

# Red Rock Data Science

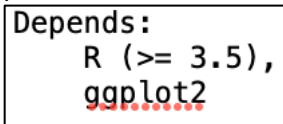
## Workshop 2 – R Packages

### Introduction

The goal of this workshop is to get you familiar with building **R** packages. If you have used Git and GitHub, then you can complete both parts. If not, I recommend you skip to Part 2. Some of these steps involve git and GitHub. This is if you want to upload your package to GitHub. If not, feel free to skip those parts. Any part that uses git or GitHub is marked with an asterisk (\*) and is skippable.

1. We will create a new package that contains some plotting functions in **R** and install it. First identify a folder on your computer to store the package. This might be your Documents, or Desktop, or somewhere else. For the sake of this workshop, I will call it “package\_directory”, but feel free to call it something more descriptive or creative.
2. In RStudio, set the working directory to the package\_directory folder. Then run the code: **devtools::create("myplots")**. RStudio may give you a message about myplots being nested in an existing project. That is fine, confirm you want to do it. This will create a new folder called myplots with the template files for creating a new package.
3. \*Create a repository on GitHub with the same name as your package folder (myplots). Do not add any files to it.
4. \*We will now initialize Git in your package folder. In the terminal, change directory into the myplots folder. Then type **git init**. Then sync it with GitHub by typing **git remote add origin https://github.com/username/repo.git**. Replace username with your GitHub username and replace repo with “myplots” without the quotes. Add all files using **git add .** (you may get a message about LF being replaced by CRLF, that is fine) then commit with **git commit -m “message”**, but change the message. Finally, push the changes to GitHub using **git push -u origin main**. If you have not set up a token in GitHub before, you will need to do that now. Feel free to Google how to do that or ask me.
5. Add a README.md file in your myplots package. You can do this by using the **touch README.md** command in the terminal or you can create one manually using TextEdit on a Mac or Notepad++ on Windows. Then edit that file to indicate that is a package containing plotting functions.
6. Now put some functions in the R subfolder that is in your myplots folder that, in turn, is in your package\_directory folder. There are a few functions you can put in there on the GitHub page in the Workshop 2 folder. The three functions are gbox.R, ggraph.R and ghist.R. Copy those into the R folder in the myplots directory.

7. You'll notice that the `gbox.R` and `ggraph.R` functions are nicely documented. Copy the documentation from one of those functions into the `ghist.R` function and edit that documentation so it makes sense for this function.
8. Let's move on to editing the DESCRIPTION file. Open the DESCRIPTION file in a text editor. Put a Title with something like "This Package Contains Useful Plotting Functions", edit the Authors@R section to include your personal information, and edit the Description section.
9. DESCRIPTION file, add a Depends section. List `ggplot2` and R version 3.5 or greater in that section. This will make sure that `ggplot2` is loaded whenever your package is loaded. You can put this on multiple lines as long as you do a four-space indent and a comma after each item except the last one. See the image below. Important: be sure to leave a blank (empty) line at the end of your DESCRIPTION file (you can probably imagine the pain that has caused me in the past before realizing that was an issue!).



```
Depends:
  R (>= 3.5),
  ggplot2
```

10. Once that is done, go to RStudio, change the working directory to the `myplots` folder and type `devtools::document()`. This will create a `man` folder (for manuals) with a couple files in it that correspond to the functions and edit the `NAMESPACE` file.
11. In RStudio, type `usethis::use_mit_license()`. That will edit the DESCRIPTION file and add a license file to indicate that the license for this package is the MIT license. That will mean that this code will be open source.
12. We are almost done! In RStudio, type `devtools::install()`. This will install your package.
13. Restart your RStudio session by either quitting and reopening or by clicking Session at the top and then Restart R.
14. Load your package by typing `library(myplots)`. This should allow you to use the functions from your package. Try using `ggraph()`. You can also see documentation by typing a question mark before the function: `?ggraph`.
15. You will need to have LaTeX installed on your system for this step to work. In RStudio, type `devtools::build_manual()` to build a manual. Then move this to the `myplots` folder. If you do not have LaTeX installed, you can either skip this step or you can install the `tinytex` package using these steps:
  - a. Type `install.packages("tinytex")` in RStudio and then run `tinytex::install_tinytex()`. This will take a few minutes and should install a light version of LaTeX that should be enough to create the manual.

- b. Once you have done that, type **tinytex::tlmgr\_install("makeindex")**. That should put tinytex in the correct path.
  - c. Restart your computer (annoying, I know, but necessary), open RStudio, make sure the working directory is in "myplots", and run **devtools::build\_manual()**. This should now work and a new manual file with the name "myplots\_0.0.0.9000" in the parent folder of myplots will be created, which should be your package\_directory folder on your computer.
  - d. Add this manual in your myplots folder.
16. \*Go ahead and add, commit, and push all of this to GitHub.
17. \*Finally, use **devtools::install\_github("username/myplots")** to install this package from GitHub and make sure you get no errors. Quit RStudio and the reopen it (or just Restart R from the Session drop down at the top), load your myplots package, and verify that it is working. If it is, you are done!