Creating R Packages

Heeju Lim

University of Connecticut
Statistical Computing

October 28, 2025

Introduction to R Packages

- **What is an R package?**
  - A collection of R functions, data, and documentation.
  - Used for sharing code, organizing projects, and making functions reusable.
- **Why Build a Package?**
  - Easier code maintenance.
  - Simplifies sharing with others.
  - Streamlines reproducible research.

Introduction to R Packages

Packages developed by others need to be installed. Often these are installed from either CRAN (Comprehensive R Archive Network) or GitHub.

- To install from CRAN

```
install.packages("packageName")
```

- To install from GitHub

```
remote::install_github("packageName")
```

- To install vignettes associated with packages on GitHub

```
remotes::install_github("packageName",buid_vignettes=TRUE)
```

Introduction to R Packages

The whole package can be loaded into memory using

```
library(packageName)
```

All functions in the packages can then be used by its functionName
For example, the package readxl has a dozen functions that help you get data out of Excel files and into R. To load all the functions into memory:

```
library(readxl)
```

Individual functions from a package can be loaded without having to load the entire library. This becomes important when writing your packages. Use de syntax:

```
packageName::functionName
```

For example, if you want to read a legacy excel file into R:

```
readxl::read_xls("filename.xls")
```

Why create your own R package?

- Keeps you organized
- Reduce copy and pasting of functions from project to project
- Allows to keep track of code and easily reuse it
- Allow you to easily share your code (e.g. through a repository on GitHub). Important for publications.
- Great option for documentation (`roxygen`, `rmarkdown`, `pkgdows`)
- Better understanding of how R works

Your own package will be organized in a way akin to folders on your computer. You will keep R code in one folder, data in another, documentation in another etc.

Pre-requisites: devtools, remortes & usethis

Three key packages you will need (if you don't have them already) are:

1. devtools
2. remotes
3. usethis

usethis is a package to aid in development of packages. To ckeck you have it installed, type the following line:

```
"usethis" %in% row.names(installed.packages())
```

A result of FALSE means you don't have it. Install it
The devtools package was split into remotes and usethis. However there are still a few cases in which you will need devtools. It is advisable to install all three.

```
install.packages(c("remotes","devtools","usethis"))
```

Pre-requisites: Rtools, XCode

You will also need to make sure you have another component installed (This is not an R package) and this component differs between operating sytems.

1. Windows: Rtools
2. Mac: XCode (free in app store)
3. Linux: install R development tools.

To check you have everything installed type in the console:

```
devtools::has_devel()
```

if the result is: `Your system is ready to build packages!`, then you have all required tools.

Pre-requisites: Other packages

- `Rmarkdown` package (to creating general documentation)
- `Knitr` package (to convert documentation into `html` or `pdf`)
- `Roxygen` package (for function documentation)

To install all at once:

```
1 install.packages(c("knitr","roxygen2","rmarkdown","devtools","
      remotes","usethis"))
```

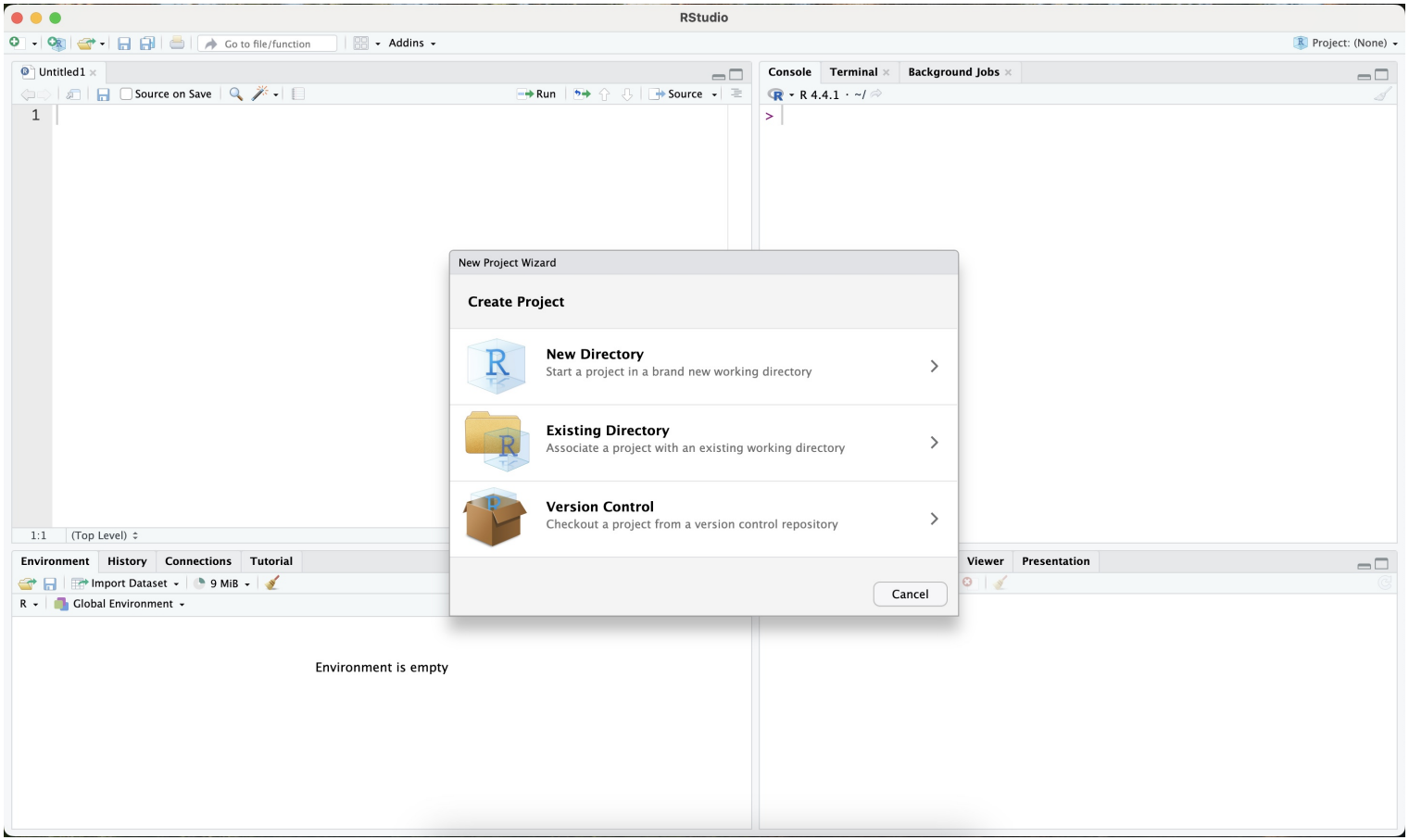
Naming your package

- Keep it concise, unique, but descriptive.
- only use letters, numbers.
- no underscores or hyphens.
- use abbreviations or acronyms (we should be good at that!)

Create your package

To create a project from scratch

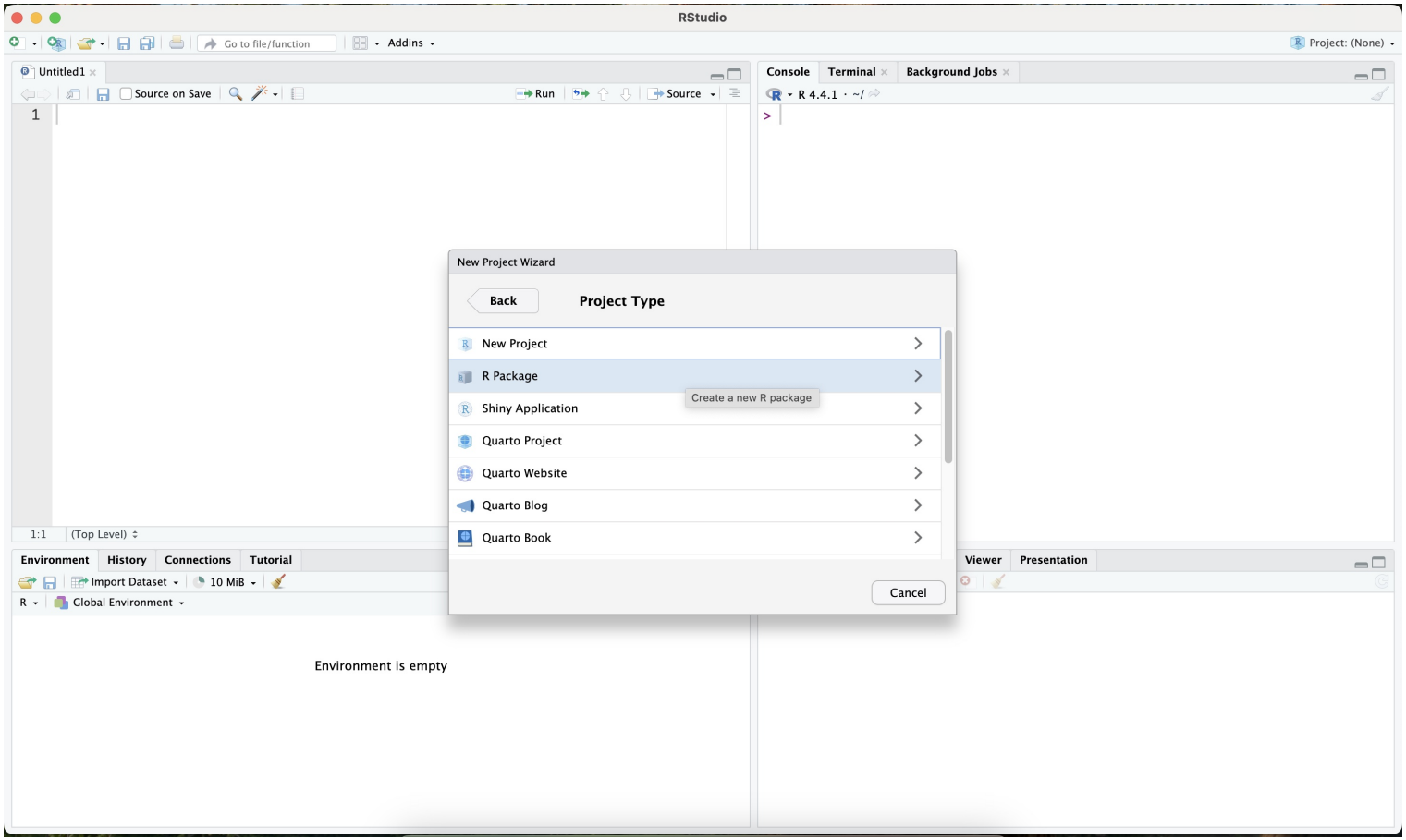
File $\longrightarrow$ New Project $\longrightarrow$ New Directory $\longrightarrow$ R Package



Create your package

To create a project from scratch

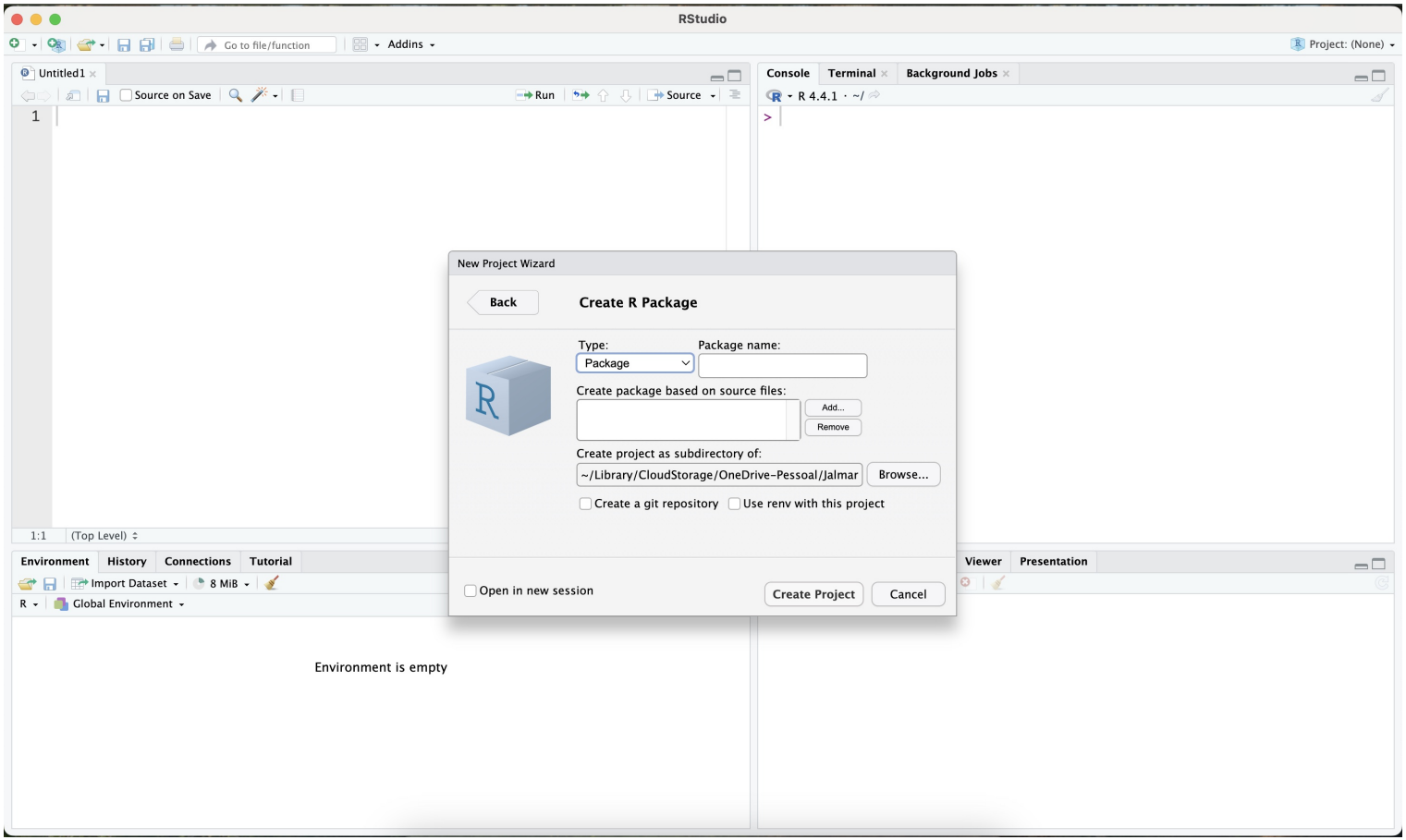
File $\longrightarrow$ New Project $\longrightarrow$ New Directory $\longrightarrow$ R Package



Create your package

To create a project from scratch

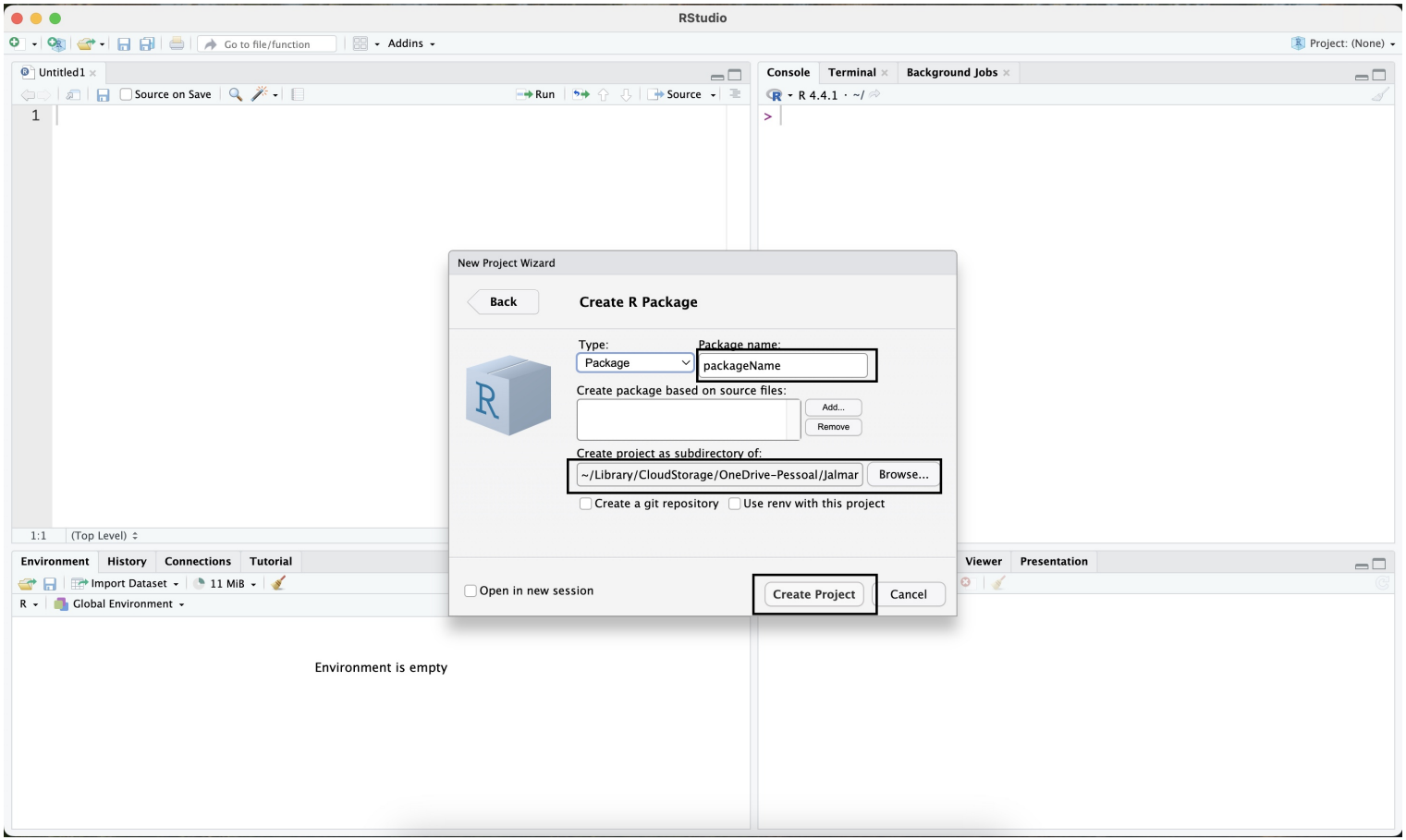
File $\longrightarrow$ New Project $\longrightarrow$ New Directory $\longrightarrow$ R Package



Create your package

To create a project from scratch

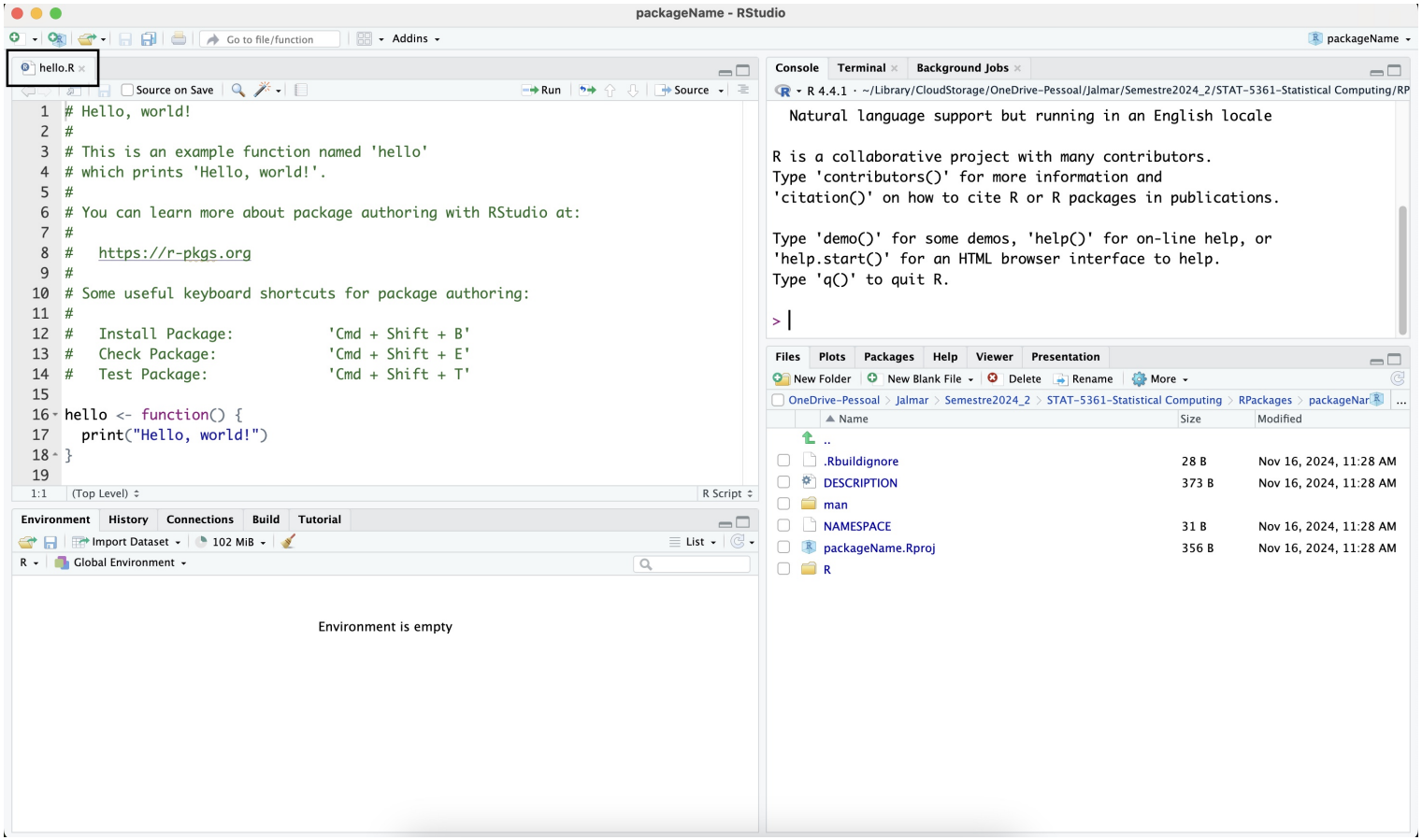
File ⟶ New Project ⟶ New Directory ⟶ R Package



Create your package

To create a project from scratch

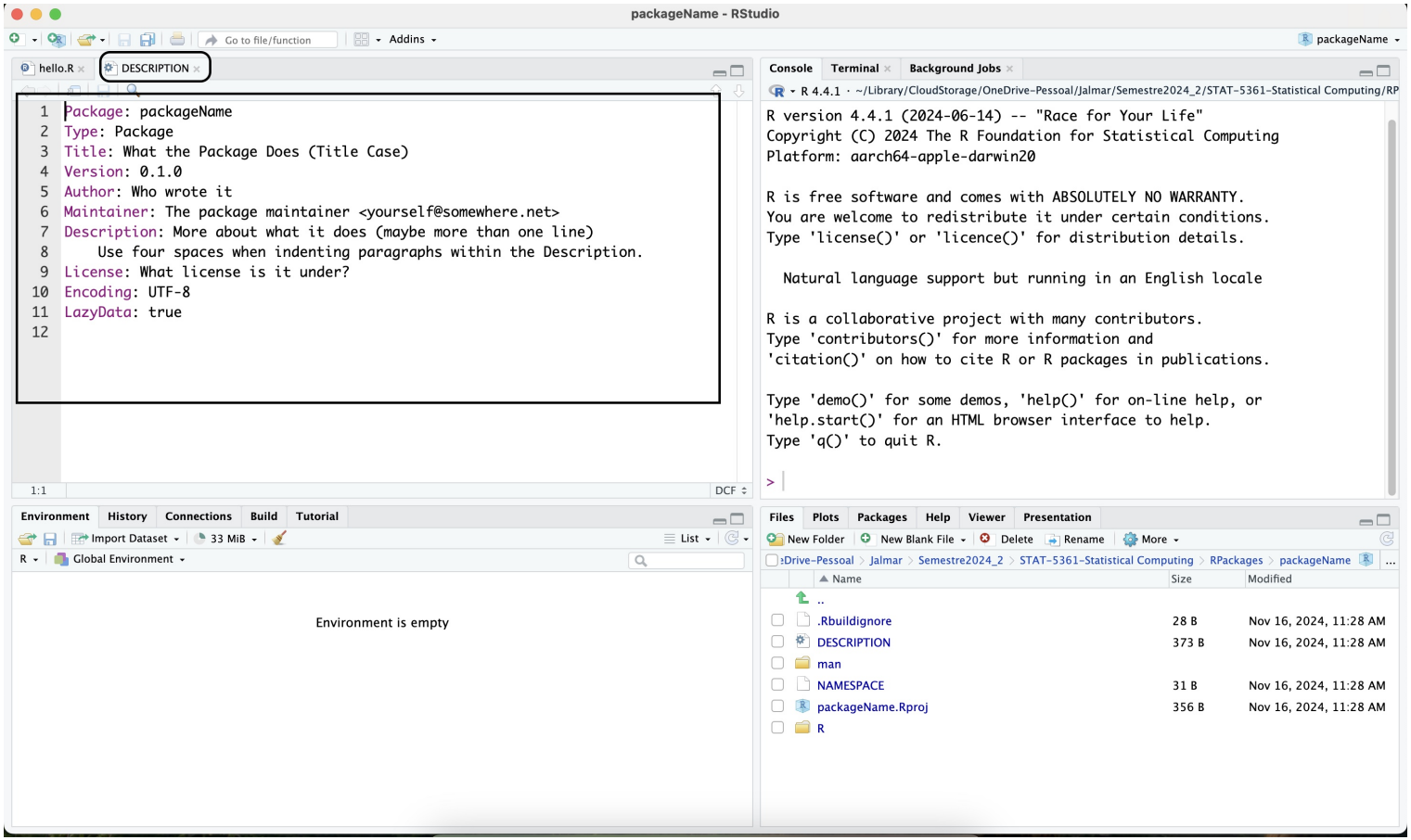
File $\longrightarrow$ New Project $\longrightarrow$ New Directory $\longrightarrow$ R Package



Create your package

To create a project from scratch

File $\longrightarrow$ New Project $\longrightarrow$ New Directory $\longrightarrow$ R Package

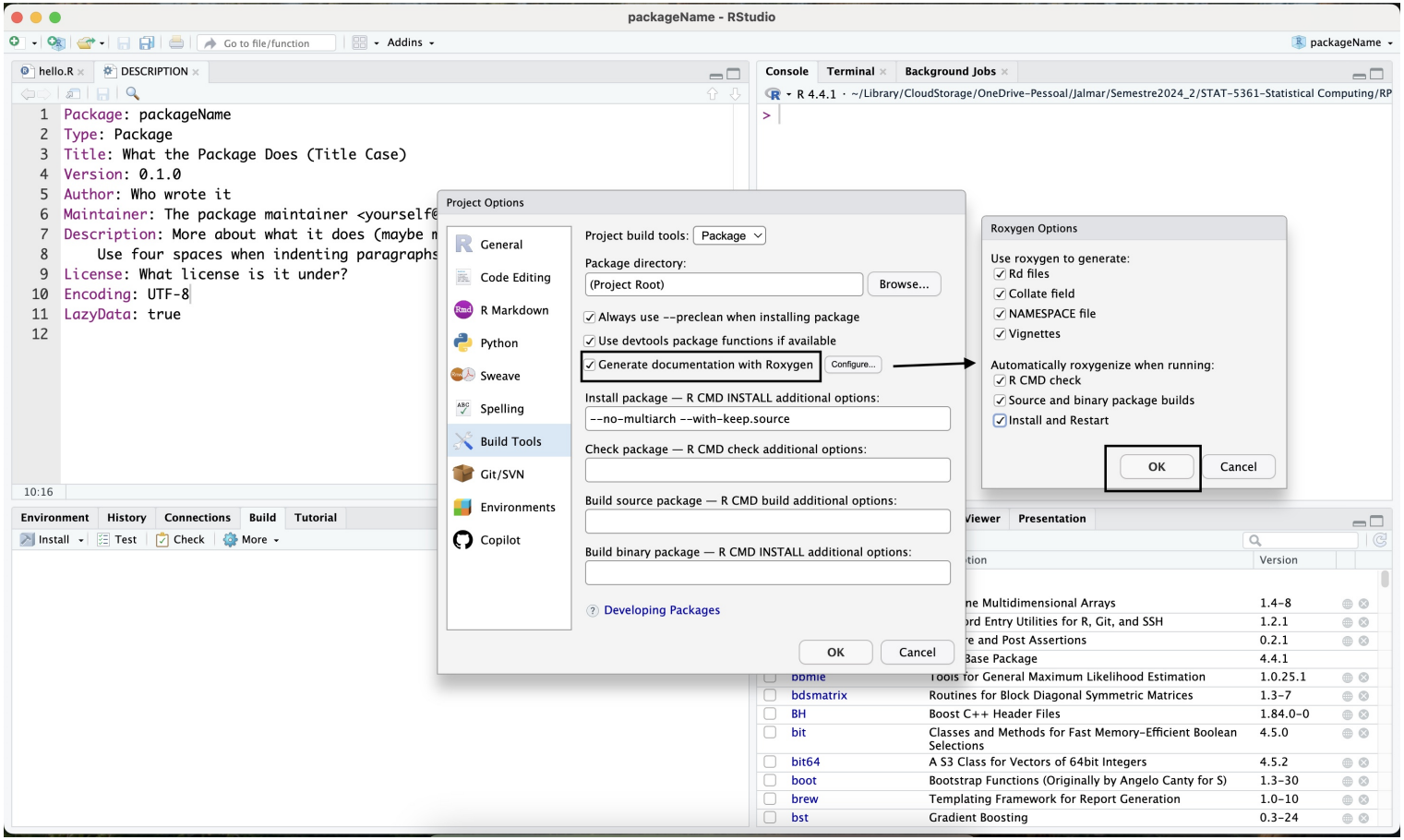


Package structure

- A DESCRIPTION file - Basic information about the package, authors, contributors, version, title, description, the type of license, and your packages dependencies (a list of other packages that your package depends on!)
- A NAMESPACE file - defines the functions in your packages made available to other users. It also defines external functions and packages that are important. This is automatically generated if using `roxygen2` for documentation so don't worry about this.
- An R/ directory where your `R` code will reside
- A man/ directory where functions documentation will reside (this is handled by `roxygen2`)
- An .rbuildignore file where you can specify which files you don't want to include in the package.

Project preferences

You will also need to change some preferences in the project options.
Tools $\longrightarrow$ Project Options $\longrightarrow$ Build tools.



Packages: Now what?

First you should edit the DESCRIPTION file to give some basic information about your packages. However, this can be done at anytime prior to sharing.

- Package: Enter the name of your package
- Title: Enter a single line describing your project
- Author: Enter the authors, email, and their roles e.g.
- Description: Paragraph describing your packages
- License: Licensing info

Packages: Workflow

At this point you are ready to enter the workflow loop:

1. Developed your code
2. Document it (Don't forget to @export)
3. Build your packages
4. See if the code does what you expect
5. Look at documentation using ? to see if it looks ok
6. Repeat

Packages: Workflow

At this point you are ready to enter the workflow loop:

1. Developed your code
2. Document it (Don't forget to @export)
3. Build your packages
4. See if the code does what you expect
5. Look at documentation using ? to see if it looks ok
6. Repeat

As you continue in this workflow loop you may realize you need to:

Packages: Workflow

At this point you are ready to enter the workflow loop:

1. Developed your code
2. Document it (Don't forget to @export)
3. Build your packages
4. See if the code does what you expect
5. Look at documentation using ? to see if it looks ok
6. Repeat

As you continue in this workflow loop you may realize you need to:

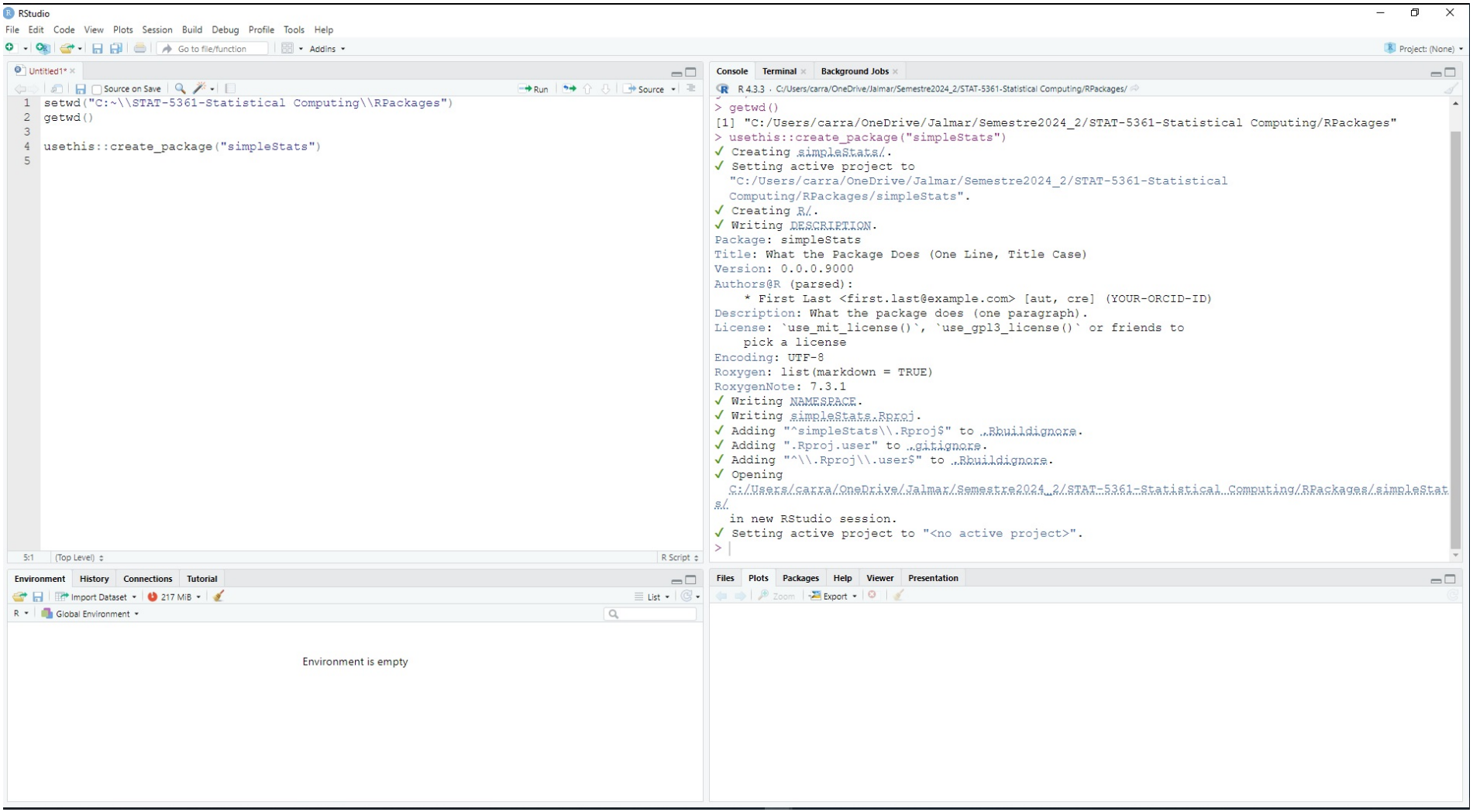
- Include data in your package. (`usethis::use_data`)
- Do some preprocessing of data that you don't want in your packages (just the result of the processing)(`usethis::use_data_raw`)
- Add an overview of how to use your package (`usethis::use_vignette`)
- Utilize functions in external packages (`usethis::use_package`)

Example

Lets create a package called simpleStats. It has a function called calc_stats which given a vector as input calculates the mean, range, and standard deviation. The result is a list containing these 3 results. Create a data set to be bunded with the package called `myData.Rdata`. Make `myData` be a numeric vector of size 100. Import the package `ggplot2` and plot the data.

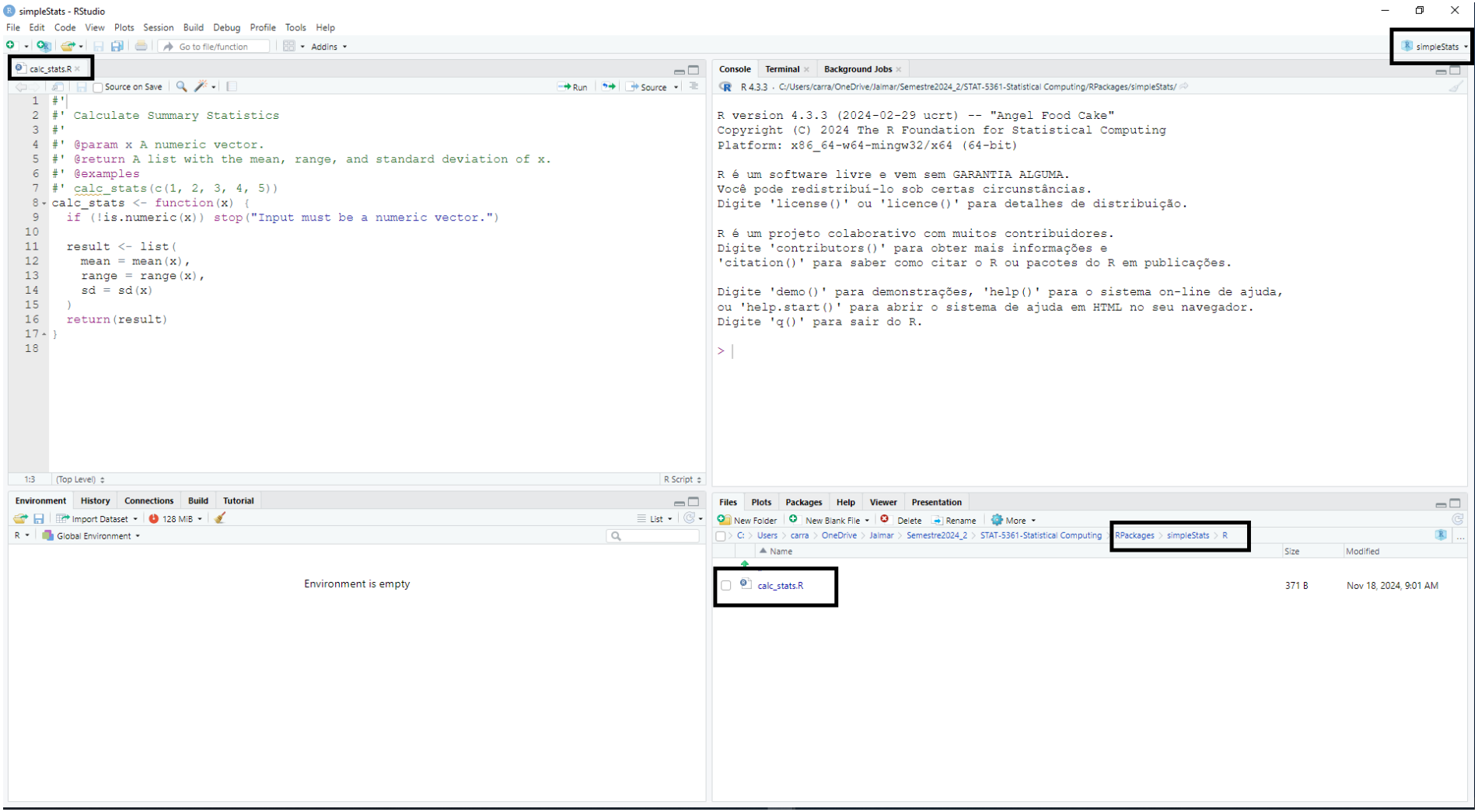
Example

File $\longrightarrow$ New Project $\longrightarrow$ New Directory $\longrightarrow$ R Package, or



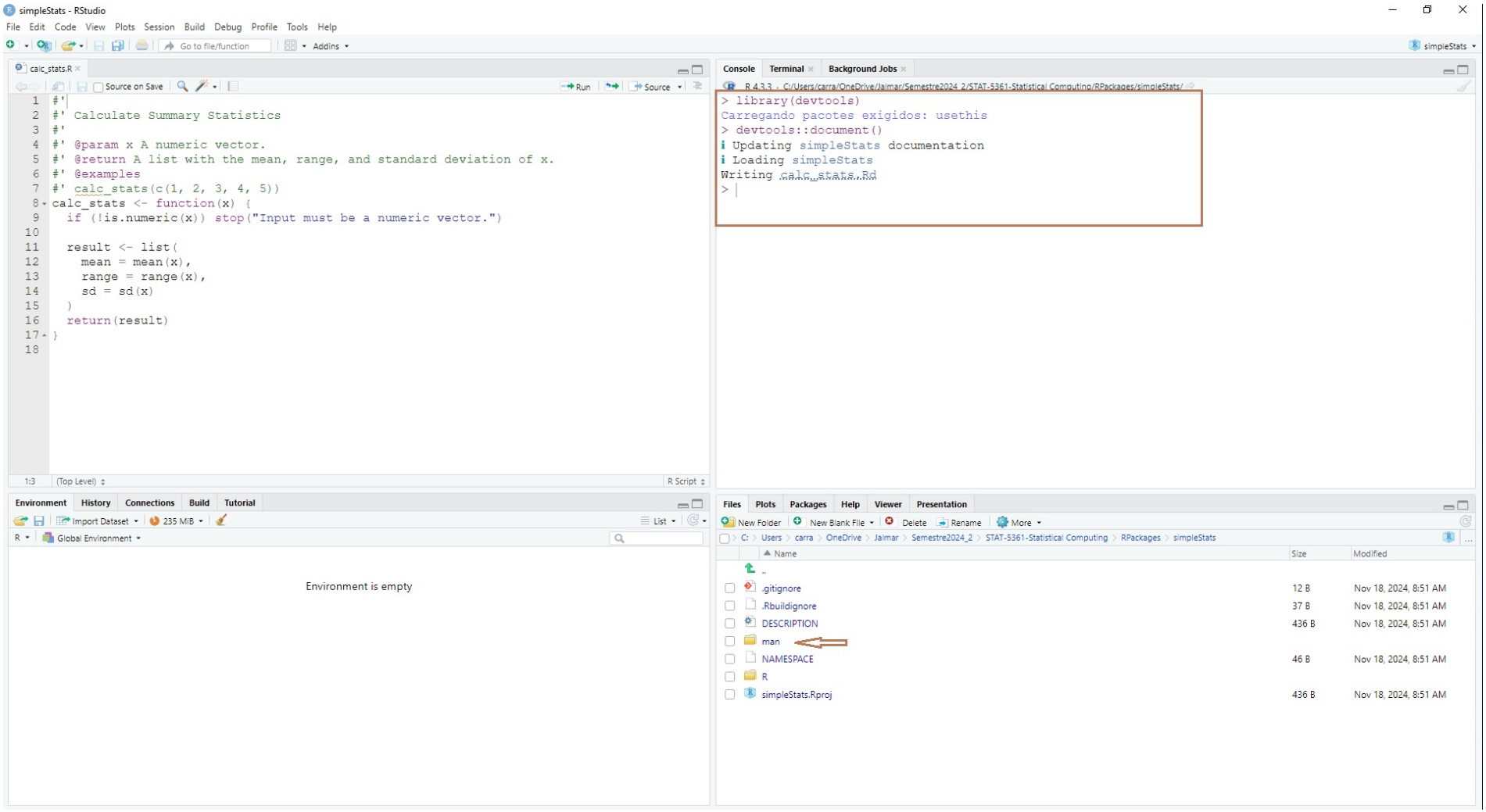
Example

File ⟶ New Project ⟶ New Directory ⟶ R Package, or



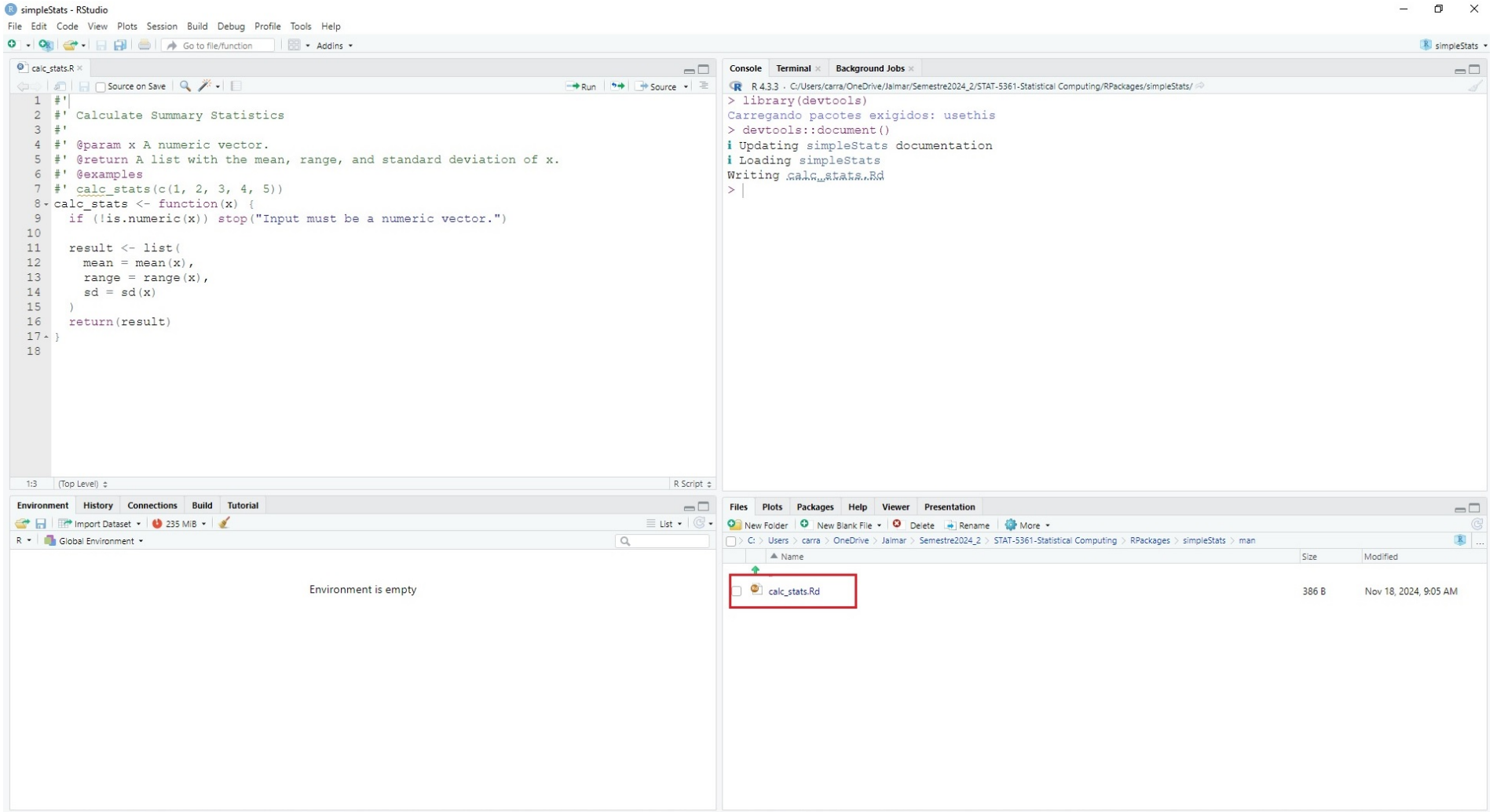
Example

Run `devtools::document()` to generate the documentation for the function.



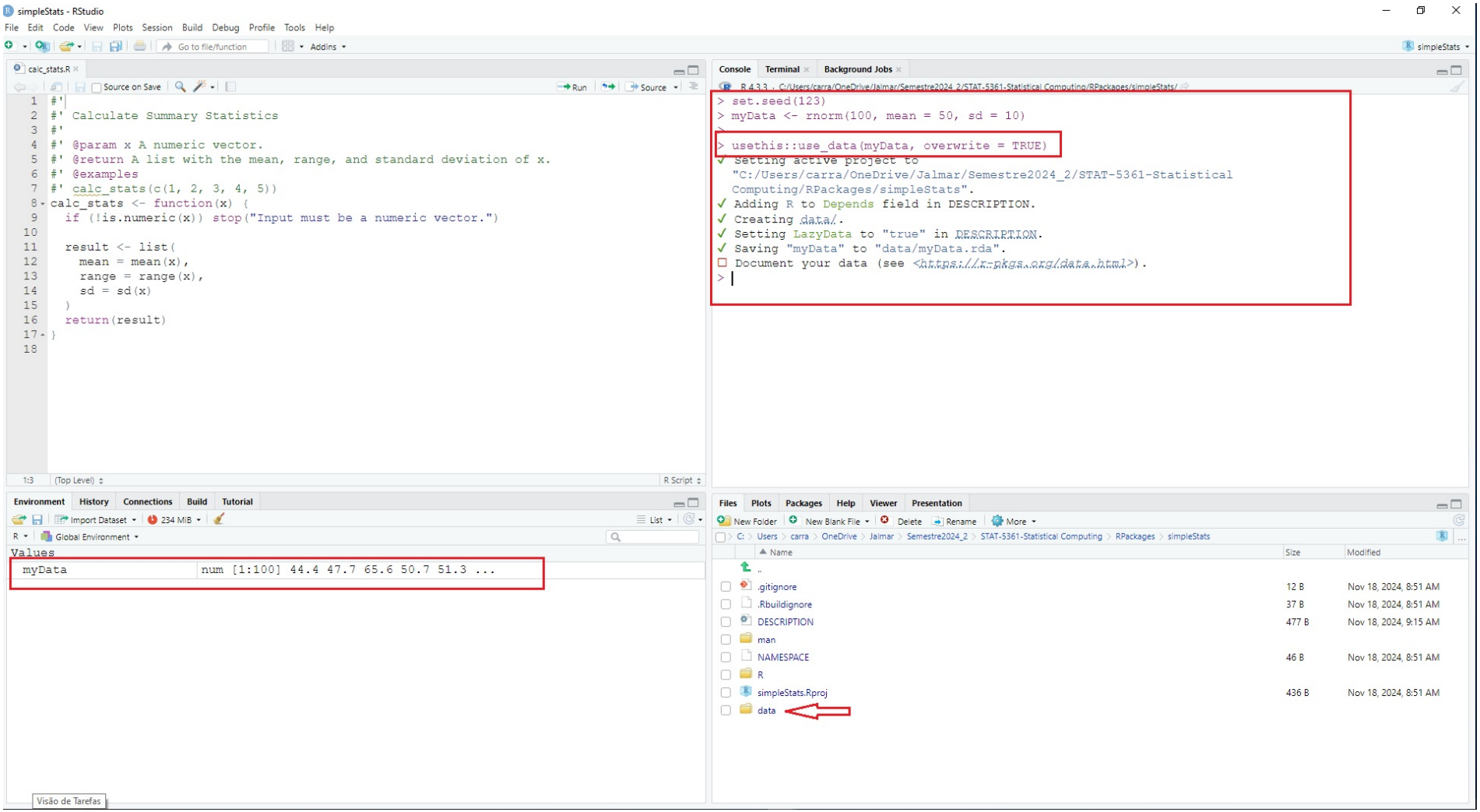
Example

Run `devtools::document()` to generate the documentation for the function.



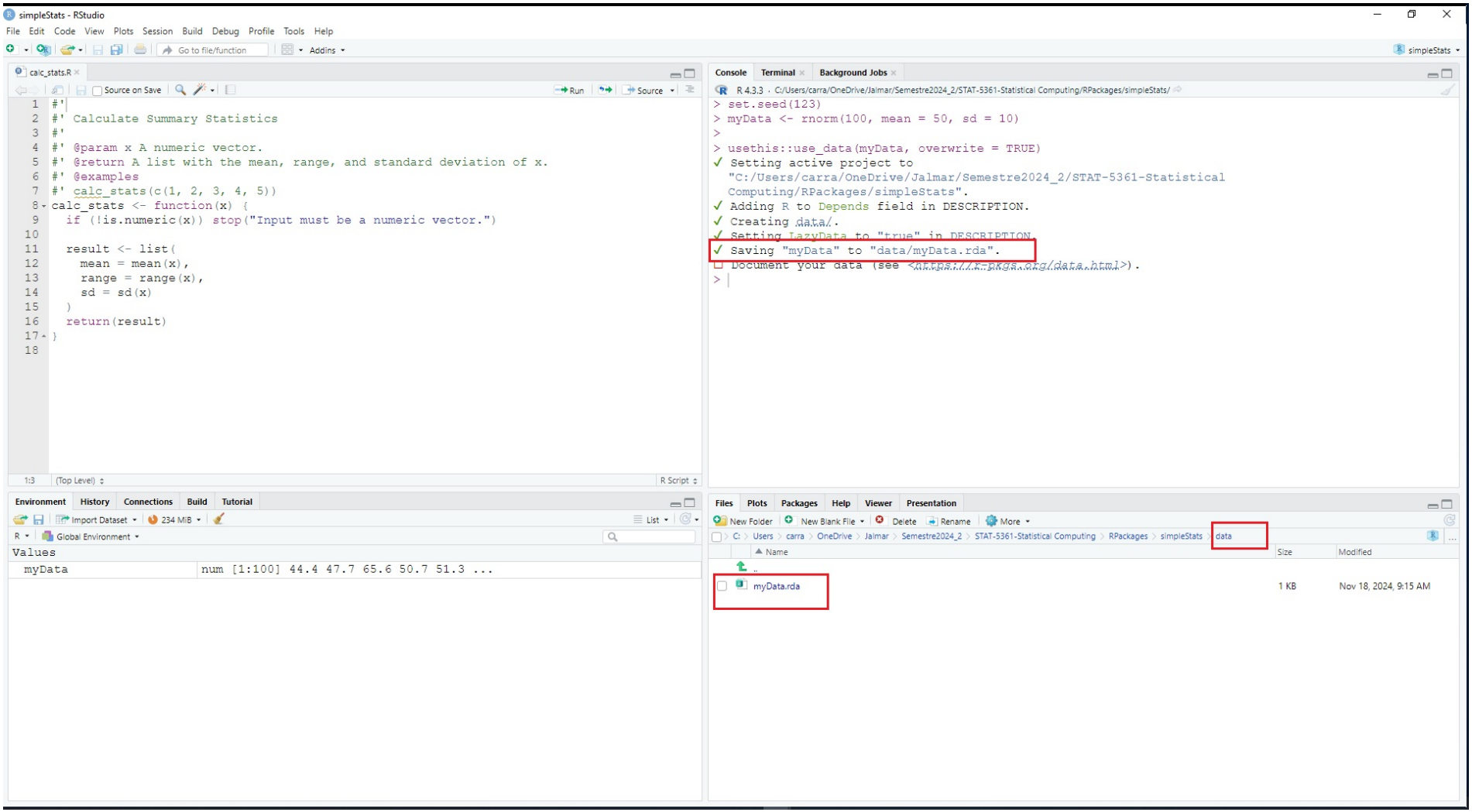
Example

Create a numeric vector `myData` and save it to the package's `data/` directory:



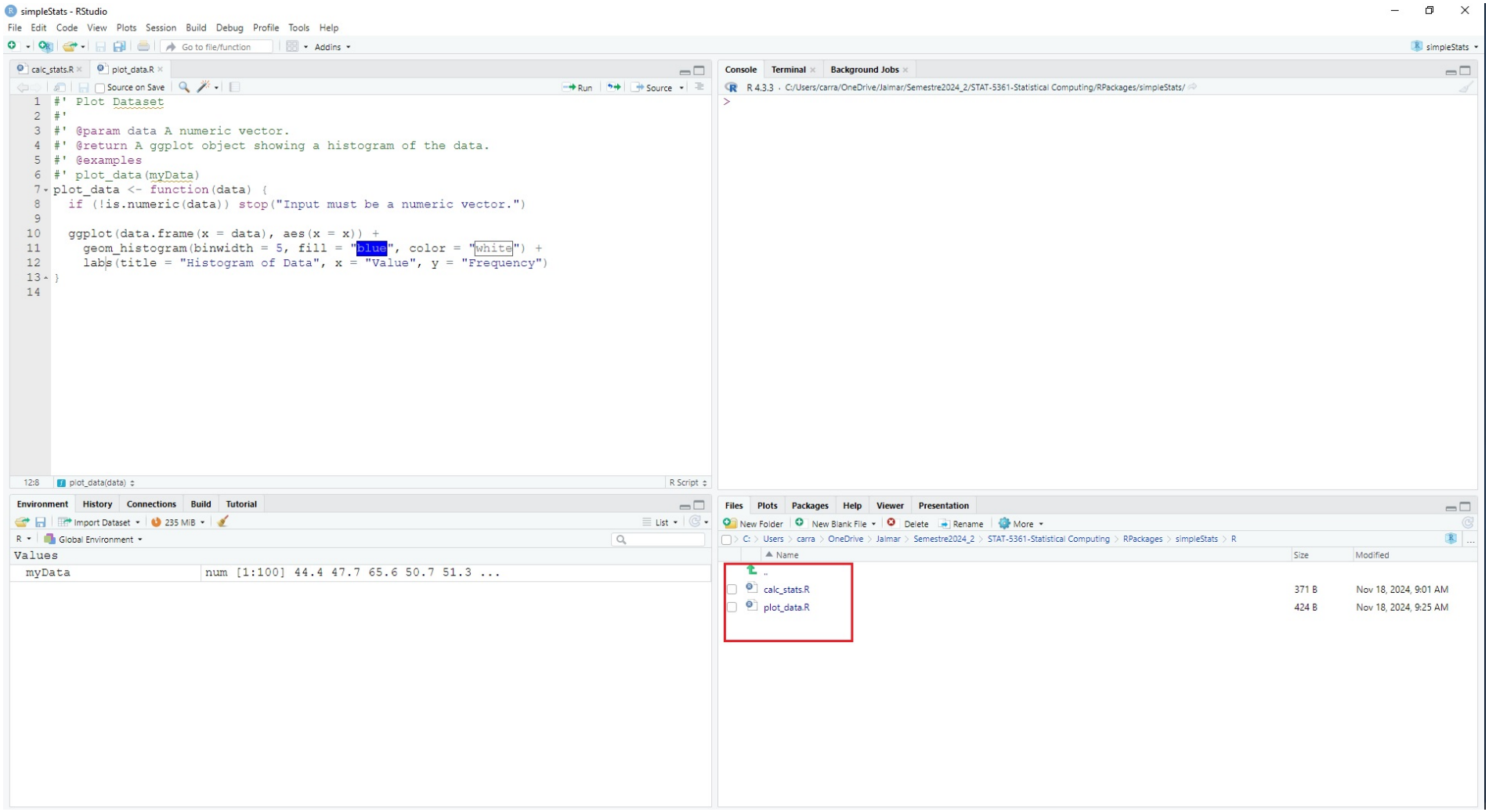
Example

Create a numeric vector `myData` and save it to the package's `data/` directory:



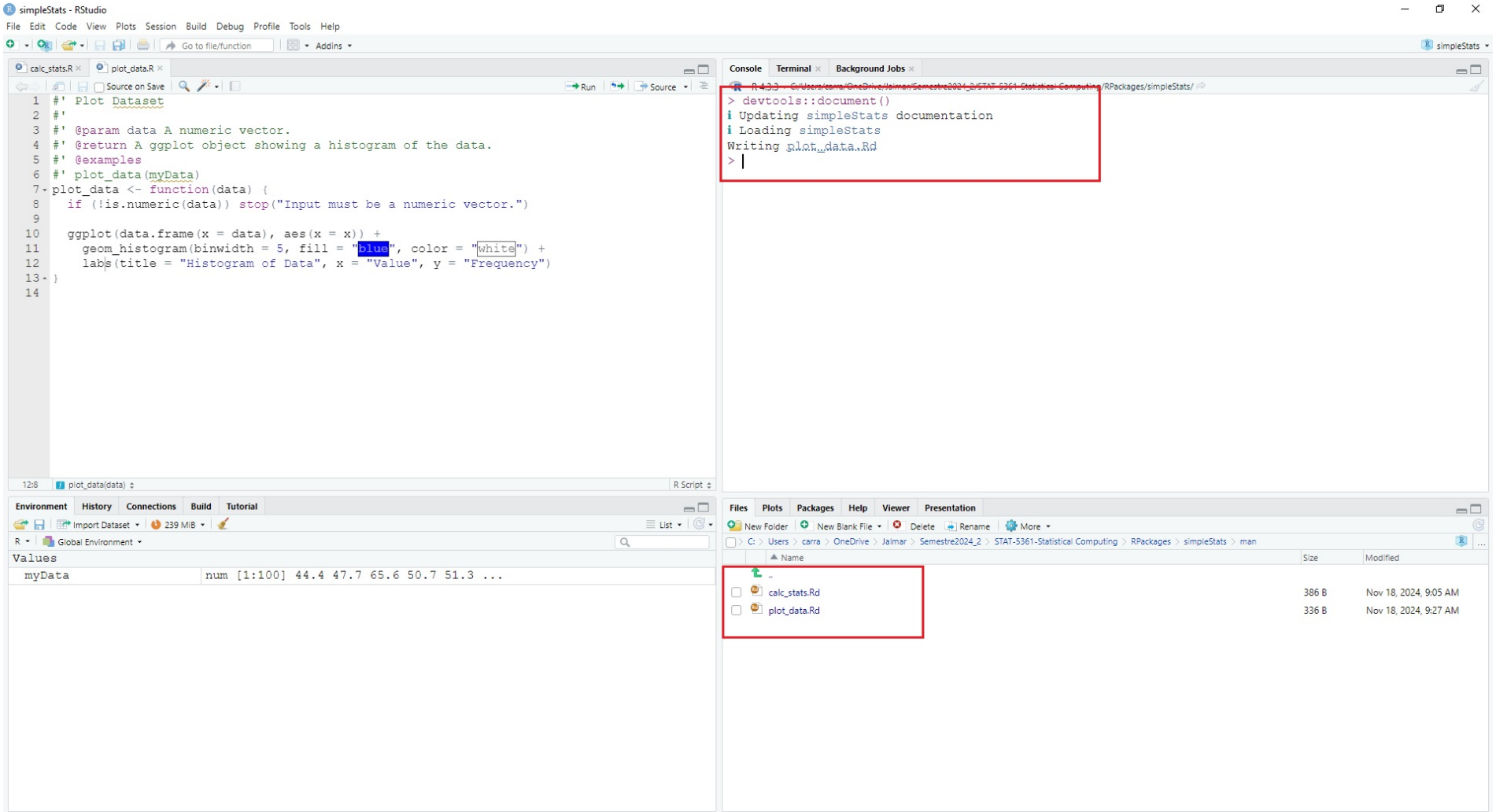
Example

Add a plotting function that uses `ggplot2`:



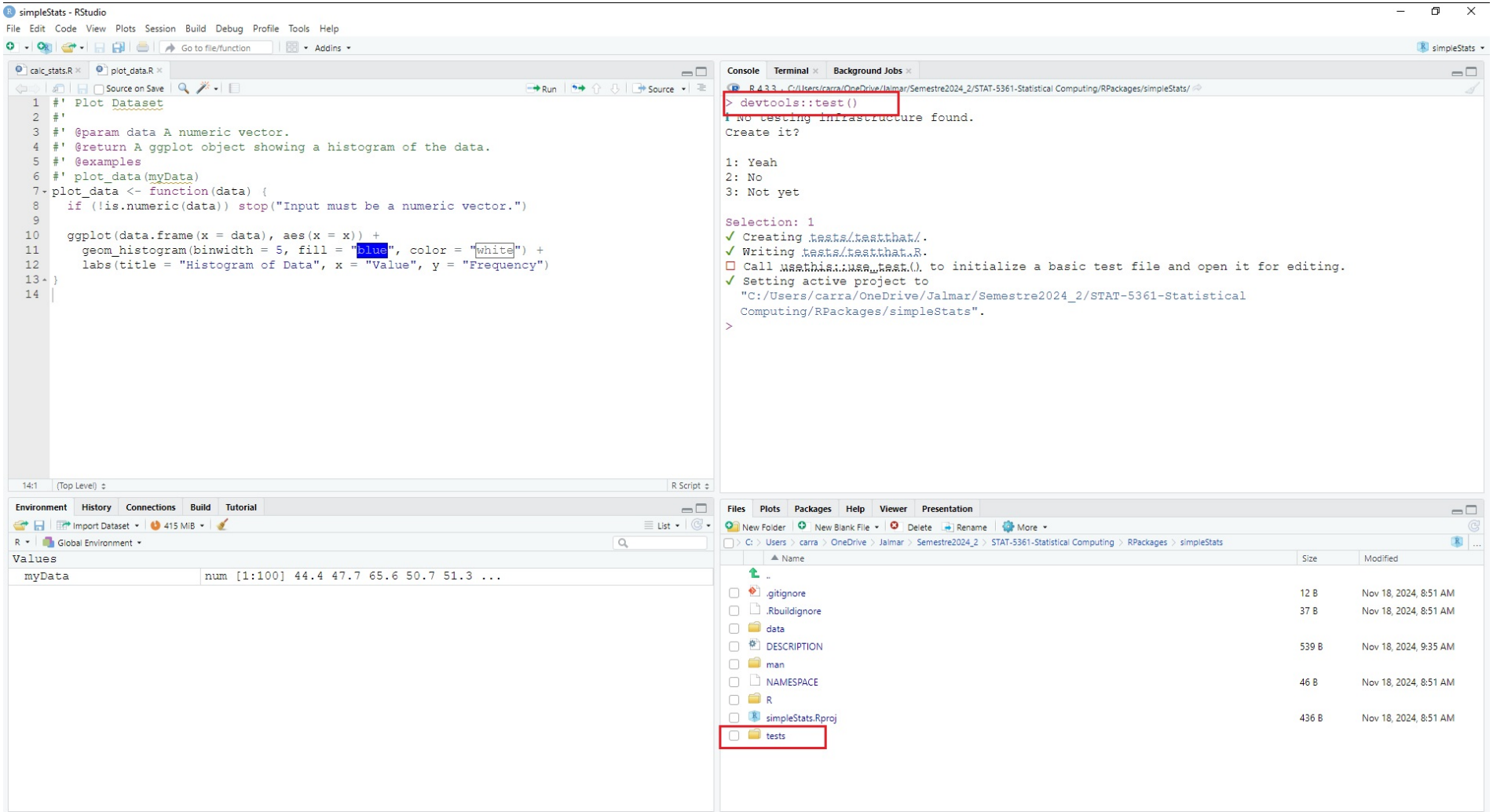
Example

Add a plotting function that uses `ggplot2`:



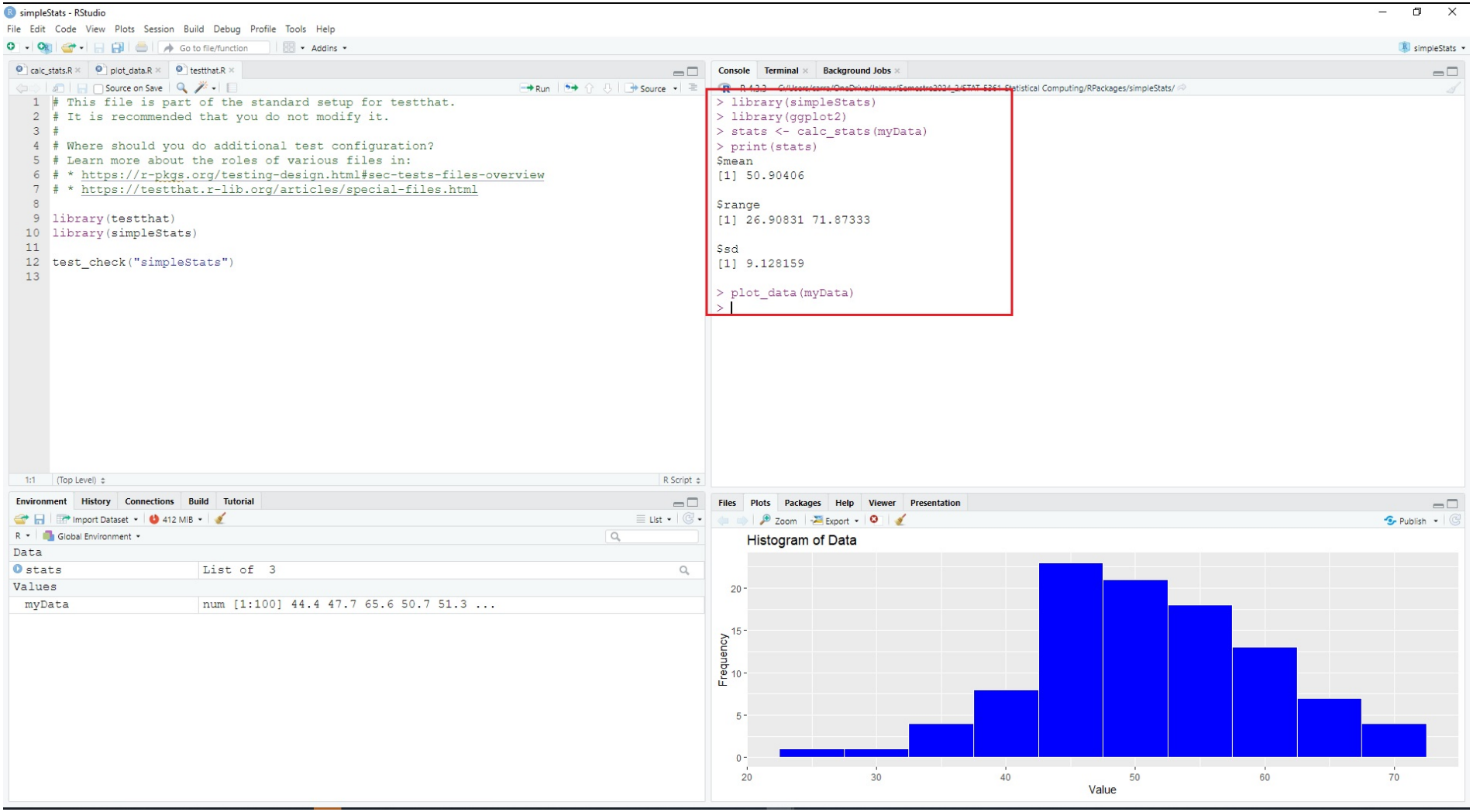
Example

Run the tests with:



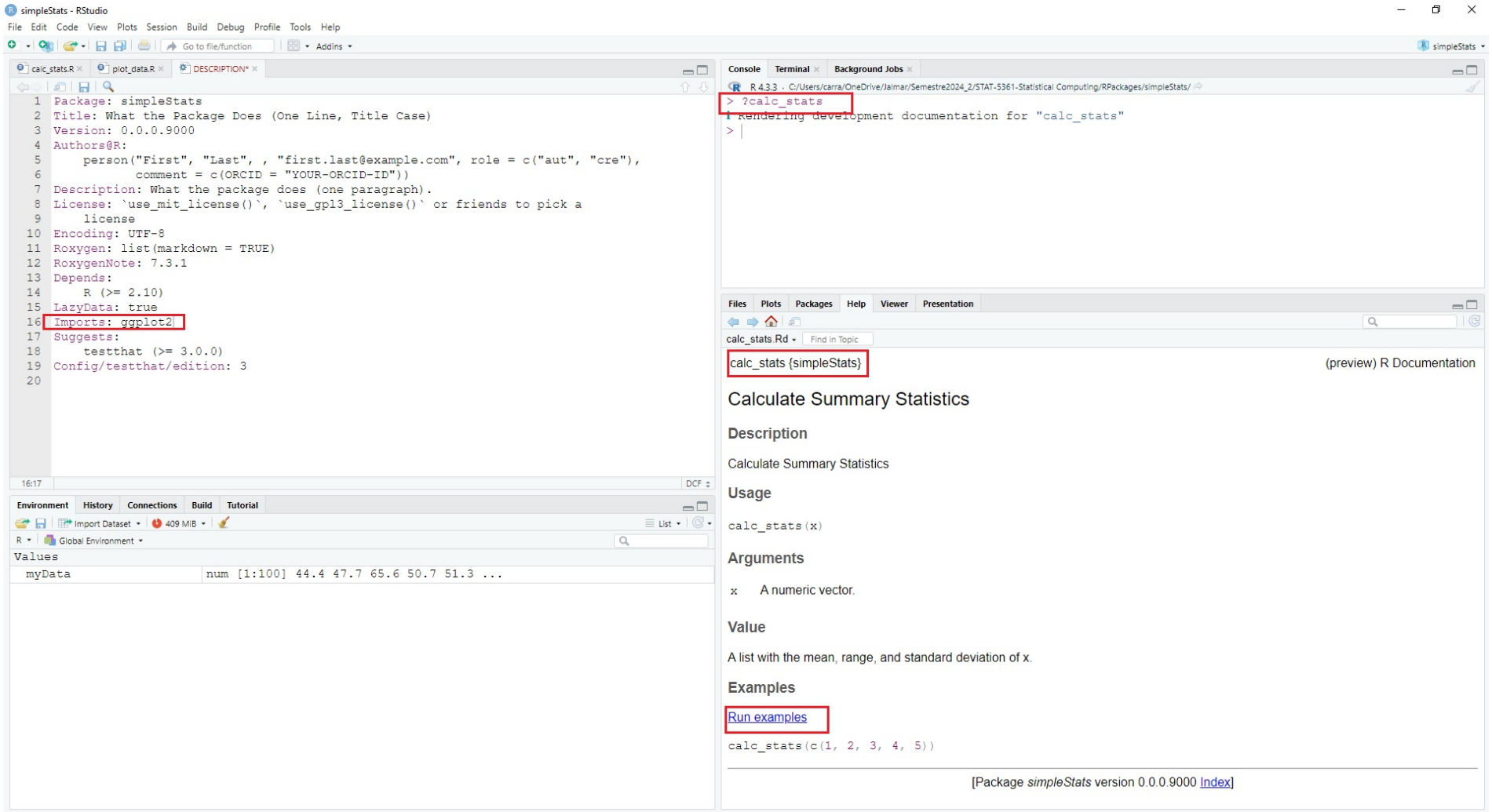
Example

Run the tests with:



Example

Run the tests with:



Adding Dependencies

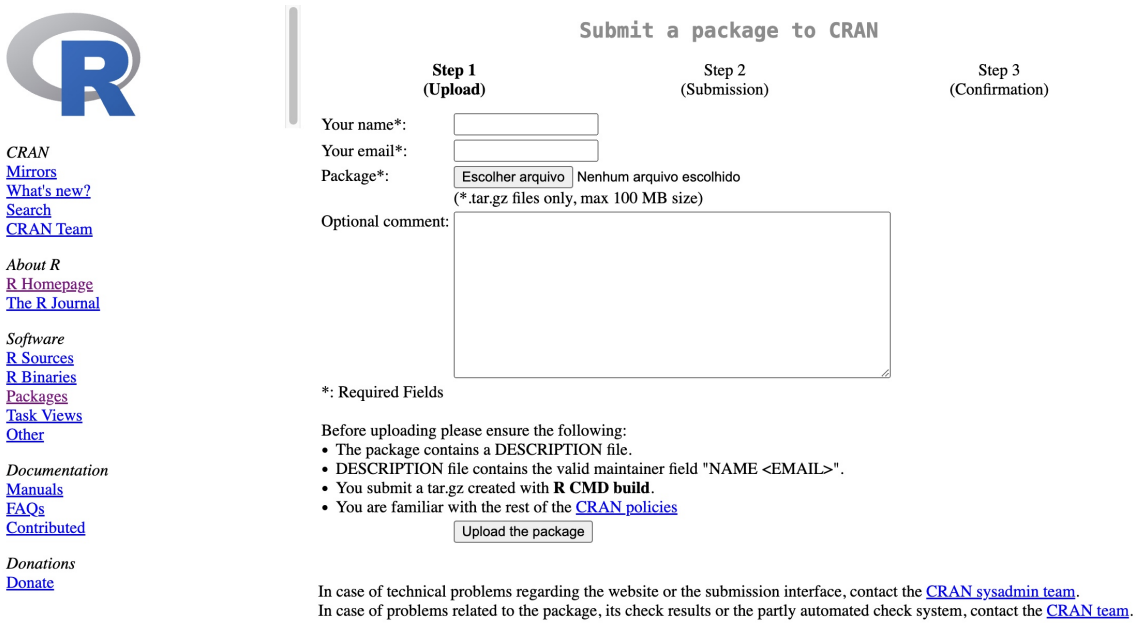
- **Manage Dependencies in the `DESCRIPTION` File**
  - Add packages under `Imports` (for required packages) or `Suggests` (for optional)

Creating Vignettes and README

- **Add Vignettes for Detailed Examples:**
  `usethis::use_vignette("Introduction")`
- **README File:**
  - Use `usethis::use_readme_rmd()` to add a README with usage instructions and examples

Sharing Your Package

- **GitHub**
  - Use `usethis::use_github()` to upload the package
- **CRAN Submission**
  - Review CRAN policies
  - Use `devtools::check()` to ensure the package meets CRAN requirements



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Donations
Donate

Submit a package to CRAN

Step 1 (Upload) | Step 2 (Submission) | Step 3 (Confirmation)

Your name*:
Your email*:
Package*: Escolher arquivo Nenhum arquivo escolhido
(*.tar.gz files only, max 100 MB size)
Optional comment:

*: Required Fields

Before uploading please ensure the following:
- The package contains a DESCRIPTION file.
- DESCRIPTION file contains the valid maintainer field "NAME <EMAIL>".
- You submit a tar.gz created with **R CMD build**.
- You are familiar with the rest of the CRAN policies

Upload the package

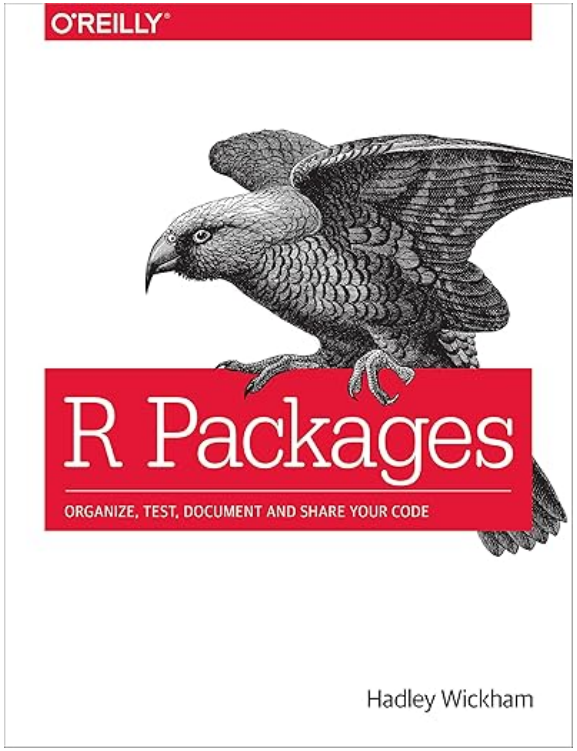
In case of technical problems regarding the website or the submission interface, contact the CRAN sysadmin team.
In case of problems related to the package, its check results or the partly automated check system, contact the CRAN team.

Homework

Select a function developed during the Statistical Computing course, create an R package for it, and publish the package in a GitHub repository.

- Provide the GitHub link to your package for heeju.lim@uconn.edu.
- Provide a script file with R code to install your package and demonstrate the use of its functions.

Good reference!



www.r-pkgs.org

Q&A / Discussion

Questions?