Extending R using packages, creating and manipulating objects

Learning the basics of R - Part 2

Ernest Guevarra

2023-01-17

Outline

1. Base functions in R

    - What is a function?
    - Basic function syntax

2. Extending R using packages

    - What are packages?
    - How to install packages
    - Loading packages to the environment

3. Accessing and reading data into R

What is a function?

- A set of statements organized together to perform a specific task.
- R has a large number of in-built functions.
- In R, a function is an object so the R interpreter is able to pass control to the function, along with arguments that may be necessary for the function to accomplish the actions.
- The function in turn performs its task and returns control to the interpreter as well as any result which may be stored in other objects.

Functions in R

Base functions

- Term we use for builtin functions in R.
- These functions cover a wide range of purposes, use cases, and applications one of which is for statistical analysis (probably the most common builtin functions in R)
- Everyting we do in R is almost always mediated/made possible by using functions

Basic function syntax

```
function_name(argument1, argument2, ...)
```

Using functions - accessing R builtin dataset

- First let us use some sample/toy data. R has builtin datasets for teaching/testing purposes. We will continue on the BMI theme from yesterday by accessing the `women` builtin dataset in R. This dataset is of average height (inches) and weight (lbs) of women age 30-39 years old.

- We access this data using the `data()` function as follows:

```
data("women")
```

```
women
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Using functions - exploring data structure

- Being able to understand the **data structure** of a dataset helps us make good decisions on how to work with data or how to analyse data.

- There are several R functions that gives us the characteristics and structure of a dataset such as:

    - The shape of the data
    - The number of records in the data
    - The variables of the data
    - The number of variables in the data
    - The values of variables in the data

Using functions - describing the shape of the data

- We use the `class()` function to know the **class** attribute of an R object.
- Knowing the **class** of an R object give us information on what kind of object it is and how we can work with it in R

Task:

- Using the `women` dataset that we just loaded, apply the `class()` function:

```
## Get class of women dataset
class(women)
```

```
## [1] "data.frame"
```

Using functions - number of records in the data

- We often need to know how many records are in the dataset that we are working on.
- This is useful for various statistical analysis that we perform on data.
- The function `nrow( )` gives us the number of rows of a `data.frame` R object

Task:

- Using the `women` dataset, apply the `nrow( )` function to get the number of rows:

```
## Get number of rows of women dataset
nrow(women)
```

```
## [1] 15
```

Using functions - number of records in the data

Bonus question:

- How many columns does the `women` dataset have?

```
ncol(women)
```

```
## [1] 2
```

Using functions - variable names of a dataset

- We often need to know the variables of the dataset that we are working on.
- This is useful for various statistical analysis that we perform on data.
- The function `names()` gives us the variable names of a `data.frame` R object

Task:

- Using the `women` dataset, apply the `names()` function to get the variable names:

```r
## Get variable names of women dataset
names(women)
```

```
## [1] "height" "weight"
```

Using functions - variable names of a dataset

Bonus questions:

- Can you describe the shape and structure of the output of `names(women)`?

```r
## Get class of variable names of women dataset
class(names(women))
```

```
## [1] "character"
```

- Can you get how *LONG* (how many variable names) the output of `names(women)` is?

```r
## Get length of the variable names of women dataset
length(names(women))
```

```
## [1] 2
```

Using functions - describing the structure of a dataset

- Another approach to get a full description of the structure of a dataset object in R is by using the `str` function

```
str(women)
```

```
## 'data.frame':    15 obs. of  2 variables:
##  $ height: num  58 59 60 61 62 63 64 65 66 67 ...
##  $ weight: num  115 117 120 123 126 129 132 135 139 142 ...
```

- The output of using `str()` function is comprehensive.
    - It gives us the class of the object
    - It gives us the number of records/observations
    - It gives us the number of variables
    - It gives us the names of the variables
    - It gives us the class of each of the variables
    - It gives us a glimpse of the values of each of the variables

Using functions - accessing the variables of a dataset

- When working with `data.frame` objects, we often need to use/access only a specific variable in that `data.frame` object

- Knowing how to access a specific variable in a `data.frame` object is one of the most important skill im R

- There are several ways to access a specific variable in a `data.frame` object

Using functions - accessing the variables of a dataset

Using the $ operator

- Access the **`height`** variable using the **`$`** operator

```
women$height
```

```
##  [1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
```

- Now try to access the **`weight`** variable using the **`$`** operator

```
women$weight
```

```
##  [1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164
```

Using functions - accessing the variables of a dataset

Using the indexing method - `[ ]`

- Access the **`height`** variable using `[ ]`

```
women[ , "height"]
```

```
##  [1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
```

```
women[ , 1]
```

```
##  [1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
```

```
women[[1]]
```

```
##  [1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
```

Using functions - accessing the variables of a dataset

Using the indexing method - `[ ]`

- Now try to access the **`weight`** variable using **`[ ]`**

```
women[ , "weight"]
```

```
##  [1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164
```

```
women[ , 2]
```

```
##  [1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164
```

```
women[[2]]
```

```
##  [1] 115 117 120 123 126 129 132 135 139 142 146 150 154 159 164
```

Using functions - accessing other values of a dataset

Bonus question:

- Access the **`height`** value for the **third** row/record of the dataset

```r
women[3, "height"]
```

```
## [1] 60
```

```r
women[ , "height"][3]
```

```
## [1] 60
```

Using functions - accessing other values of a dataset

Bonus question:

- Access the **`height`** value for the **third** row/record of the dataset

```
women[3, ]["height"]
```

```
##   height
## 3     60
```

```
women[3, ][["height"]]
```

```
## [1] 60
```

```
women$height[3]
```

```
## [1] 60
```

Using functions - some basic statistical functions

Function	Description
mean()	Get the mean value of a set of numbers
median()	Get the median value of a set of numbers
var()	Get the estimated variance of the population from which you sampled
sd()	Get the standard deviation of the population from which you sampled
scale()	Get the z-scores for a set of numbers

Using functions - application of some basic statistical functions

1. Get the mean height in the women dataset

2. Get the median weight in the women dataset

```
mean(women$height)
```

```
## [1] 65
```

```
median(women$weight)
```

```
## [1] 135
```

Extending R using packages

- There are times that we need functions that are not builtin to R but are available through external **R packages**
- **R packages** are collections of functions and data sets developed by the community.
- **R packages** increase the power of R by improving existing base R functionalities, or by adding new ones.
- For this project, majority of the statistical tools/functions we need are already builtin to R.
- However, most of the tools we need for data access and loading, data manipulation, data processing, creating reports, reproducibility, and automation will rquire us to extend R using these additional **R packages**

Extending R using packages

- We usually have our data in different files and these files can be in different file formats.
- Depending on the file format of your data, different functions are used to read these files into R.
- Base (builtin) functions in R have a limited types of data that it can read.
- We often need to install additional **R packages** to read other types of data e.g., `.XLSX`, `.dta`, `.sav`, etc.

Extending R using packages

- Using base functions in R to read a `text` type of data file such as CSV

```r
read.table(
  file = "data/women.csv",
  header = TRUE, sep = ","
)
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Extending R using packages

- Using base functions in R to read a `text` type of data file such as CSV

```r
read.csv(file = "data/women.csv")
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Extending R using packages

- We should assign this data to an object. Let us call this object women_csv

```r
women_csv <- read.csv("data/women.csv")
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Extending R using packages

- Using the R package **openxlsx** to read .XLSX type of data file

- We first need to install the openxlsx package

```
install.packages("openxlsx")
```

- We then need to load the package into the current working environment. We use the library() function for this:

```
library("openxlsx")
```

Extending R using packages

- Using the R package **openxlsx** to read .XLSX type of data file
- We are now ready to use the function read.xlsx( ) from the openxlsx package to read the women.xlsx file:

```
read.xlsx(
  xlsxFile = "data/women.xlsx",
  sheet = 1
)
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Extending R using packages

- We should assign this data to an object. Let us call this object `women_xlsx`

```r
women_xlsx <- read.xlsx(
  xlsxFile = "data/women.xlsx",
  sheet = 1
)
```

```r
women_xlsx
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Coding challenge

Given what you have learned so far, I would like you to write an R script (add this to your script in `bmi.R`) that would:

- Calculate the BMI of each of the 15 records in the `women` dataset. For this, remember that the units of the weight in the `women` dataset is in lbs and the units of the height is in inches. So you need to do conversions. You can use the following:

  1 inch = 0.0254 metres
  1 lb = 0.453592 kgs

Compared to our exercise earlier, we are now working on 15 records. Think very well how you would apply the calculations!

- Determine which record has the highest BMI and which record has the lowest BMI using R code

- Determine how many records have a BMI higher than 23.

- Calculate the mean and median BMI of the 15 records

Questions?

Thank you!

Slides can be viewed at https://oxford-ihtm.io/open-reproducible-science/session3.html

PDF version of slides can be downloaded at https://oxford-ihtm.io/open-reproducible-science/pdf/session3-r-basics-part2.pdf

R scripts for slides available here