

R

If you already have R installed on your computer:

Check that you have a version greater than 3.6.0. I will be using R 4.0.2, and there are a lot of differences in the latest version. It is a good idea to update.

If you have never installed R:

Go to <https://www.r-project.org/> and click on **CRAN**. CRAN is a network of servers that hosts R products to download. You will see a list of mirror locations. Scroll down to USA and click a mirror location. Michigan or Kansas are great options. Generally, it is good to choose a location that is close to you, but we don't see much of a difference. In the rare event that one of these locations is unavailable and you get an error message, you can always try a different mirror than the one you selected. You will see a download link for your operating system. Follow the instructions for installing the latest version.

RStudio

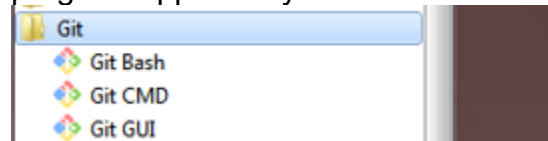
Go to <https://rstudio.com/products/rstudio/#rstudio-desktop> and choose the download RStudio desktop (free, basic version). Follow the download instructions and installer prompts.

Git

Installation

Using Git on a Windows operating system:

I find the Git program is the best program for interfacing with GitHub and version controlling code. You can download it here: <https://git-scm.com/download/win>. Don't change any of the installation defaults. Once you install the program, you will see a new program appear in your start menu:



You won't really work with the program directly here, but this is a good way to check if you aren't sure. You will want to access git from the command prompt. Go to your search bar in or next to your start menu (location depends on operating system version). You can verify that it is working properly by typing git into the command prompt. You should see a list of operations appear like below:

```
Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\hkropp>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

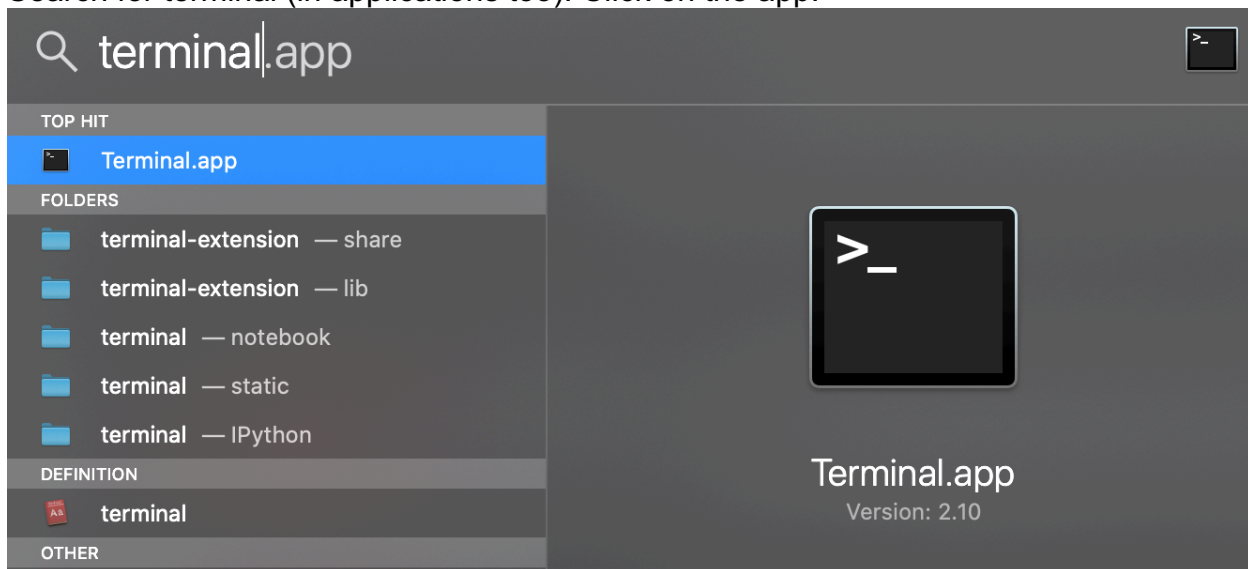
These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
    clone             Clone a repository into a new directory
    init              Create an empty Git repository or reinitialize an existing
one

work on the current change (see also: git help everyday)
    add               Add file contents to the index
    mv                Move or rename a file, a directory, or a symlink
    restore            Restore working tree files
    rm                Remove files from the working tree and from the index
    sparse-checkout    Initialize and modify the sparse-checkout
```

Using Git on a Mac operating system:

Search for terminal (in applications too). Click on the app.



The terminal window will appear and follow the instructions below. Copy and paste this code into the terminal window. You will now see homebrew install. Homebrew is program that helps provide a lot of coding resources for macs.

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Once homebrew finishes installation, type or copy this line into the prompt:

```
brew install git
```

You can now use git in your terminal window. Type git to verify that the program is running in the terminal. You will see a long print out with documentation about different commands that git will run.

```
(base) Mac-23712:~ hkropp$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
    clone                Clone a repository into a new directory
    init                 Create an empty Git repository or reinitialize an existing
one

work on the current change (see also: git help everyday)
    add                 Add file contents to the index
```

Sign up for GitHub

GitHub is a website that helps display and manage your git repositories. It is frequently used in professional data science and programming settings. You will need to set up a user profile. You will use this profile to store and catalog your code. You may set up any username that you wish. I used my name because I wanted my username to be readily identifiable for my peers and fellow researchers. *However, should you wish to maintain privacy, you can set up a username that does not reveal your identity.* You can also use any email that you prefer for the account. You can enter as much or as little personal information that you wish in setting up your account. All you need is a username and email.

Try to keep track of the password that you set because you will need to use it on Friday.

There are many ways to interface with Git and GitHub including through Rstudio. While this interface may not look very user friendly, it is the easiest to troubleshoot and avoid running into errors. Don't hesitate to reach out with any questions or concerns. This approach to GitHub is also general and will provide you with skills for version controlling for any context and coding language.

Git: helpful reminders for using Git after Activity 1

Part 1: Storing repositories:

You will want to work with repositories in a folder that is local to your operating system. I recommend choosing your documents folder. **This folder should not be synced by a back-up program (e.g. iCloud or Google Sync).** Make a folder called GitHub

On a pc this will look like:

`c:\Users\username\Documents\GitHub`

on a mac:

`/Users/username/Documents/GitHub`

Your username is whatever you set up personally when you set up your laptop. It will not be associated with your Hamilton info. For example, on my personal laptop I set up my username as kroppheather. My file path on a pc would be:

`c:\Users\kroppheather\Documents.`

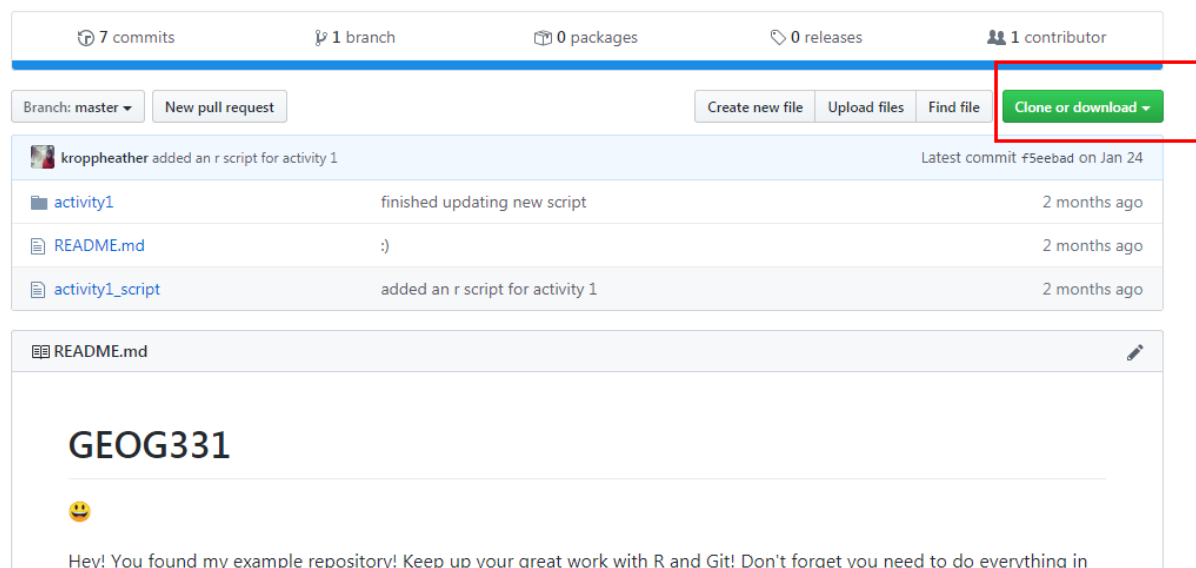
Part 2: Getting started

You will clone your repository when setting it up for the first time. It will have the most up to date version that you pushed on whatever computer you used previously. In your command window, change your directory to your GitHub folder using `cd` and the directory:

```
C:\Users\hkropp>cd c:\Users\hkropp\Documents\GitHub  
c:\Users\hkropp\Documents\GitHub>
```

Go to your repository on GitHub and click on the green button and then the clipboard. This will copy the repository URL.

[Manage topics](#)



7 commits 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file **Clone or download**

File	Commit Message	Time
activity1	finished updating new script	2 months ago
README.md	:)	2 months ago
activity1_script	added an r script for activity 1	2 months ago

README.md

GEOG331

😊

Hey! You found my example repository! Keep up your great work with R and Git! Don't forget you need to do everything in

In your command window type: `git clone` and right click to paste your URL

```
c:\Users\hkropp\Documents\GitHub>git clone https://github.com/kroppheather/GEOG331.git
Cloning into 'GEOG331'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.

c:\Users\hkropp\Documents\GitHub>
```

You will see a new folder appear in your GitHub folder with the name of your repository and all of the up to date contents inside.

Part 3: Committing changes

This is the same on any computer. Here's a refresher:

1. Type **git add -A**
 - This will tell git to add changes to all files for tracking.
2. Type **git commit -m "description of changes"**
 - This command helps label all of your changes
3. Type **git push**
 - This command sends the changes to github
 - You will have to enter your username and password for GitHub here

```
c:\Users\hkropp\Documents\GitHub\GEOG331>git add -A

c:\Users\hkropp\Documents\GitHub\GEOG331>git commit -m "started a script for activity 1"
[master e8ad999] started a script for activity 1
1 file changed, 2 insertions(+)
create mode 100644 activity1/activity1_script.r

c:\Users\hkropp\Documents\GitHub\GEOG331>git push
fatal: HttpRequestException encountered.
An error occurred while sending the request.
Counting objects: 4, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 384 bytes | 384.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/kroppheather/GEOG331.git
bb62613..e8ad999 master -> master
```

Part 4: Updating a local repository from GitHub (a helpful reminder after Activity 1):

You may run into this situation if you work from two computers. Let's say you work on a repository in one computer (**first computer**) and push all of your changes to GitHub. You then clone it to a **second computer** and work from it and push all changes to GitHub. When you go back to your **first computer** to work on the repository, you will need to **update** it. Keep in mind this is for updating an **existing** repository to have the most recent changes. Simply type **git pull** and hit enter, and all new changes will update in your repository:

```
c:\Users\hkropp\Documents\GitHub\GEOG331_answers>git pull
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 98 (delta 25), reused 38 (delta 24), pack-reused 59
Unpacking objects: 100% (98/98), 9.39 MiB | 699.00 KiB/s, done.
From https://github.com/kroppheather/GEOG331_answers
872ace8..7cc97db master -> origin/master
Updating 872ace8..7cc97db
```

For any other issues including conflicts, please email me. Include a screenshot and a description of what you did.