# Lab 2
## R Packages

## Goal

The goal of this lab is to get you familiar with modifying and then building **R** packages. In the first lab, you forked the mypackage directory and then cloned it to your local system. You may have called the forked repo "mypackage" to match mine or something different. Either way is fine, but I will refer to it as "mypackage" in this lab.

## Updating mypackage

1. First, on GitHub, navigate to your repos and to your mypackage repo. In that repo, because you forked it from mine, you will have a Sync fork button. Click that button. There you will have two options: one to discard changes and one to open a pull request. These are your options because your forked repo is both ahead and behind mine. To the left of the Sync fork button, you will see how many commits ahead and behind you are. You will probably only be one commit ahead and that is when you added the folder with the file explaining what experience you had with Git and GitHub before the first lab. Since this is not a needed file in your package, click on the Discard commits button (if you want to keep that file for some reason, you can move it out of the folder on your local computer). If you had important changes you did not want to discard, you would need to resolve them using the terminal. While these are not needed now, here are instruction in case you are interested: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/syncing-a-fork.

2. Now that your GitHub forked repo is synced with mine, you can navigate to your mypackage folder on your computer. In the terminal, make sure you are in your main branch (you can type **git checkout main**) and then type **git pull**. This will sync all changes from GitHub to your computer.

3. Now let's write a couple new functions that we can add to the package. First, in the R/ folder, copy the add.R function and change the name to multiply.R. Edit this function and the documentation to indicate this function is used to multiply two numbers. Of course, this is a practically useless function since it is easy enough to multiply two numbers!

4. Write one more simple function called divide.R that takes two numbers and divides the first number by the second number. Make sure the documentation is updated as well. In addition to dividing, include a check in this function. If the second number is 0, make this function output an error message saying "You cannot divide by 0." The **stop()** function is good for this. You can create an if statement and then put

**stop("You cannot divide by 0")**.

5. Now let's add a new data set this package. Go to my Math3190_Sp24 GitHub repo: https://github.com/rbrown53/Math3190_Sp24 and find the brainbody.txt data file in the Data folder. Download this and put it in data-raw folder.

6. Copy the cranes.R file and rename it to brainbody.R. Then edit it so it reads in the brainbody.txt file. Then edit the usethis line so we have a brainbody object being created instead of cranes.

7. In RStudio, set your working directory to be your mypacakge folder. Then run the code in the brainbody.R file in RStudio. This should add the brainbody.rda object in the data folder.

8. Now edit the data.R file in the R folder. Copy and paste lines 1-9, but leave a blank line between the cranes documentation and the brainbody documentation. Then edit those lines to be about the brainbody data set. Of course, this data set has 5 variables instead of 2.

9. Now edit the DESCRIPTION file to include yourself as an author. Check out this question on statoverflow.com: https://stackoverflow.com/questions/54133136/how-to-include-authorsr-in-the-manual-of-r-package for info on how to add a second person as an author. Your role would just be "aut" not "cre" since you were not the original creator of the package.

10. In the DESCRIPTION file, also edit the Description section to indicate that you added functions about multiplying and dividing.

11. In the DESCRIPTION file, also edit the Version number to 0.0.1.0000.

12. Now edit the my-vignette.Rmd file to include one example of the **multiply()** function and one of the **divide()** function. Updated steps below:
    a. Install the devtools library: **install.packages("devtools")**.
    b. Be sure your R working directory is in the mypackage folder and then run **devtools::document()**. This will create manuals for each of your new functions and will add them and the brainbody info to the NAMESPACE file. If there were any errors in those functions or your editing of the DESCRIPTION file, it will let you know.
        i. If it gives you grief about a package needing to be a more modern version, like maybe the cli package, try to remove the package using **remove.packages("cli")** and then installing it again: **install.packages("cli")**. If that doesn't work, let me know.
    c. Run **devtools::install()**. It is okay if it mentions something about the package masking other functions.

d. Then Knit the vignette.

By the way, as an alternative to installing your package and then Knitting the vignette, you could instead run `devtools::build_rmd("vignettes/my-vignette.Rmd")`. This will install your package temporarily, then Knit the vignette, and then uninstall your package. Either option is fine, but if you do this option, then you should also do part 13.

~~13. Now that all of this is done, you can install the devtools library. Then run the **devtools::document()** command in RStudio (again, your working directory should be the mypackage folder). This will create manuals for each of your new functions and will add them and the brainbody info to the NAMESPACE file.~~

14. Now run **devtools::install(build_vignettes = TRUE)** in RStudio. This will install the mypackage package, but will also build the vignette.

15. Quit RStudio and reopen it or go to Session at the top and then Restart R. Then type **library(mypackage)** to load your package. Then try to use the **multiply()** and **divide()** functions to verify they work and the package was installed correctly. Then type **?multiply()** in RStudio to verify the help page for that function works. Then type **?brainbody** to verify the help page for that data set is working.

16. Now type **utils::browseVignettes("mypackage")** in RStudio. This should launch the vignettes page. Click on the HTML to view the vignette and verify that your new examples are showing.

17. If everything is good, let's build a new manual. Type **devtools::build_manual()** in RStudio. This will create a new manual file with the name "mypackage_0.0.1.0000" in the parent folder of mypackage, which should be your Math_3190 folder on your computer. Add this manual in your mypackage folder. You can delete the old manual with the name "mypackage_0.0.0.9000".

18. If everything is working properly, you should add, commit, and push all changes in Git. Go to the terminal, type **git status** to see the changes. Then type **git add .** to add them all. Then **git commit -m "message here"** to commit them with a message. Edit the message to indicate that you added two new functions, a new data set, edited the description, and created a new manual. Then **git push** them to GitHub.

19. Finally, try to install your package from GitHub using the code **devtools::install_github("your_username/mypackage", build_vignettes = TRUE)**. If that installs properly, then you are done with this portion of the lab!

## Creating a New Package

20. Now we will create a new package and install it. We will run through many of the R Packages Recap slides at the end of Notes 2.
    First, in RStudio, set the working directory to the Math_3190 folder. Then run the code: **devtools::create("myplots.in")**, but change the "in" to your initials. For example, if I were doing this, I would type **devtools::create("myplots.rb")**. 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.in (but with your initials) with the template files for creating a new package.

21. Create a repository on GitHub with the same name as your package folder. Do not add any files to it.

22. 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.in" without the quotes and with your initials instead of the "in". Add all files using **git add .** then commit with **git commit -m "message"**, but change the message. Finally, push the changes to GitHub using **git push -u origin main**.

23. Add a README.md file in your myplots.in package by using the **touch README.md** command in the terminal. Then edit that file to indicate that is a package containing plotting functions.

24. Now put some functions in the R folder. The first function you should add is the ggraph.R function that was already used in mypackage. Copy it into the R folder in the myplots.in directory. The second file you should add is the influence_plots.R function from my GitHub page. Go to https://github.com/rbrown53/Math3190_Sp24 and then into the Labs folder, and then into the Lab 2 folder. There you will find the influence_plots.R file. Download it and put it into the R folder in your myplots.in directory.

25. We will not add a data set or vignette to this package at this time. So 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.

26. Like in the mypackage DESCRIPTION file, add a Depends section. List ggplot2 and R version 3.5 or greater in that section. Important: be sure to leave a blank (empty) line at the end of your DESCRIPTION file.

27. Once that is done, go to RStudio, change the working directory to the myplots.in folder and type **devtools::document()**. This will create a man folder with a couple files in it that correspond to the functions and edit the NAMESPACE file.

28. In RStudio, type **usethis::use_gpl_license()**. That will edit the DESCRIPTION file and add a license files to indicate that the license for this package is GPLv3. That will mean that this code and all derivatives of this code will be open source.

29. We are almost done! In RStudio, type **devtools::install()**. We do not need to include **build_vignettes = TRUE** since we did not make a vignette.

30. In RStudio, type **devtools::build_manual()** to build the manual. Then move this to the myplots folder.

31. Go ahead and add, commit, and push all of this to GitHub.

32. Finally, use **devtools::install_github("username/myplots.in")** to install this package from GitHub and make sure you get no errors. Quit RStudio and the reopen it, load your myplots.in package, and verify that it is working. If it is, you are done! We will be adding more functions in this package later.