Wrapping R packages demonstration

Nnenna Asidianya

Introduction

This is a demonstration of how to build an R package through a series of steps.

The tutorial will require the use of the following packages `usethis` and `royxygen2`. In order to share your package externally, you will need `devtools`.

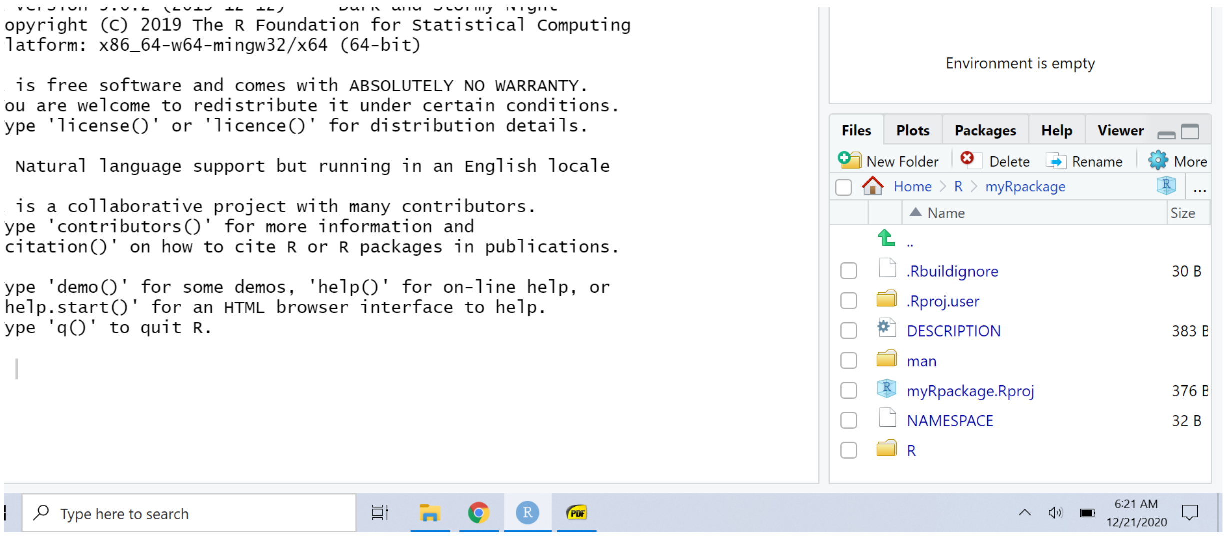
The outline is as follows:

1. Set up the package directory
2. Describe the package
    - ‘DESCRIPTION‘ is already set up
    - README.Rmd to display the contents of the package
3. Add data to a package (optional)
4. Create and add functions to R directory

Set up the package directory

Start by setting up a new project using R studio: File -> New Project… -> New Directory” -> R Package.

This brings up a menu where you give your package a name, and specify where to create it on your hard drive.



```
opyright (C) 2019 The R Foundation for Statistical Computing
latform: x86_64-w64-mingw32/x64 (64-bit)

 is free software and comes with ABSOLUTELY NO WARRANTY.
ou are welcome to redistribute it under certain conditions.
ype 'license()' or 'licence()' for distribution details.

 Natural language support but running in an English locale

 is a collaborative project with many contributors.
ype 'contributors()' for more information and
citation()' on how to cite R or R packages in publications.

ype 'demo()' for some demos, 'help()' for on-line help, or
help.start()' for an HTML browser interface to help.
ype 'q()' to quit R.
```

At this point your package infrastructure is there, but there is nothing in the package as of yet.

Describe the package

This file is created initially when you create your R package folder. The details are outlined below. Since it is created during the R package creating process, the file must therefore remain in this machine-readable. Here's an example for myRpackage (currently empty):

```
Package: myRpackage
Type: Package
Title: What the Package Does (Title Case)
Version: 0.1.0
Author: Who wrote it
Maintainer: The package maintainer <yourself@somewhere.net>
Description: More about what it does (maybe more than one line)
    Use four spaces when indenting paragraphs within the Description.
License: What license is it under?
Encoding: UTF-8
LazyData: true
```

One section called "Imports" is missing when you generate this file. If your package depends on an external R package then specify this in imports after 'LazyData'. e.g..

```
Imports:
      tidvyerse
```

Readme

I recommend installing `usethis` at this point. Open an R markdown file and run the following lines of code. This will create a README.Rmd file that can be used as a base to explain the features of your package. This cannot be knit until you *Install and Restart* your package once the contents are inside.

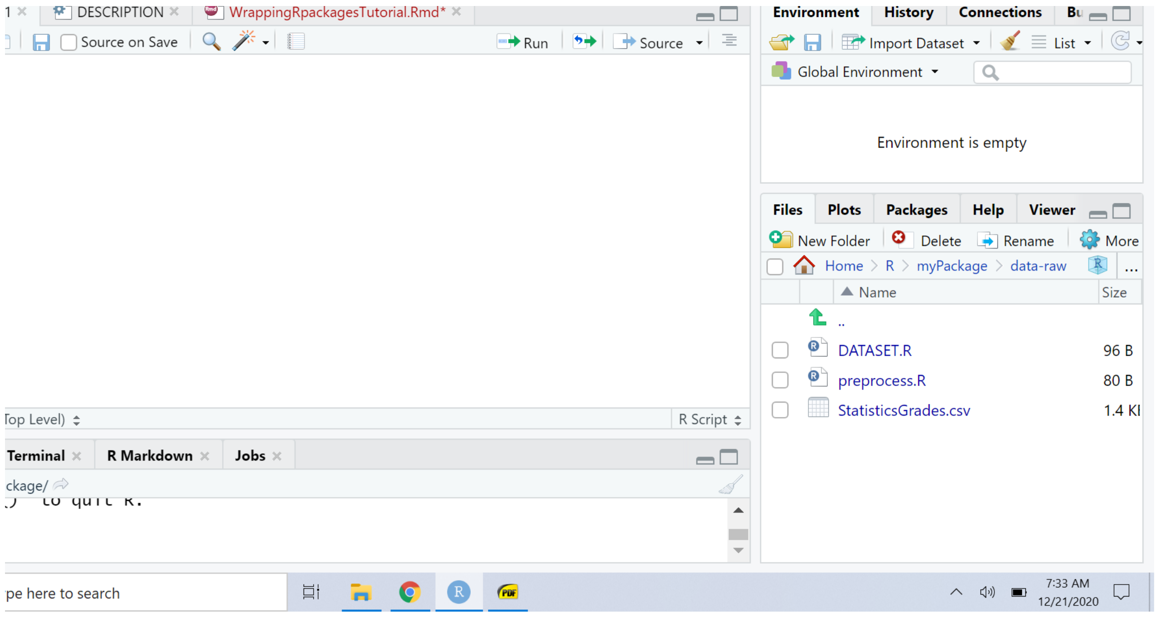
```
#library(usethis)
#use_readme_rmd()
```

#Add Data

The `usethis` package has a function called `use_data_raw()`. In my R markdown file I will run this line to create a working directory for my data:

```
#use_data_raw()
```

This will create a new folder called `data-raw` in your package folder. Prior to this tutorial I created a mini-package in a folder called "myPackage." The screen shot below shows the contents inside of the daw-rata file. I manually moved an example data file (statistics term test 1 and 2 grades from a previous course stripped of identifying characteristics) into the data-raw file.



Name	Size
..	
DATASET.R	96 B
preprocess.R	80 B
StatisticsGrades.csv	1.4 KI

Notice that in the same directory is a folder called "preprocess.R." I wrote a small script found below that when run would create a dataset R file that will be read when you build and install your function. The function `use_data` is from the `usethis` package. It will transfer the data file that is created to the folder called `data`.

```
library(tidyverse)
data_yours-read_csv("data_yours.csv")
use_data(data_yours)
```

It is also possible that your data is simulated and therefore already within the R environment. Changing the code should not be that big of an issue. Ditto if you are using a .txt file for your data rather than a csv, etc.

Here is what my data looks like in the package I created:

```
library(tidyverse)
library(myPackage)
glimpse(Grades)
#> Rows: 103
#> Columns: 4
#> $ `Term Test 1 (/505)`        <dbl> 34.0, 36.0, 26.0, 45.0, 18.0, 34.0, -1.0...
#> $ `Term Test 1 Adjust (/50)` <dbl> 36.0, 38.0, 28.0, 47.0, 20.0, 36.0, -1.0...
#> $ `Term Test 2 (/57)`         <dbl> 40, 42, 29, 48, -1, 40, -1, 45, 28, 43, ...
#> $ `Term Test 2 Adjust (/55)` <dbl> 40, 42, 29, 48, -1, 40, -1, 45, 28, 43, ...
```

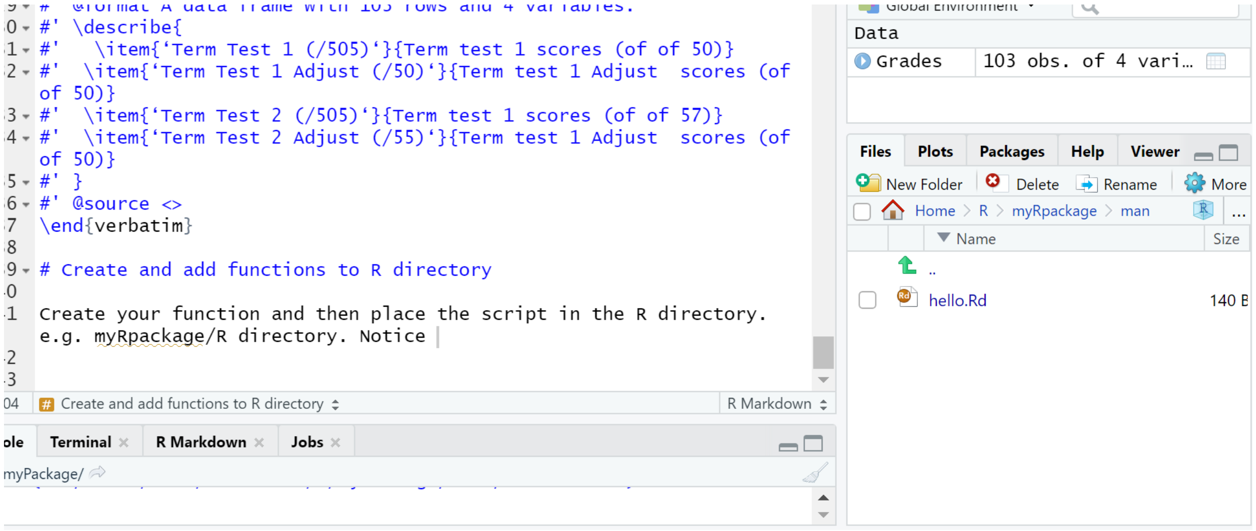
In the last step you will need to install roxygen2. The intent is to describe your functions in comments next to their definitions. When you compile your entire package the description of what the function does alongside the function will be processed in the roxygen2 folder. Create a “data.R” file and place it in the R directory. When you compile your package roxygen2 will add the description of your data alongside the data frame. Here is what mine looks like:

```
#' @title Scores of undergraduate statistics students at UofT
#'
#' @description A data set with the scores of two groups.
#'
#' @format A data frame with 103 rows and 4 variables:
#' \describe{
#'   \item{`Term Test 1 (/505)`}{Term test 1 scores (of of 50)}
#'  \item{`Term Test 1 Adjust (/50)`}{Term test 1 Adjust  scores (of of 50)}
#'  \item{`Term Test 2 (/505)`}{Term test 1 scores (of of 57)}
#'  \item{`Term Test 2 Adjust (/55)`}{Term test 1 Adjust  scores (of of 50)}
#' }
#' @source <>
```

Create and add functions to R directory

Create your function and then place the script in the R directory. e.g. myRpackage/R directory. Notice when you first look inside the man directory, you see the hello.Rd script.

This is the default function inside your folder when you create a new project. I deleted it and added my own function to the R directory so that when I Installed my package my own function would be in the myPackage/man folder instead.



```
#' @format A data frame with 103 rows and 4 variables.
#' \describe{
#'   \item{‘Term Test 1 (/505)‘}{Term test 1 scores (of of 50)}
#'  \item{‘Term Test 1 Adjust (/50)‘}{Term test 1 Adjust  scores (of
of 50)}
#'  \item{‘Term Test 2 (/505)‘}{Term test 1 scores (of of 57)}
#'  \item{‘Term Test 2 Adjust (/55)‘}{Term test 1 Adjust  scores (of
of 50)}
#' }
#' @source <>
\end{verbatim}

# Create and add functions to R directory

Create your function and then place the script in the R directory.
e.g. myRpackage/R directory. Notice
```

```
Data
Grades    103 obs. of 4 vari…
```

```
Home > R > myRpackage > man
hello.Rd    140 B
```

Once you have added your own function then you would want to use roxygen2 syntax to document your function. In the R studio menu click the following path: Code" -> "Insert Roxygen Skeleton. The last steps are as follows (and in order):

- Go to Tools -> Project Options... -> Build Tools, and make sure that "Generate documentation with roxygen" is checked, and that "Automatically run roxygen when running install and restart" is checked in the subsequent "Configure" menu.
- Then, delete the two files, man/hello.Rd and NAMESPACE, which R Studio created automatically when you started your package.
- In R Studio's "Build" tab, click "Install and Restart".

Share the package in GitHub

I have not done this yet, because I actually want to finish my package. However I looked into ways in which you can share your code and one common one is through github. Installation of devtools is required here. I believe the command is the following:

```
devtools::install_github("github_name/myRPackage")
```