Intro2R-Git

Version control using Git and Github

Song, Xiao Ping
xp.song@u.nus.edu

Course materials: https://github.com/xp-song/Intro2R-Git
updated 2020-05-19

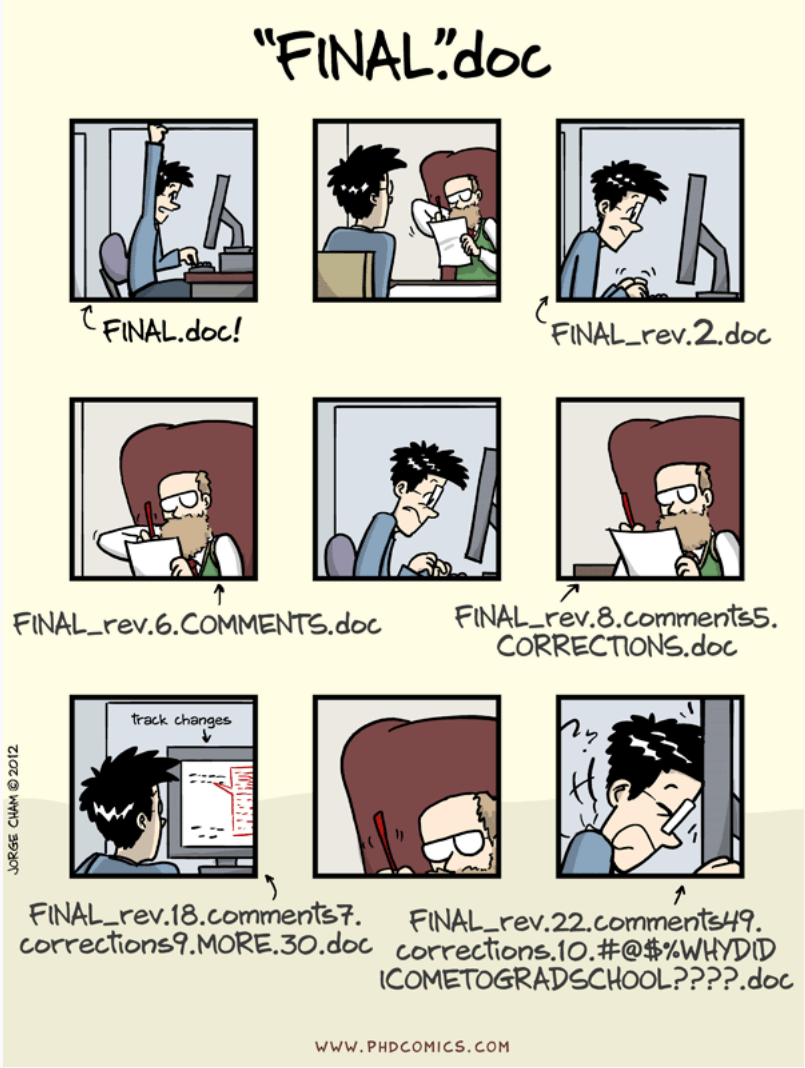
Intro2R-Git

1. Navigate to course webpage for instructions and background information
   https://github.com/xp-song/Intro2R-Git

2. Download workshop materials
   (green button on webpage)

Outline

Why use version control?



Why use version control?

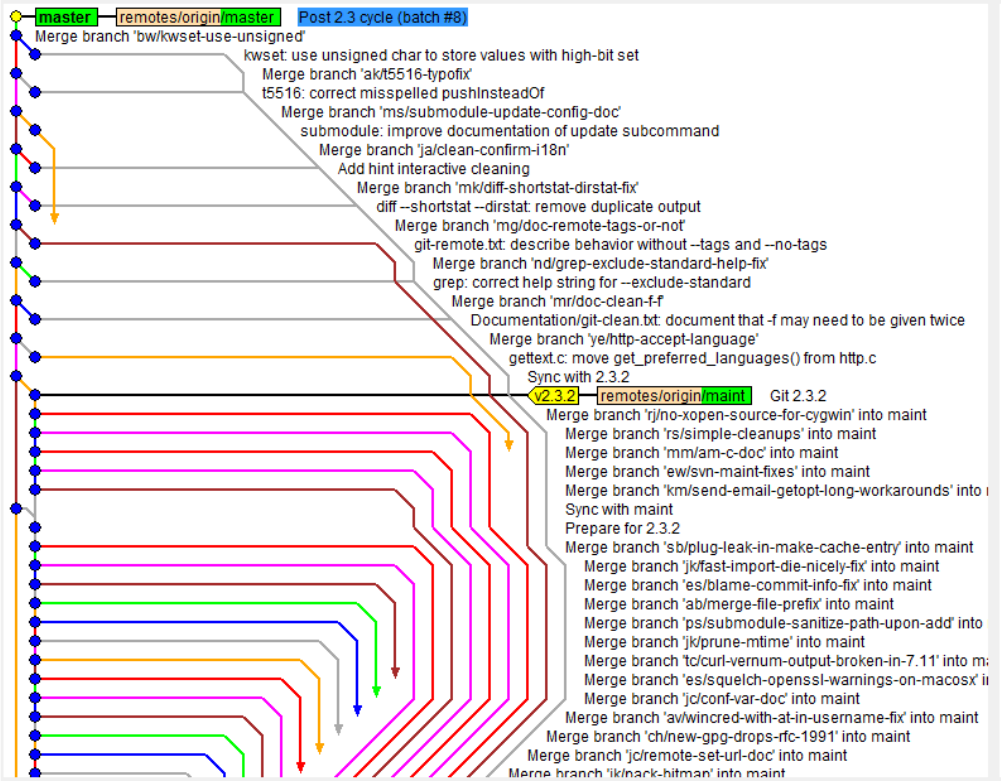
Git

- Version control system
- *Track* changes to your code
- *Visualise* changes with GUI clients (e.g. RStudio, Github Desktop, etc.)
- *Share* changes & collaborate (e.g. Github, Gitlab, Bitbucket)
- Changes are combined automatically
- Systematic way to de-conflict changes

Git + Github

- Widely used in software development
- Most popular for developing R packages

A bit about Git



Git hell (tugberkugurlu.com)

Outline

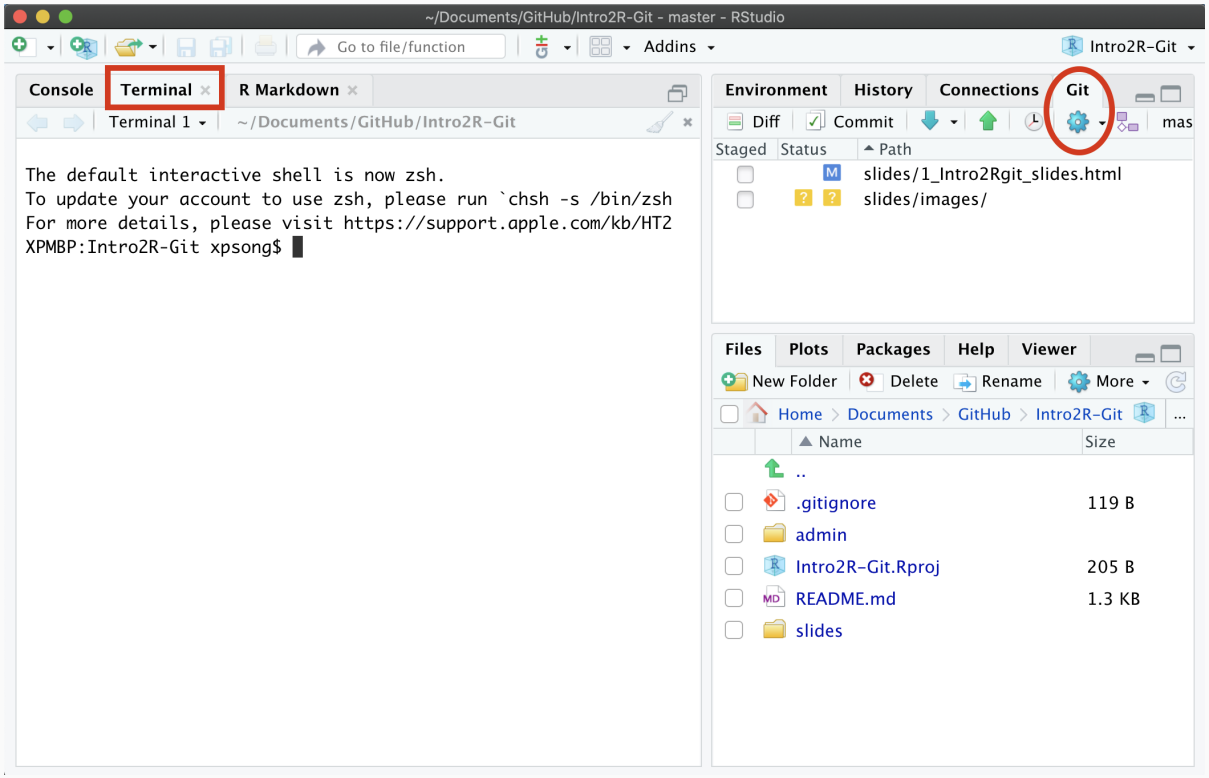
Git with RStudio Projects

- RStudio Projects help organise your work into separate 'R sessions'.
- Each project has it's own workspace a.k.a. 'working directory' (separate configuration files, history, etc.)
- The location of the '*.RProj*' file defines the 'working directory' *and the* ***'Git repository'***

🌟 Best Practice

- Use *relative* paths in your script, based on *.RProj* file location
- Keep all project items in the working directory

Git with RStudio Projects



- If needed, you can access the Shell via *Tools* > *Shell* (or the *'Terminal'* tab)
- Commands will be executed in the working directory (location of the '*.RProj*' file)

Important Shell commands: `pwd` to print working directory; `cd <name>` : change directory, or `cd ..` to move up the directory hierarchy; `ls` : list files in the current directory.

Outline

Initial set up

Install Git

- OS X: http://git-scm.com/download/mac
- Windows: http://git-scm.com/download/win
- Debian/Ubuntu: sudo apt-get install git-core
- Other Linux distros: http://git-scm.com/download/linux

Initial set up

Install Git

Set-up Git

In the *Shell*, configure your name and email for Git:

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

All commits (i.e. tracked changes) you make will be labelled with this information.

You can check if you're set up correctly by running:

```
git config --global --list
```

Initial set up

Install Git

Set-up Git

Github

Create an account at https://github.com

Initial set up

Install Git

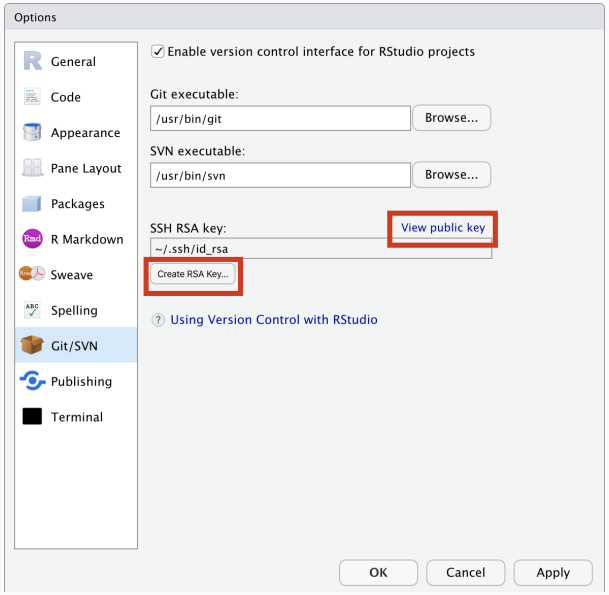
Set-up Git

Github

Security

Use an *SSH key-pair* to reduce the need to key in your password

- The *public* key encrypts data to be read by those with *private* key
- In R, check if it is already present: `file.exists("~/.ssh/id_rsa.pub")`
- Go to *RStudio Preferences* to create/view the public key:



Initial set up

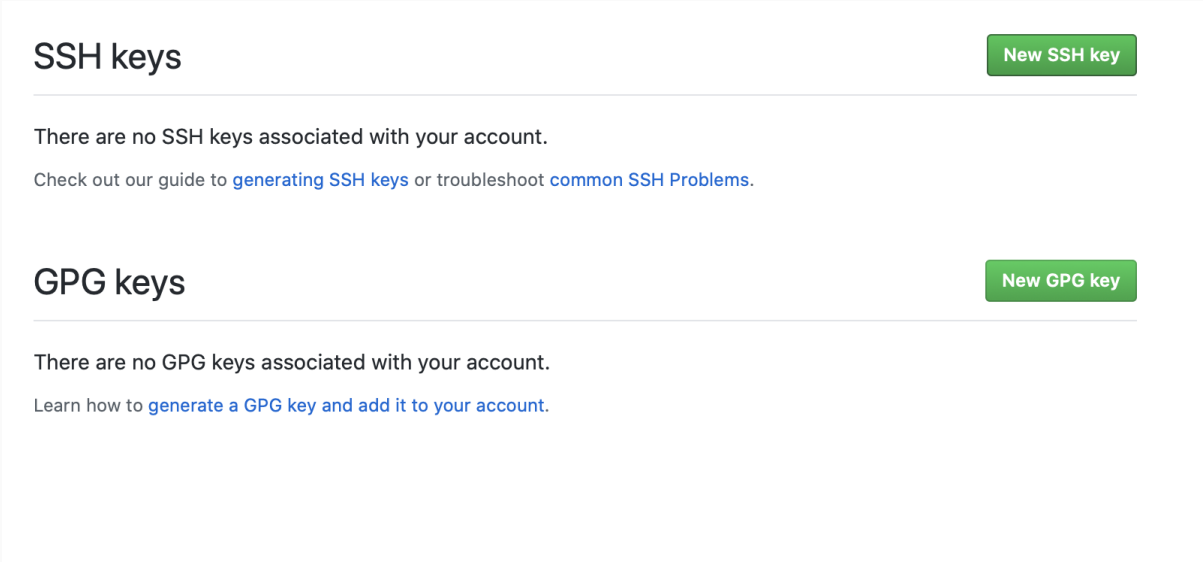
Install Git

Set-up Git

Github

Security

Add SSH key to Github at https://github.com/settings/ssh



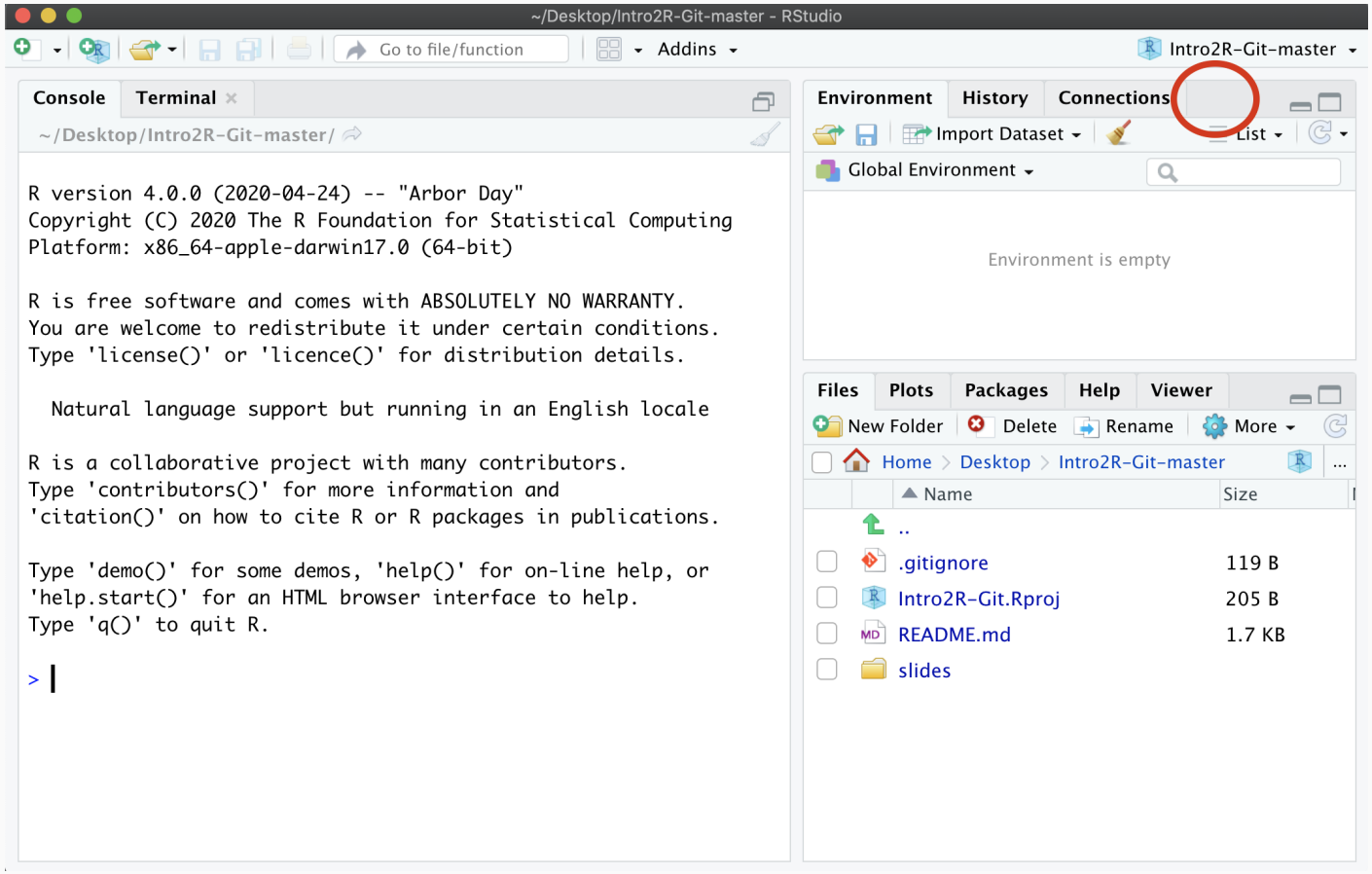
Note: If you've turned on 2FA on Github, create a Personal Access Token (PAT) under the *'Developer settings'* of your *'Account settings'*. Use the generated PAT as the password within RStudio, instead of your regular password.

Outline

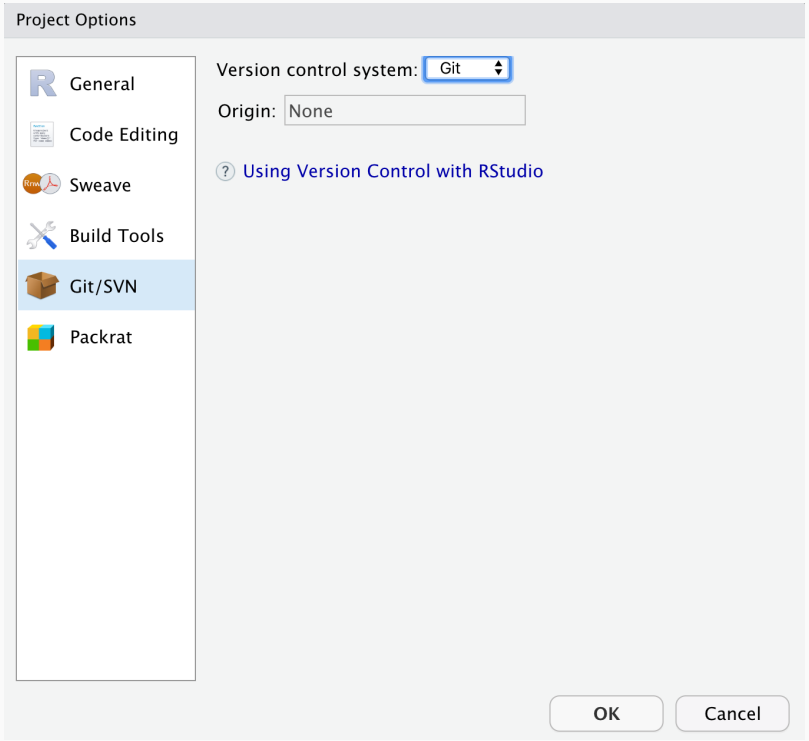
Intro2R-Git

Unzip downloaded workshop materials and open the RStudio Project file *Intro2R-Git.Rproj*

Initialise Git repo



Initialise Git repo



Enable Git with this RStudio Project

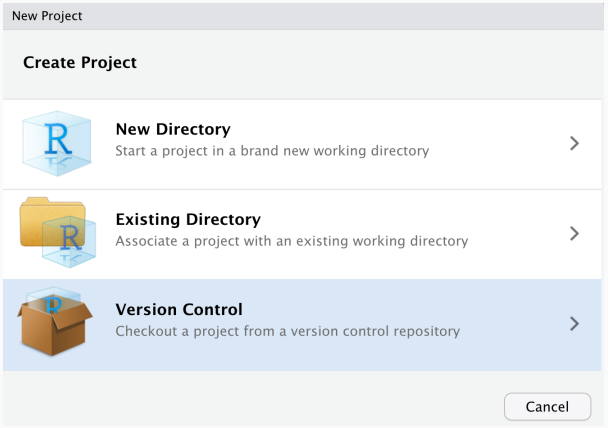
Go to *Tools* > *Project Options* > *Git/SVN* (you will be prompted to restart R).

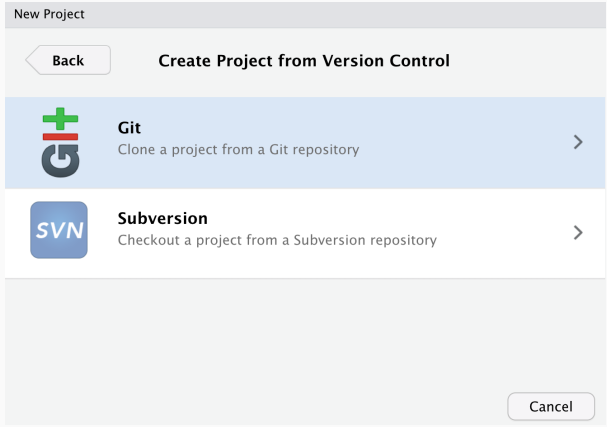
Congrats, Git is now enabled for your project folder!

Initialise Git repo

Alternative method: Clone the online repo directly from Github using RStudio

Go to *File* > *New Project*

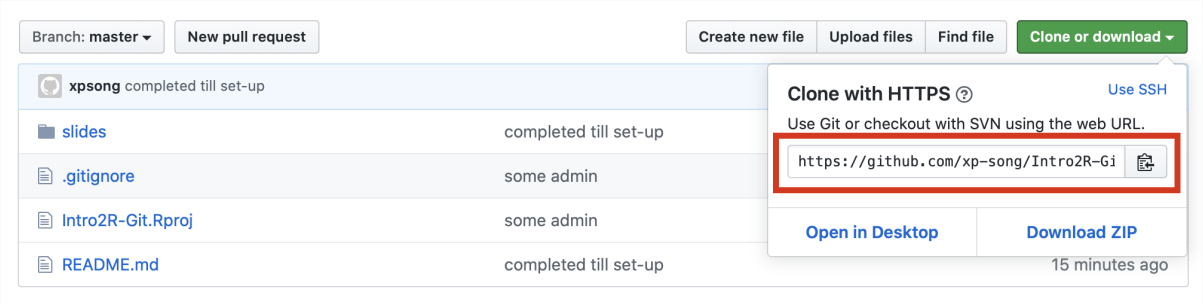


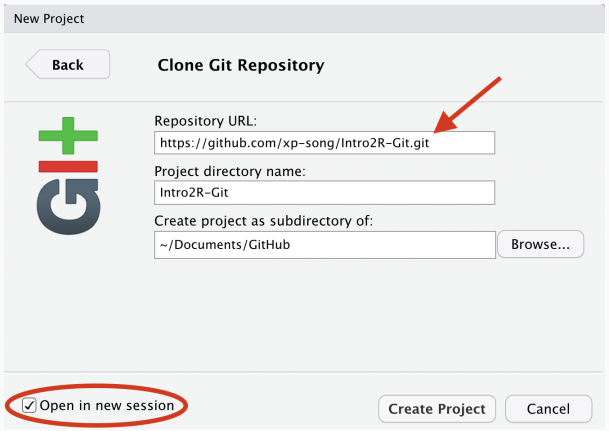


Initialise Git repo

Alternative method: Clone the online repo directly from Github using RStudio

Go to *File* > *New Project*





Initialise Git repo

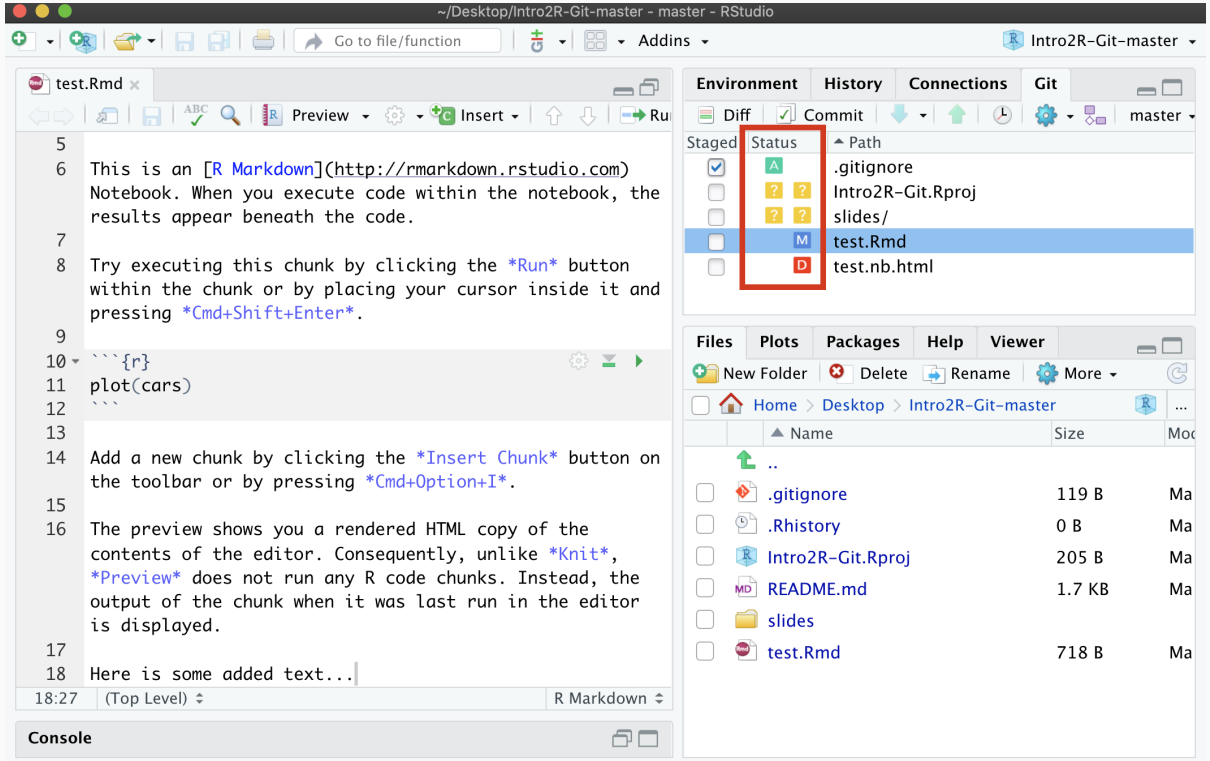
Important‼

- Both these methods enable Git version control *for the downloaded folder*.
- Version control is on your ***local*** computer, and has no links with your online Github account!
- We will learn how to connect/collaborate via Github in a later section.

Outline

Git basics

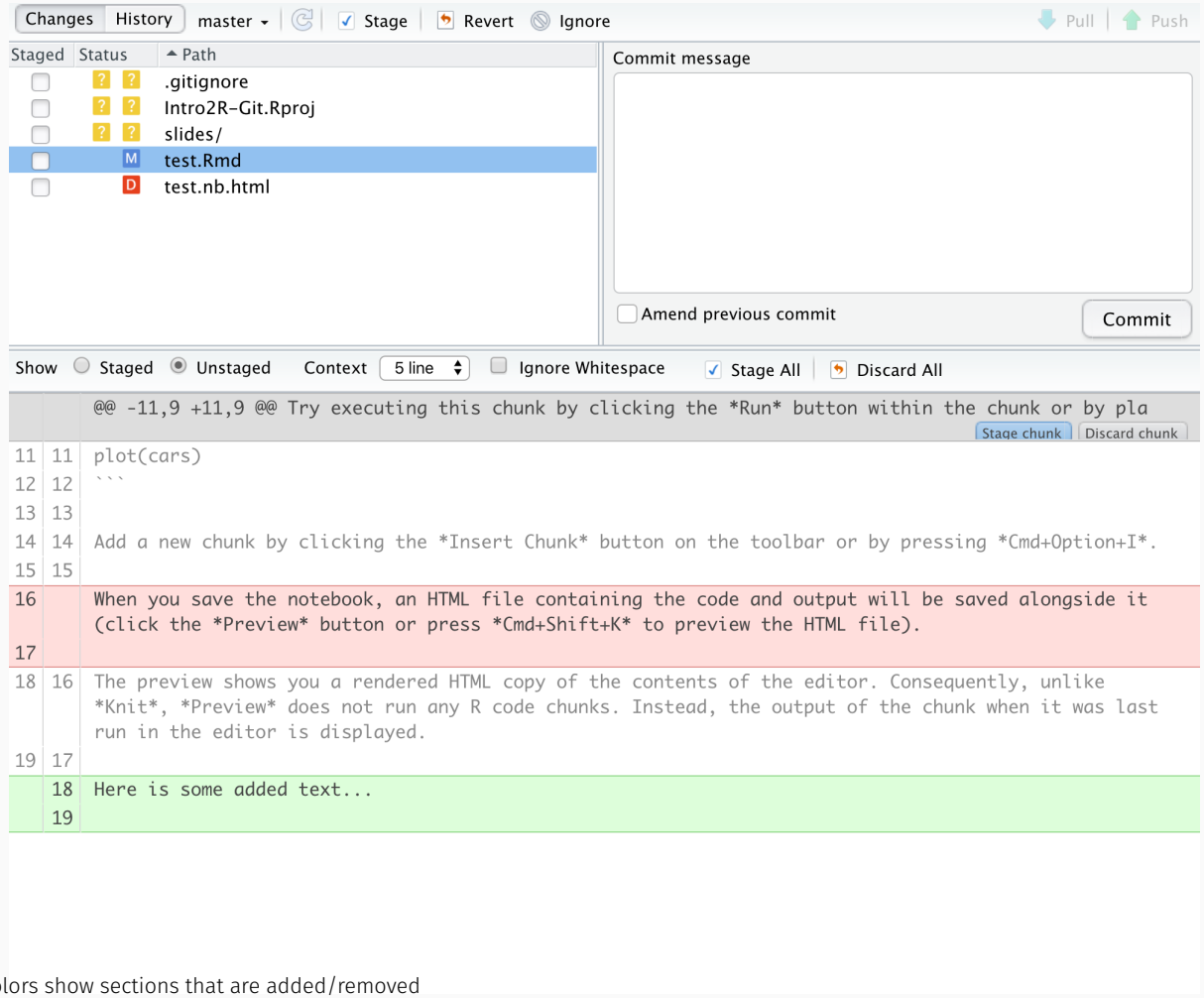
Create a new R Notebook at *File > New File > R Notebook*, and save it as *test.Rmd*



`git status`

- Yellow : New untracked file
- Green: File added
- Red: File deleted
- Blue: File modified

Git basics



`git diff`

- Background colors show sections that are added/removed
- Shows old/new line numbers
- Greyed-out sections give you context

Git basics

Commit changes

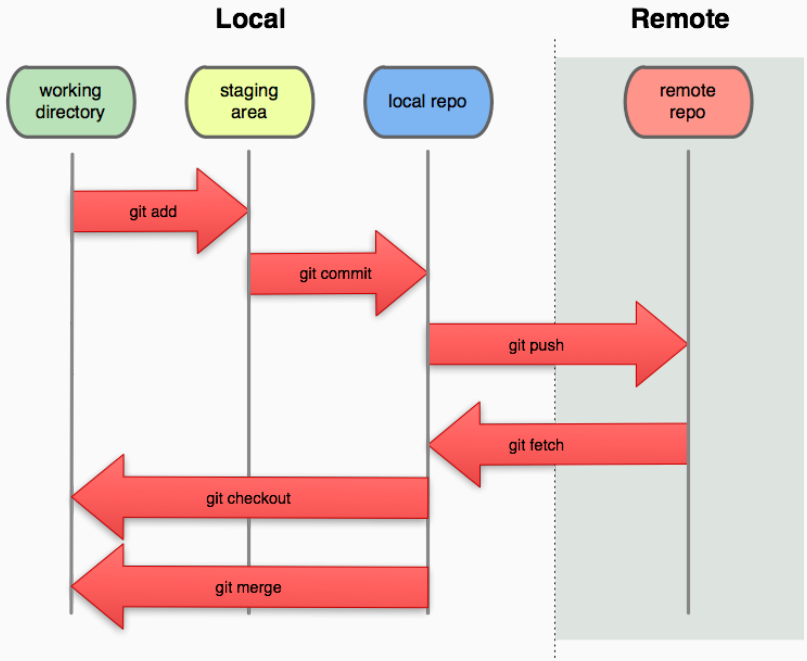
`git commit` has five components:

1. Unique ID aka SHA (secure hash algorithm)
2. Author
3. Commit message
4. Changeset that describes files that were added/modified/deleted
5. One parent (i.e. previous) commit, unless merging commits from multiple parents or if this is the initial commit

Git basics

Commit changes

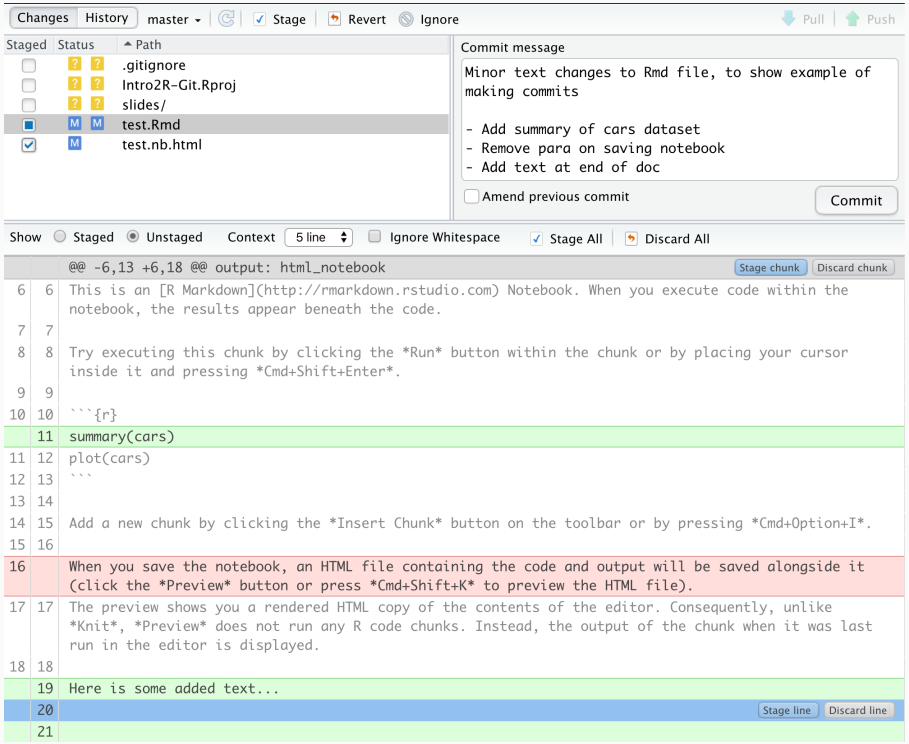
Changes must be *'staged'*, in preparation for a *'commit'*.



Git basic operations (kevintshoemaker.github.io)

Git basics

Commit changes



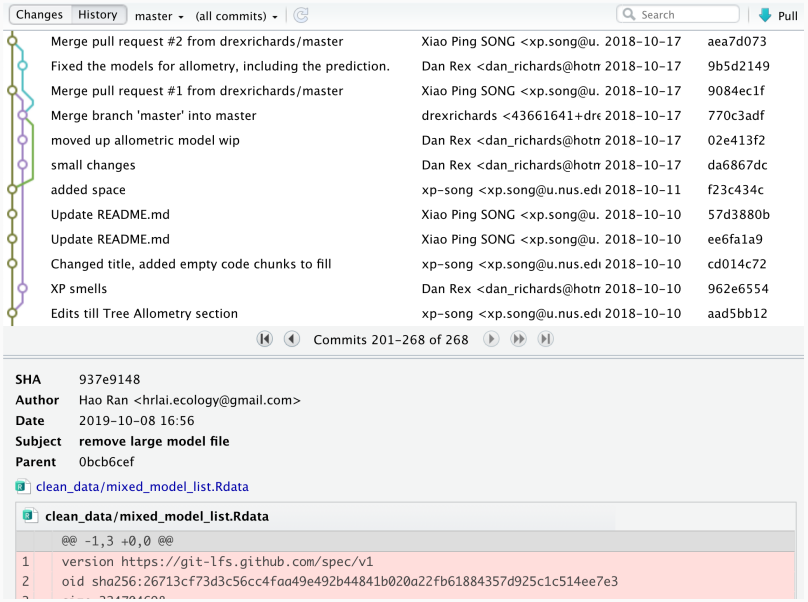
1. Save changes
2. Select files or sections of a file to stage
3. Write commit message (<50 char header; list details)

Git basics

Commit changes

Git basics

Historic changes

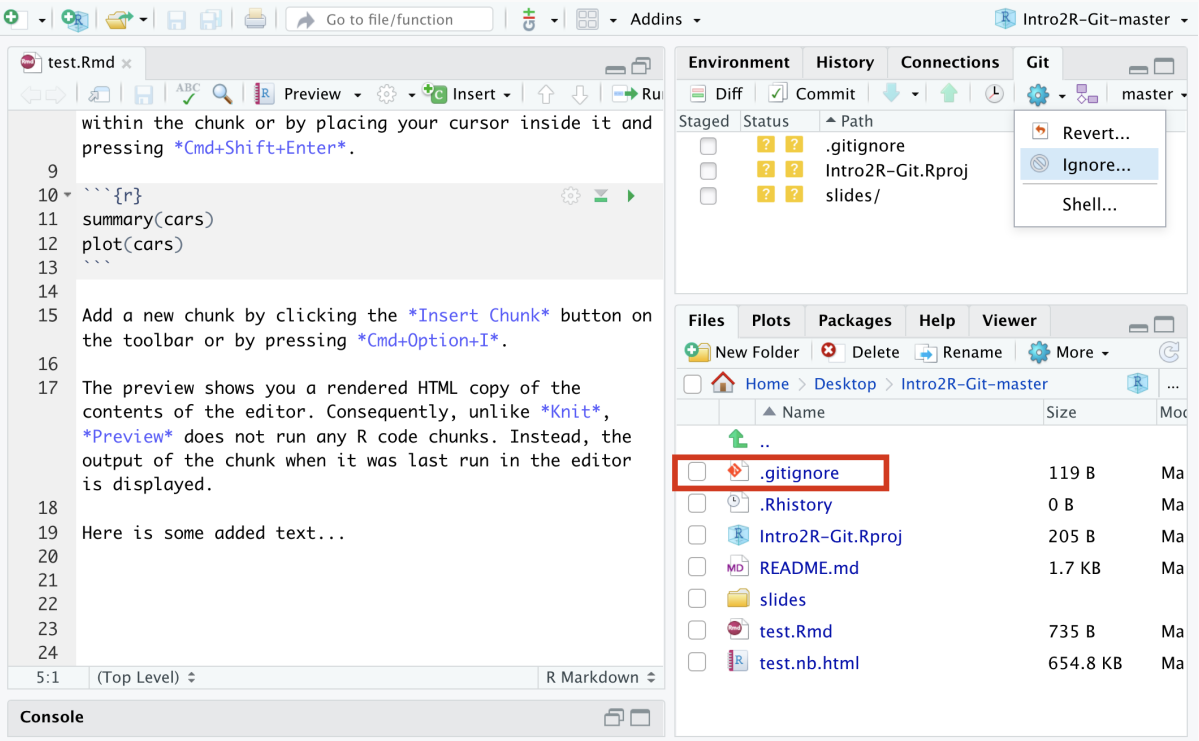


To revert to a previous commit:

1. Copy the SHA
2. Run `git checkout <SHA> <filename>` in the Shell

Git basics

Ignore files specified in `.gitignore`



Outline

Git collaborating

Link Github

1. **Create a new repo on Github**

Git collaborating

Link Github

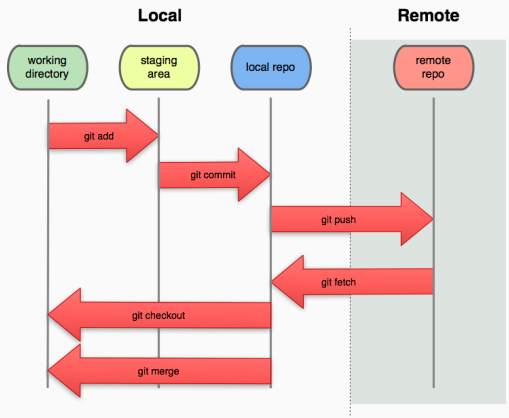
2. Create linked local repo

Create new RStudio project and clone the online repo to your computer (as shown previously)

If you've already been working on a local repo, *'push'* (i.e. publish) content to the new online repo. Run in Shell:

```
git remote add origin git@github.com:YOUR USERNAME/REPO NAME.git
```

```
git push -u origin master
```



Git basic operations (kevintshoemaker.github.io)

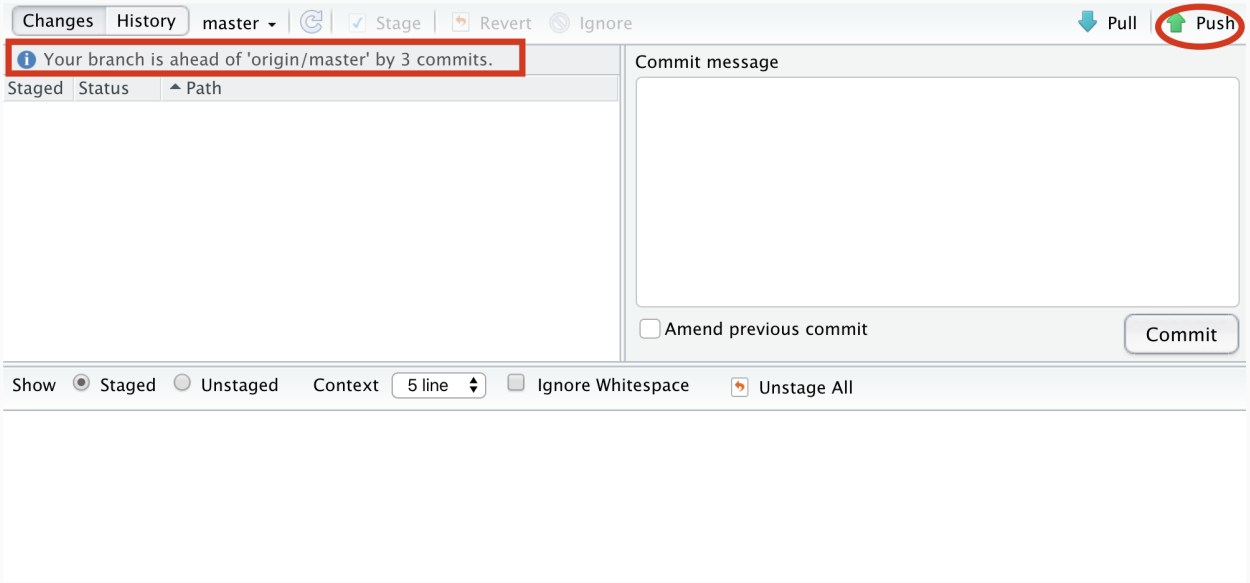
Git collaborating

Link Github

Push

Push code when you're ready for collaborators to view changes.

Each push usually includes multiple commits.



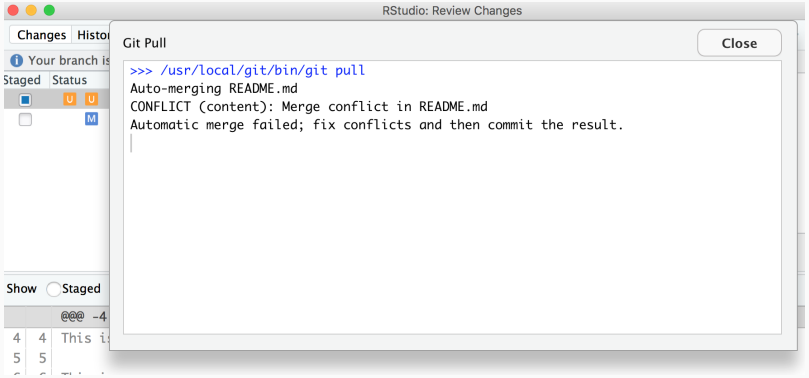
🌟 **Best Practice:** *Always 'Pull' before you 'Push'*, in case collaborators have made changes online!

Git collaborating

Link Github

Push

Conflicts 😱



Set style to `diff3`

This shows local/remote changes *with the original code*. Run in the Shell:

```
git config --global merge.conflictstyle diff3
```

If you're in the middle of a merge conflict, change the setting before re-trying the merge:

```
git merge --abort
```

```
git config --global merge.conflictstyle diff3
```

```
git pull
```

Git collaborating

Link Github

Push

Conflicts 😱

De-conflict regions with conflict marker

```
<<<<<<< HEAD

# This code section shows my local changes...

||||||| merged common ancestors

=======

# This code section shows remote changes...

>>>>>>> remote
```

The middle section shows *original* code before the split

***Delete all conflict markers* before staging/commit/pushing the new de-conflicted file!**

If the whole thing is one hot mess, abort the merge with `git merge --abort` & try again `git pull`.

Git collaborating

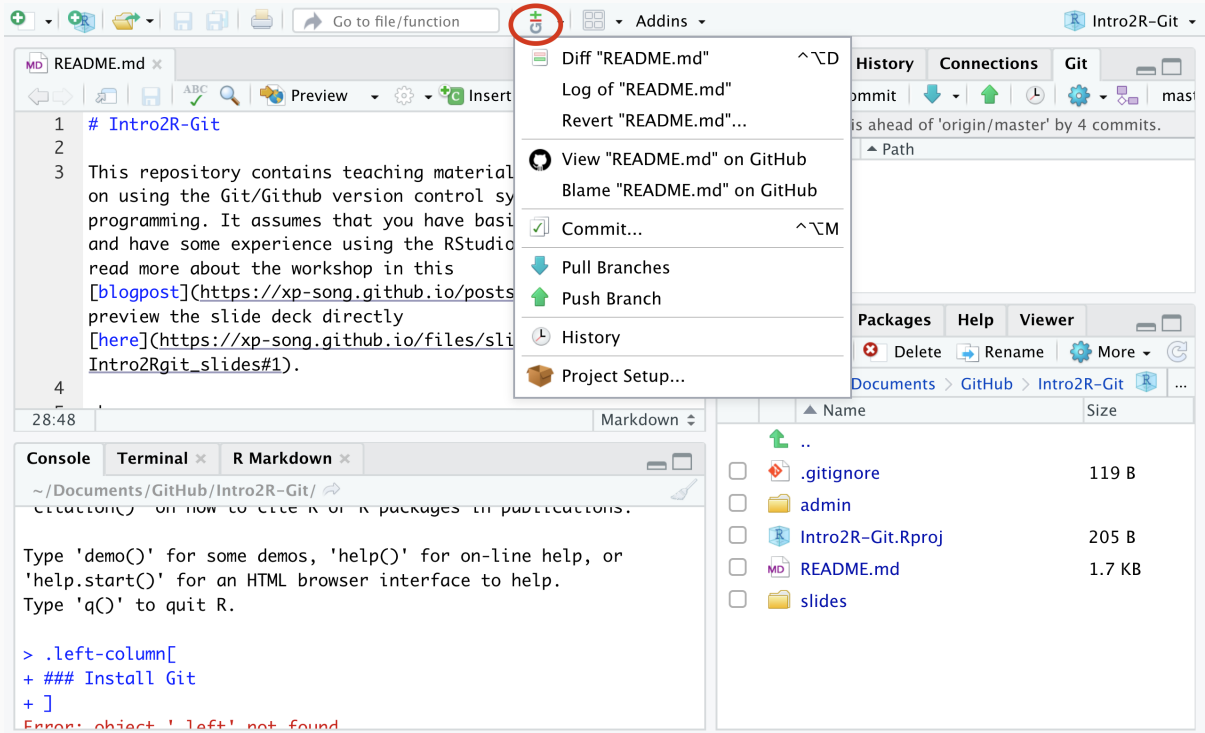
Link Github

Push

Conflicts

Investigate

Track the history of a *specific* file using the Git dropdown menu



'Blame' shows the last change made to each line of code, who made the change, and the commit the change belongs to.

Git collaborating

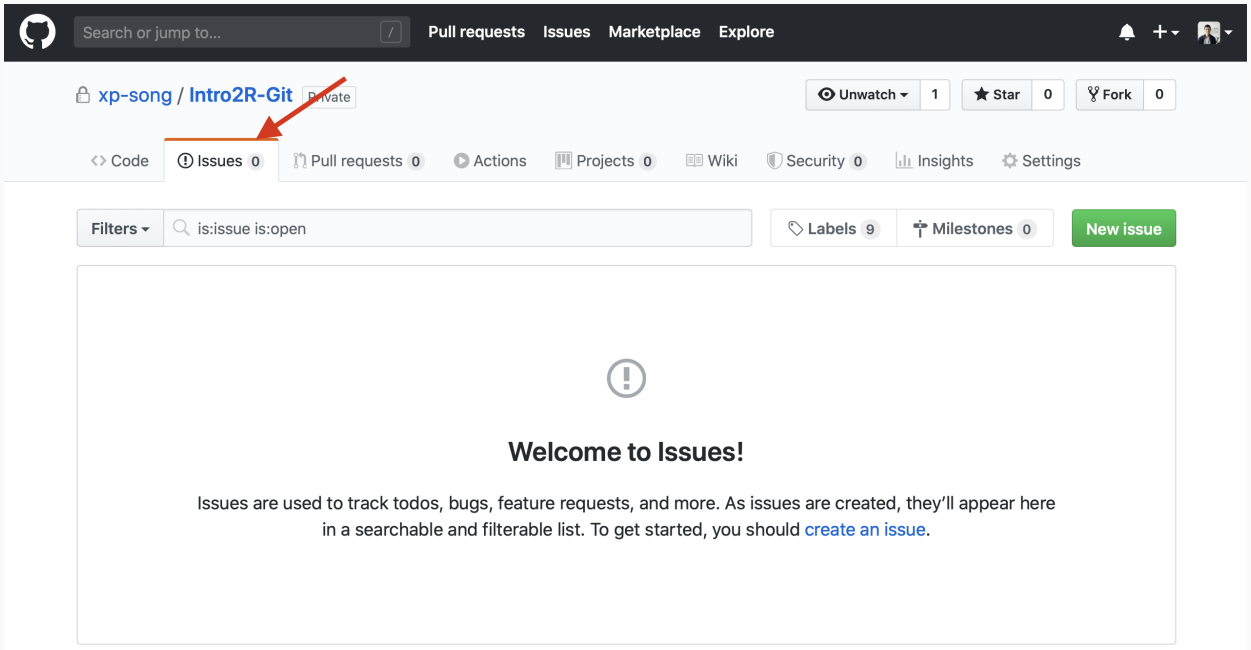
Link Github

Push

Conflicts

Investigate

Discuss & track issues on Github



Note: Automatically link to an issue by typing `#<issue number>` in your commit message. You can also close an issue by typing `Closes #<issue number>`.

Git collaborating

Link Github

Push

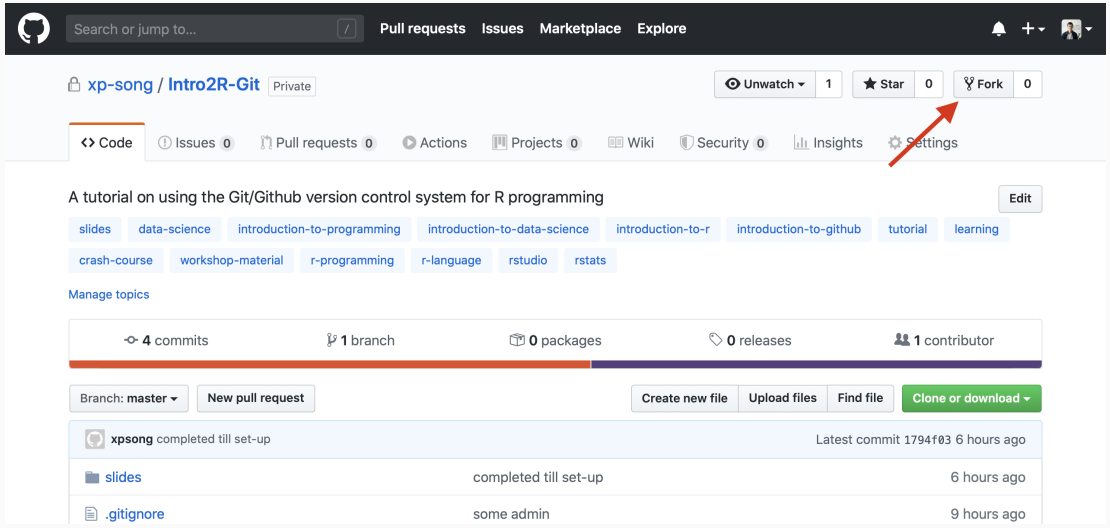
Conflicts

Investigate

Branches

What if you do not own a repo?

'Fork' (i.e. make a copy of) that repo onto your own Github account, then *clone* it to your local computer (as shown previously)



A *forked* repo is a ***static*** copy. Even if you sync between your local and forked repo, the *original* repo may undergo changes

Git collaborating

Link Github

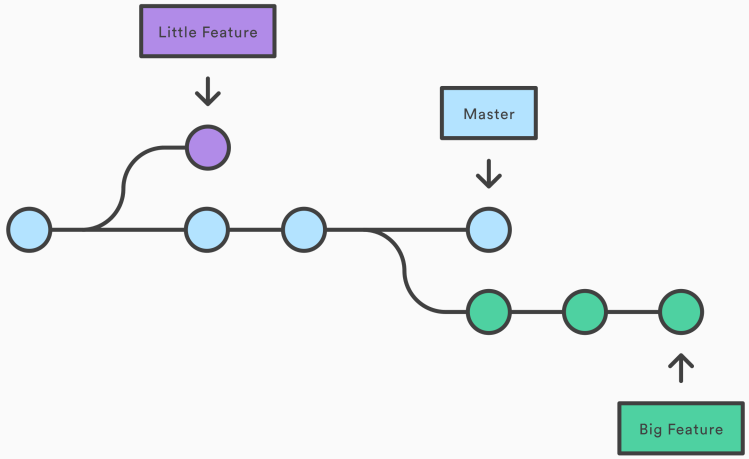
Push

Conflicts

Investigate

Branches

Branching in Git



Example showing git branches (atlassian.com)

Git collaborating

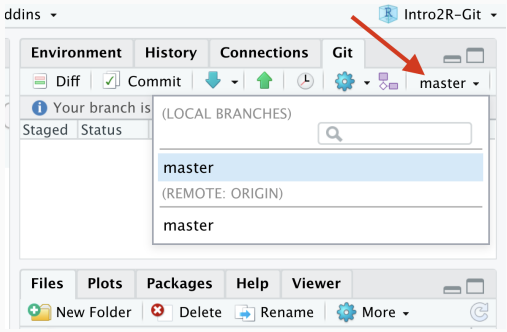
Link Github

Push

Conflicts

Investigate

Branches



Resolve conflicts regularly if you're working on a branch, by running in the Shell:

```
git merge master
```

To merge a branch back to the master:

```
git checkout master
```

```
git merge <branch-name>
```

```
git branch -d <branch-name>
```

Git collaborating

Link Github

Push

Conflicts

Investigate

Branches

If using a *forked* repo (you're not the owner), there are extra steps:

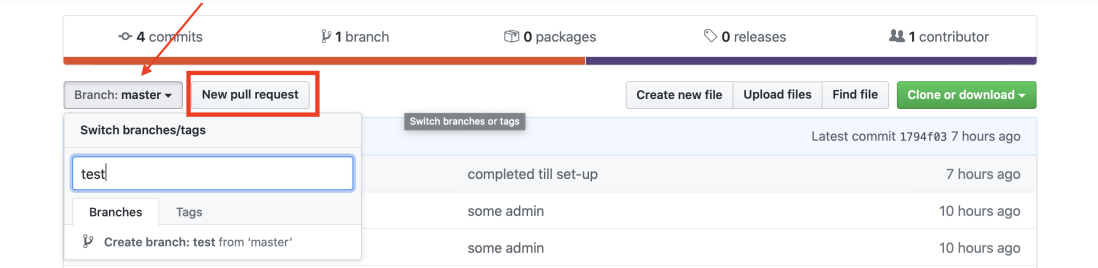
- Sync *forked* repo on Github with *original* repo: [1]

```
git remote add upstream git@github.com:<original-name>/<repo>.git
```

```
git fetch upstream
```

```
git merge upstream/master
```

- Create a branch and **pull request** to propose/discuss changes [1]



[1] If you always *submit pull requests in branches*, run `git branch -u upstream/master` to create branch on your ***local*** repo, then sync:

```
git checkout master
```

```
git pull
```

Check for remote changes if they occur while you work on your branch:

```
git checkout <my-branch>
```

```
git merge master
```

Git collaborating

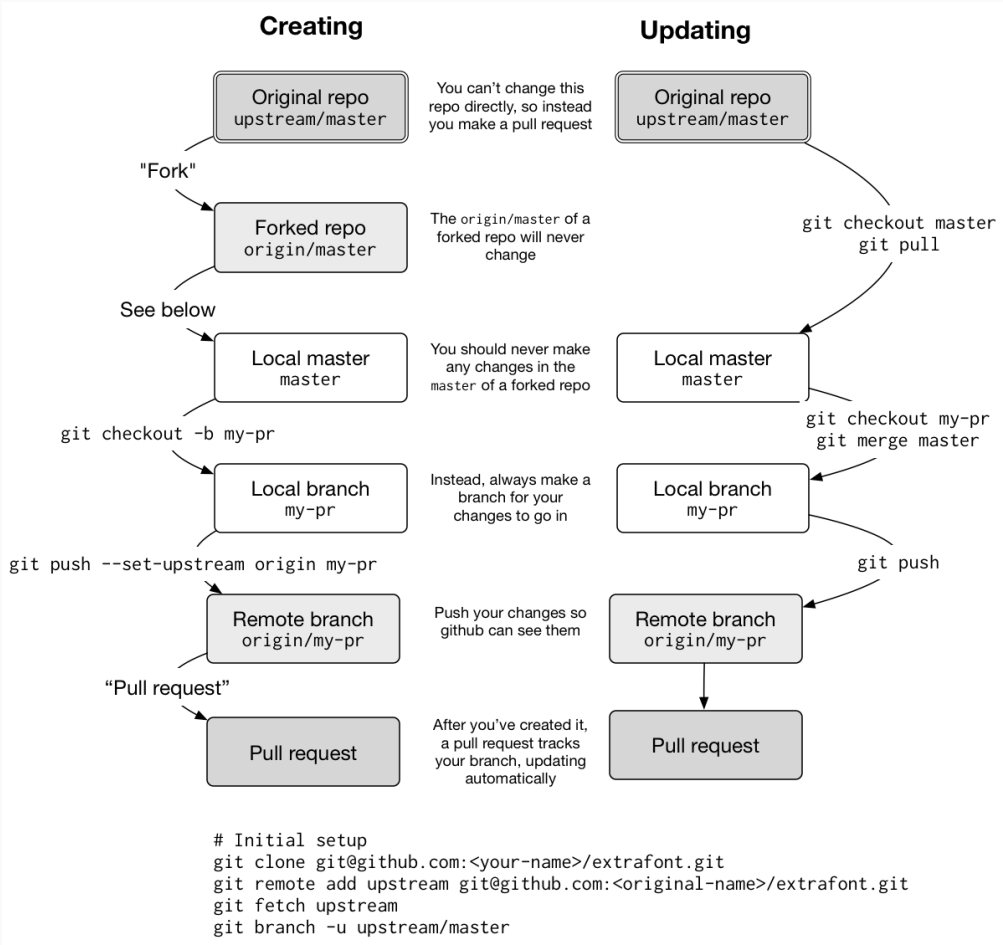
Link Github

Push

Conflicts

Investigate

Branches



Overview of pull requests (http://r-pkgs.had.co.nz/git.html)

Outline

Useful Resources

One-page guide by Hadley Wickham

Full e-book by Jenny Bryan

Infographic on fixing common problems