RStudio, R packages, and R project

A typical data science workflow in R

Tuesday, October 20, 2020

About Me

About Me

Name: Ezekiel Adebayo Ogundepo

About Me

Name: Ezekiel Adebayo Ogundepo

Twitter: @gbganalyst

About Me

Name: Ezekiel Adebayo Ogundepo

Twitter: @gbganalyst

GitHub: www.github.com/gbganalyst

About Me

Name: Ezekiel Adebayo Ogundepo

Twitter: @gbganalyst

GitHub: www.github.com/gbganalyst

Website: https://bit.ly/gbganalyst

Table of contents

Section 1

R and RStudio

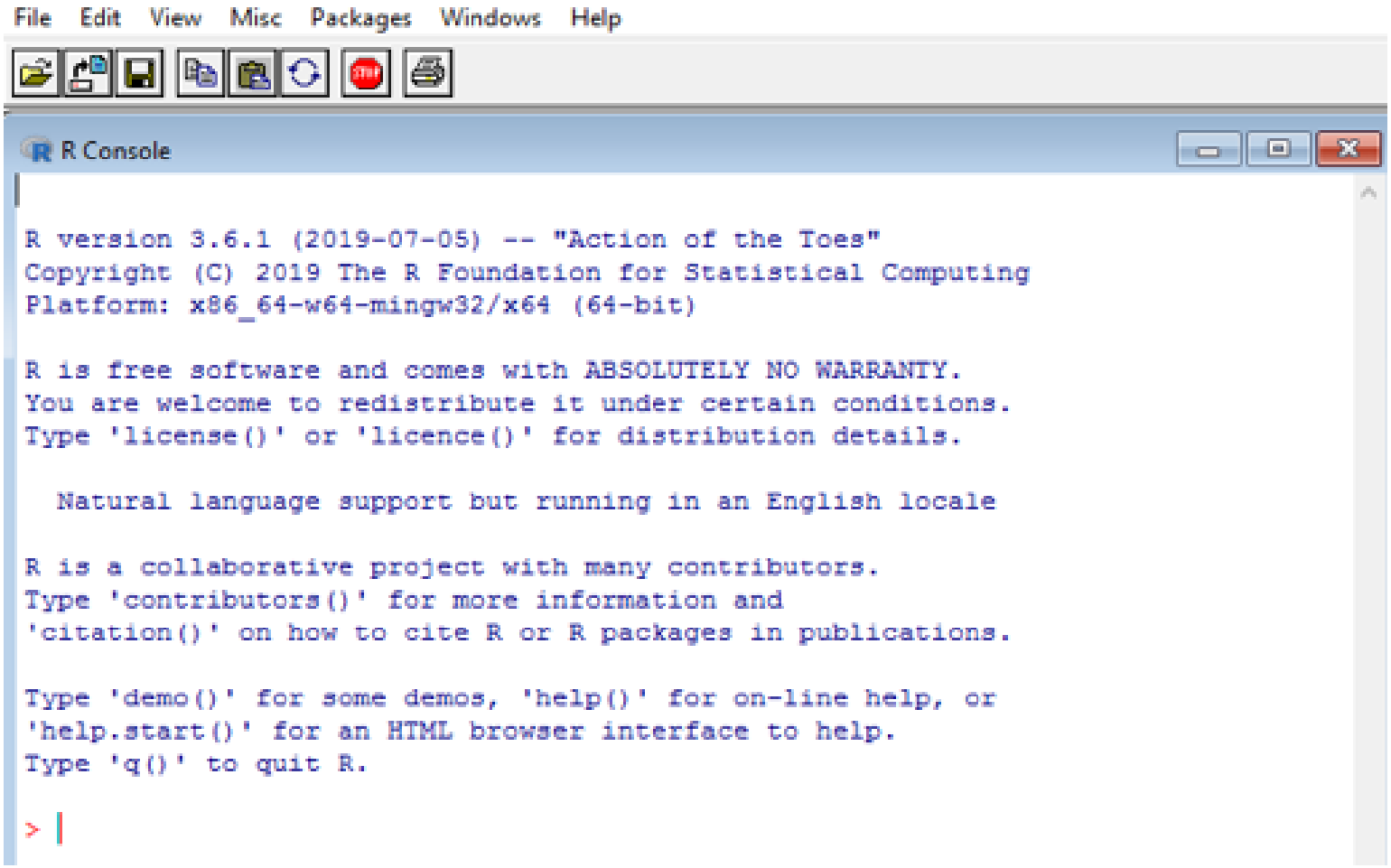
What is R programming?

What is R programming?

R is a statistical programming language for data cleaning, analysis, visualization, and modelling.

What is R programming?

R is a statistical programming language for data cleaning, analysis, visualization, and modelling. The current version of R programming is 4.0.3.



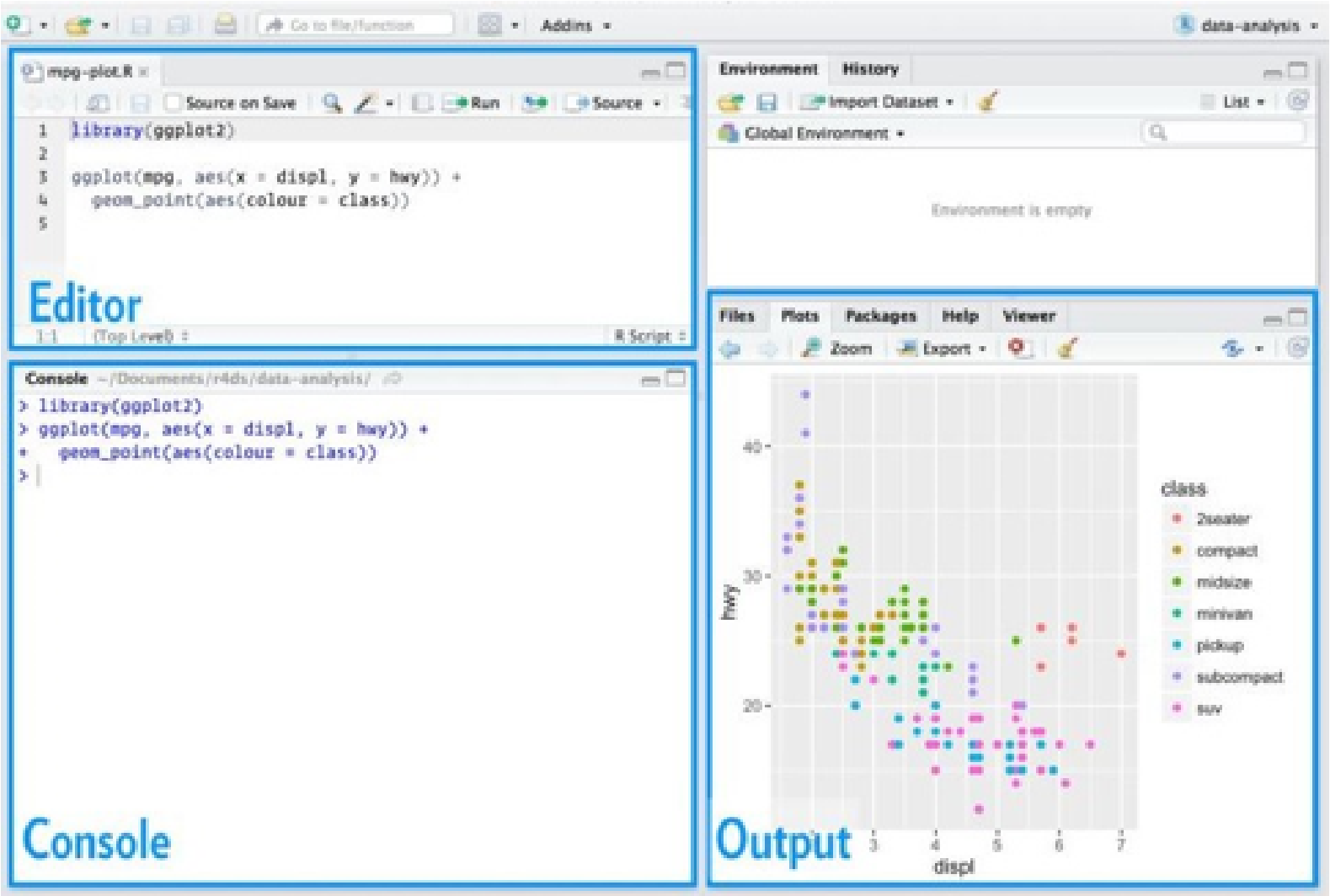
What about RStudio?

What about RStudio?

R Studio is an integrated development environment (IDE) for R programming. R Studio makes programming easier and friendly in R.

R studio

The current version of RStudio is 1.4.869.



Section 2

R packages and library

R packages and library

A package is a collection of R functions that extends basic R functionality (base::functions).

R packages and library

A package is a collection of R functions that extends basic R functionality (base::functions).

A package can contain a set of functions relating to a specific topic or tasks. For example, data wrangling packages include tidyr, janitor, etc.

R packages and library

A package is a collection of R functions that extends basic R functionality (base::functions).

A package can contain a set of functions relating to a specific topic or tasks. For example, data wrangling packages include tidyr, janitor, etc.

The location where the packages are stored is called the library. If there is a particular package that you need, you can install the package from the Comprehensive R Achieve Network (CRAN) by using:

R packages and library

A package is a collection of R functions that extends basic R functionality (base::functions).

A package can contain a set of functions relating to a specific topic or tasks. For example, data wrangling packages include tidyr, janitor, etc.

The location where the packages are stored is called the library. If there is a particular package that you need, you can install the package from the Comprehensive R Achieve Network (CRAN) by using:

```
install.packages("pkg_name")
```

R packages and library

R packages and library

Other packages that are not yet on CRAN can also be installed from GitHub by using devtools package e.g. fakir

R packages and library

Other packages that are not yet on CRAN can also be installed from GitHub by using devtools package e.g. fakir

```
library(devtools)
install_github("ThinkR-open/fakir")
```

Import or load a package

Import or load a package

To actually use the package, you need to use the command

Import or load a package

To actually use the package, you need to use the command

```
library("pkg_name")
```

Import or load a package

To actually use the package, you need to use the command

```
library("pkg_name")
```

which makes that package functions available to you at the R session.

Library

Library

Library is a directory where the packages are stored. You can have multiple libraries on your hard disk.

Library

Library is a directory where the packages are stored. You can have multiple libraries on your hard disk.

To see which libraries are available (which paths are searched for packages):

Library

Library is a directory where the packages are stored. You can have multiple libraries on your hard disk.

To see which libraries are available (which paths are searched for packages):

```
.libPaths()
```

```
## [1] "C:/Users/OGUNDEPO EZEKIEL .A/Documents/R/win-library/3
## [2] "C:/Program Files/R/R-3.6.1/library"
```

And to see which packages are there:

And to see which packages are there:

```r
lapply(.libPaths(), dir)
```

```
## [[1]]
##   [1] "abind"              "acepack"            "ada"
##   [4] "askpass"            "assertthat"         "attempt"
##   [7] "AUC"                "babynames"          "backports"
##  [10] "bartMachine"        "bartMachineJARs"    "base64enc"
##  [13] "BBmisc"             "BH"                 "bibtex"
##  [16] "bit"                "bit64"              "bitops"
##  [19] "blob"               "blogdown"           "bookdown"
##  [22] "brew"               "broom"              "BSDA"
##  [25] "bst"                "C50"                "callr"
##  [28] "car"                "carData"            "caret"
##  [31] "catboost"           "caTools"            "cellranger"
##  [34] "charlatan"          "checkmate"          "citr"
##  [37] "classInt"           "cli"                "clipr"
##  [40] "clisymbols"         "coin"               "colorspace"
```

`library(x)` or `require(x)`?

`library(x)` or `require(x)`?

`library(package)` and `require(package)` both load the namespace of the package with name package and attach it on the search list. require is designed for use inside other functions; it returns FALSE and gives a warning (rather than an error as `library()` does by default) if the package does not exist.

Remove installed packages

Remove installed packages

Removes installed packages/bundles and updates index information as necessary.

```
remove.packages("pkg_name")
```

Using functions in other packages with Double Colon operator

Using functions in other packages with Double Colon operator

There are many ways to make use of functions in other packages. You can load the package with `library(pkg_name)` and then just use the functions. Or you can use the :: operator, for example writing `janitor::clean_name()` rather than `library(janitor)` and then `clean_name()`.

Using functions in other packages with Double Colon operator

There are many ways to make use of functions in other packages. You can load the package with `library(pkg_name)` and then just use the functions. Or you can use the :: operator, for example writing `janitor::clean_name()` rather than `library(janitor)` and then `clean_name()`.

The move is towards the latter, where only the necessary functions will be loaded, rather than attaching the whole package. So to carry the reader of your article on which function belongs to a particular package, it is better to use `package_name::function()`

Section 3

RStudio project

Where Does Your Analysis Live?

Where Does Your Analysis Live?

The working directory is where R looks for files that you ask it to load, and where it will put any files that you ask it to save.

Where Does Your Analysis Live?

The working directory is where R looks for files that you ask it to load, and where it will put any files that you ask it to save.

RStudio shows your current working directory at the top of the console:

Where Does Your Analysis Live?

The working directory is where R looks for files that you ask it to load, and where it will put any files that you ask it to save.

RStudio shows your current working directory at the top of the console:

And you can also print this out by using:

```
getwd()
```

```
"C:/Users/OGUNDEPO EZEKIEL .A/Desktop/R programming end-to-end
```

If you have specific directory and you want to use that as your working directory, in R you can do that with the command `setwd()` e.g.

If you have specific directory and you want to use that as your working directory, in R you can do that with the command `setwd()` e.g.

```
setwd("/path/to/my/data_analysis")
```

or by using the keyboard shortcut with `Ctrl+Shift+H` and choose that specific directory (Folder).

Paths and Directories

Paths and Directories

- Absolute paths: This looks different in every computer. In Windows they start with a drive letter (e.g., C:). In my R working directory I have `"C:/Users/OGUNDEPO EZEKIEL .A/Desktop/R programming end-to-end"` as absolute path.

Paths and Directories

- Absolute paths: This looks different in every computer. In Windows they start with a drive letter (e.g., C:). In my R working directory I have `"C:/Users/OGUNDEPO EZEKIEL .A/Desktop/R programming end-to-end"` as absolute path.

You should never use absolute paths in your scripts, because they hinder sharing and no one else will have exactly the same directory configuration as you.

Paths and Directories

- Absolute paths: This looks different in every computer. In Windows they start with a drive letter (e.g., C:). In my R working directory I have `"C:/Users/OGUNDEPO EZEKIEL .A/Desktop/R programming end-to-end"` as absolute path.

You should never use absolute paths in your scripts, because they hinder sharing and no one else will have exactly the same directory configuration as you.

- Relative paths: With the help of library `here::here()` or `R project` we can have a relative path like `data/submission_format.csv` that allow for file sharing and collaboration.

RStudio Projects

RStudio Projects

For a typical data science workflow, you should use Rstudio project.

R experts keep all the files associated with a project together—like data folder, R scripts folder, analytical results folder, figures folder. This is such a wise and common practice.

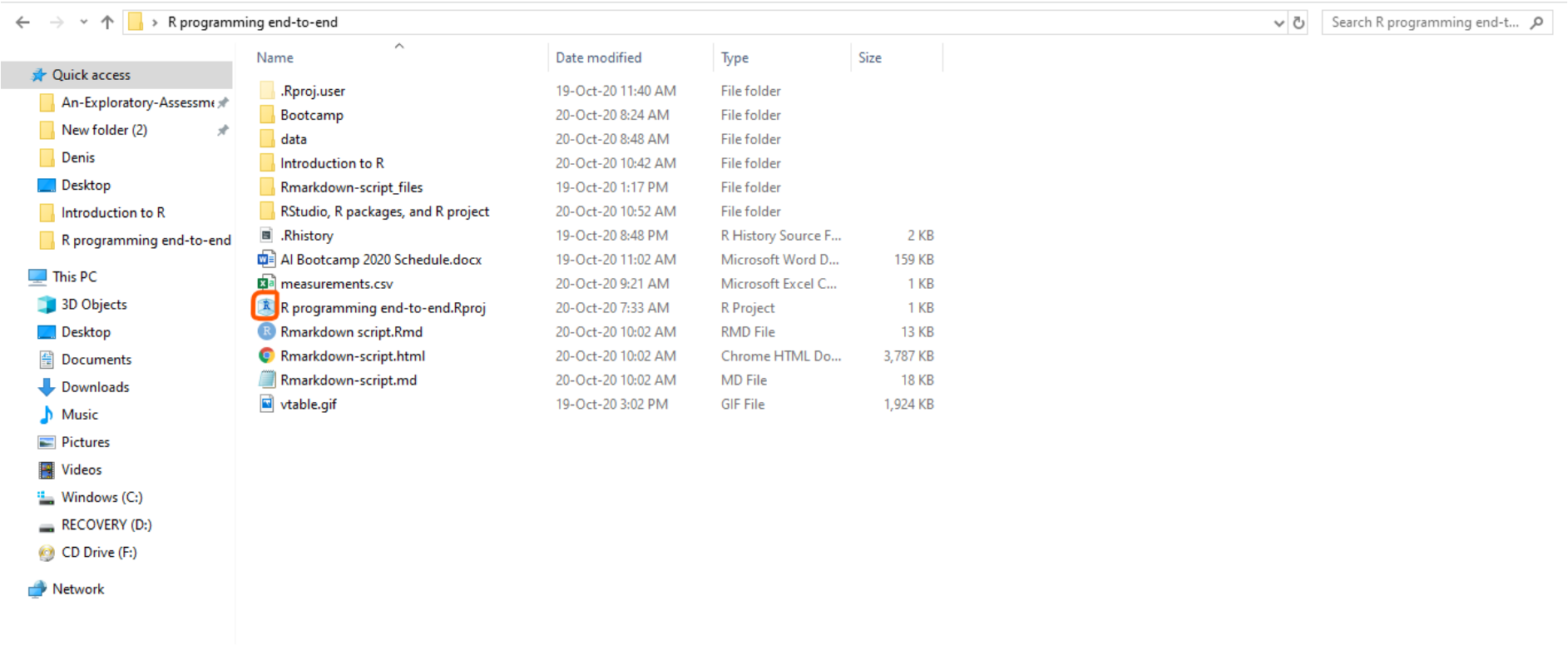
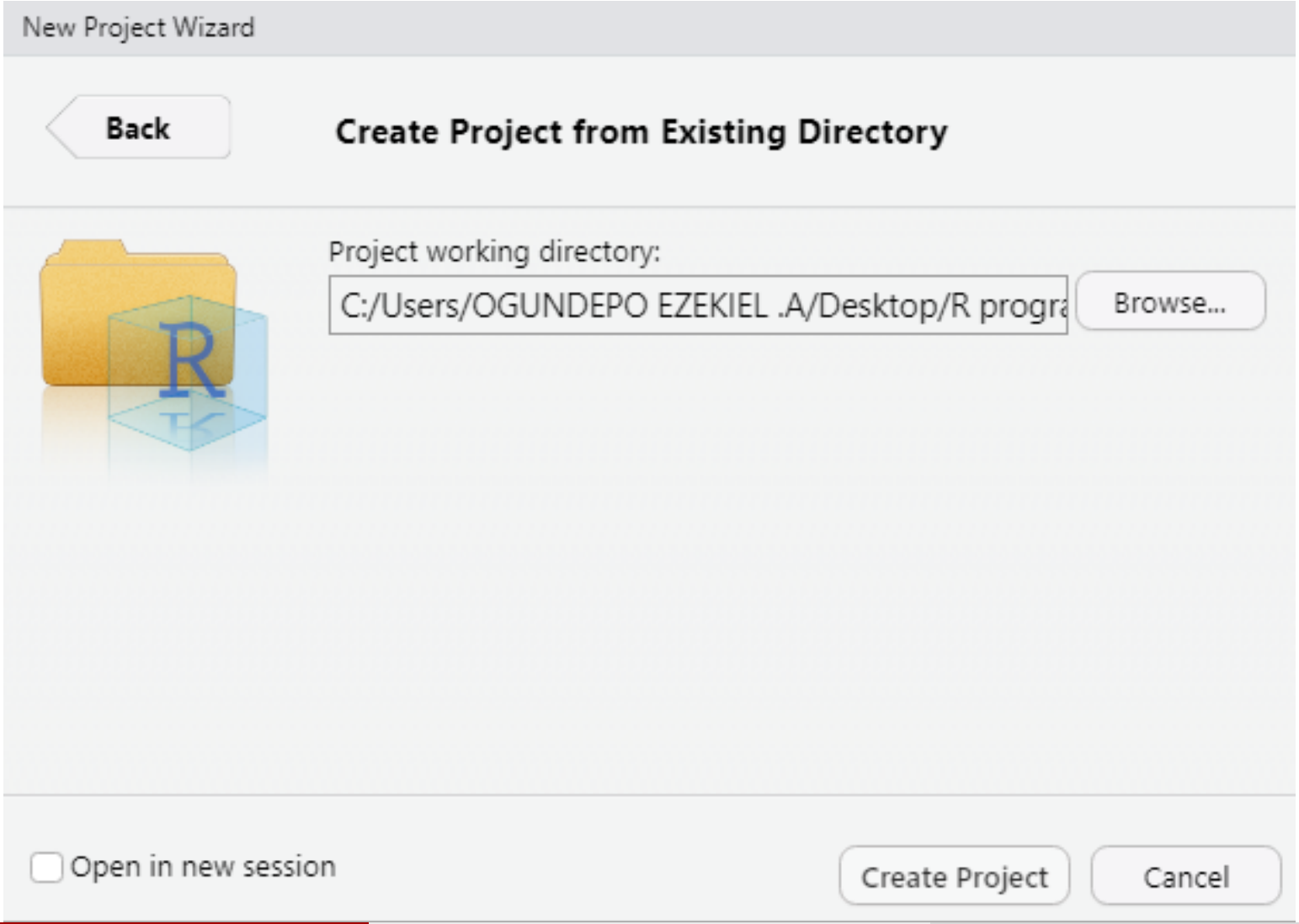


Figure 3: Example of Rstudio project

Creating a new R project

Creating a new R project

Click File → New Project, then choose Existing Directory:



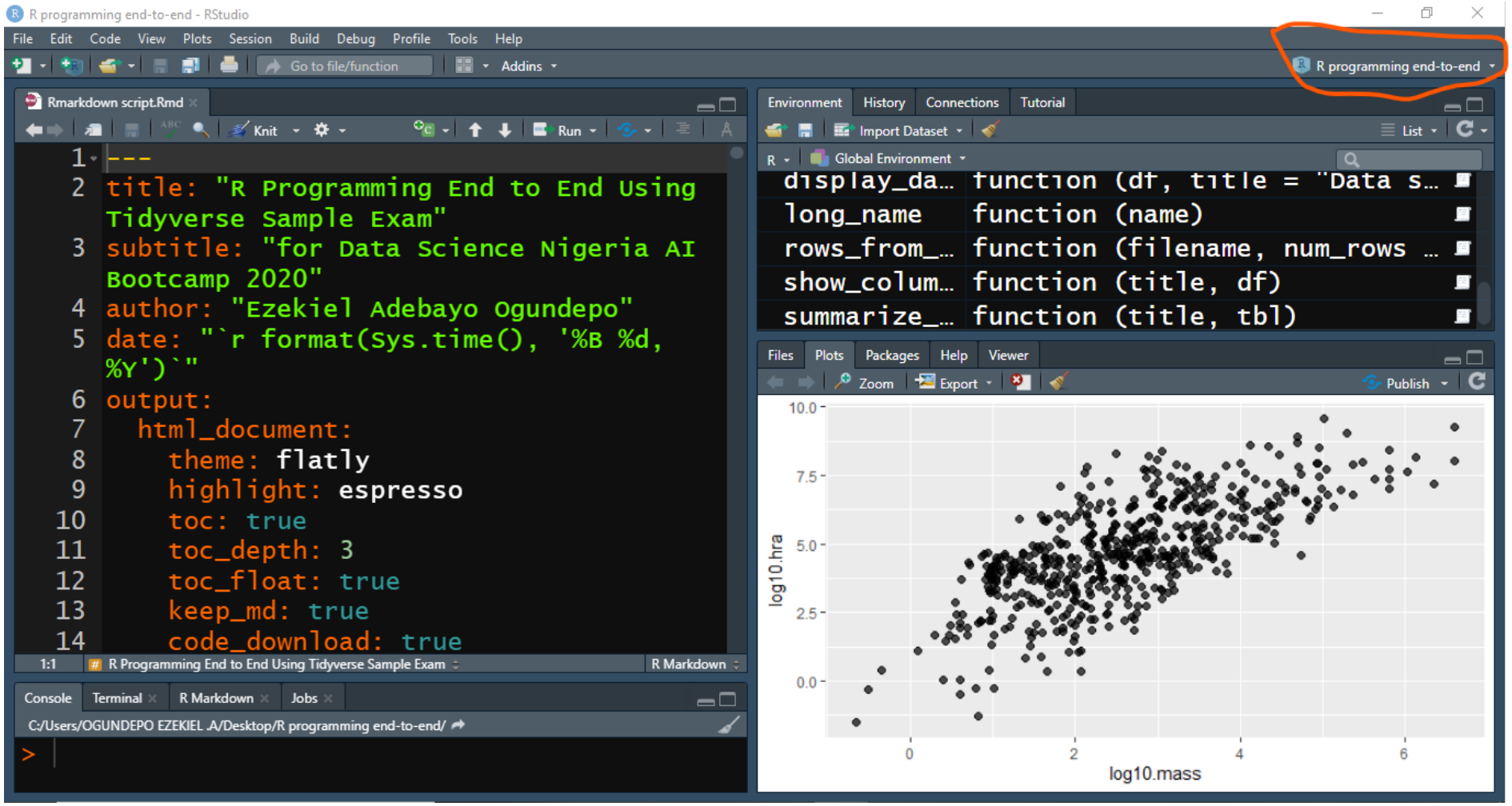


Figure 4: RStudio project

Hurray! We are in the RStudio project.

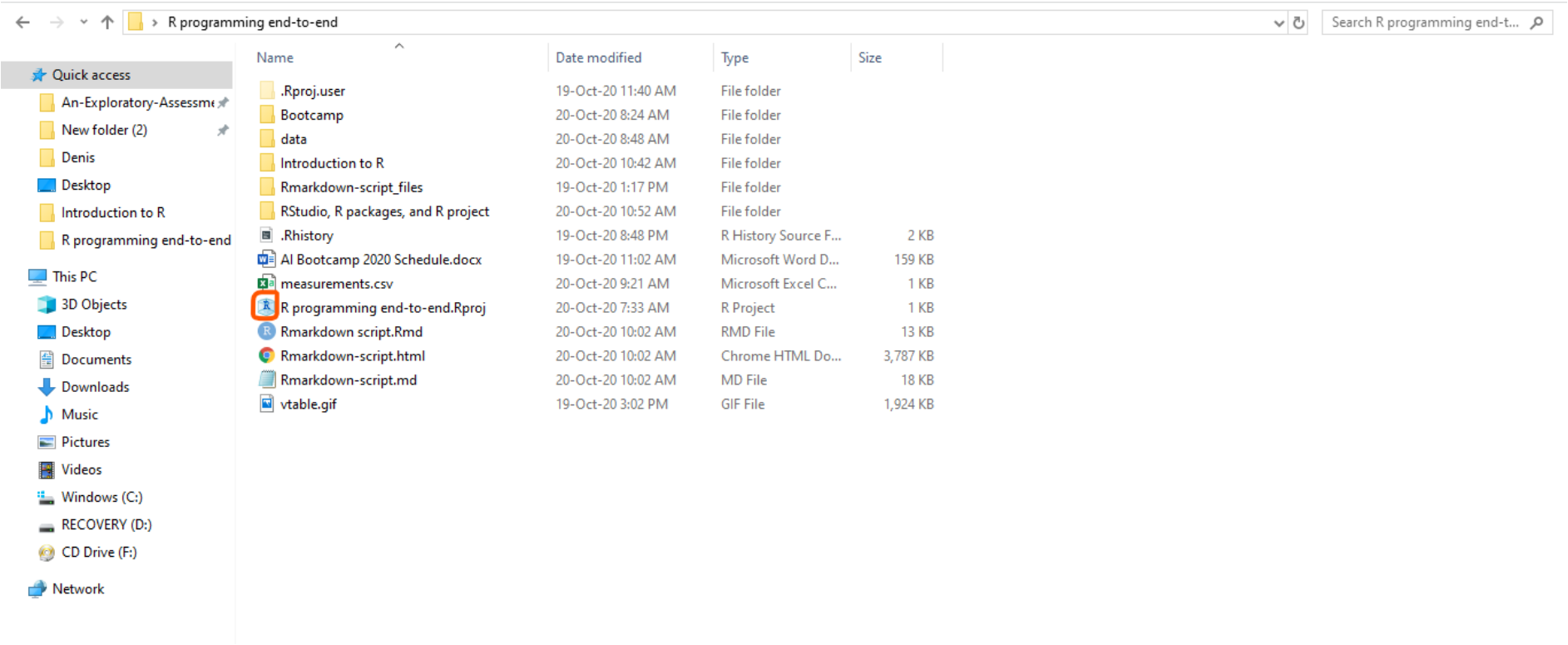


Figure 5: Meetup R project directory

Henceforth you will click **.Rproj** to open RStudio project.

Section 4

Summary

Summary

Data science workflow can be done in Rstudio, and we talked about R packages, how to install them and how to load them.

Summary

Data science workflow can be done in Rstudio, and we talked about R packages, how to install them and how to load them.

We also learnt about Rstudio project that enables us to organize our files i.e. keep data files, the script, save the outputs and by using only relative path.

Summary

Data science workflow can be done in Rstudio, and we talked about R packages, how to install them and how to load them.

We also learnt about Rstudio project that enables us to organize our files i.e. keep data files, the script, save the outputs and by using only relative path.

Everything you need is in one place, and cleanly separated from all the other projects that you are working on.

Thank you!

References

Wickham, H., & Grolemund, G. (2016). R for data science: import, tidy, transform, visualize, and model data. " O'Reilly Media, Inc.".