Use Git in Rstudio

Randy Jin

2021022

coloradoSPH

Version Control

“Did you think the lion was sleeping because he didn't roar?”
— Friedrich Schiller, Die Verschwörung des Fiesco zu Genua

“Coding is like climbing; coding without version control is like free-climbing: you can travel much faster in the short-term, but in the long-term the chances of catastrophic failure are high!” --”*R for Data Science*”

Git and GitHub

- **Git**
  - an open-source, version control tool
  - created in 2005
  - on the Linux operating system;
- **GitHub**
  - a company founded in 2008
  - Tool to integrate with **git**
  - You **do** not need **GitHub** to **use git**
  - But you cannot **use GitHub without using git**.

Git is most useful when combined with GitHub

Git

- Git is a totally different language
  - pwd: print working directory. This tells you which directory you're currently in.
  - cd <name>: change directory. Use cd .. to move up the directory hierarchy.
  - ls: list files. Shows all files in the current directory.

- recommend playing Terminus
- Philip Guo's Basic Unix-like command line tutorial

- Windows: https://git-scm.com/download/win.
- OS X: https://git-scm.com/download/mac.

Why Git Rules the world

Save Time: Use your time for something more useful than waiting for your version control system to get back to you.

Work Offline: With Git, almost everything is possible simply on your local machine: make a commit, browse your project's complete history, merge or create branches... Git let's you decide where and when you want to work.

Undo Mistakes Git rarely really deletes something. This is peace of mind.

Don't Worry In Git, every clone of a project that one of your teammates might have on his local computer is a fully usable backup. That means that losing data or breaking a repository beyond repair is really hard to do.

Don’t Mix Things Up Branching is the answer for all problems.

https://www.git-tower.com/learn/git/ebook/en/command-line/appendix/why-git

Why Git + GitHub in Rstudio

- Multiple user work on the same file at the same time
  - combine changes automatically
  - the ambiguities and conflicts
  - record everything and blames
- Readers can easily browse code (Markdown)
  - report bugs with GitHub issues
  - propose with pull requests.
- It makes sharing your package easy.
  - Any R user can install your codes or packages:

```
install.packages("devtools“)
devtools::install_github("username/packagename")
```

RStudio, Git and GitHub

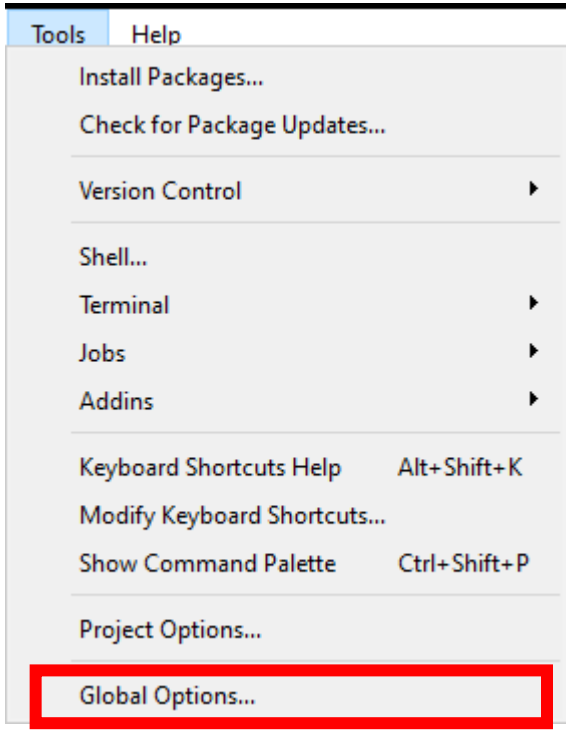
1. Install Git:
   - Windows: https://git-scm.com/download/win.
   - OS X: https://git-scm.com/download/mac.
2. Tell Git your name and email address

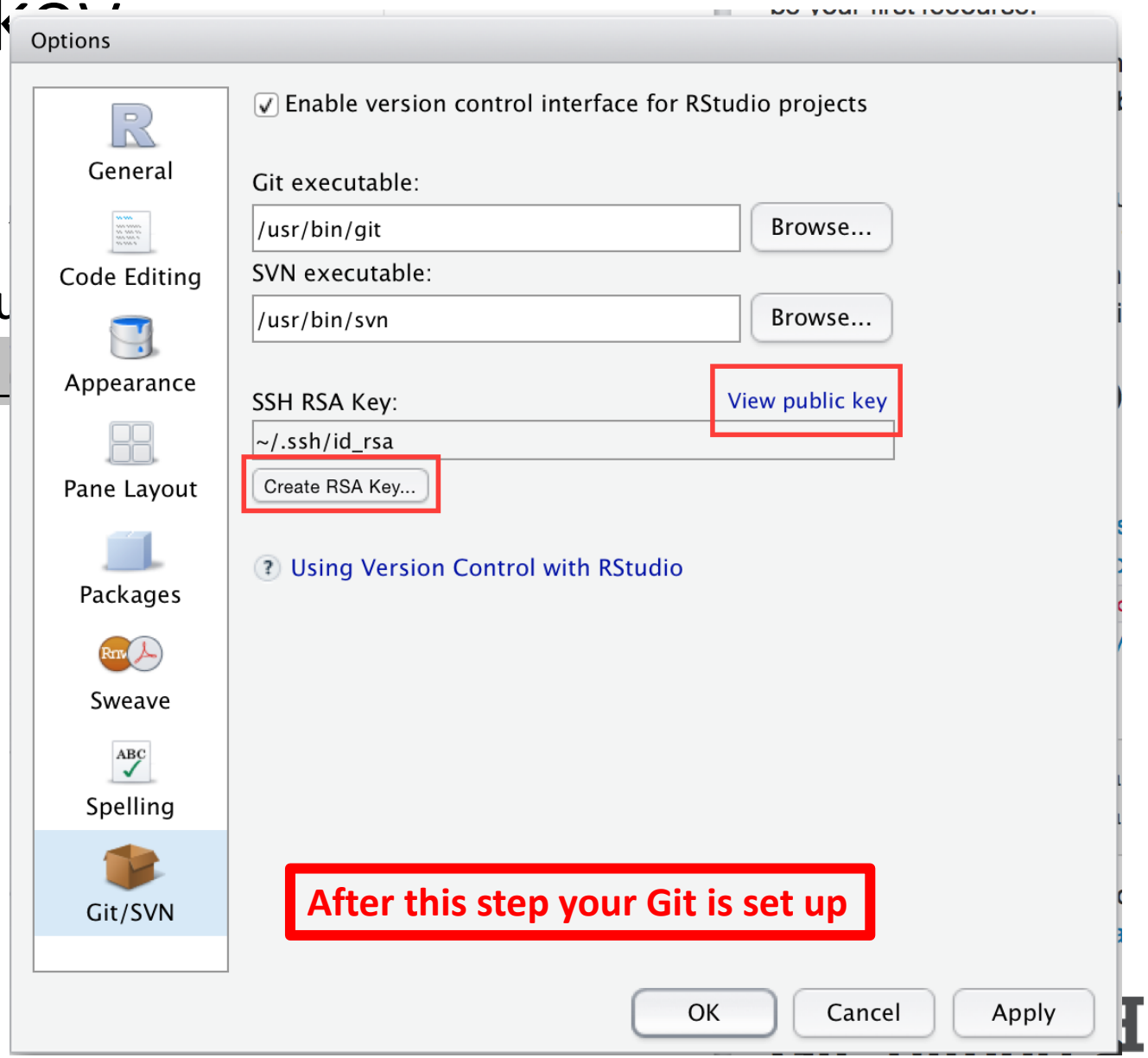
```
git config --global user.name "YOUR FULL NAME"
git config --global user.email "YOUR EMAIL ADDRESS”
```

3. Create an account on GitHub <https://github.com>
4. Generate a SSH key (next slide)
5. Give GitHub the key <https://github.com/settings/ssh>

sidenotes for SSH key

- SSH keys
  - securely communicate with
  - two parts to SSH key: one pu
  - In R run this `file.exists("~/.ssh/id_`



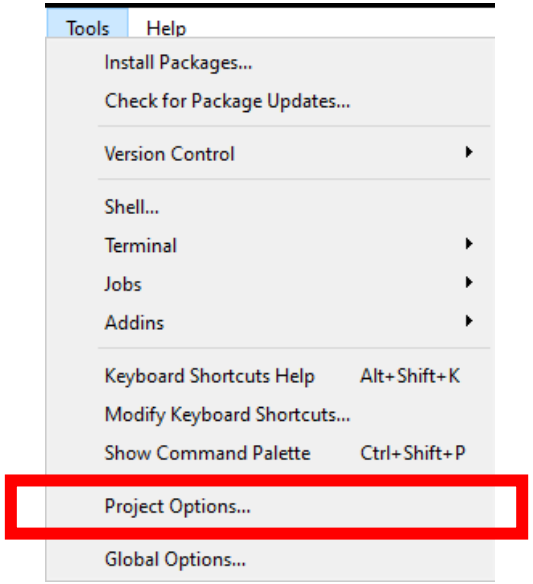


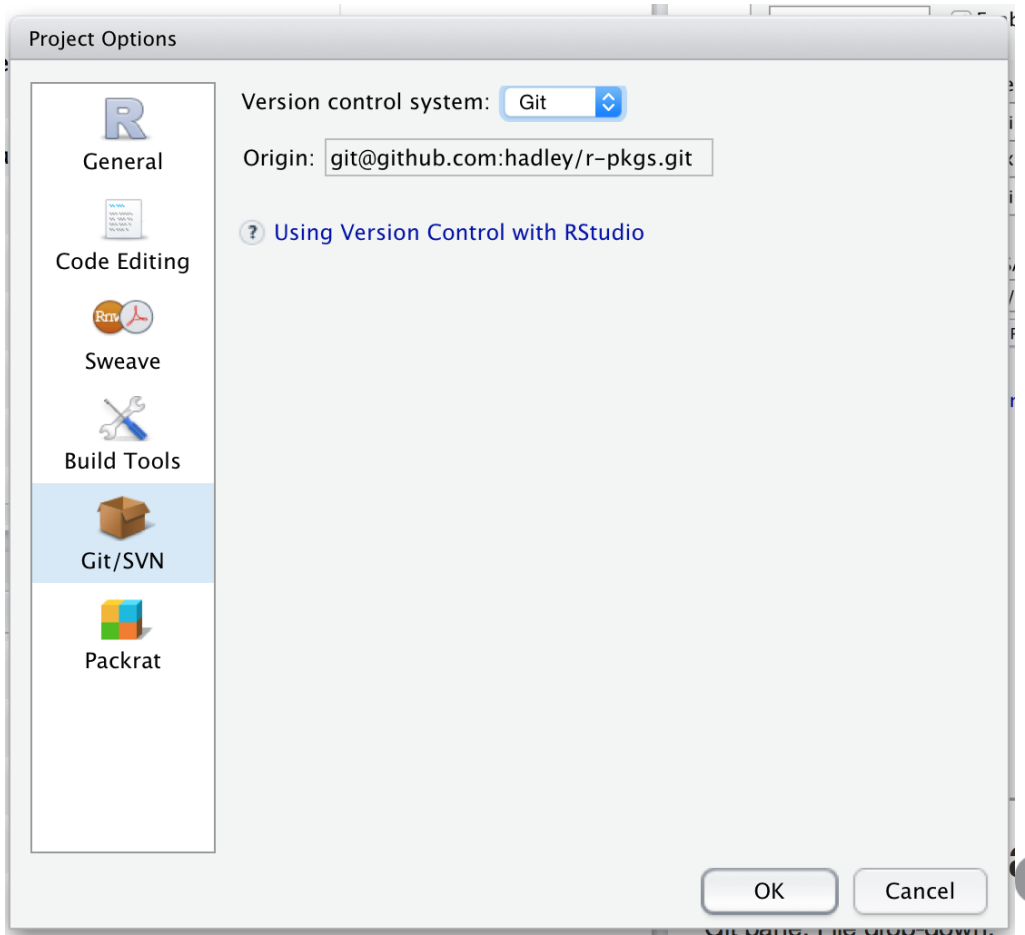
sidenotes for SSH key

- SSH keys
  - securely communicate with websites without a password
  - two parts to SSH key: one public, one private
  - In R run this `file.exists("~/.ssh/id_rsa.pub")`

Create a local Git repository

- "Version control system" from "None" to "Git"



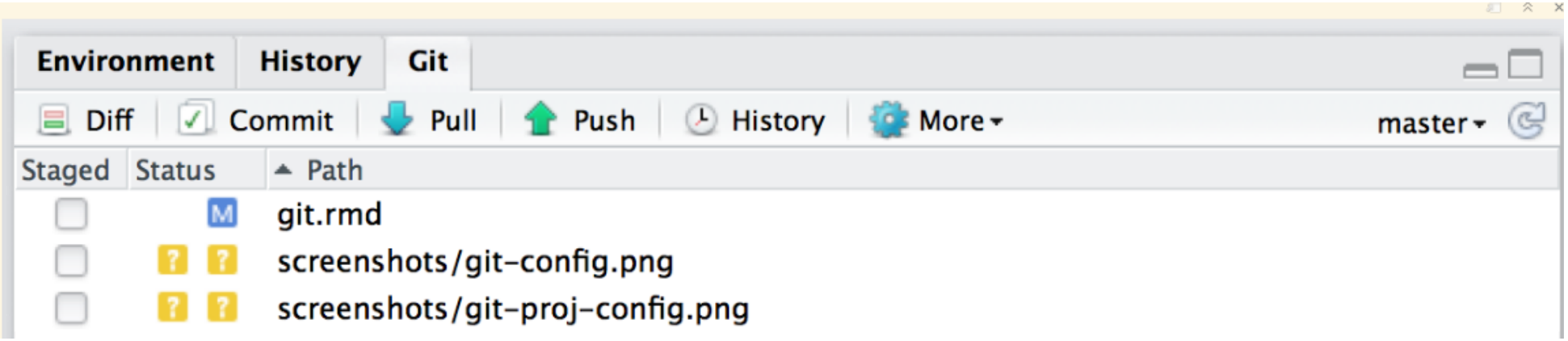


Create a local Git repository

- You'll then be prompted to restart RStudio

  Bash shell run git init

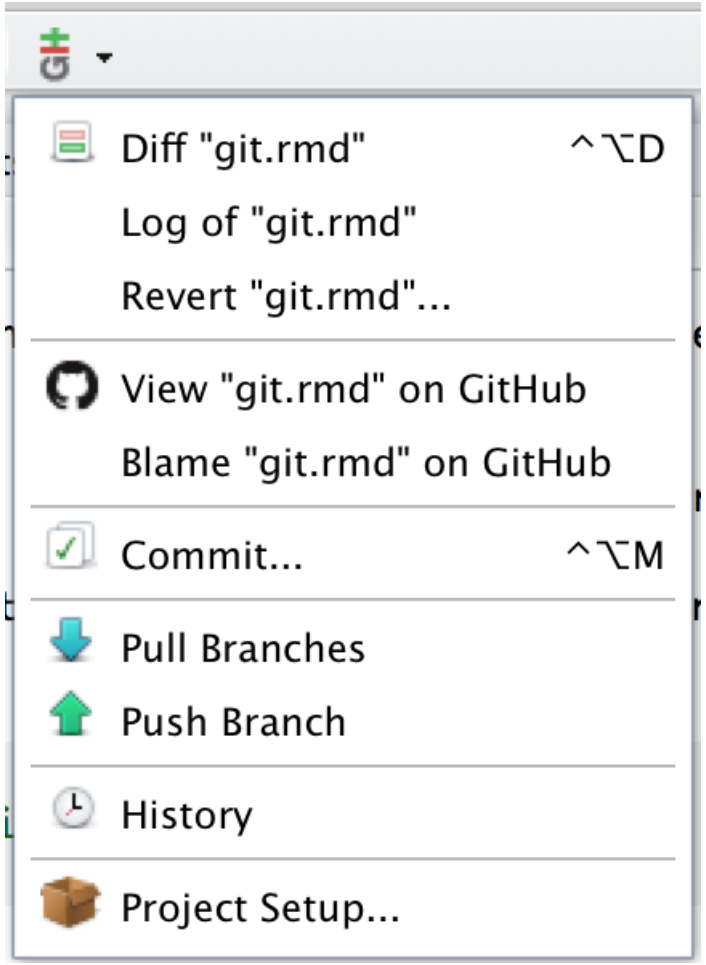
- Restart RStudio and reopen your package
- After Git has been initialized,
  two new components:



Create a local Git repository

- You'll then be prompted to restart RStudio

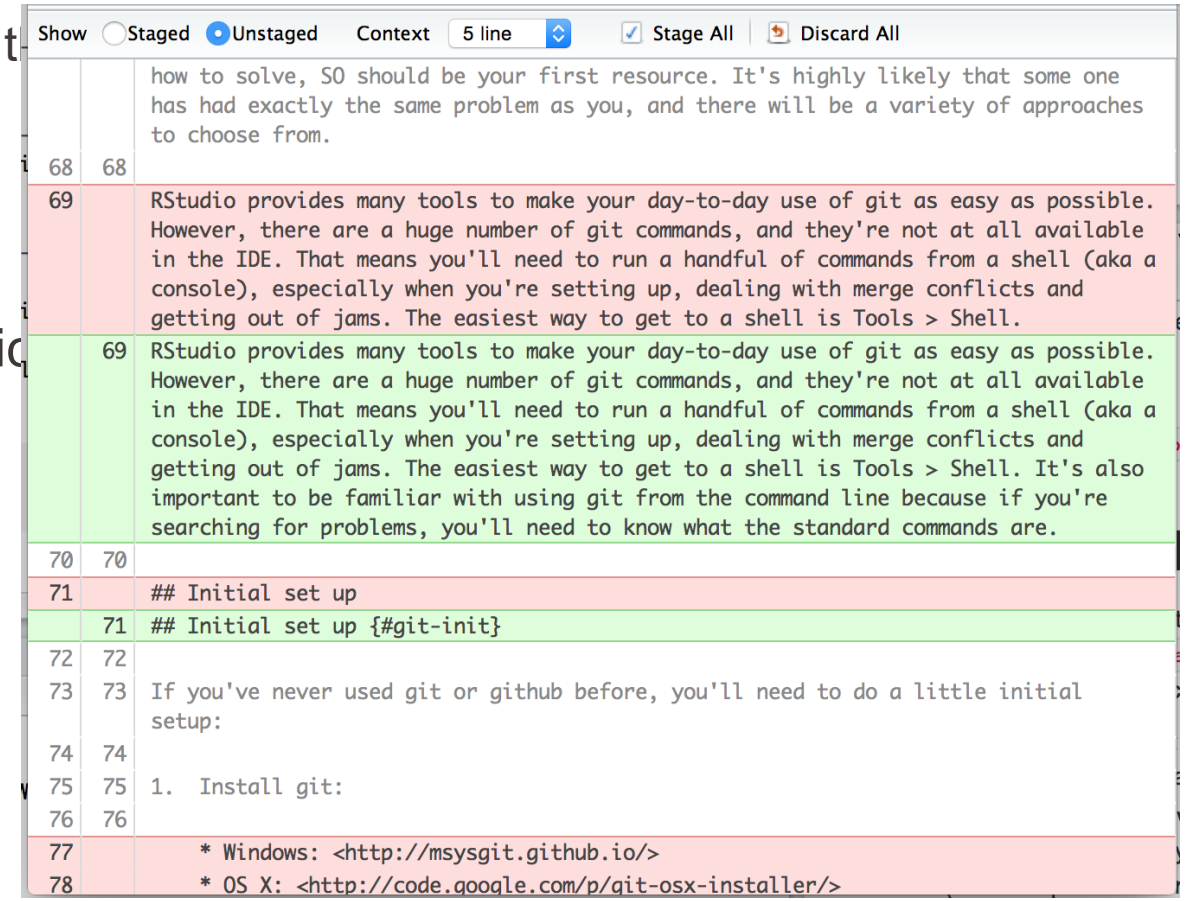
  Bash shell run git init
- Restart RStudio and reopen your package
- After Git has been initialized,
  two new components:



What has been changed

- , **Modified**. You've changed the contents of the file.
- , **Untracked**. You've added a new file t
- , **Deleted**. You've deleted a file.

Diff get more details about modific



stuff

coloradoSPH

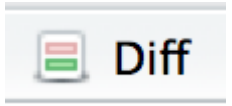
What has been changed

- 

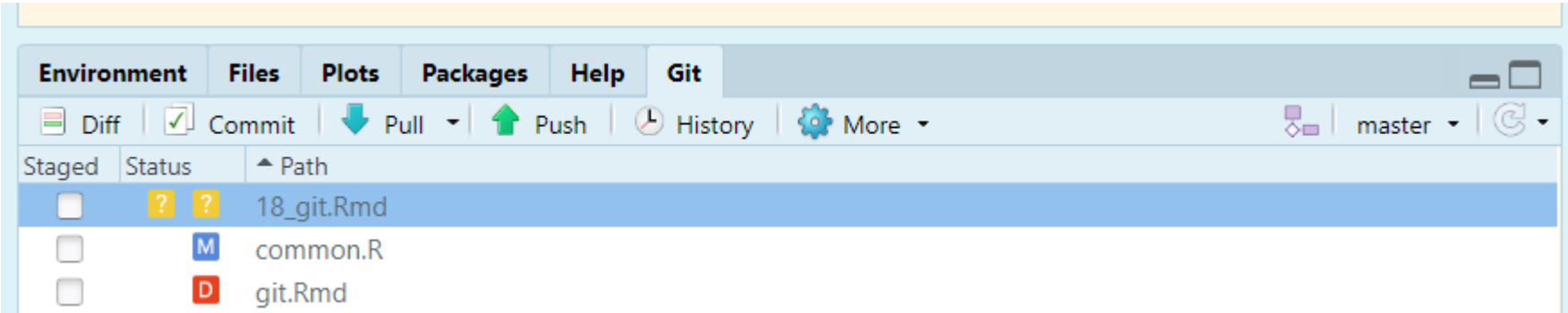
, **Modified**. You've changed the contents of the file.
- 

, **Untracked**. You've added a new file that Git hasn't seen before.
- 

, **Deleted**. You've deleted a file.



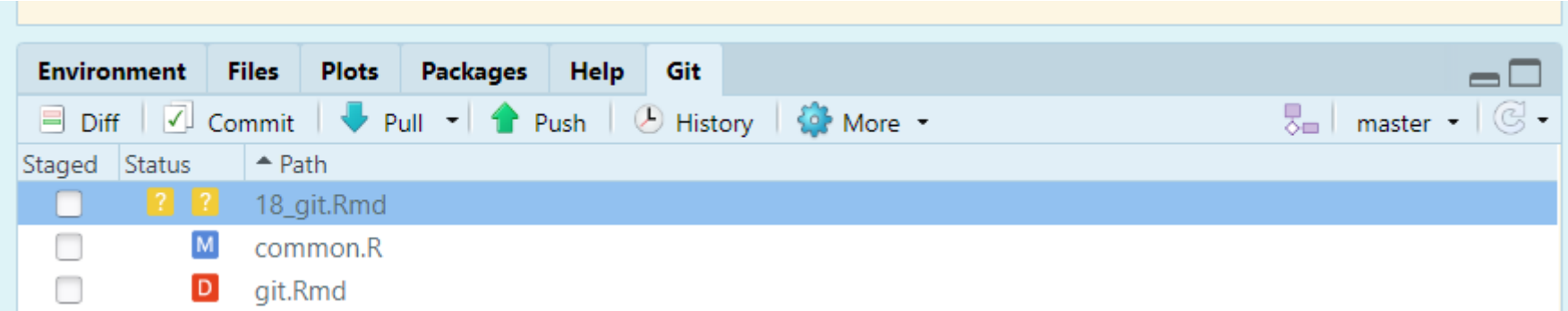
get more details about modifications with a "diff"



Commit changes

- The fundamental unit of work in Git is a **commit**
  - a snapshot of your code
  - at a specified point in time
  - like an anchor when climbing

"Committing too frequently will slow your progress; use more commits when you're in uncertain or dangerous territory. Commits are also helpful to others, because they show your journey, not just the destination." – Hadley Wickham



Commit components

- SHA (short for secure hash algorithm)
- Changeset: files were added, modified and deleted
- Human-readable commit message
- Parent commit that came before current comment
- An author (remember the user name and email?)

Commit steps

1. You **stage** files

   telling Git which changes should be included in the next commit

2. You **commit** the staged files

- the current as the Git pane in RStudio window.
- the bottom the diff pane shows of the currently file.
- the top-right pane for the commit message

Commit steps

- 1. save your changes
- 2. open the commit window
- 3. select files or select all
- 4. stage files
- 5. MUST commit message
- 6. commit confirmation

- Added: A : after staging an untracked file, Git now knows that you want to add it to the repo.
- Renamed: R : If you rename a file, Git initially sees it as a deletion and addition. Once you stage both changes, Git will recognise that it's a rename.

Sometimes you'll see a status in both columns, e.g. M M. This means that you have both staged and unstaged changes in the same file. This happens when you've made some changes, staged them, and then made some more. Clicking the staged checkbox will stage your new changes, clicking it again will unstage both sets of changes.

Staging files is a little more complicated in the shell.
You use git add to stage new and modified files,
and git rm to stage deleted files. To create the commit, use git commit -m <message>.

Sync with GitHub

- To publish, or **push**, your code to GitHub:

1. Create a new repo on GitHub: https://github.com/new.

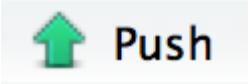
   Give it the same name as your package

   Include the package title as the repo description

   Leave all the other options as is, then click Submit.

2. Open a shell, too complicated for me to use

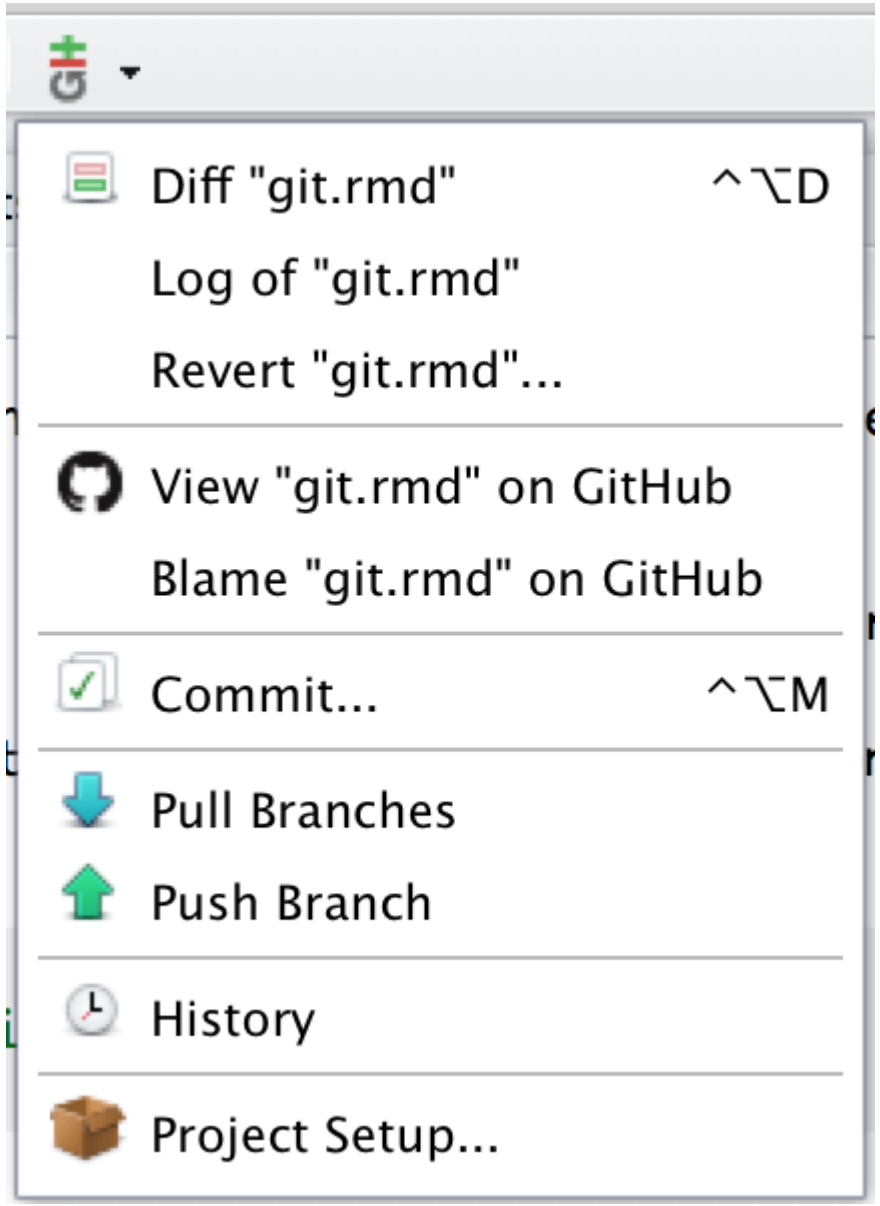
Sync with GitHub

- Modify something
- add URL and BugReports fields link to new GitHub site
- Save the file and commit 

- **Push** your changes to GitHub
- Go to your GitHub page and look

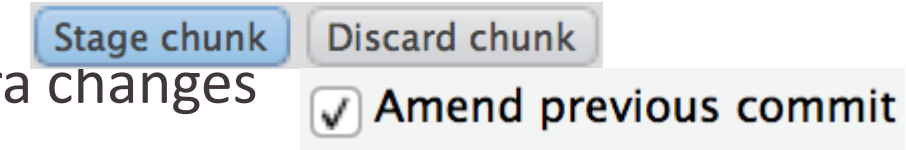
Check point

- Tracking history
- Blame view
- Comments on commits



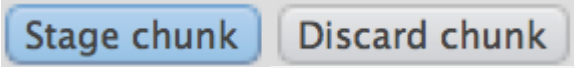
Check point

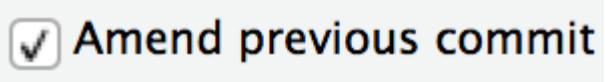
- To undo the change you've just made
  - right click on the file select "revert".
  - diff panel to change previous changes
  - modify the previous commit with extra changes



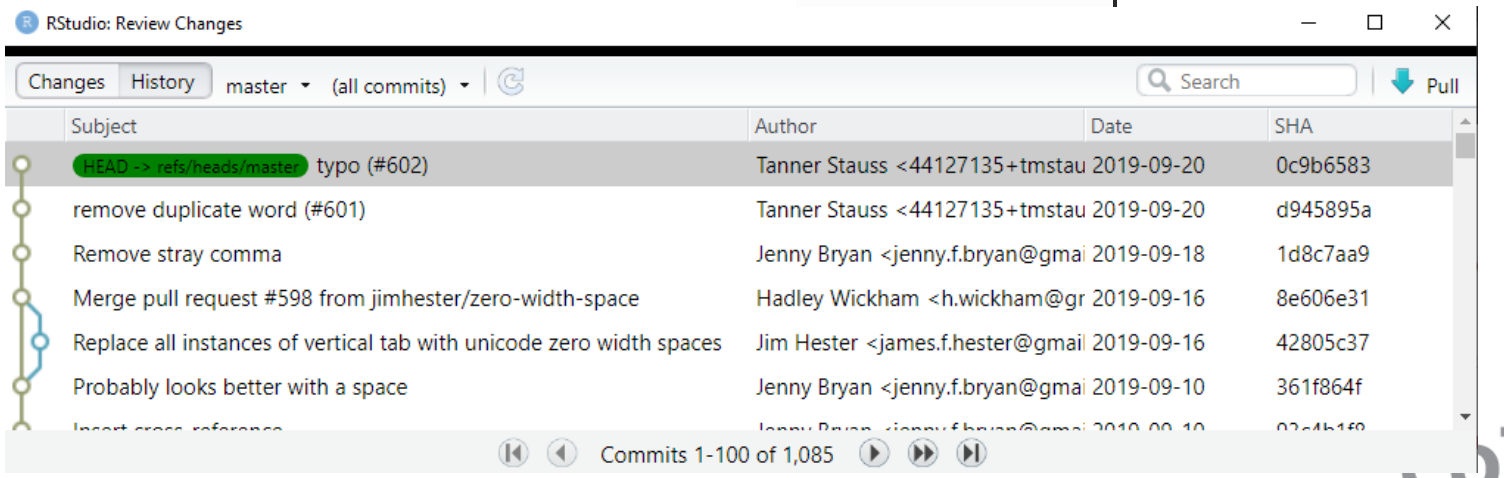
- What if everything goes wrong
  - The top part lists every commit to your repo.
  - The bottom part shows you the commit:
    - SHA (the unique id), the author, the date, the parent and the changes in the commit.
    - This is an advanced technique called **rebasing history**. As you might imagine, going back in time to change the past can have a profound impact on the present.

Check point

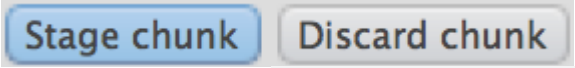
- To undo the change you've just made
  - right click on the file select "revert".
  - diff panel to change previous changes 

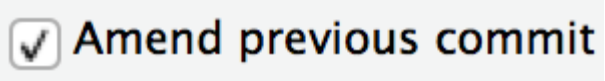
  - modify the previous commit with extra changes 

- What if everything goes wrong 



Check point

- To undo the change you've just made
  - right click on the file select "revert".
  - diff panel to change previous changes 

  - modify the previous commit with extra changes 

- What if everything goes wrong
- **rebasing history**
  as you might imagine going back in time to change the past can have a profound impact on the present.

Working with others

- You use **push** to send your changes to GitHub.
- Locally you'll need to **pull** their changes from GitHub.
  - Git first downloads (**fetches**) all of the changes
  - then **merges** them with the changes that you've made
  - A merge is a commit with two parents.
  - You'll need to resolve the **merge conflict** yourself

To resolve a merge conflict, you need to open every file with the status U U . In each file, you'll find a conflict marker that looks like this:

```
<<<<<<< HEAD

||||||| merged common ancestors

=======

>>>>>>> remote
```

Working with others

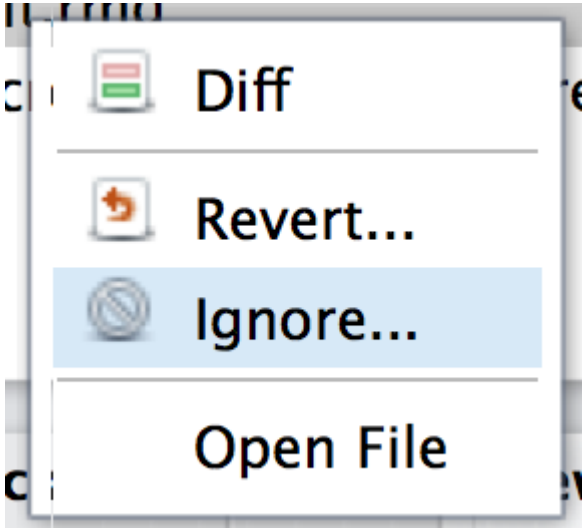
- At the top, your local code.
- In the middle, the code from the last commit before the split between the two lines of development
- At the bottom, the remote code that you pulled down from GitHub.

To resolve a merge conflict, you need to open every file with the status U U . In each file, you'll find a conflict marker that looks like this:

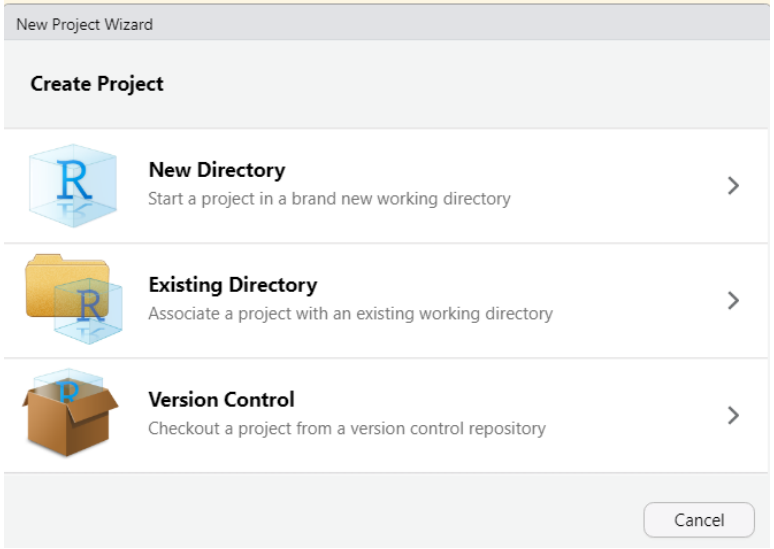
```
<<<<<<< HEAD

||||||| merged common ancestors

=======

>>>>>>> remote
```

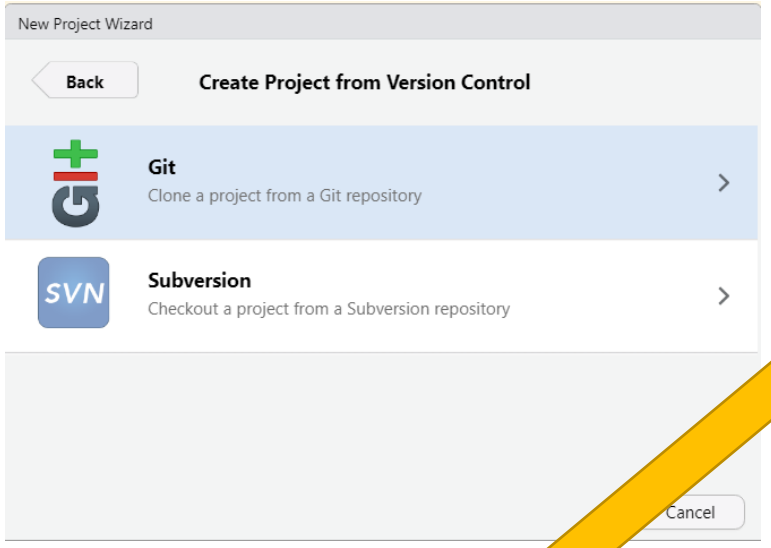
Ignore files

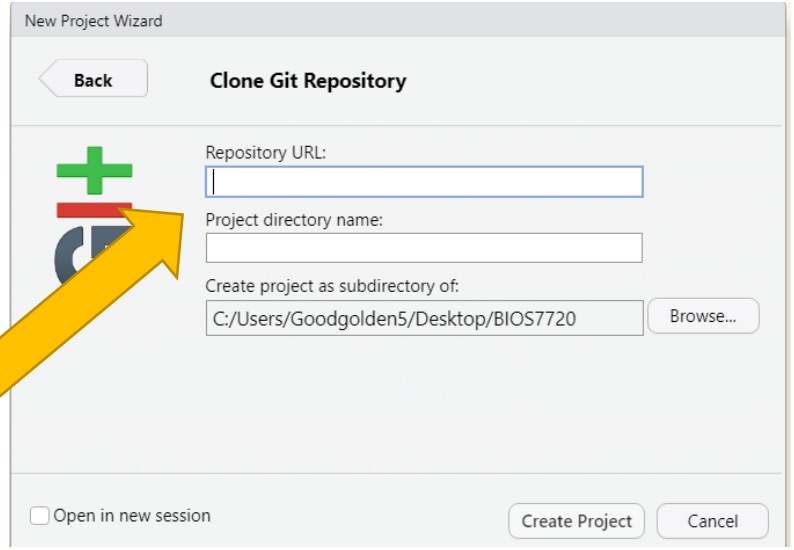
- data you don’t want to include in the repository.
- rather than carefully not staging them each time
- you should instead add them to .gitignore.
  - Using ignore panel
  - Adding into .gitignore file

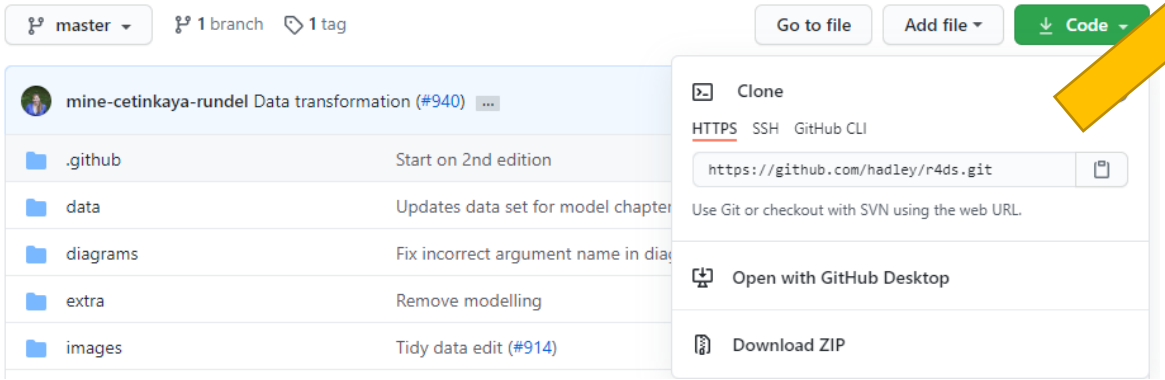


Get Repo from GitHub







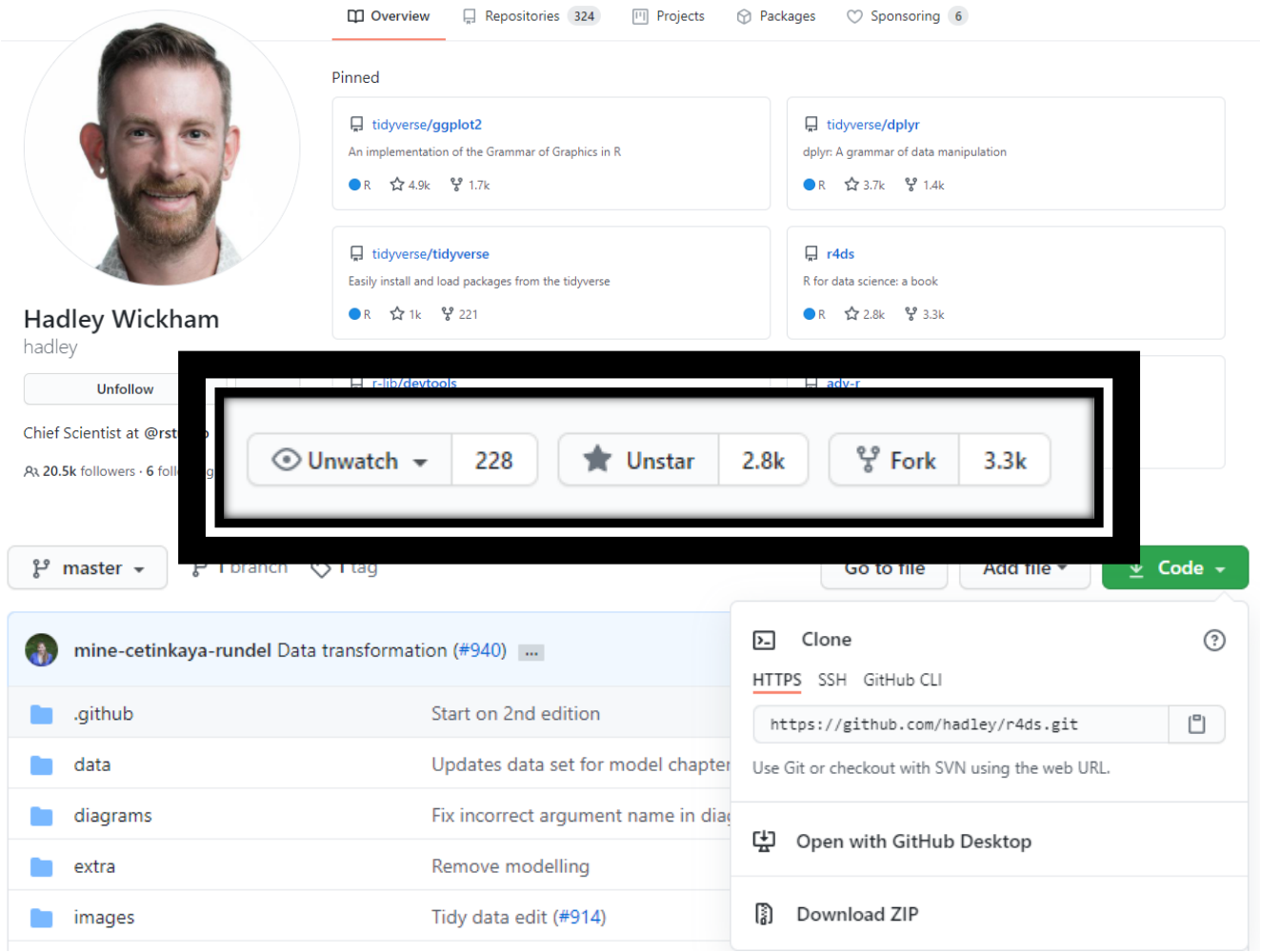


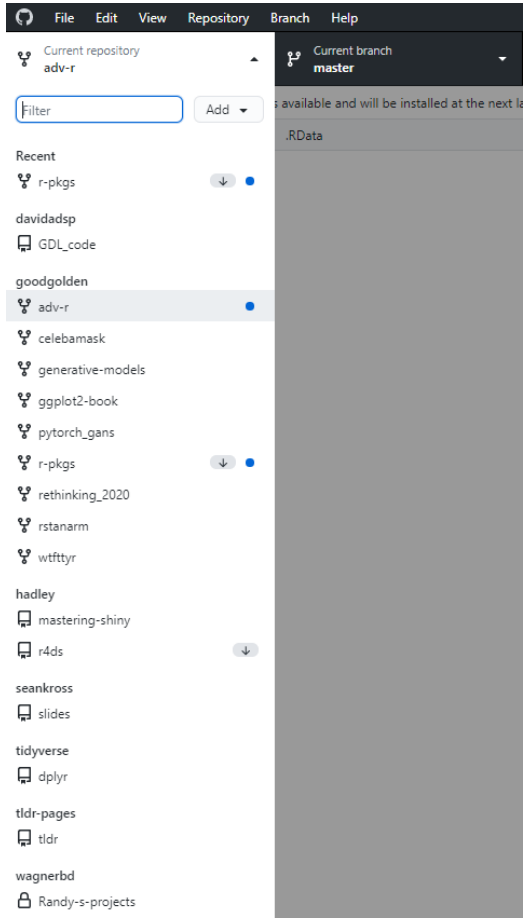
stuff

coloradoSPH

https://desktop.github.com/

4. My own library

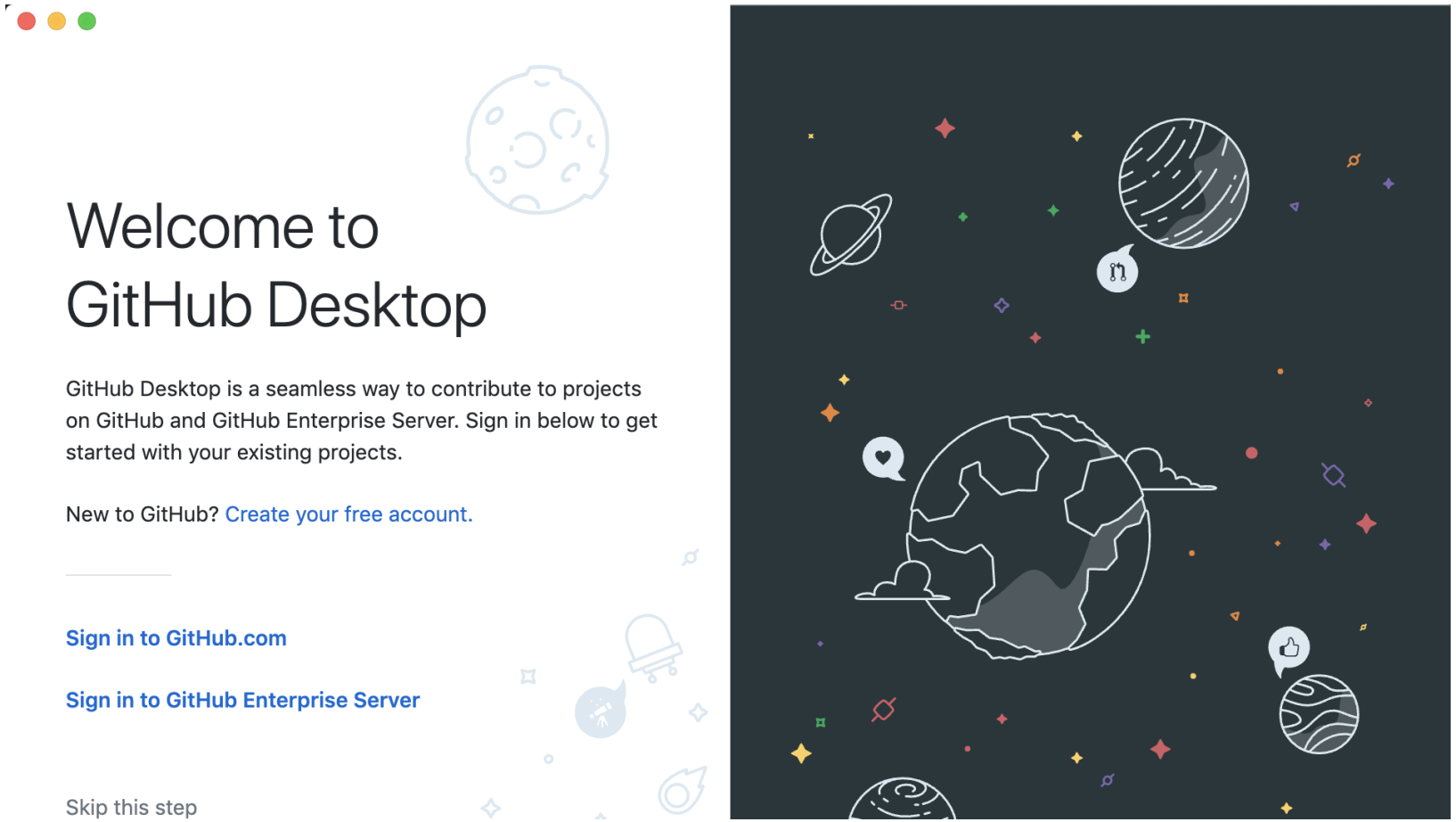




stuff

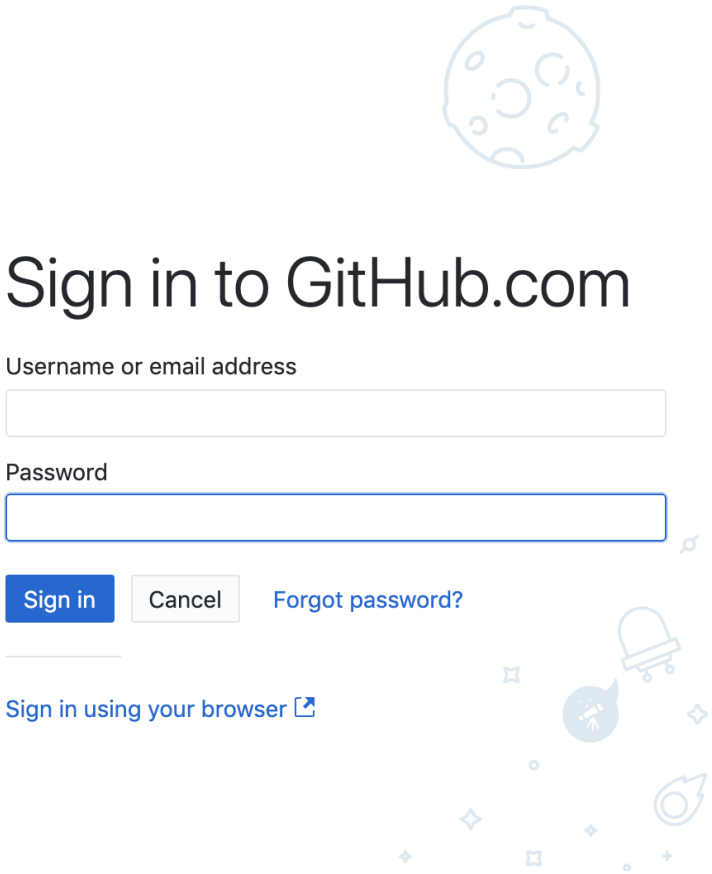
coloradoSPH

https://desktop.github.com/



stuff

coloradoSPH



coloradoSPH

Your Identity

The first thing you should do when you install Git is to set your user name and email address. This is important because every Git commit uses this information, and it's immutably baked into the commits you start creating:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Configure Git

This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.

Name

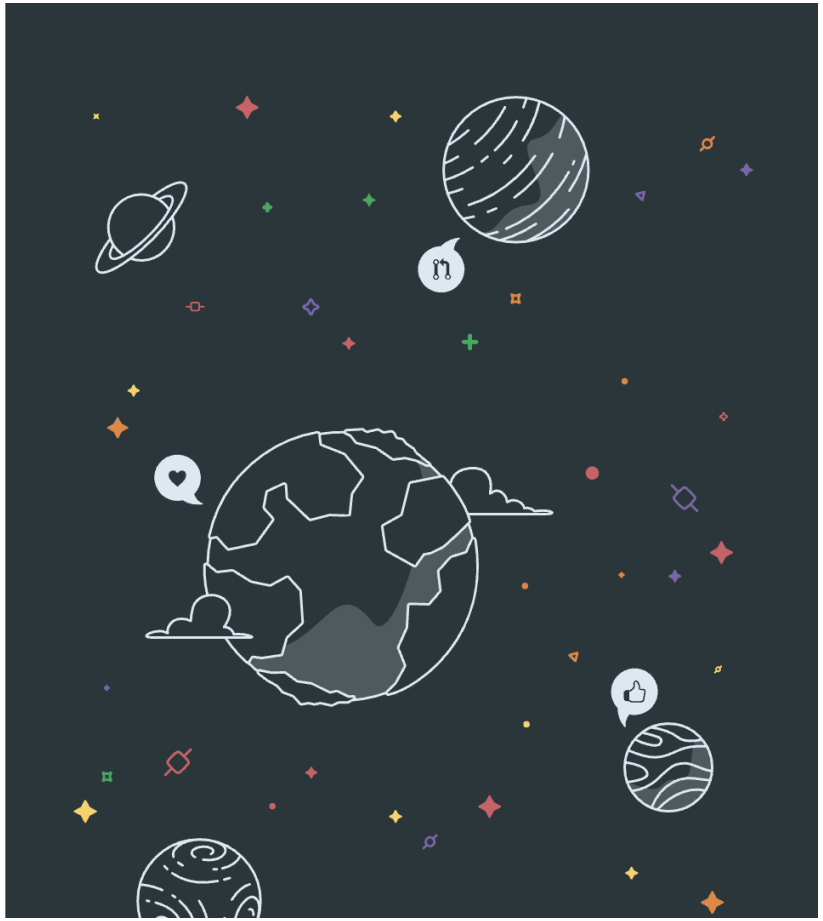
Email

Continue Cancel

Example commit

Fix all the things

Seth Russell committed 30 minutes ago



Let's get started!

Add a repository to GitHub Desktop to start collaborating

Create a Tutorial Repository...

Clone a Repository from the Internet...

Create a New Repository on your Hard Drive...

Add an Existing Repository from your Hard Drive...

Filter your repositories

Your Repositories

- balinterdi/rarwe-code
- callahantiff/Abra-Collaboratory
- magic-lantern/abc-test
- magic-lantern/Abra-Collaboratory
- magic-lantern/altair_exploration
- magic-lantern/AntimicrobialCDS
- magic-lantern/becausalfrence
- magic-lantern/bios6645
- magic-lantern/biostatistics-seminar-2017

Current Repository
testing

Current Branch
master

Publish repository
Publish this repository to GitHub

Changes | History

0 changed files

Summary (required)

Description

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Publish your repository to GitHub
This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.
Always available in the toolbar for local repositories or ⌘ P

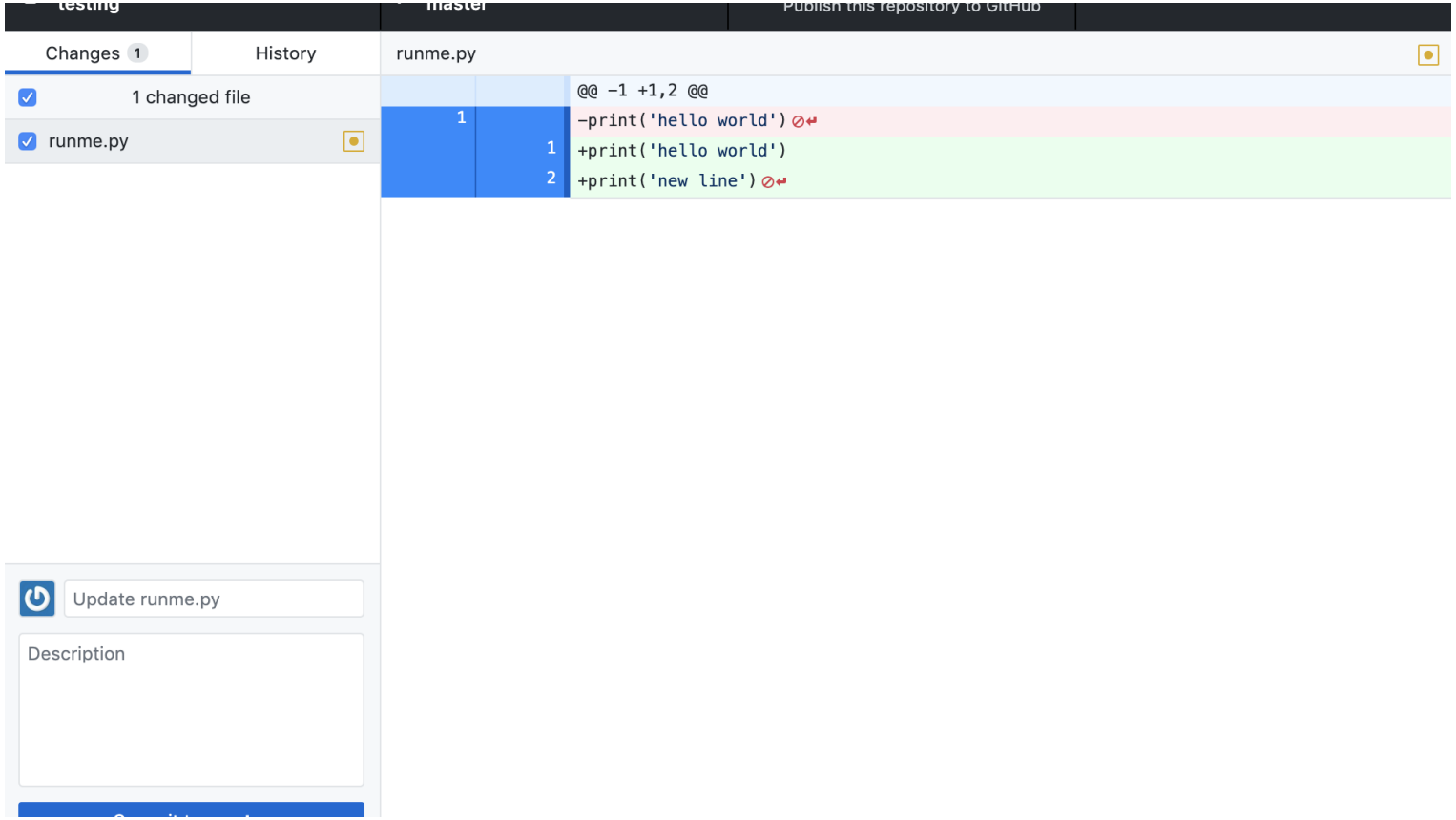
Publish repository

Open the repository in your external editor
Select your editor in Preferences
Repository menu or ⌘ ⇧ A

Open in Visual Studio Code

View the files of your repository in Finder
Repository menu or ⌘ ⇧ F

Show in Finder



coloradoSPH