Installing R

<https://cran.r-project.org/>

Installing RStudio

<https://rstudio.com/products/rstudio/download/>

**R Packages**

<https://www.rdocumentation.org/>

What are repositories?

A repository is a central location where many developed packages are located and available for download.

There are three big repositories:  
1. [**CRAN (Comprehensive R Archive Network):**](https://cran.r-project.org/web/packages/) R’s main repository (>12,100 packages available!)

install.packages("ggplot2")

install.packages(c("ggplot2", "devtools", "lme4")) - multiple packages at once

2. **[BioConductor:](https://bioconductor.org/packages/release/BiocViews.html" \l "___Software" \t "_blank)** A repository mainly for bioinformatic-focused packages

source("https://bioconductor.org/biocLite.R")

biocLite("GenomicFeatures")

With R version 3.5 or greater, install Bioconductor packages using BiocManager; see <https://bioconductor.org/install>

3. [**GitHub:**](https://github.com/collections) A very popular, open source repository (not R specific!)

1. install.packages("devtools")  - only run this if you don’t already have devtools installed. If you’ve been following along with this lesson, you may have installed it when we were practicing installations using the R console
2. library(devtools)  - more on what this command is doing immediately below this
3. install\_github("author/package")  - replacing “author” and “package” with their GitHub username and the name of the package

Loading Packages

library(ggplot2)

Updating Packages

old.packages() - check what packages need an update with a call to the function

update.packages() - update all packages, use

install.packages("packagename") - update a specific package, just use once again

Unloading Packages

detach("package:ggplot2", unload=TRUE)

Uninstalling Packages

remove.packages("ggplot2") - or uncheck the box

Checking R Version

version

sessionInfo() - lists R version, operating system, and loaded packages

Getting Help

help(package = "ggplot2")

browseVignettes("ggplot2")

**GitHub**

<https://github.com/>

Installing git:

<https://git-scm.com/download>

<https://www.atlassian.com/git/tutorials/install-git>

Configruing git:

To do so, in the command prompt (either Git Bash for Windows or Terminal for Mac), type:

git config --global user.name "Jane Doe" with your desired username in place of “Jane Doe.” This is the name each commit will be tagged with.

Following this, in the command prompt, type:

git config --global user.email [janedoe@gmail.com](mailto:janedoe@gmail.com)

At this point, you should be set for the next step, but just to check, confirm your changes by typing: git config –list

Once you are satisfied that your username and email is correct, exit the command line by typing exit and hit Enter.

### Linking RStudio and Git

In RStudio, go to Tools > Global Options > Git/SVN

In that same RStudio option window, click “Create RSA Key” and when this completes, click “Close.”

Following this, in that same window again, click “View public key” and copy the string of numbers and letters. Close this window.

Go to [github.com/](https://github.com/), log-in if you are not already, and go to your account settings. There, go to “SSH and GPG keys” and click “New SSH key”. Paste in the public key you have copied from RStudio into the Key box and give it a Title related to RStudio. Confirm the addition of the key with your GitHub password.

### Create a new repository and edit it in RStudio

On GitHub, create a new repository (github.com > Your Profile > Repositories > New). Name your new test repository and give it a short description. Click Create repository. Copy the URL for your new repository.

In RStudio, go to File > New Project. Select Version Control. Select Git as your version control software. Paste in the repository URL from before, select the location where you would like the project stored. When done, click on “Create Project”. Doing so will initialize a new project, linked to the GitHub repository, and open a new session of RStudio.

Create a new R script (File > New File > R Script) and copy and paste the following code:

print("This file was created within RStudio")

print("And now it lives on GitHub")

Save the file. Note that when you do so, the default location for the file is within the new Project directory you created earlier.

Once that is done, looking back at RStudio, in the Git tab of the environment quadrant, you should see your file you just created! Click the checkbox under “Staged” to stage your file.

Click “Commit”. A new window should open, that lists all of the changed files from earlier, and below that shows the differences in the staged files from previous versions. In the upper quadrant, in the “Commit message” box, write yourself a commit message. Click Commit. Close the window.

So far, you have created a file, saved it, staged it, and committed it. If you remember your version control lecture, the next step is to push your changes to your online repository. Push your changes to the GitHub repository.

**Creating a project that is not under version control**

We’ve now created an R Project that is not currently under version control. Let’s fix that. First, let’s set it up to interact with Git. Open Git Bash or Terminal and navigate to the directory containing your project files. Move around directories by typing cd ~/dir/name/of/path/to/file

When the command prompt in the line before the dollar sign says the correct directory location of your project, you are in the correct location. Once here, type git init followed by git add . - this initializes (*init*) this directory as a git repository and *adds* all of the files in the directory (.) to your local repository. Commit these changes to the git repository using git commit -m "Initial commit"