

# Intro2R-Git

Version control using Git and Github

---

Song, Xiao Ping

[xp.song@u.nus.edu](mailto: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?**

Git with RStudio Projects

Initial set up

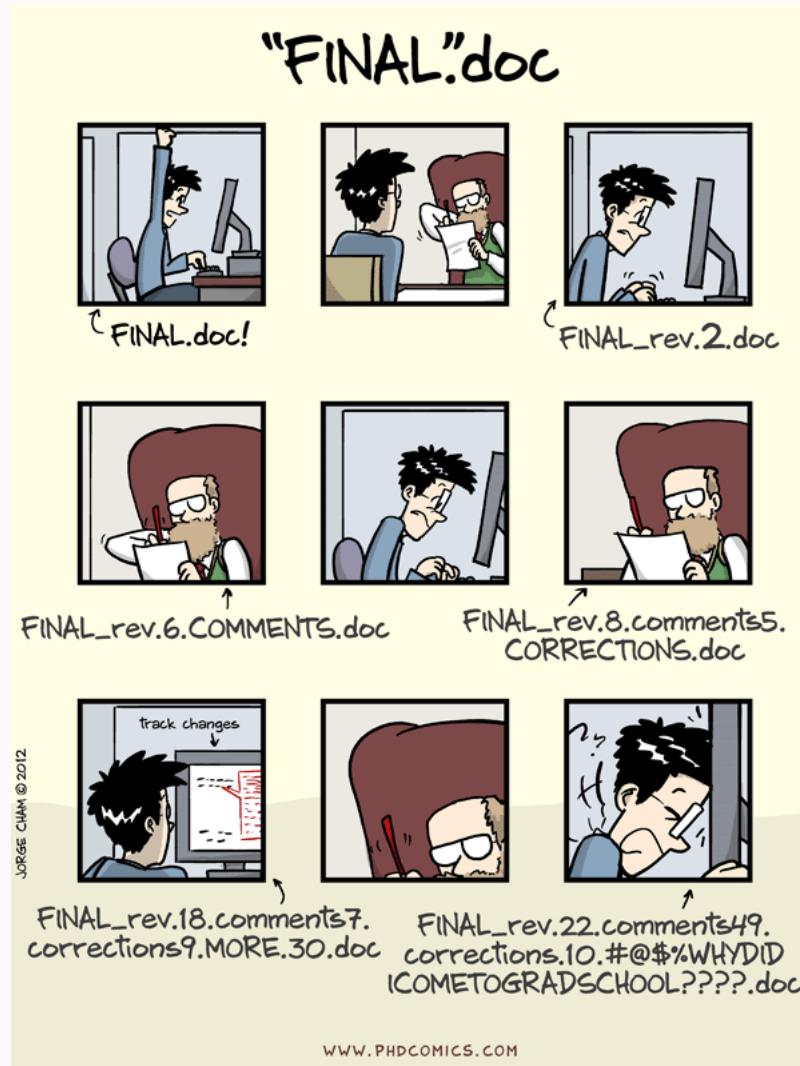
Git it going

Git basics

Git collaborating

Useful resources

# Why use version control?

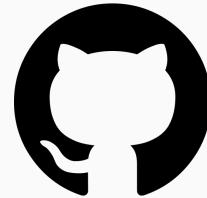


# Why use version control?



## Git

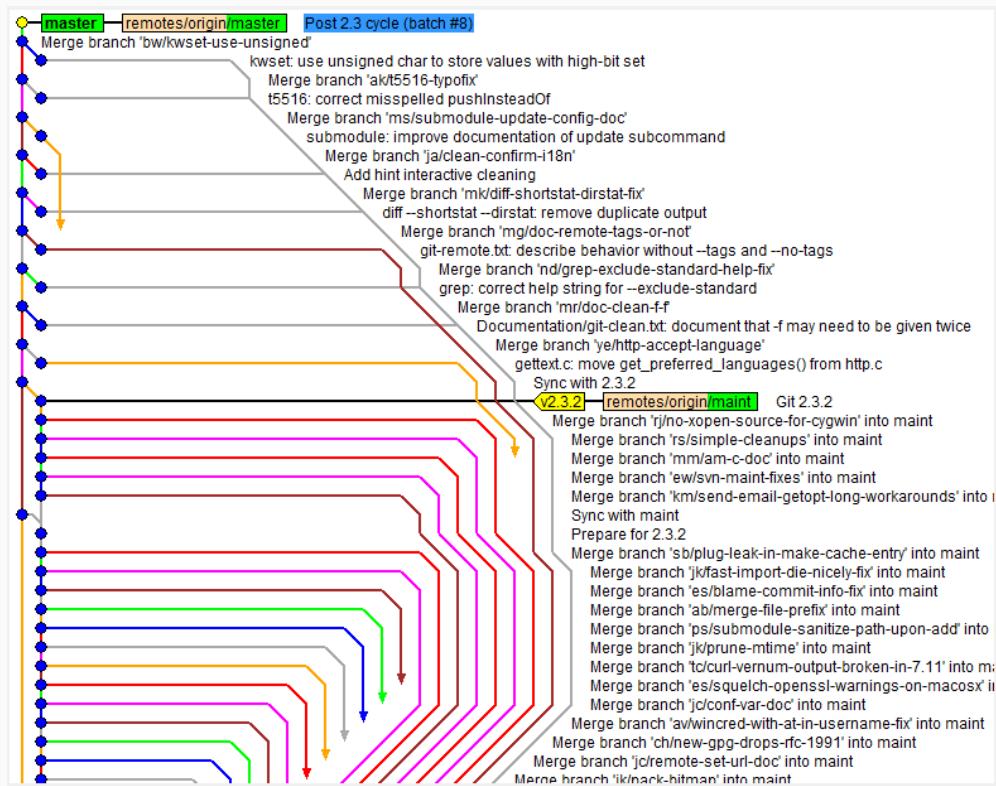
- Version control system
- *Track* changes to your code
- *Visualise* changes with GUI clients  
(e.g. RStudio, Github Desktop, etc.)
- *Share* changes & collaborate  
(e.g. Github, Gitlab, Bitbucket)
- Changes are combined automatically
- Systematic way to de-conflict changes



## Git + Github

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

# A bit about Git



Git hell ([tugberkugurlu.com](http://tugberkugurlu.com))

# Outline

Why use version control?

## **Git with RStudio Projects**

Initial set up

Git it going

Git basics

Git collaborating

Useful resources

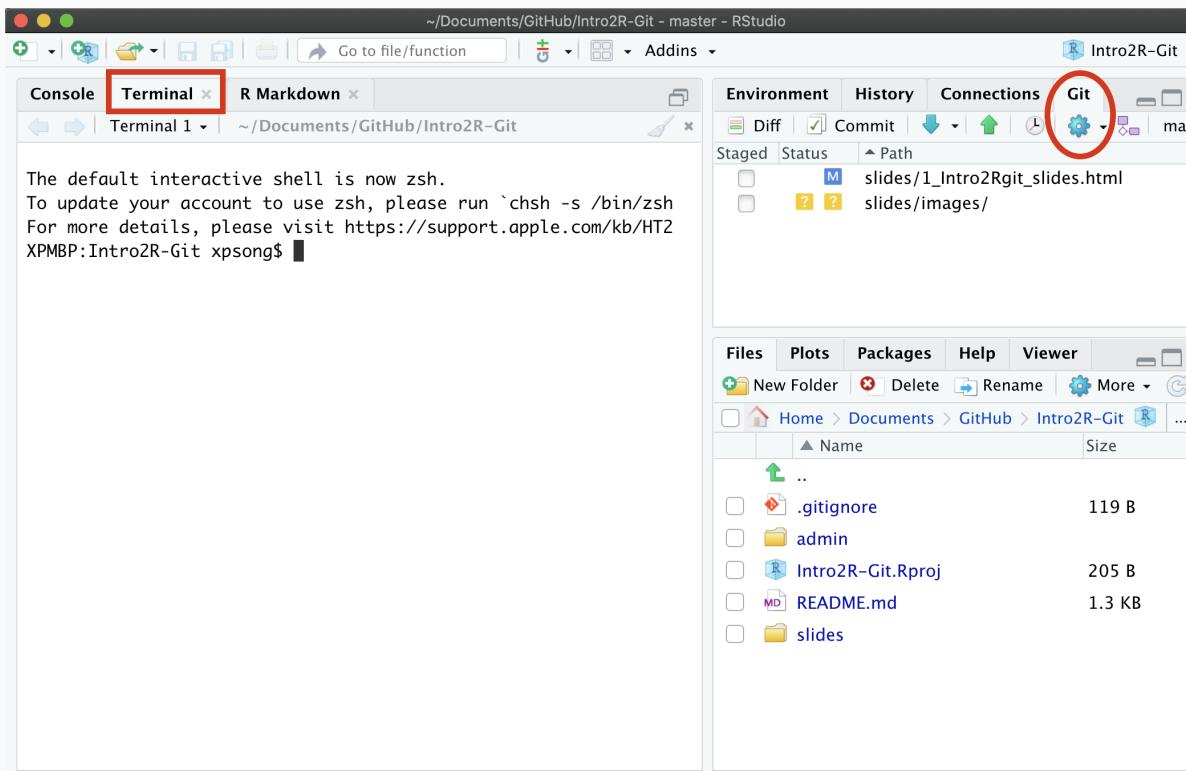
# Git with RStudio Projects

- 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

Why use version control?

Git with RStudio Projects

## **Initial set up**

Git it going

Git basics

Git collaborating

Useful resources

# 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

**Create an account at <https://github.com>**

Set-up Git

Github

# Initial set up

Install Git

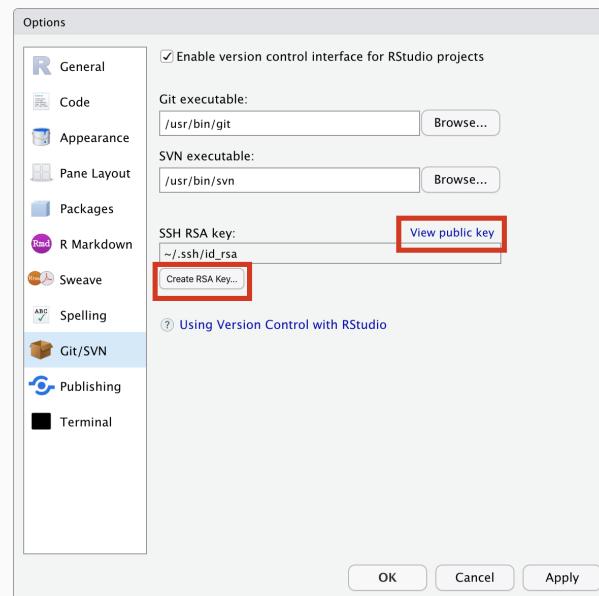
Set-up Git

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>

The screenshot shows the GitHub 'SSH keys' settings page. At the top, there is a green button labeled 'New SSH key'. Below it, a message states: 'There are no SSH keys associated with your account.' A link to 'generating SSH keys' and another to 'common SSH Problems' are provided. Below this section is another green 'New GPG key' button. It displays a similar message: 'There are no GPG keys associated with your account.' A link to 'generate a GPG key and add it to your account' is included.

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

Why use version control?

Git with RStudio Projects

Initial set up

## **Git it going**

Git basics

Git collaborating

Useful resources

# Intro2R-Git

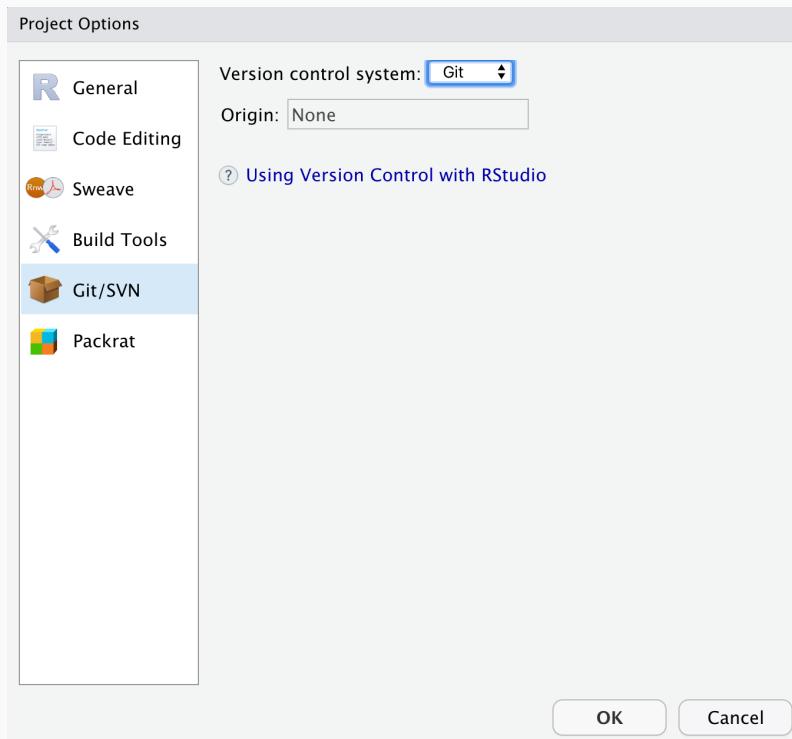
Unzip [downloaded](#) workshop materials and open the RStudio Project file  
*Intro2R-Git.Rproj*

# Initialise Git repo

The screenshot shows the RStudio interface with the following details:

- Top Bar:** Shows the path `~/Desktop/Intro2R-Git-master - RStudio` and the project name `Intro2R-Git-master`.
- Console Tab:** Displays the R startup message for version 4.0.0 (2020-04-24) "Arbor Day".
- Environment Tab:** Shows the message "Environment is empty". A red circle highlights the "List" button in the toolbar above this tab.
- Files Tab:** Shows a file tree with the following contents:
  - .. (Parent directory)
  - .gitignore (119 B)
  - Intro2R-Git.Rproj (205 B)
  - README.md (1.7 KB)
  - slides (Folder)

# Initialise Git repo



## Enable Git with this RStudio Project

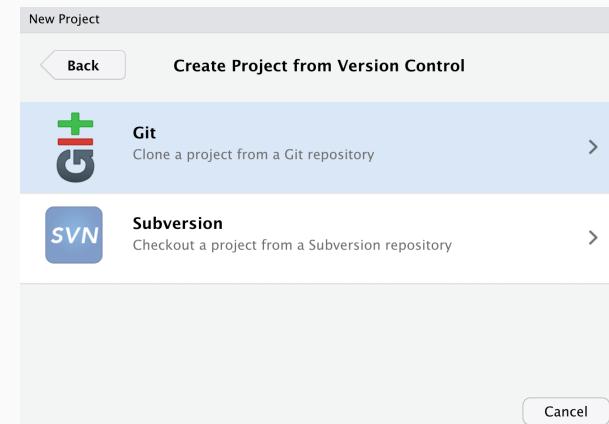
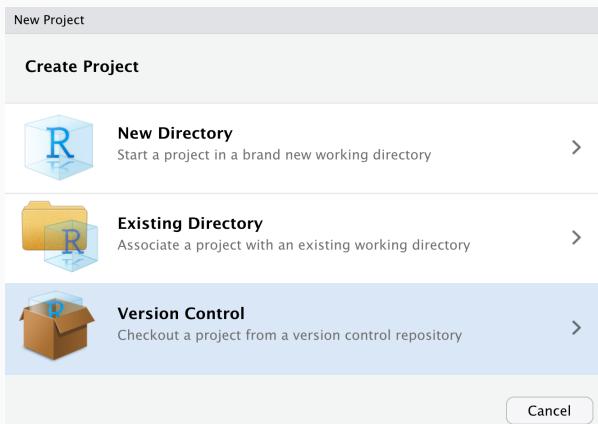
Go to *Tools > Project Options > Git/SVN* (you will be prompted to restart R).

**Congrats, Git is now enabled for your project folder!**

# Initialise Git repo

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

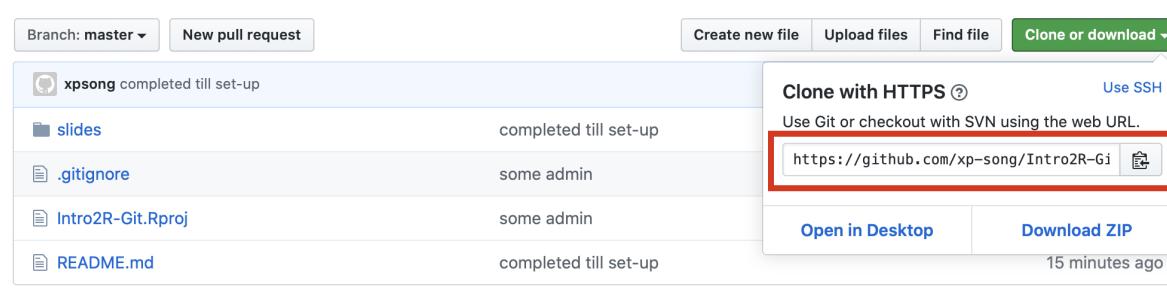
Go to *File > New Project*



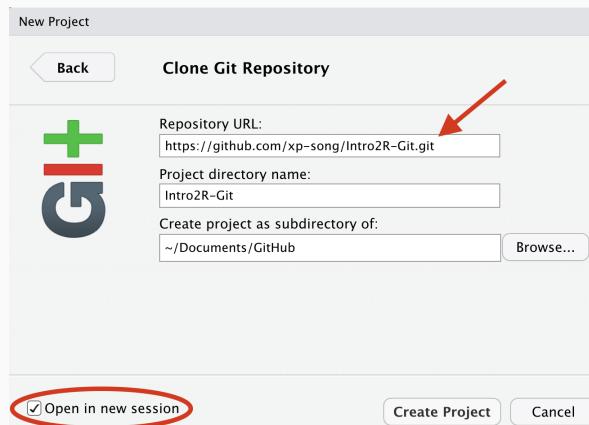
# Initialise Git repo

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

Go to *File > New Project*



A screenshot of a GitHub repository page for 'xp-song' named 'Intro2R-Git'. The page shows files like 'slides', '.gitignore', 'Intro2R-Git.Rproj', and 'README.md'. In the top right, there are buttons for 'Create new file', 'Upload files', 'Find file', 'Clone or download', 'Use SSH', and 'Clone with HTTPS'. A red box highlights the 'Clone with HTTPS' button and the resulting URL 'https://github.com/xp-song/Intro2R-Git.git'.



A screenshot of the RStudio 'New Project' dialog. It shows the 'Clone Git Repository' tab selected. The 'Repository URL:' field contains 'https://github.com/xp-song/Intro2R-Git.git', which has a red arrow pointing to it. Below it, the 'Project directory name:' field is 'Intro2R-Git'. Underneath, there's a checkbox 'Create project as subdirectory of:' with the value '/Documents/GitHub' and a 'Browse...' button. At the bottom, there's a checked checkbox 'Open in new session' with a red oval around it, and buttons for 'Create Project' and 'Cancel'.

# 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

Why use version control?

Git with RStudio Projects

Initial set up

Git it going

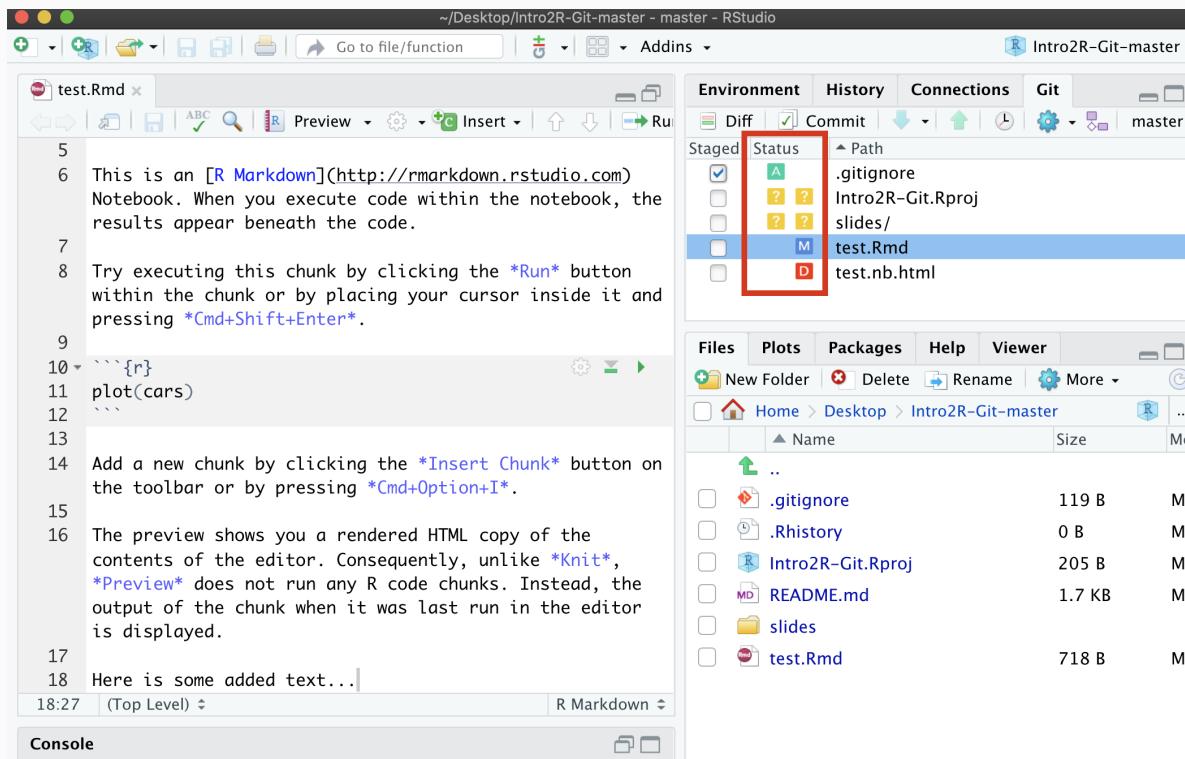
## **Git basics**

Git collaborating

Useful resources

# 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

The screenshot shows a Git commit interface with the following elements:

- Top Bar:** Changes, History, master, Stage, Revert, Ignore, Pull, Push.
- Staged Changes List:** .gitignore, Intro2R-Git.Rproj, slides/, test.Rmd (selected), test.nb.html.
- Commit Message:** A large text area for entering the commit message, with an "Amend previous commit" checkbox and a "Commit" button.
- Show Options:** Staged (radio selected), Unstaged, Context, 5 line, Ignore Whitespace, Stage All (checkbox checked), Discard All.
- Diff View:** Shows the differences between two versions of a file. Lines 11-15 show added text: "plot(cars)", "```,", "Add a new chunk by clicking the \*Insert Chunk\* button on the toolbar or by pressing \*Cmd+Option+I\*.", and "15". Line 16 shows a note about saving an HTML file. Line 18 shows a note about previewing the HTML file. Line 18 also contains the text "Here is some added text...".

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



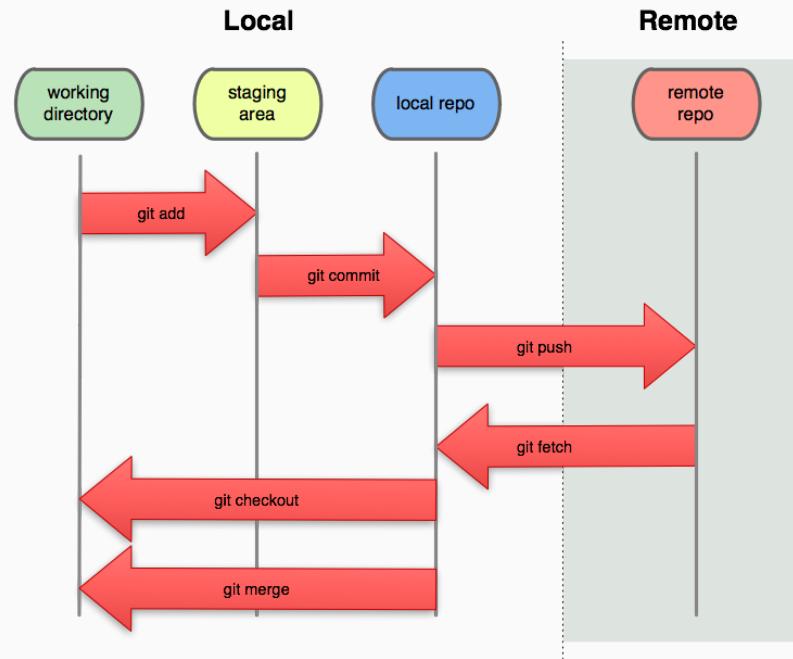
`git commit` has five components:

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](http://kevintshoemaker.github.io))

# Git basics

## Commit changes

The screenshot shows a Git commit interface with the following details:

- Staged Status:** Shows files .gitignore, Intro2R-Git.Rproj, slides/, test.Rmd, and test.nb.html. test.Rmd is selected.
- Commit message:** Minor text changes to Rmd file, to show example of making commits
  - Add summary of cars dataset
  - Remove para on saving notebook
  - Add text at end of doc
- Commit button:** A large "Commit" button is visible.
- Show/Status buttons:** Show, Staged, Unstaged, Context, 5 line, Ignore Whitespace, Stage All, Discard All.
- Diff View:** Shows the difference between lines 6 and 13 of the output. Lines 11 through 16 are highlighted in green, while lines 17 through 21 are highlighted in red. Lines 19 and 20 are highlighted in blue.
- Buttons in diff view:** Stage chunk, Discard chunk, Stage line, Discard line.

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

The screenshot shows a Git commit history interface with a timeline of commits. The commits are listed in chronological order from bottom to top. Each commit includes the author, date, subject, and SHA-1 hash. A detailed view of a specific commit is shown at the bottom.

Author	Date	Subject	SHA-1
Xiao Ping SONG <xp.song@u.nus.edu>	2018-10-17	Merge pull request #2 from drexrichards/master	aead7d073
Dan Rex <dan_richards@hotn>	2018-10-17	Fixed the models for allometry, including the prediction.	9b5d2149
Xiao Ping SONG <xp.song@u.nus.edu>	2018-10-17	Merge pull request #1 from drexrichards/master	9084ec1f
drexrichards <43661641+dre>	2018-10-17	Merge branch 'master' into master	770c3adf
Dan Rex <dan_richards@hotn>	2018-10-17	moved up allometric model wip	02e413f2
Dan Rex <dan_richards@hotn>	2018-10-17	small changes	da6867dc
xp-song <xp.song@u.nus.edu>	2018-10-11	added space	f23c434c
Xiao Ping SONG <xp.song@u.nus.edu>	2018-10-10	Update README.md	57d3880b
Xiao Ping SONG <xp.song@u.nus.edu>	2018-10-10	Update README.md	ee6fa1a9
xp-song <xp.song@u.nus.edu>	2018-10-10	Changed title, added empty code chunks to fill	cd014c72
Dan Rex <dan_richards@hotn>	2018-10-10	XP smells	962e6554
xp-song <xp.song@u.nus.edu>	2018-10-10	Edits till Tree Allometry section	aad5bb12

Commits 201-268 of 268

**Commit Details:**

SHA: 937e9148  
Author: Hao Ran <hrlai.ecology@gmail.com>  
Date: 2019-10-08 16:56  
Subject: remove large model file  
Parent: 0bcb6cef  
File: clean\_data/mixed\_model\_list.Rdata  
Content:  
@@ -1,3 +0,0 @@  
1 version https://git-lfs.github.com/spec/v1  
2 oid sha256:26713cf73d3c56cc4faa49e492b44841b020a22fb61884357d925c1c514ee7e3  
3

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`

The screenshot shows the RStudio interface with the following details:

- Editor Area:** Displays an R Markdown file named `test.Rmd`. The code includes chunks for `summary(cars)` and `plot(cars)`, and a note about adding new chunks via the toolbar or keyboard.
- Git Tab:** Shows the `master` branch. The `.gitignore` file is listed under the `Staged` section, indicating it has been added to the repository.
- File List:** Shows the contents of the `Intro2R-Git-master` directory. A red box highlights the `.gitignore` file in the list, which has a size of 119 B.

Name	Size	Last Modified
<code>.gitignore</code>	119 B	Mar 20 2024
<code>.Rhistory</code>	0 B	Mar 20 2024
<code>Intro2R-Git.Rproj</code>	205 B	Mar 20 2024
<code>README.md</code>	1.7 KB	Mar 20 2024
<code>slides</code>		
<code>test.Rmd</code>	735 B	Mar 20 2024
<code>test.nb.html</code>	654.8 KB	Mar 20 2024

# Outline

Why use version control?

Git with RStudio Projects

Initial set up

Git it going

Git basics

**Git collaborating**

Useful resources

# Git collaborating

Link Github

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

# Git collaborating

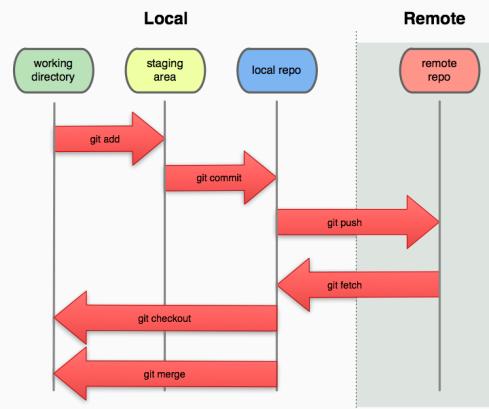
## Link Github

### 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
```



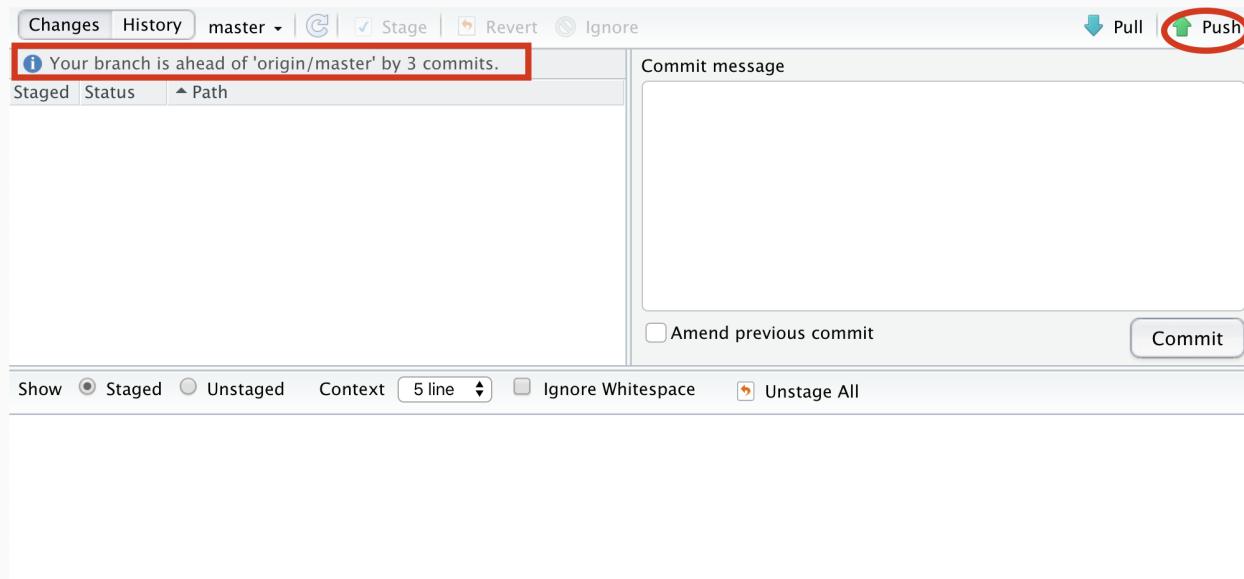
# Git collaborating

## Link Github

## Push

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

Each push usually includes multiple commits.



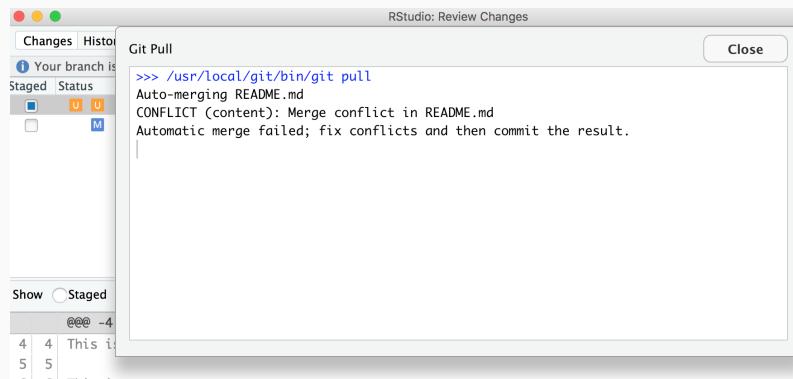
🌟 **Best Practice:** Always 'Pull' before you 'Push', in case collaborators have made changes online!

# 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

## De-conflict regions with conflict marker

Push

Conflicts 😱

```
<<<<< 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

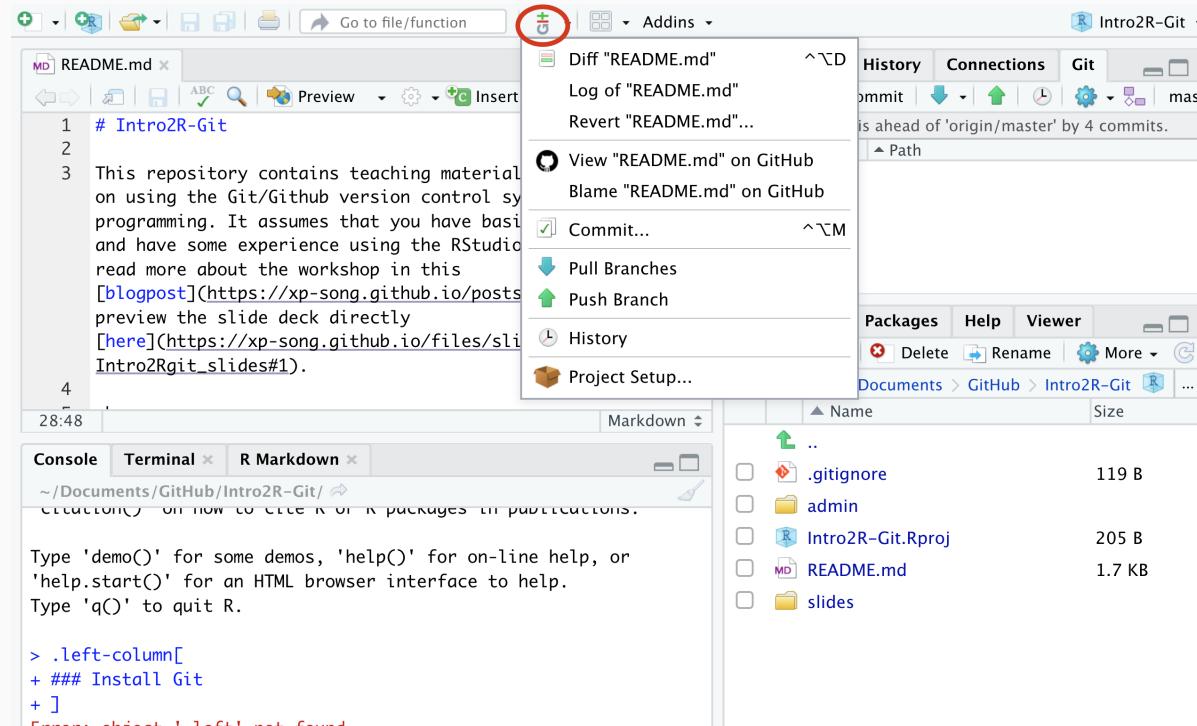
Link Github

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

## Discuss & track issues on Github

Push

Conflicts

Investigate

The screenshot shows the GitHub interface for the repository `xp-song / Intro2R-Git`. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. On the right, there are buttons for Unwatch (1), Star (0), Fork (0), and Settings. Below the navigation, there are tabs for Code, Issues (0, highlighted with a red arrow), Pull requests (0), Actions, Projects (0), Wiki, Security (0), Insights, and Settings. A search bar at the top allows filtering by 'is:issue is:open'. The main content area displays a message: 'Welcome to Issues! Issues are used to track todos, bugs, feature requests, and more. As issues are created, they'll appear here in a searchable and filterable list. To get started, you should [create an issue](#)'. At the bottom, a note explains how to link to an issue in commit messages.

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

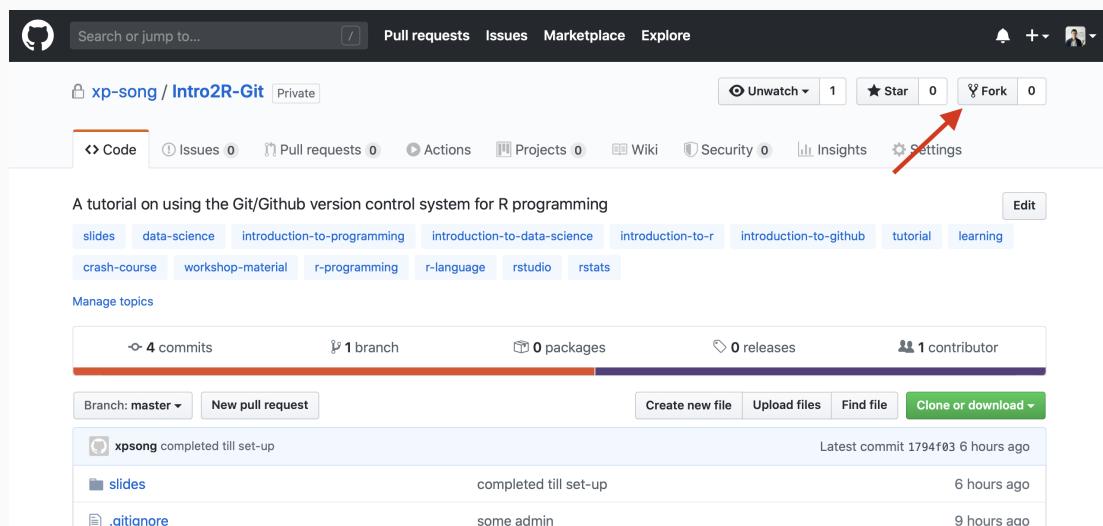
Conflicts

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 screenshot of a GitHub repository page for 'xp-song / Intro2R-Git'. The page shows basic repository statistics: 4 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. A red arrow points to the 'Fork' button in the top right corner of the header bar.

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

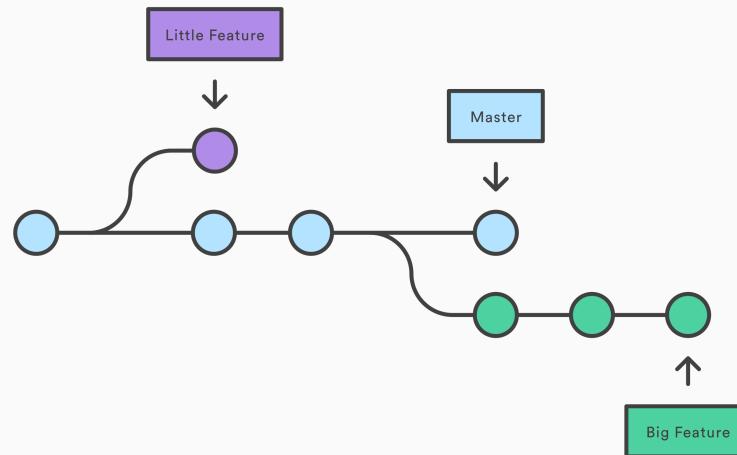
## Branching in Git

Push

Conflicts

Investigate

Branches



Example showing git branches (atlassian.com)

# Git collaborating

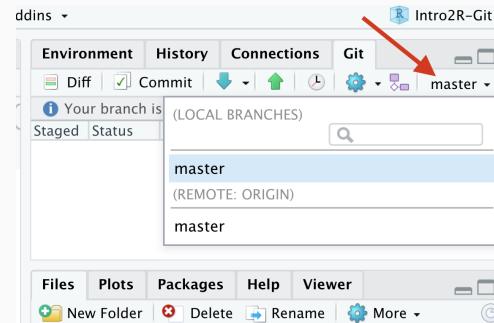
Link Github

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:

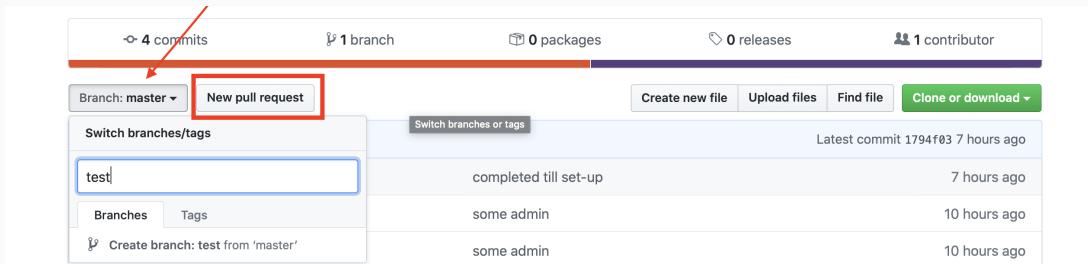
- Sync *forked* repo on Github with *original* repo:<sup>1</sup>

```
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<sup>1</sup>



<sup>1</sup> 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

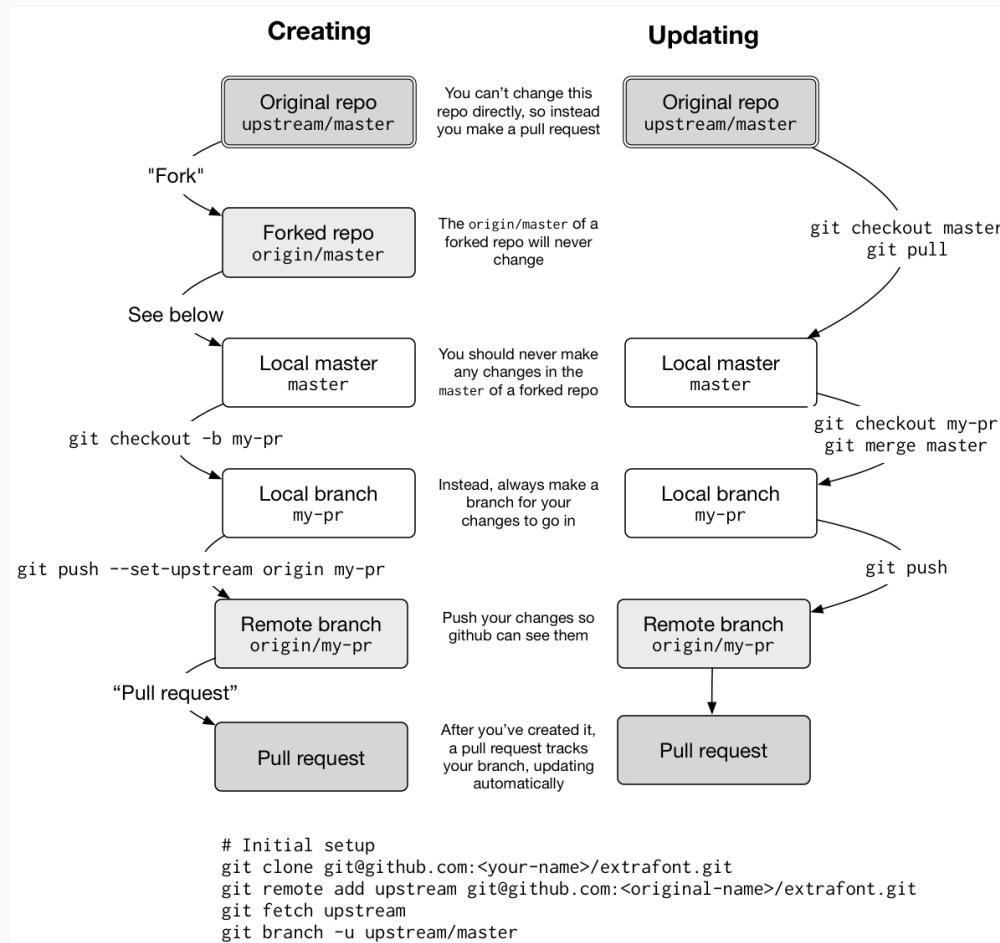
Link Github

Push

Conflicts

Investigate

Branches



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

# Outline

Why use version control?

Git with RStudio Projects

Initial set up

Git it going

Git basics

Git collaborating

**Useful resources**

# Useful Resources

[\*\*One-page guide\*\*](#) by Hadley Wickham

[\*\*Full e-book\*\*](#) by Jenny Bryan

[\*\*Infographic on fixing common problems\*\*](#)