

A home for your functions:

CREATING AN R PACKAGE

CORINNE LEOPOLD

DATA SCIENTIST AT IBM



corinnovation



corinneleopold

“

PACKAGES

ARE THE FUNDAMENTAL UNITS OF REPRODUCIBLE
R CODE. THEY INCLUDE **REUSABLE** R FUNCTIONS,
THE **DOCUMENTATION** THAT DESCRIBES HOW
TO USE THEM, AND **SAMPLE DATA**.

”

Hadley Wickham, in *R Packages*

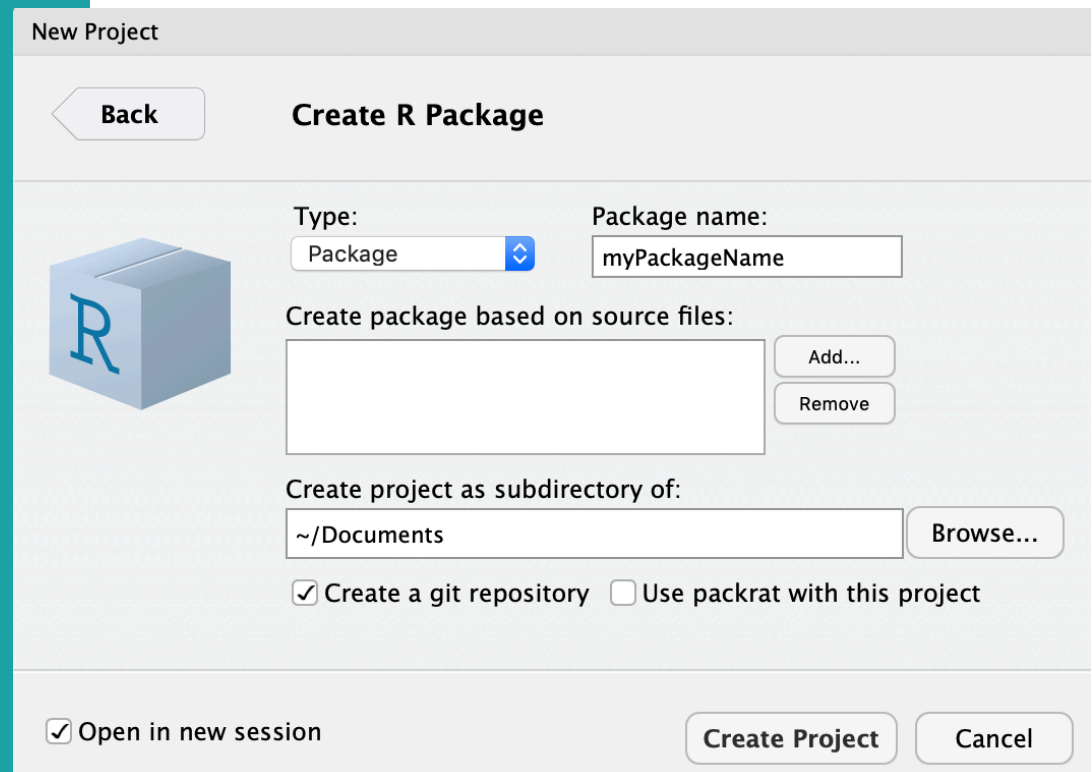
YOU SHOULD CREATE A PACKAGE IF YOU...

- Want to easily reuse code
- Spend time copying and pasting functions across files
- Need to organize your code in a central place
- Collaborate w/ others on ongoing projects
- Want to standardize your code (e.g. consistent plot themes)

STEP #1

In RStudio, go to:

File → New Project
→ New Directory
→ R Package



The image shows the 'New Project' dialog box in RStudio. The title bar says 'New Project'. Inside, there's a 'Back' button on the left. The main heading is 'Create R Package'. On the left side of the dialog is a 3D blue cube with a white 'R' on it. To the right of the cube, there are two input fields: 'Type:' with a dropdown menu showing 'Package' and a blue arrow, and 'Package name:' with a text box containing 'myPackageName'. Below these is a section 'Create package based on source files:' with a large empty text box and two buttons: 'Add...' and 'Remove'. Underneath is 'Create project as subdirectory of:' with a text box showing '~/Documents' and a 'Browse...' button. At the bottom of this section are two checkboxes: 'Create a git repository' (checked) and 'Use packrat with this project' (unchecked). At the very bottom of the dialog are three buttons: 'Open in new session' (checked), 'Create Project', and 'Cancel'.

New Project

Back

Create R Package

Type: Package

Package name: myPackageName

Create package based on source files:

Add...

Remove

Create project as subdirectory of: ~/Documents

Browse...

☒ Create a git repository ☐ Use packrat with this project

☒ Open in new session

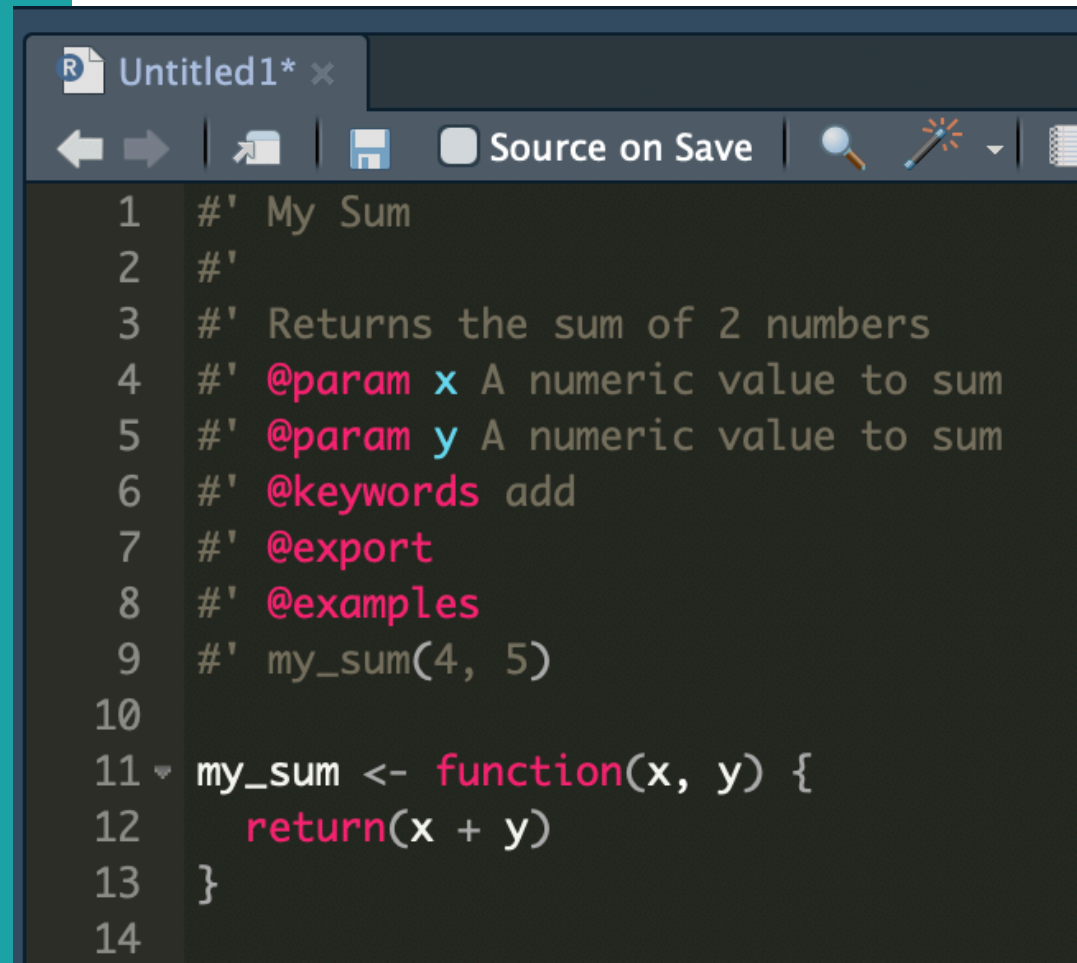
Create Project

Cancel

STEP #2

Add your functions to
new R scripts

Add documentation via
roxygen comments



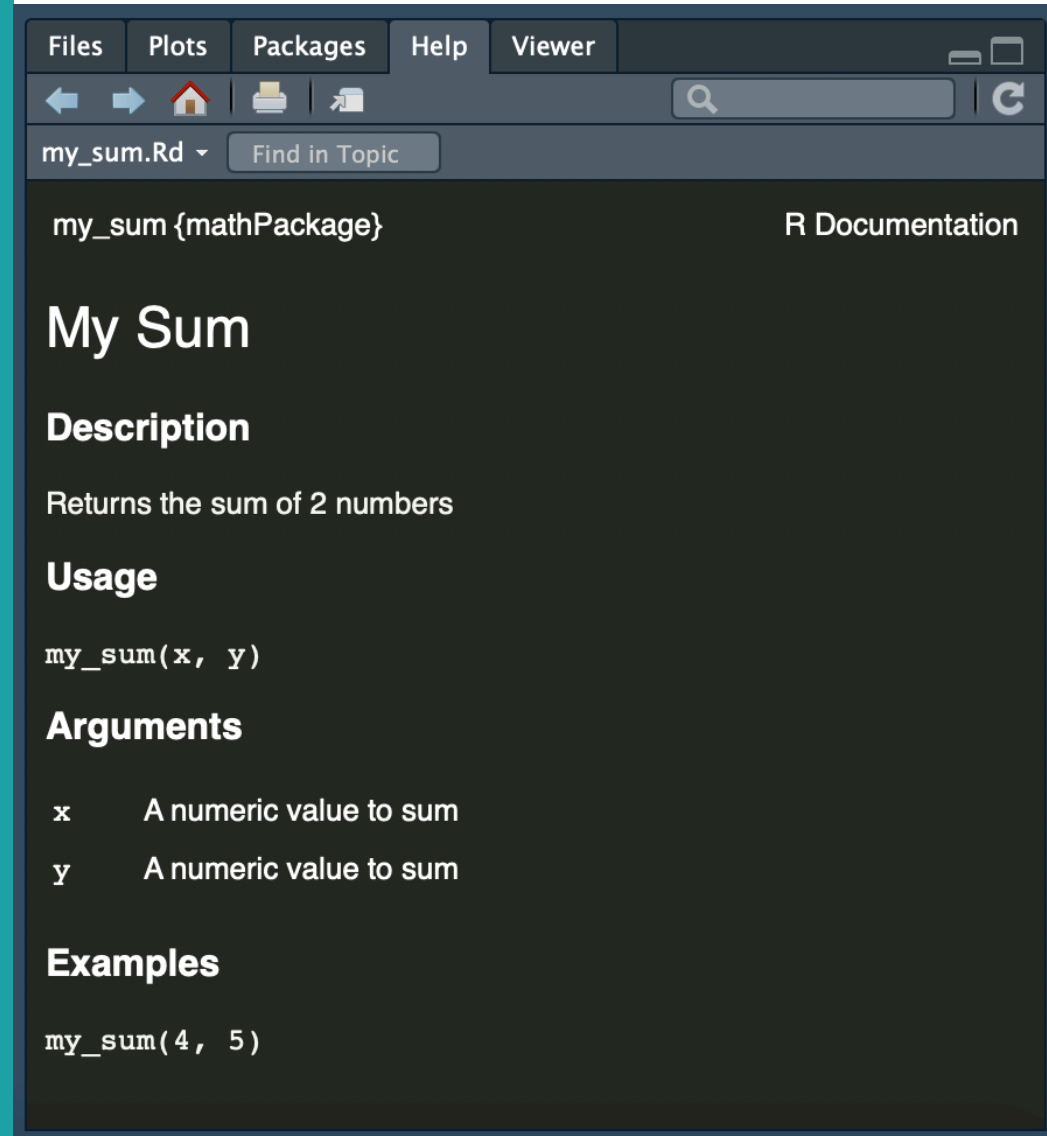
The screenshot shows an R script editor window titled 'Untitled1*'. The script contains a function definition for 'my_sum' with roxygen comments. The comments are: '# My Sum', '#', '# Returns the sum of 2 numbers', '# @param x A numeric value to sum', '# @param y A numeric value to sum', '# @keywords add', '# @export', and '# @examples'. The function body is: 'my_sum <- function(x, y) {', ' return(x + y)', '}'.

```
1 #' My Sum
2 #'
3 #' Returns the sum of 2 numbers
4 #' @param x A numeric value to sum
5 #' @param y A numeric value to sum
6 #' @keywords add
7 #' @export
8 #' @examples
9 #' my_sum(4, 5)
10
11 my_sum <- function(x, y) {
12   return(x + y)
13 }
14
```

STEP #3

Generate .Rd files via
`devtools::document()`

View your docs via
`?<function>`



The screenshot shows the RStudio interface with the help viewer open. The viewer displays the documentation for the `my_sum` function from the `mathPackage`. The documentation includes a title, a description, usage instructions, arguments, and examples.

```
my_sum {mathPackage} R Documentation
```

My Sum

Description

Returns the sum of 2 numbers

Usage

```
my_sum(x, y)
```

Arguments

<code>x</code>	A numeric value to sum
<code>y</code>	A numeric value to sum

Examples

```
my_sum(4, 5)
```

STEP #4

Test via `devtools::check()`,
then use your package!

```
> library(mathPackage)
> my_sum(10, 20)
[1] 30
>
```





thank
you!

CORINNE LEOPOLD



corinnovation



corinneleopold