Creating an R Package

rugB

2024-10-23

Step 1: Create a version control `R` package.

- There are a couple of ways to do this step. For this demo, we are going to clone a starter repo from GitHub.
- In the future, you should consider using `usethis::create_package()`.

Step 2: Create a version control RStudio Project with the same name as the GitHub repo.

- Can't do this step if you don't yet have git installed on your computer.
- Should now have a **Git** pane and a **Build** pane in the upper-right window of your RStudio session.
  - Restart your `R` Session if either is missing.

Step 3: If you want to include data in the package, add the raw data and wrangle it into the version the package will share.

- Run `usethis::use_data_raw()` to create a `data-raw` folder and the `DATASET.R` file.
- Add the raw data files to the `data-raw` folder.
- Use the file `DATASET.R` to load and wrangle the raw data.
  - Instead of loading packages with `library(package_name)`, use `package_name::function_name()`.
  - At the bottom, include the following code to create a tidy `.Rda` file:

```r
usethis::use_data(insert_data_name, overwrite = TRUE)
```

- Run the code in `DATASET.R` to create a new folder called `data` that contains the tidy data.
- Run `usethis::use_r("insert_data_name")` to create a blank script file.
  - We will add the data codebook to this file in Step 5.

Step 4: Add your functions to the `R` folder.

- For each user-facing function in your package, run `usethis::use_r("insert_function_name")` to create a new script file.
- Within each function, instead of loading packages with `library(package_name)`, use `package_name::function_name()`.
- For any packages your functions depends on, run `usethis::use_package("package_name")`.

Step 5: Create documentation.

- For each of your **function** scripts in the `R` folder, add some template documentation code by going to `Code > Insert roxygen skeleton`.
- For each of your **data** scripts, you will need to write it from scratch.
  - Mimic examples!
  - Make sure to include `@format` and `@source`.
  - Include the dataset name in quotes at the bottom of the script.
- Add `roxygen` comments that document your function or dataset.
  - Here's a data example.
  - Here's a function example.
  - See the Object Documentation Chapter of R packages for more information on the syntax.
- To create the output help files, run `devtools::document()`.
  - Notice that there is now a man folder with `Rd` help files.

Step 6: Test drive your package functions.

- Restart your R Session and run `devtools::load_all()` to make the package functions and data available.
- Test out the functions.
  - Return to earlier steps if you find any bugs.
- Type `?insert_function_name` or `?insert_data_name` to make sure the help file pops up and to see if it is formatted correctly.

Step 7: Run a more formal check of your package with `devtools::check(document = FALSE)`.

- Fix any errors or warnings. (Note: The package will still compile when there are warnings and notes.)

Step 8: Try installing the package with `devtools::install()`.

Step Often: Commit and push the changes to the GitHub repository.

Additional Components:

- Package metadata
  - DESCRIPTION
  - NAMESPACE
  - License
- Documentation
  - Best practices for the help files
  - Vignettes
  - Effective Readme
  - Website
- Testing