Git + R = collaborative coding

Ainhoa Leger
Gonzalo Garcia

April 2021

Introduction

- **What is GitHub?** It is an online place to
  - ❖ store your code
  - ❖ track the changes that we have made on the code
  - ❖ avoid multiple copies or versions
  - ❖ roll back my code to a previous version
  - ❖ collaborate effectively with others

- **Workshop aim**: Learn how to version control an R project using GitHub within Rstudio

Roadmap

- Part 1: Ainhoa
  - ❖ **Git + R for individual use**

- Part 2: Gonzalo
  - ❖ **Git + R for collaborative use**
  - ❖ **Sneak into more advanced Git + R features**

- Both parts :Guidelines (15 min) + Hands on! (15 min)

PART 1
# (individual use)

Topics

Individual use

    1. Install Git + Setup a GitHub account

    2. Create a GitHub repository

    3. Connect a GitHub repository to an RStudio project

    4. Version control the files

        a. Commit action

        b. Push action

Collaborative use

    5. Fork GitHub repositories

    6. Create Pull Request

    7. Accept/Merge changes

    8. Refresh forked repo

1.1 Install Git

- In order to use the Git version control system we first need to install it
- Git is the machinery which works behind Github and Rstudio
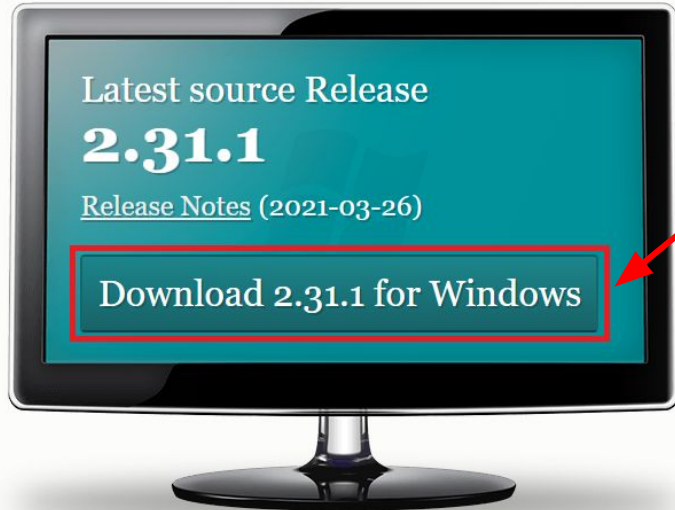
Steps to install Git on your machine

→   Go to https://git-scm.com/downloads

→   Download executable file (.exe)

1.1 Install Git

➔ Execute the file and follow the instructions (by clicking next) to complete the installation
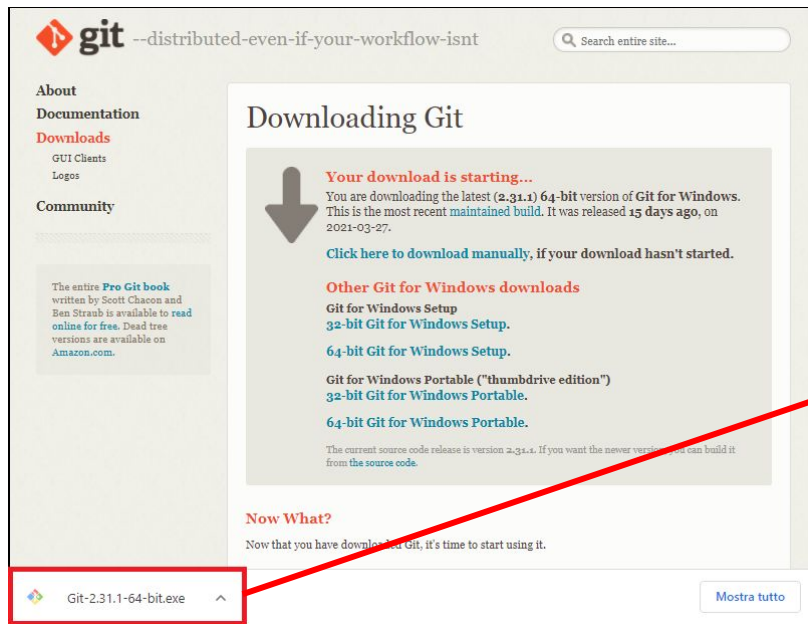
1.1 Install Git
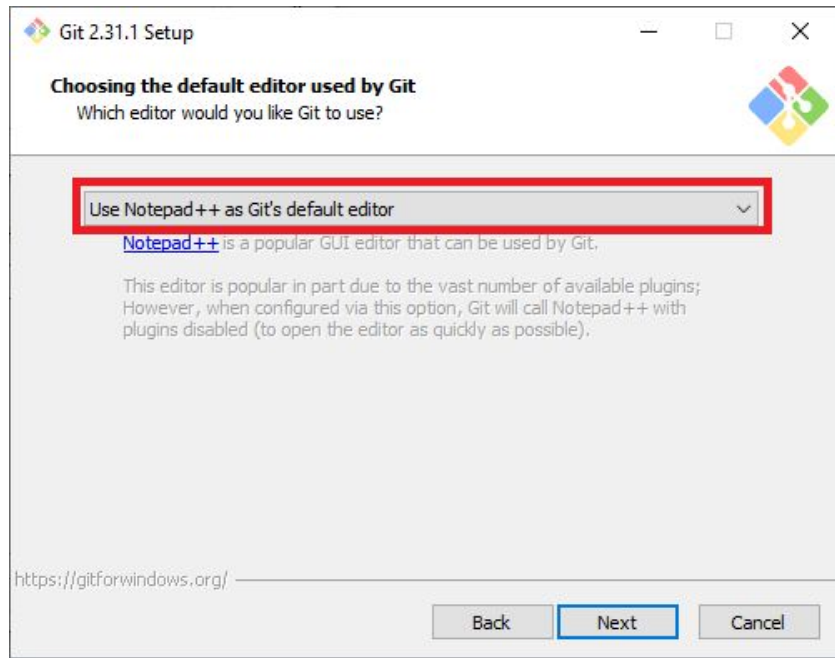
Advice: There are several editors to choose from. Notepad++ is a simple editor that can be useful in general to have installed on your machine. However please note that when coding in R we use RStudio to edit our scripts.

1.2 Setup a GitHub account

- We need to create the online platform where you have your repositories, version control your projects and collaborate with others
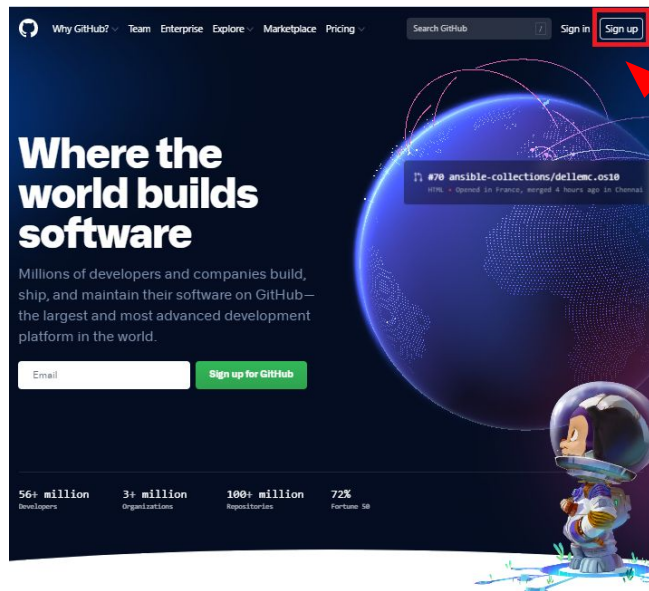
Steps to setup a Github account

➜ Go to https://github.com/

➜ Just follow the Sign up process (top right of the page)

1.2 Setup a GitHub account

- ➜ Type a username, email address and password

  It is a good idea to use a username that can easily be associated with you.

  For example: first name + last name

  Tim Riffe => **timriffe**

  Diego Alburez-Gutierrez => **alburezg**

- ➜ Do the verification process

Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

1.2 Your GitHub account

- You can sign in to your GitHub account from the webpage
- The **drop down menu** (topright corner) gives access to the profile and repositories
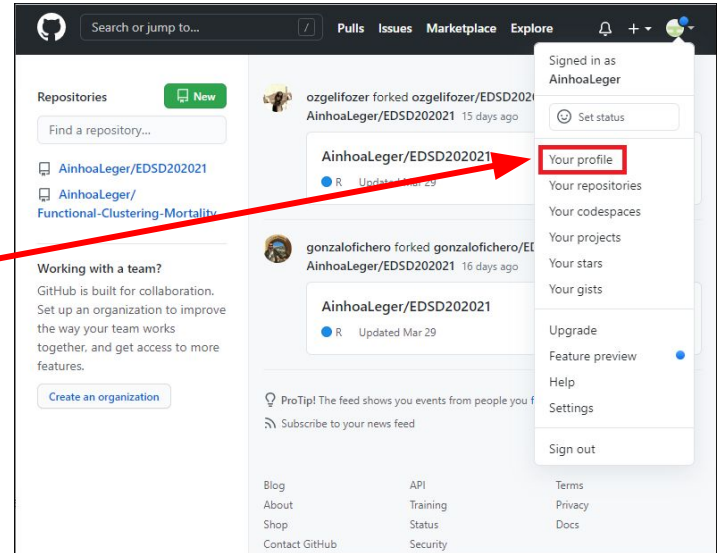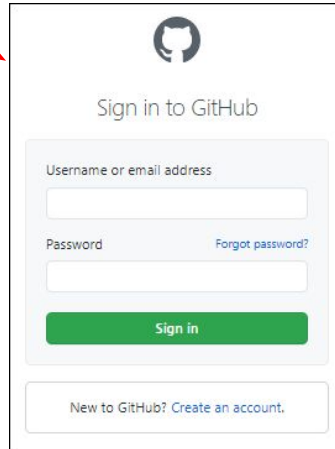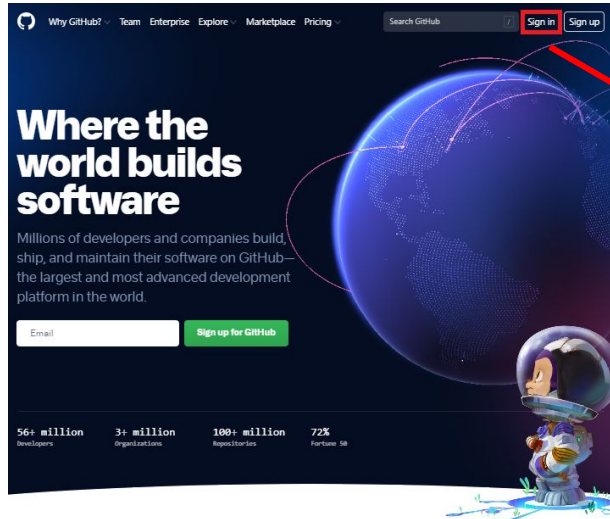
1.2 Your GitHub account

This is the **URL** that others can go to and see your public activities

This is the **contribution graph**; the darker green the square the more contributions for that day
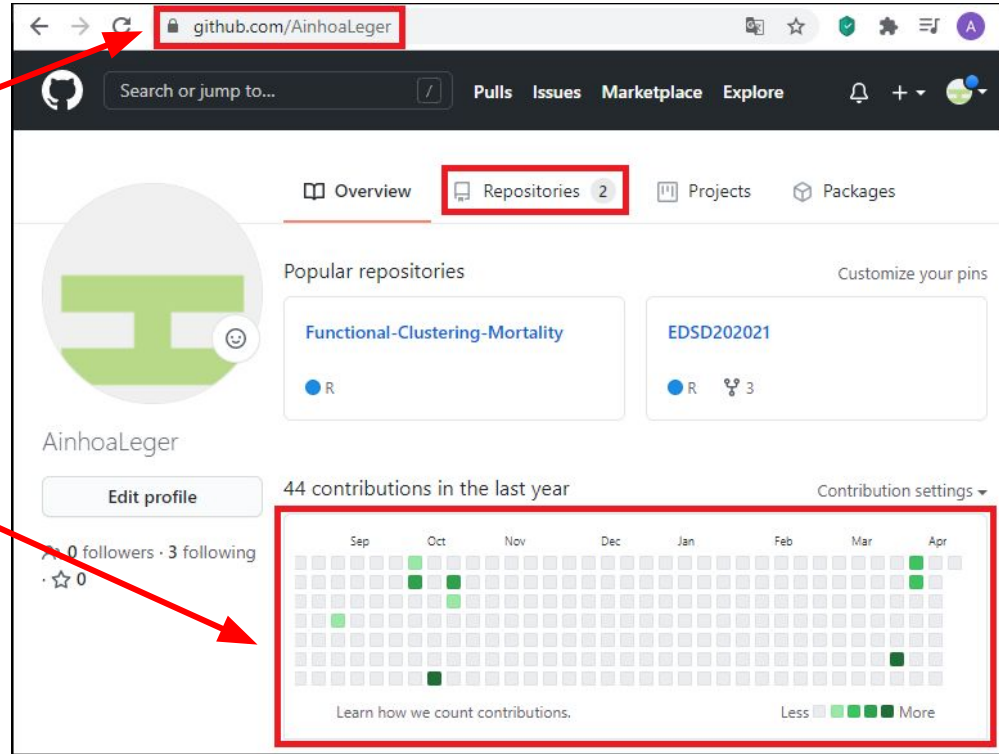
Topics

Individual use

    1. Install Git + Setup a GitHub account

    2. Create a GitHub repository

    3. Connect a GitHub repository to an RStudio project

    4. Version control the files

   a. Commit action

        b. Push action

Collaborative use

    5. Fork GitHub repositories

    6. Create Pull Request

    7. Accept/Merge changes

    8. Refresh forked repo

2. Create a GitHub repository

- We can now create a new **public repository**, where your folder and files (data, R scripts, images) are stored and version controlled
- Each project that you work on usually has its own dedicated repository
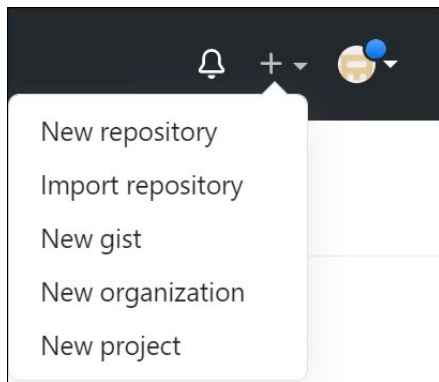
The **"+" button** (topright corner) allows you to create a new repository

2. Create a GitHub repository

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Owner *   Repository name *

🌐 AinhoaLeger ▾   /

Great repository names are short and memorable. Need inspiration? How about verbose-octo-spoon?

Description (optional)

⦿ 📖 **Public**
        Anyone on the internet can see this repository. You choose who can commit.

◯ 🔒 **Private**
        You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
     This is where you can write a long description for your project. Learn more.

☐ **Add .gitignore**
     Choose which files not to track from a list of templates. Learn more.

☐ **Choose a license**
     A license tells others what they can and can't do with your code. Learn more.

Create repository

This is the repository name. This is also the URL address that others can use to go straight into that project

For example: Workshop_GitHubR

Select the repository type, in this workshop we will work with a public repository

2. Create a GitHub repository
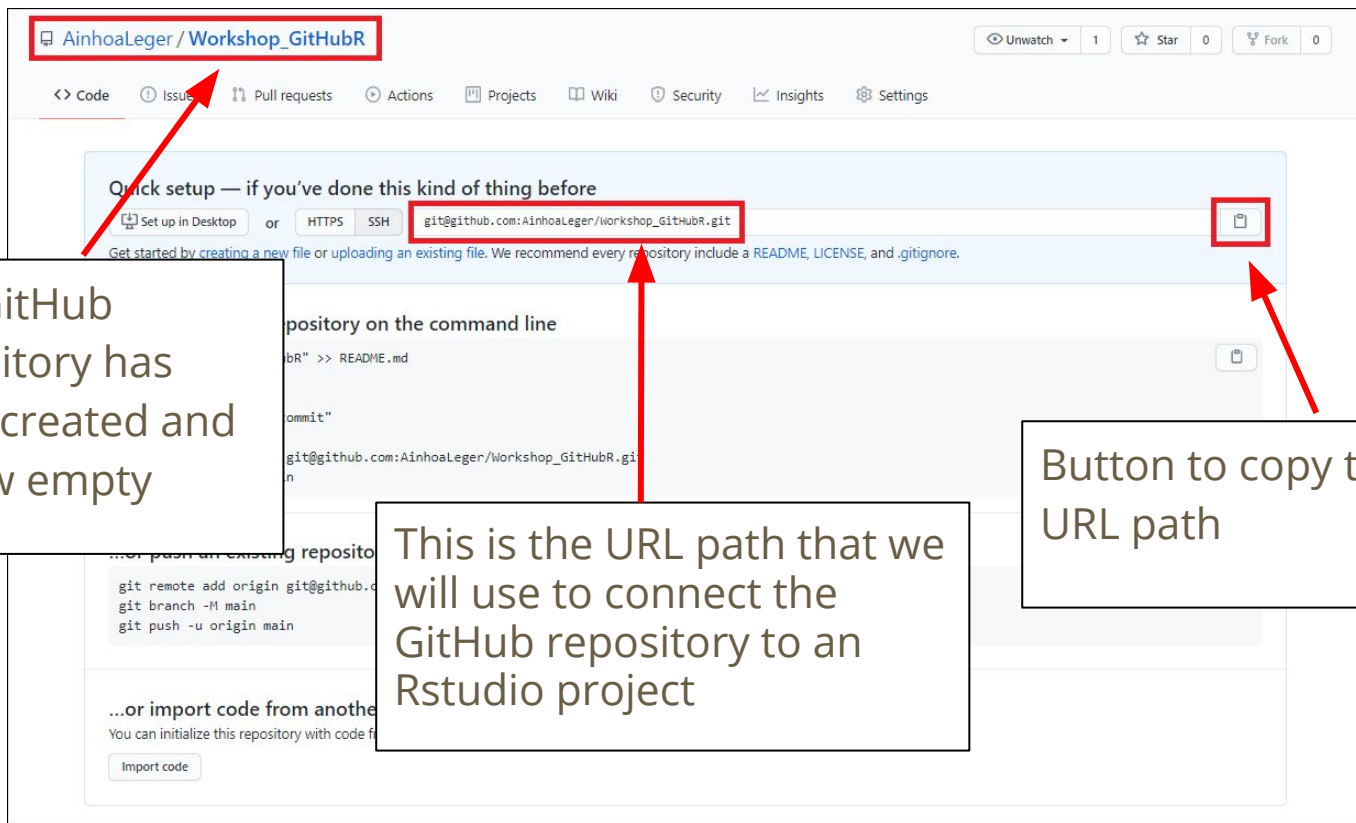
Topics

Individual use

    1. Install Git + Setup a GitHub account

    2. Create a GitHub repository

    3. Connect a GitHub repository to an RStudio project

    4. Version control the files

        a. Commit action

        b. Push action

Collaborative use

    5. Fork GitHub repositories

    6. Create Pull Request

   7. Accept/Merge changes

    8. Refresh forked repo

3. Connect a GitHub repository to an RStudio project

- In order to link up our GitHub account with RStudio (desktop) we need to establish a secure connection. This is done by creating and using an **SSH key** and it only needs to be done once.

- SSH stands for Secure Shell and is a "cryptographic network protocol for operating network services securely over an unsecured network".

➤ **Create an SSH key from your RStudio desktop**
Rstudio > Tools > Global options > GIT/SVN > Create RSA key > View public key > copy the key

➤ **Associate the SSH key with your GitHub account**
Github > Settings > SSH and GPG keys > New SSH key > paste the key to the clipboard

➤ **Clone a GitHub repository to an R project**
Rstudio > File > New Project > Version Control > Git
Github > Copy the URL path to clipboard
Rstudio > Paste the URL path to Repository URL

3. Connect a GitHub repository to an RStudio project

Create an SSH key from your RStudio desktop
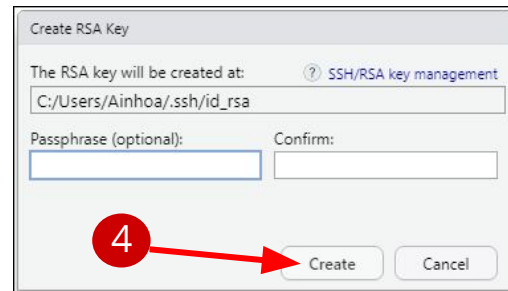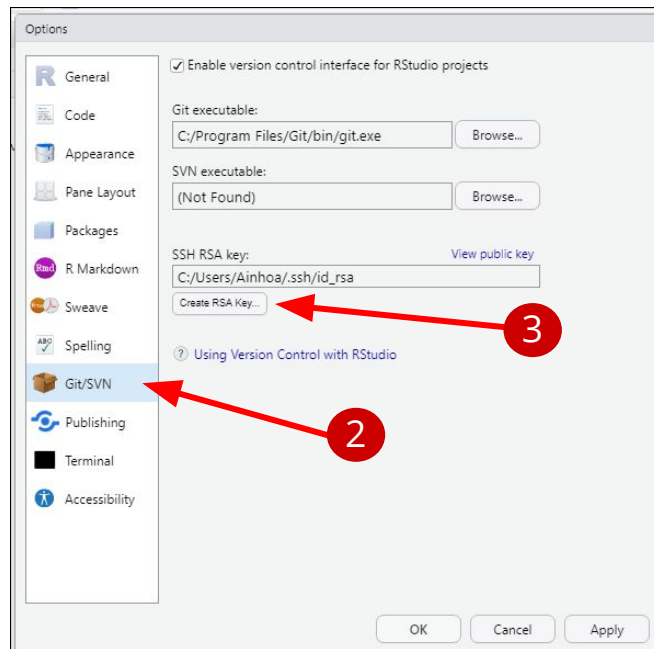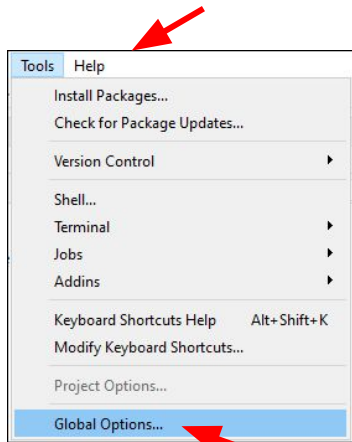
3. Connect a GitHub repository to an RStudio project

Your SSH key

6 Press Ctrl+c to copy the key to the clipboard
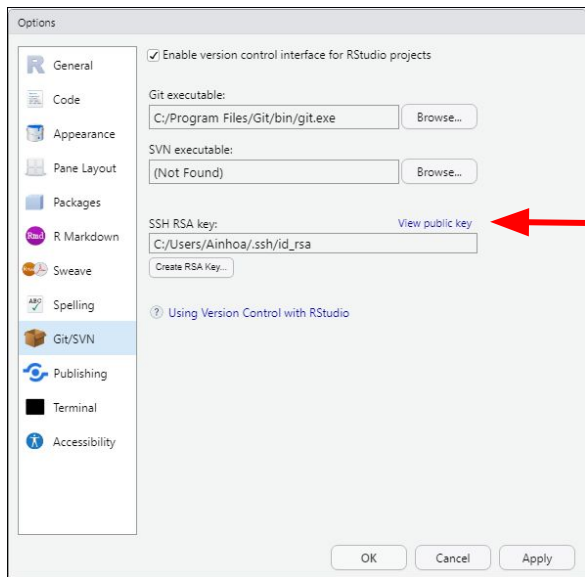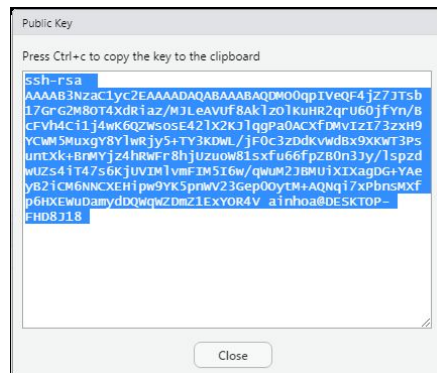
7 Press Ctrl+c to copy the key to the clipboard
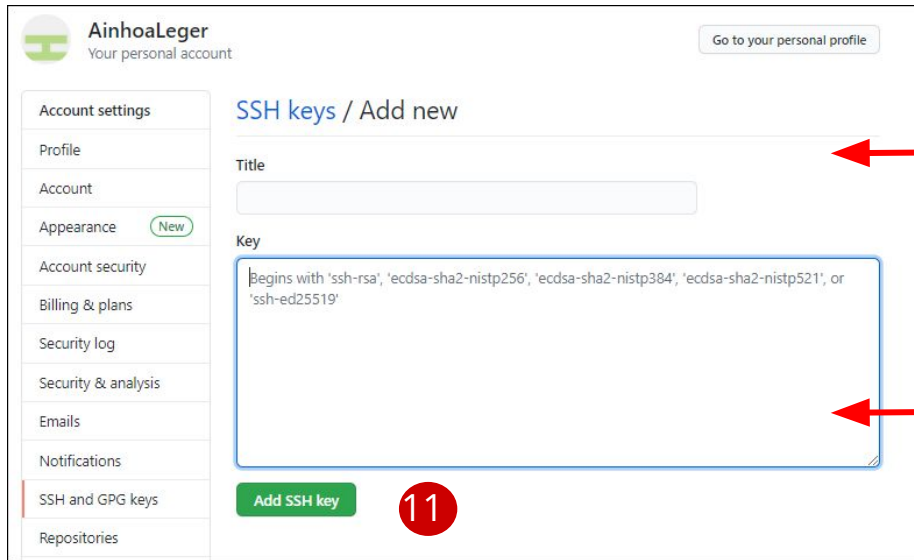
3. Connect a GitHub repository to an RStudio project

Associate the SSH key with your GitHub account

3. Connect a GitHub repository to an RStudio project

Associate the SSH key with your GitHub account

Give this key a title

Paste here the SSH key that you copied earlier from RStudio

3. Connect a GitHub repository to an RStudio project

Clone a GitHub repository to an R project

- We now want to connect the GitHub repository that we have just created with an R project that will live on your machine

- R projects allow you to divide your work in a structured way.

- Each project has its own setup such as a working directory, workspace and history.

- This is even better and more efficient when it is version controlled!

3. Connect a GitHub repository to an RStudio project

Clone a GitHub repository to an R project
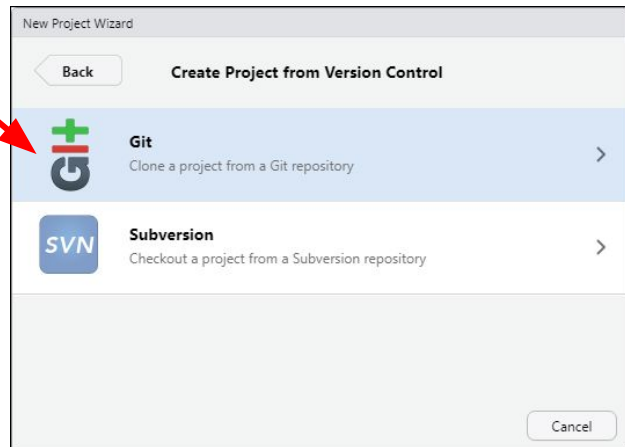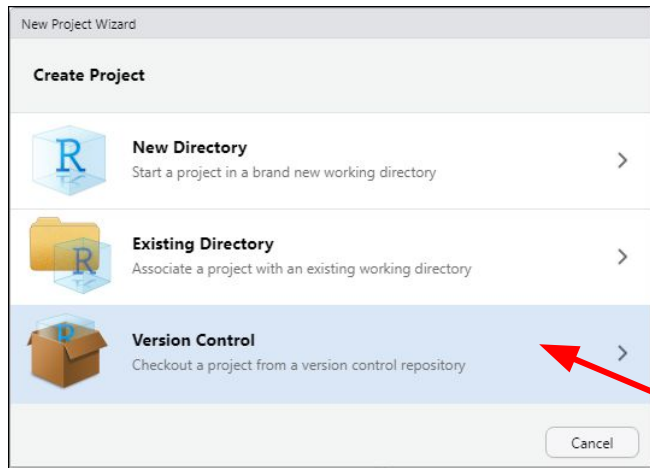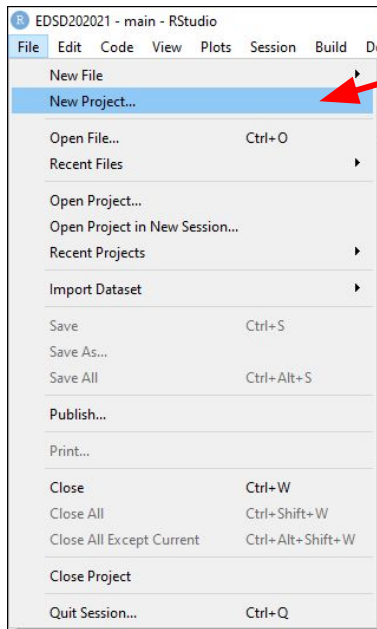
3. Connect a GitHub repository to an RStudio project

Clone a GitHub repository to an R project
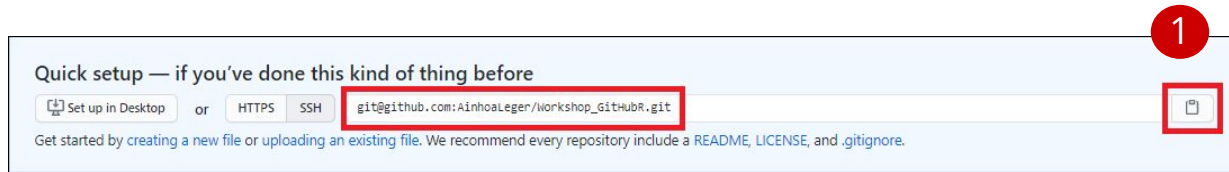
1. Copy the URL path to clipboard
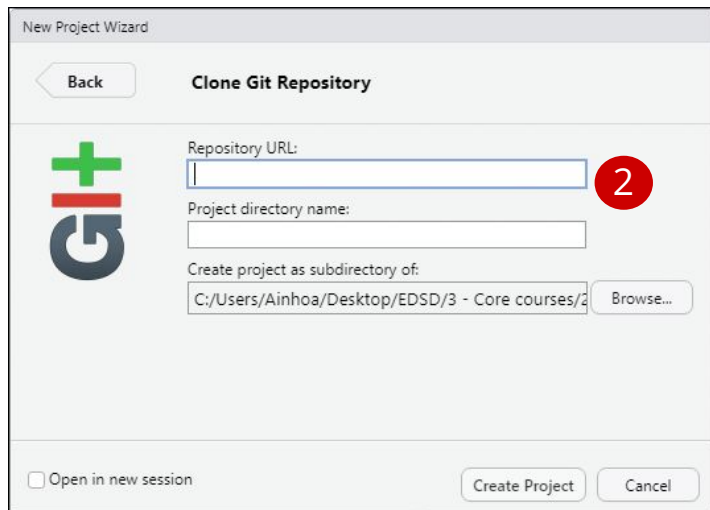
2. Paste the URL path to Repository URL. You can edit the project directory name and the folder

3. Click on create project

3. Connect a GitHub repository to an RStudio project

Clone a GitHub repository to an R project

This tells us that we are inside the R project

This is the "new" tab that should appear that tells you that your R project has added Git functionality, i.e. it is version controlled

This is the .Rproj file associated with this project

Topics

Individual use

    1. Install Git + Setup a GitHub account

    2. Create a GitHub repository

    3. Connect a GitHub repository to an RStudio project

    4. Version control the files

        a. Commit action

        b. Push action

Collaborative use

    5. Fork GitHub repositories

    6. Create Pull Request

    7. Accept/Merge changes

    8. Refresh forked repo

4. Version control the files

Basic version control processes

Now that our project is version controlled let's cover some of the basic processes to version control a file, keeping things as simple as possible. A file in our repository can be one of four statuses:

1. **Untracked**: The file is not being tracked, i.e. it is not version controlled

2. **Unmodified**: The file is tracked but has not changed

3. **Modified**: The file is tracked and has been changed

4. **Staged**: The file is tracked and the changes are ready to be version controlled

4. Version control the files

- If we go in "Git" in the top right panel we can see that we have two files with a yellow question mark
- This means that the files are unknown to Github (untracked)
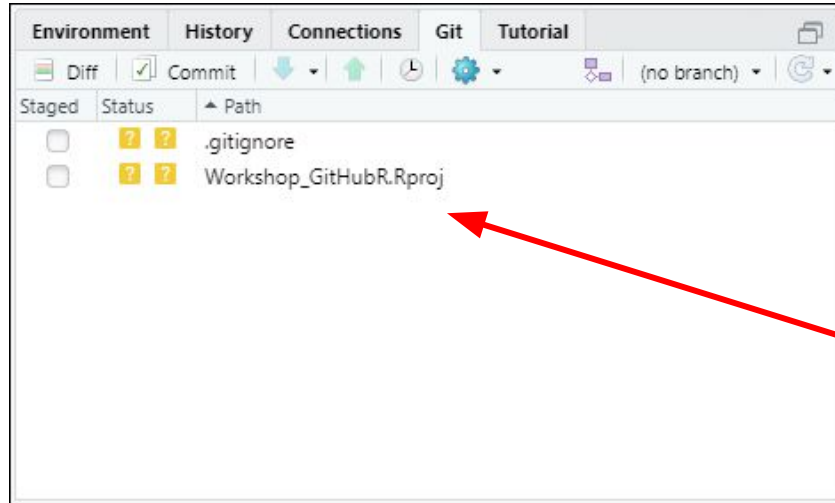
These files are currently untracked

4. Version control the files

- We want to "send" the files stored locally (local repository) or the changes to the file into the remote (online) repository in GitHub
- In this way the file is being version controlled
- We need to add the file to the "staging area", commit the file and all of its changes, and push them into the GitHub repository
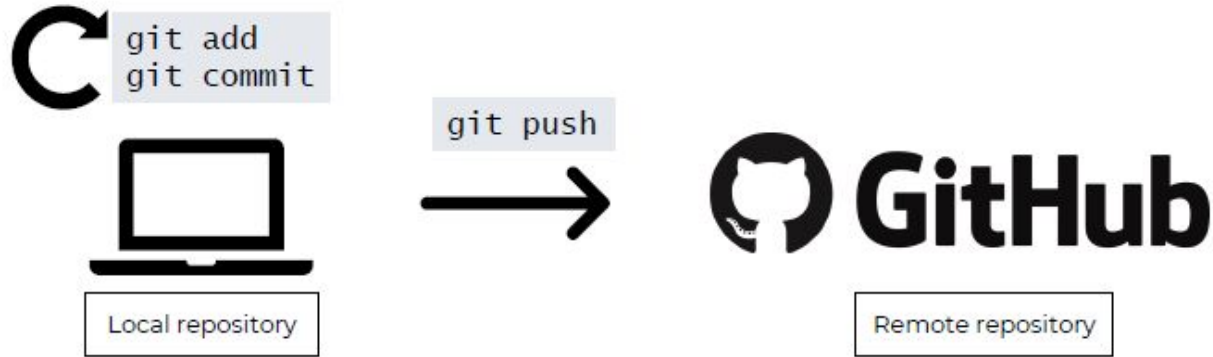
4.a. Commit action

- We will "send" to GitHub the .gitignore file
- The .gitignore is a very useful (hidden) file that communicates to our GitHub repository what files/folders should be ignored and not version controlled
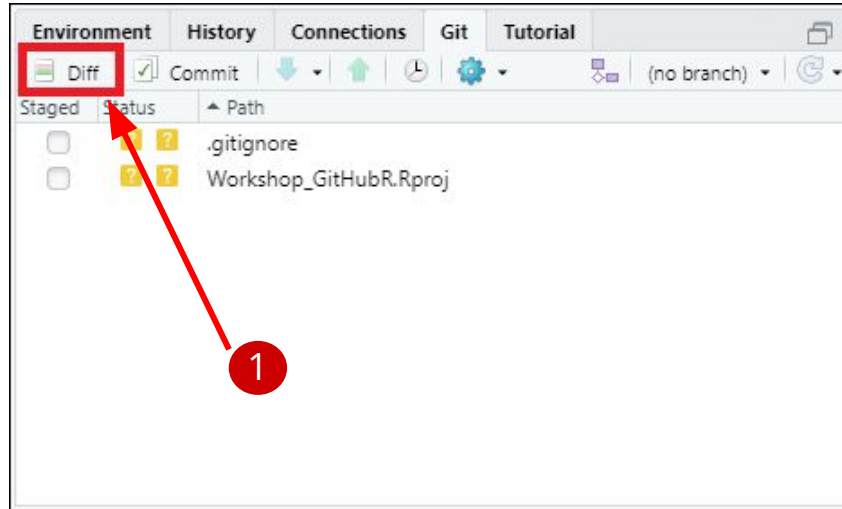
Steps to commit a file/change

1. Click on the Diff button in the Git environment

4.a. Commit action
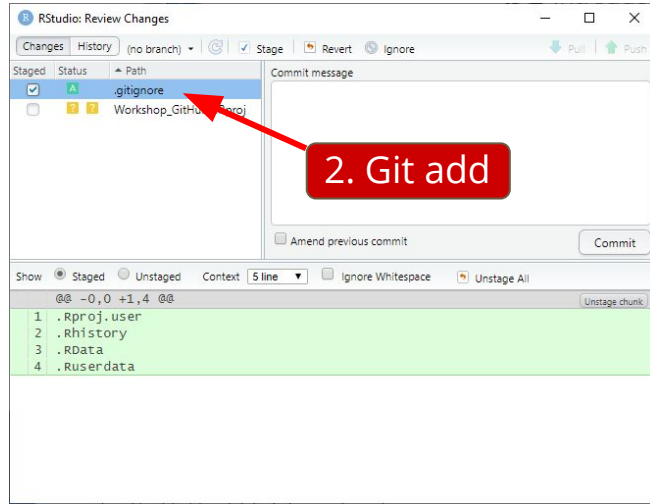
2. Click on the file or changes you want to add to "staging area" (green highlight is added changes, red highlight is deleted changes)

3. Commit the file and all of its changes with some useful comment about the changes

4. Close the pop-up window
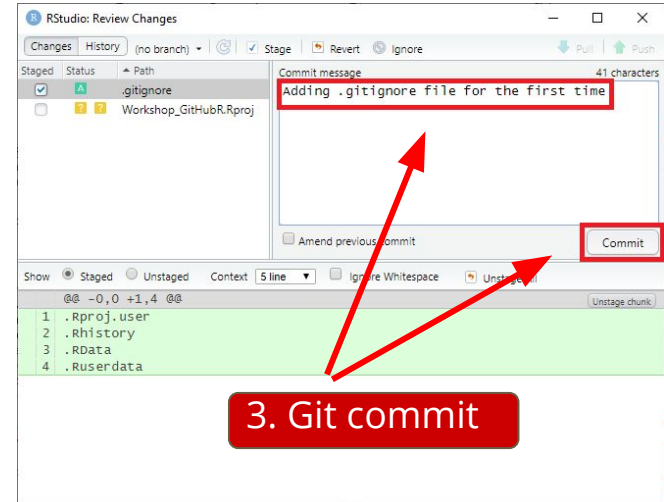
4.b. Push action
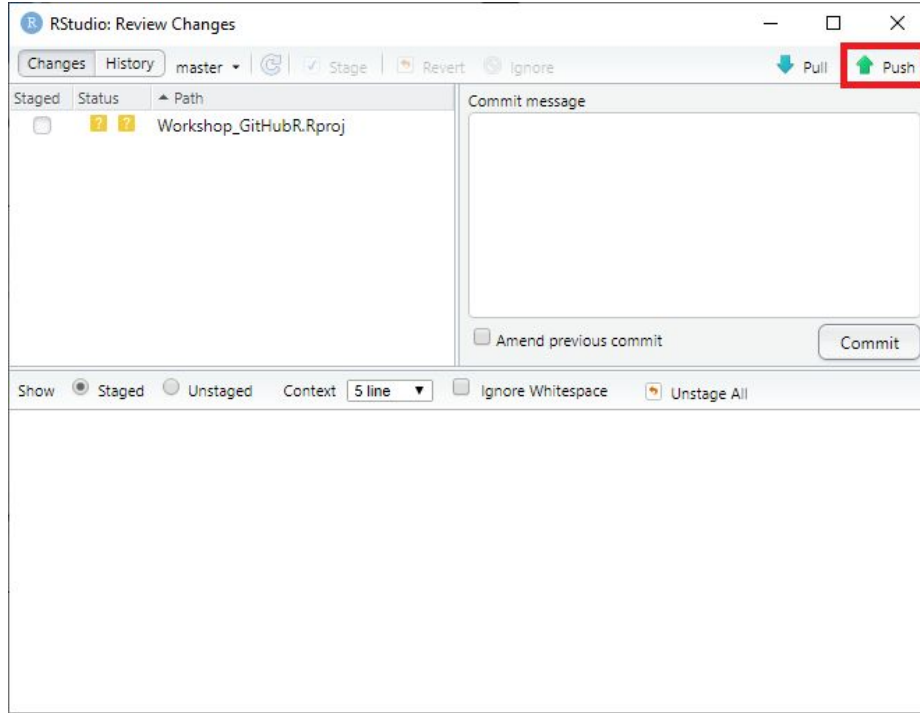
4. Git push

Click the push button to send the files/changes to your remote (online) repository in GitHub

A master branch has been created

4.b. Push action

Congratulations! You have just added, committed and pushed a file into a GitHub repository within an R project in RStudio desktop!
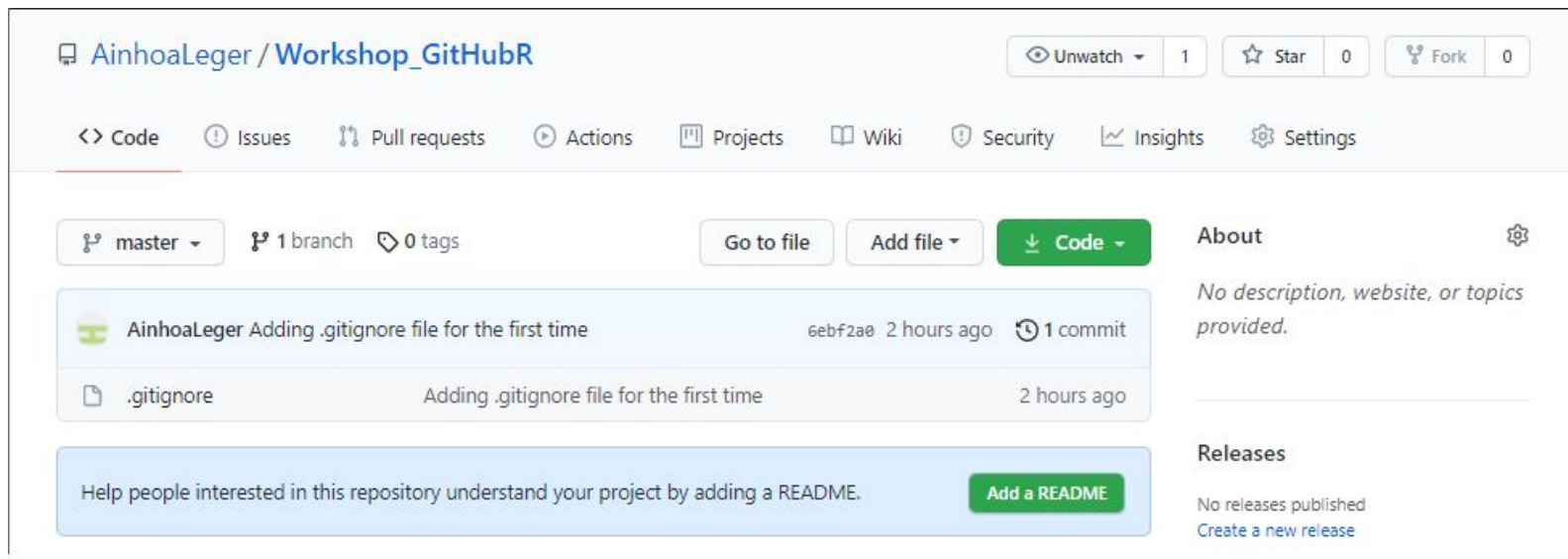
PART 2
Now it gets collaborative

Topics

Individual use

    1. Install Git + Setup a GitHub account

    2. Create a GitHub repository

    3. Connect a GitHub repository to an RStudio project

    4. Version control the files

        a. Commit action

        b. Push action

Collaborative use

    5. Fork GitHub repositories

    6. Create Pull Request

    7. Accept/Merge changes

    8. Refresh forked repo

5. Forking* repos

Fork a repo

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.
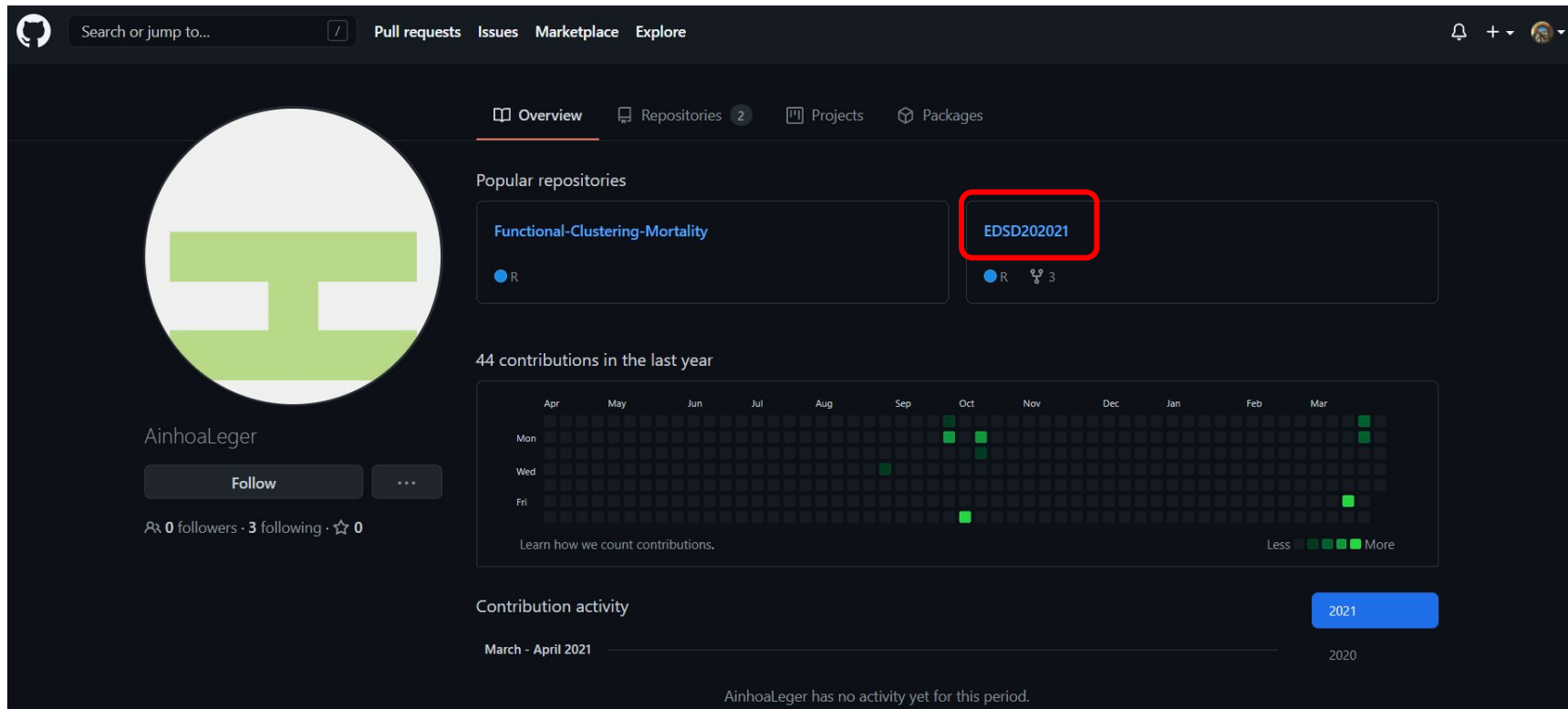
Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.

Propose changes to someone else's project

For example, you can use forks to propose changes related to fixing a bug. Rather than logging an issue for a bug you've found, you can:

- Fork the repository.
- Make the fix.
- Submit a pull request to the project owner.

5. Forking repos

5. Forking repos

5. Forking repos

5.a. Pull* action

Instead of pushing (sending), this time we need to bring information (pulling).

Just 2 simple steps:

1. Create R project just like Ainhoa showed

2.
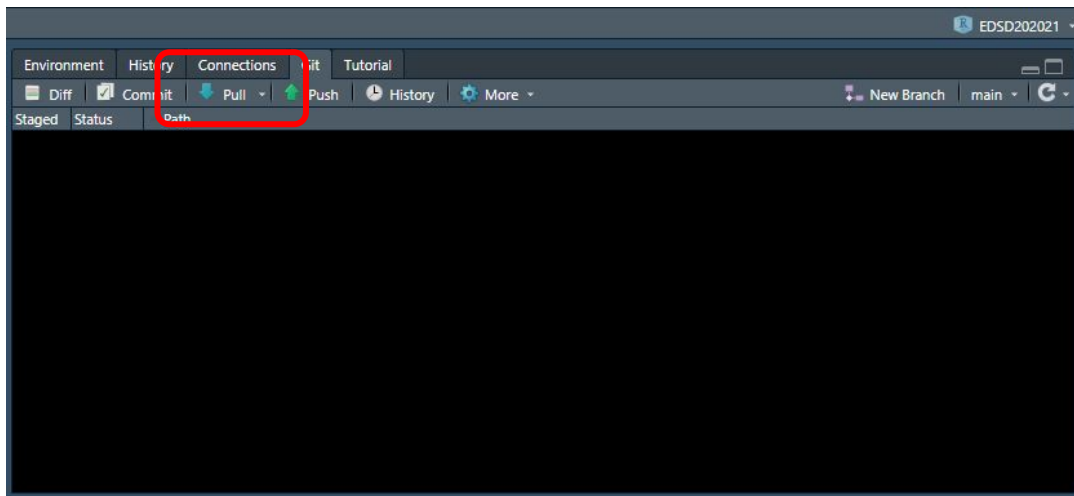
5.b. Commit + Push on forked repo

- Same as Ainhoa actions.
- What's different then?
  - Changes will be pushed to repository in our **own** Github
  - Other people won't be modified (yet)
  - We are 1 step closer to work collaborative

Topics

Individual use

    1. Install Git + Setup a GitHub account

    2. Create a GitHub repository

    3. Connect a GitHub repository to an RStudio project

    4. Version control the files

        a. Commit action

        b. Push action

Collaborative use

    5. Fork GitHub repositories

    6. Create Pull Request

    7. Accept/Merge changes

    8. Refresh forked repo

6. Pull Request*

About pull requests

Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

6. Pull Request

6. Pull Request

Now is the other guy's problem...

Topics

Individual use

    1. Install Git + Setup a GitHub account

    2. Create a GitHub repository

    3. Connect a GitHub repository to an RStudio project

    4. Version control the files

        a. Commit action

        b. Push action

Collaborative use

    5. Fork GitHub repositories

    6. Create Pull Request

    7. Accept/Merge changes

    8. Refresh forked repo

7. Accept/Merge changes

7. Accept/Merge changes

Step 1

7. Accept/Merge changes

You got Mail

7. Accept/Merge changes

7. Accept/Merge changes

7. Accept/Merge changes

7. Accept/Merge changes

7. Accept/Merge changes

7. Accept/Merge changes

Topics

Individual use

1. Install Git + Setup a GitHub account
2. Create a GitHub repository
3. Connect a GitHub repository to an RStudio project
4. Version control the files
    a. Commit action
    b. Push action

Collaborative use

5. Fork GitHub repositories
6. Create Pull Request
7. Accept/Merge changes
8. **Refresh forked repo**

8. Refresh forked repo

8. Refresh forked repo

8. Refresh forked repo

8. Refresh forked repo

8. Refresh forked repo

8. Refresh forked repo

8. Refresh forked repo

8. Refresh forked repo

Same steps as in Merging, until...

8. Refresh forked repo

8. Refresh forked repo

And don't forget to re-pull the repo in your RStudio!

As to basic Git, that's all you need to know

9.

9.a. Git commands

- We did many actions directly in the browser
- But we could use Git's shell for:
  - Fork repo
  - Create pull request
  - Merge changes
  - Refresh forked repo
  - more...
- Why? Faster!
- How?
  - https://happygitwithr.com/git-commands.html
  - https://dzone.com/articles/top-20-git-commands-with-examples
  - https://git-scm.com/docs/git

9.b. Branches

- Branch?
  - Is an independent line of development.
- Example?
  - Different regression models, but pushing just the one that fits best.
- Why?
  - More tidy, less merging problems
  - Working in parallel
- How?
  - https://happygitwithr.com/git-branches.html
  - https://www.atlassian.com/git/tutorials/using-branches

9.c Gaming with Git

- Interactive game to learn Git commands
- https://ohmygit.org/

HANDS ON!

Small exercise to feel comfortable

1. Groups of 2: 1 forker & 1 owner
2. Owner:
   a. Create Git repo
   b. Create R project linked to repo
   c. Add R file (provided) and commit + push to repo
   d. Wait for it…
3. Forker:
   a. Fork repo
   b. Pull from forked repo (should have a file in your folder now)
   c. Create change in file
   d. Commit + Push
   e. Create pull request
4. Owner:
   a. Merge changes in own repo
5. Forker:
   a. Refresh merged repo

See you back in 20'

Q&A