Absolutely — here’s a **complete, no-nonsense manual** for when you’re **adding or changing a function** in an R package that's under Git version control (e.g., hosted on GitHub).

You’ll learn:

1. 🔧 Where to put the function
2. 📝 How to document it
3. 🧪 How to test it
4. 🔁 How to regenerate package files
5. 🚀 How to commit + push to GitHub
6. 📦 How to install + verify the whole thing

**✅ R Package Update Manual (GitHub version)**

**📁 1. Place your function in the correct location**

* All function files **must go in the R/ folder**.
* Name the file after the function for clarity, e.g.:

yourpackage/

├── R/

│ └── my\_new\_function.R <- put it here

Paste your function inside this file.

**📝 2. Add Roxygen2 documentation above your function**

Example:

#' Brief Title of the Function

#'

#' Longer description of what the function does.

#'

#' @param arg1 Description of the first argument

#' @param arg2 Description of the second argument

#' @return What the function returns (e.g., data frame, list)

#' @export

#' @importFrom readxl read\_excel

#' @name my\_new\_function

my\_new\_function <- function(arg1, arg2) {

# your code here

}

Make sure:

* The block starts with #' lines (NOT just #)
* You include @export
* You include @name (same as the function name)
* The block is directly above the function (no empty lines)

**🔁 3. Document the package**

Run this in your R console, with the package **open in RStudio or working directory set**:

devtools::document()

This will:

* Add your function to NAMESPACE
* Generate .Rd help files in man/

**🧪 4. Test the function**

Run this to reload the package without reinstalling:

devtools::load\_all()

Try your function:

my\_new\_function() # or whatever your arguments are

If it works: great.

If it errors: fix it now before committing.

**📁 5. (Optional) Add a test file using testthat**

If your package uses testthat, add a test file in tests/testthat/:

test\_that("my\_new\_function works as expected", {

expect\_true(is.list(my\_new\_function("input1", "input2")))

})

Then run:

devtools::test()

**🧾 6. Add new dependencies to DESCRIPTION**

If your function uses any new packages (e.g., readxl), open the DESCRIPTION file and add them under Imports::

Imports:

readxl,

tools,

data.table

Do **not leave blank lines** and don’t manually edit NAMESPACE.

**🌐 7. Commit and push to GitHub**

In terminal or RStudio Git tab:

git add R/my\_new\_function.R man/my\_new\_function.Rd NAMESPACE

git commit -m "Add my\_new\_function"

git push origin main # or whatever branch you're on

**📦 8. Install your updated package**

If you want to install it locally (after all changes):

devtools::install()

This will build and install the package in your local library.

**🔍 9. Confirm it’s working**

library(yourpackage)

my\_new\_function()

Use ?my\_new\_function to view your documentation.

**🔚 Summary: Commands Cheat Sheet**

# After editing/adding function

devtools::document()

devtools::load\_all()

devtools::test() # optional, if using testthat

devtools::install() # to install locally

# GitHub step

git add .

git commit -m "Add or update function"

git push origin main

If you want, I can generate a Markdown .md version of this manual to include in your repo as dev-manual.md.

Want that?