابع به شکل زیر است:

```
> function(آرگومان‌ها){
+ متن برنامه
+ }
```

```
corind <-
function (df1) {
  round (cor (df1),2)
 print (round (cor (df1),2))
  ggcorrplot(cor(df1))
}
```

تابع بالا، شیئی با نام df1 را به‌عنوان آرگومان اجباری می‌گیرد و تجزیه همستگی بر روی آن انجام می‌دهد، ضرایب همبستگی را برآورد می‌کند و سپس نمودار گرمایی را با تابع ggcorrplot می‌کشد. به بیان دیگر، با اجرای تابع corind (df1)، تجزیه همبستگی و کشیدن نمودار پشت سر هم انجام می‌شود. بنابراین، این تابع ضریب همبستگی را تخمین زده و در یک ماتریس و نمودار

گرمایی ارائه می دهد. لازم است برای اجرای این تابع، بسته ggcorrplot نصب و اجرا شود. در این تابع همچنین از تابع cor از بسته Stats استفاده شده است.

در R، برای ذخیره کردن تابع‌های تازه نوشته شده می‌توان تابع dump را بکار برد که با آن تابع جدید، در فایلی با نام تابع و پسوند R. بر روی نشانی فضای کاری (Work Directory) ذخیره می‌شود. با نرم‌افزارهای پردازش متنی مانند Notepad نیز می‌توان این فایل را باز و ویرایش نمود. همچنین می‌توان با تابع source، تابع جدید را فراخواند و سپس آن را بکار برد.

```
> dump("corind ","corind.R")
> source("corind.R")
```

اکنون می‌توان هنگام اجرای بسته با devtools::load_all() یا Ctrl-Shift-L در RStudio به این تابع در کنسول R دسترسی پیدا کرد. هم اینک، هنگامی‌که شروع به تایپ کردن sum_ می‌شود، لیست تکمیل خودکار sum_nums مشاهده خواهد شد. همه توابع جدید را می‌توان از این روش پدید آورد.

بعضی اوقات نیاز به استفاده از توابع بسته‌های دیگری مانند dplyr یا ggplot2 است. بسته‌های R پس از اجرای تعاملی به این موارد دسترسی متفاوتی دارند. توضیح اینکه نمی‌توان یک عبارت library() را در توابع R داشت. بنابراین ابتدا این کار انجام می‌شود:

```
devtools::use_package('dplyr')
```

با این کار، پیامی به کنسول R منتقل می‌شود و چگونگی بکار بستن توابع بسته dplyr گفته می‌شود. همچنین dplyr را به بخش فایل DESCRIPTION اضافه می‌کند.

برای دسترسی به تابع جدید، با فشردن کلید Ctrl-Shift-L، دستور devtools::load_all() اجرا می‌شود و می‌توان در کنسول، تابع جدید را نوشت.

۳- ساختن فایل‌های با پسوند Rd. برای اضافه کردن توضیحات به ابتدای تابع

پس از ساختن بسته و ذخیره کردن فایل با پسوند R، (حتما این کار باید بعد از ذخیره کردن فایل R انجام شود) مکان‌نما درون تابع نوشته شده، قرار داده می‌شود و کلیدهای Ctrl-Shift-Alt-R. فشرده می‌شود و مواردی از قبیل زیر به ابتدای تابع اضافه می‌شود:

```
#' Title
#'
#' @param df1
```

```
#'
#' @return
#' @export
#
#' @examples
corind <-
function (df1) {
  round (cor (df1),2)
 print (round (cor (df1),2))
  ggcorrplot(cor(df1))
}
```

این برچسب‌ها، برچسب‌های roxygen نامیده می‌شوند و موارد زیر را در بر می‌گیرند:

@param: نام و توضیحات پارامتر/آرگومان؛

@return: خروجی تابع چیست. در این مثال، یک چهارچوب داده (data frame) است؛

@export: آیا این تابع به NAMESPACE. فرستاده شود. این نشان می‌دهد که آیا این تابع خارجی خواهد بود (در دسترس کاربر نهایی) یا داخلی (فقط برای توابع موجود در بسته R قابل دسترسی است، اما در دسترس کاربر نیست)؛

@examples: برای اینکه مثالی از چگونگی استفاده از تابع را نشان می‌دهد، معمولاً خوب است.

در پایین فایل با پسوند Rd. برای تابع corind (df1) نمایانده شده است:

```
% Generated by roxygen2: do not edit by hand
% Please edit documentation in R/corind.R
\name{corind}
\alias{corind}
\title{Estimation correlation coefficients and plotting the heatmap for correlation
matrixes
This function estimated the correlation coefficients and presented in a matrix and
ggcorrplot. It is need to install and running ggcorrplot packages. This function was also
used cor function from stats packages.}
\usage{
corind(df1)
}
\arguments{
\item{df1}{{df1: a data frame which obtained from indices function}}
}
\value{
```

{cor} {Correlation coefficients matrix}

{heatmap} {Plot the heatmap for correlation matrixes}
}
\description{
Estimation correlation coefficients and plotting the heatmap for correlation matrixes
This function estimated the correlation coefficients and presented in a matrix and ggcorrplot. It is need to install and running ggcorrplot packages. This function was also used cor function from stats packages.
}
\examples{
{corind (df1)}
}

پس از اصلاح و اضافه کردن موارد، تابع در زیر فولدر Man/ ذخیره می‌شود و از Ctrl-Shift-D برای ران کردن devtools::document() یا مستقیم با نوشتن خود این دستور بهره گرفته می‌شود. با انجام این دستور، پیام زیر نشان داده می شود:

```
> devtools::document()
Updating EASiP2 documentation
Loading EASiP2
Warning: The existing 'NAMESPACE' file was not generated by roxygen2, and will not be overwritten.
Writing corind.Rd
```

٤- ذخیره کردن داده‌ها در بسته ساخته شده

برای ذخیره کردن فایل داده‌ها در بسته جدید، از کتابخانه devtools و تابع use_data بهره گرفته می‌شود. نام مجموعه داده‌های خود را جایگزین coffeedata کنید (Hogervorst, 2016):

```
library(devtools)
use_data(coffeedata)
```

یا اینکه مستقیم از devtools استفاده شود:

```
devtools::use_data(coffeedata)
```

این دستور یک پوشه داده ایجاد می‌کند و فایل داده‌ها را با پسوند rda در آن قرار می‌دهد.

٥- سندیت دادن به بسته

با نوجه به اینکه بسته چقدر پیچیده باشد (و حتی اگر این‌گونه نباشد)، این کار به نوشتن یک اسناد توصیفی درباره چگونگی استفاده از بسته کمک می‌کند. به این اسناد vignettes گفته می‌شود. حتی اگر کسی این بسته را بکار نبرد، پیشنهاد شده است که دستکم مقدمه‌ای درباره آن

نوشته شود، زیرا چگونگی کاربرد بسته در آینده در صورت فراموش شدن تضمین می‌شود (فراموش کردن کار با بسته‌ای که فرد خودش نوشته است، ممکن است بارها رخ دهد). بنابراین، برای شروع نوشتن یک ویگنت، در کنسول، دستور زیر نوشته می‌شود:

```
devtools::use_vignette('introduction')
```

با این کار، یک پوشه vignette/ و یک فایل R Markdown ایجاد می‌شود، که می‌توان متن، کد و خروجی کد را برای آموزش بسته در آن جای داد. با نصب بسته می‌توان به این ویگنت دسترسی پیدا کرد:

```
vignette('introduction', package = 'your_package_name')
```

٦- نصب بسته و موارد دیگر

با اجرای دستور زیر می‌توان بسته را نصب کرد:

devtools :: install ()

با اجرای دستور بالا، بسته نصب می‌شود و امکان دسترسی به تابع‌ها با استفاده از فرمان library() فراهم می‌آید. موارد دیگری هم برای ساختن یک بسته R وجود دارد که دربرگیرنده آزمون‌هایی می‌باشد.

آزمون واحد (از طریق بسته testthat و devtools::use_testthat()) امکان آزمون‌هایی را برای تأیید کار تابع را می‌دهد (ران کردن Ctrl-Shift-T برای اجرای دستور devtools::test() بکار می‌رود که آزمون Unit را انجام می‌دهد).

٧- انتشار بسته

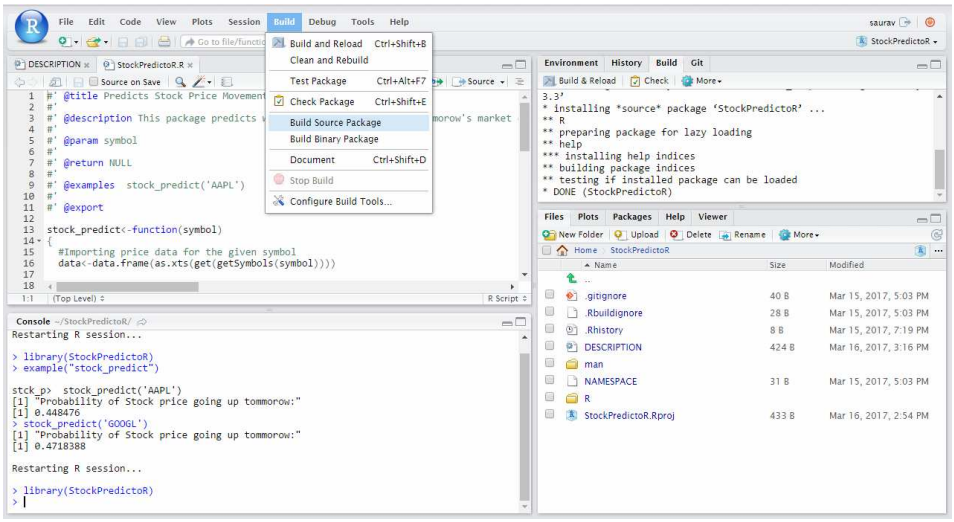
پس از ایجاد موفقیت‌آمیز یک بسته در R، گام بعدی به اشتراک گذاردن آن با دیگران است تا به آنها اجازه داده شود که از توابع موجود در بسته استفاده کنند. برای فرآیند انتشار یک بسته، دو سیستم GitHub و CRAN وجود دارد.

٧-١ انتشار بسته در CRAN (Comprehensive R Archive Network)

انتشار بسته در CRAN به دلیل آزمایش‌های گسترده و دقیق که روی بسته‌ها پیش از انتشار آنها انجام می‌شود، کار سختی است. در کنار گذراندن این آزمون‌ها، نیاز به vignetteهای جامعی

است که عملکرد بسته را توصیف کنند. این vignette در پوشه vignette ذخیره می‌شود، که می‌توان آن را در دایکتوری اصلی پروژه ایجاد کرد.

هنگامی که بدست آوردن اطمینان از پشت سر گذراندن بسته در برابر آزمون‌های شبیه‌سازی محلی و به‌خوبی مستند شدن آن، باید با رفتن به Build> Build Source Package، منبع بسته ایجاد شود (شکل ۲):



شکل ۲– ایجاد منبع بسته

پس از آنکه منبع بسته پدید آورده شد، می‌توان آن را با دستور https://cran.r-project.org/submit.html به CRAN فرستاد تا مراحل بررسی در آنجا انجام شود.

۲.۷– انتشار بسته در GitHub

به‌طور کلی، روشی بسیار آسان‌تر برای تبلیغ عمومی یک بسته، انتشار آن در GitHub است. ساده‌ترین روش برای انتشار بسته در GitHub، ایجاد یک repository جدید و بارگذاری محتوای پوشه اصلی (بسته به‌تازگی ساخته شده) در آن است. برای نمونه، بسته EASiP برای برآورد شاخص‌های مقاومت و حساسیت به تنش‌های محیطی به‌همراه نمودارهای مربوطه در آدرس https://github.com/kadose/EASiP جای دارد و هر کسی می‌تواند با استفاده از دستور زیر این بسته را نصب و از آن استفاده کند:

devtools :: install_github ("kadose / EASiP ")

افزون بر آدرس بالا، اپلیکیشن Openhub نیز برای سیستم‌های آندروید طراحی شده است که از GitHub حمایت می‌کند.

منابع بکار برده شده

Hogervorst, R.M. (2016). Creating a package for your data set. Available in: https://www.r-bloggers.com › creating-a-package-for-your-data-set

Johnston, L. (2019). R package development. Available in: https://github.com/UofTCoders/studyGroup/tree/gh-pages/lessons/r/packages

Wickham, H. (2015). R Packages: Organize, Test, Document, and Share Your Code. 1st Edition, Kindle Edition. Available in: http://r-pkgs.had.co.nz/