

# R With Git and Github

Aris Paschalidis

September 02, 2021

1 Setup

2 Git Basics

3 Exercise: Use Git

4 Working With Others

5 Exercise: Create a Pull Request

6 Geography Fun Facts!

# Section 1

## Setup

# Install Git

Git allows us to track changes to our documents (i.e. Git is version control).

- Check if git is already installed: `which git`
- Install git for Windows: <https://gitforwindows.org/>
- Install git for Mac: `brew install git`
- Install git for Linux:
  - Ubuntu or Debian: `sudo apt-get install git`
  - Fedora or RedHat: `sudo yum install git`

# Configure Git

- Introduce yourself to git

```
useThis::use_git_config(user.name = "first last",  
                        user.email = "github email")
```

- Ensure setup was successful

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

# Managing Git(Hub) Credentials<sup>1</sup>

- Adopt HTTPS
- Secure your account with 2FA
- Create a PAT: `usethis::create_github_token()`
- Store PAT into the Git credential store: `gitcreds::gitcreds_set()`

---

<sup>1</sup>Following the usethis guide

## Section 2

### Git Basics

# Stage & Commit

- We can create and edit files within our project
- We can then **stage** and **commit** these changes

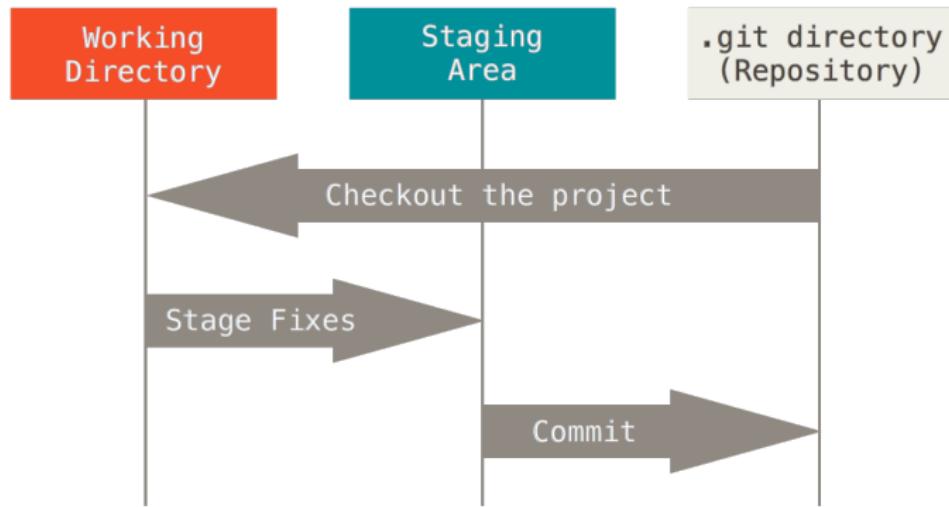


Figure 1: Git Workflow

# Diff

- We can look at the set of *differences* between files to see what has changed

The screenshot shows the RStudio interface for reviewing changes. The title bar says "RStudio: Review Changes". The top menu has tabs for "Changes" (selected), "History", and "master". Below the tabs are buttons for "Stage", "Revert", "Ignore", "Pull", and "Push". A message box says "Your branch is ahead of 'origin/master'". The left pane shows a tree view of files: README.md (modified), images/ (modified), r-with-git.Rmd (modified), and r-with-git.pdf (modified). The right pane is a "Commit message" field with a "Commit" button. Below the message field is an "Amend previous commit" checkbox. The bottom pane shows a diff view with lines 39-47. Lines 39-43 are greyed out. Line 44 is highlighted in red and contains "- Ensure set-up was successful". Line 45 is greyed out. Line 46 is highlighted in green and contains "- Ensure setup was successful". Lines 47-48 are greyed out.

```

@@ -39,11 +39,11 @@ which git
39 39 ``{r, eval = F}
40 40 usethis::use_git_config(user.name = "first last",
41 41             user.email = "github email")
42 42 ``
43 43
44 - Ensure set-up was successful
44 - Ensure setup was successful
45 45 ``{bash}
46 46 git config --global --list
47 47 ``

```

Figure 2: Example Diff

# Commit Best Practices

Each commit should be:

- *Minimal*: A commit should only contain changes related to a single problem or feature
- *Complete*: A commit should add the functionality it claims to add
- *Concise, yet evocative*: At a glance, you should be able to understand what a commit does, but you should include enough detail so you can remember what was done

# Time Travel

- Can look at old commits and access old code
- Can use your git client or Github
- Can revert back to a previous commit

# Pull & Push Changes

- **Pulling** changes from Github allows us to access stored changes
- **Pushing** changes stores changes on Github

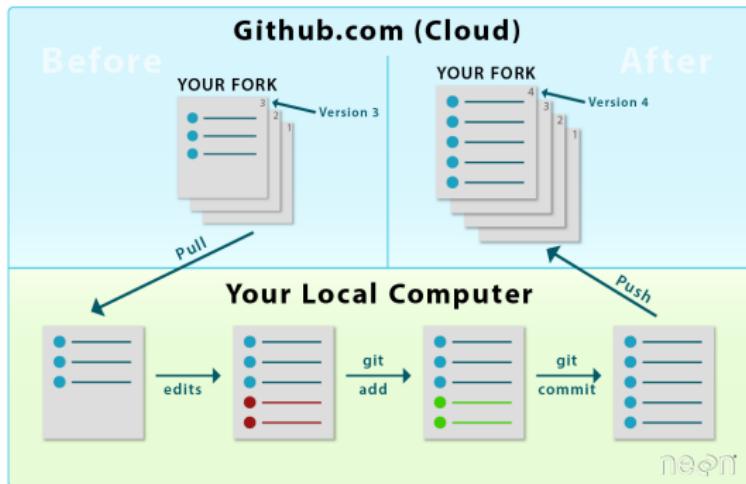


Figure 3: Accessing the Cloud

## Section 3

Exercise: Use Git

# Make a RStudio Project and Connect to Github

- Create a project: `usethis::create_project(path)`
- Use git: `usethis::use_git()`
- Connect a Github repo: `usethis::use_github()`

# Commit & Push

- Make a .Rmd file and edit it
- Commit your change
- Make another change and commit it
- Push your changes

## Section 4

Working With Others

# Overview

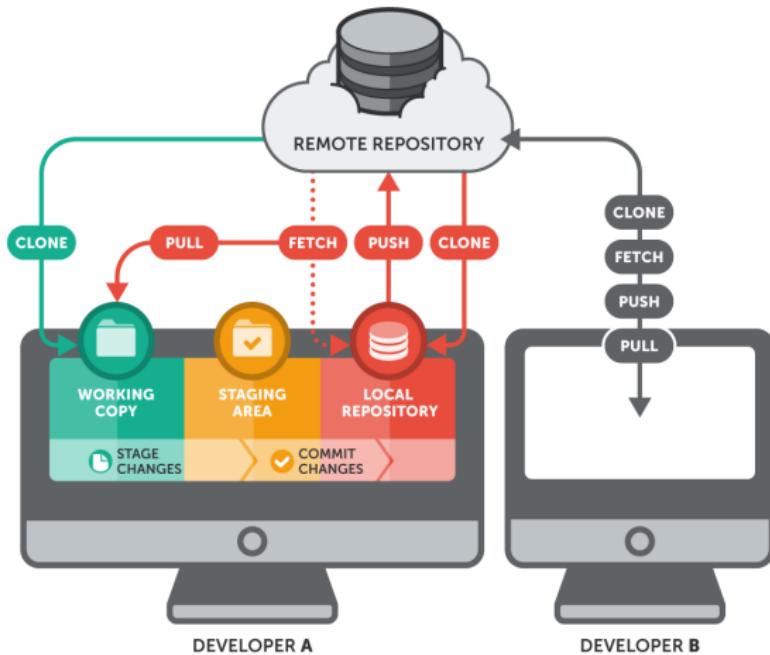


Figure 4: Multiple Developers

# Fork & Clone

- Developer A makes a repo
- Developer B wants to make some changes
- Developer B **forks and clones** the repo:

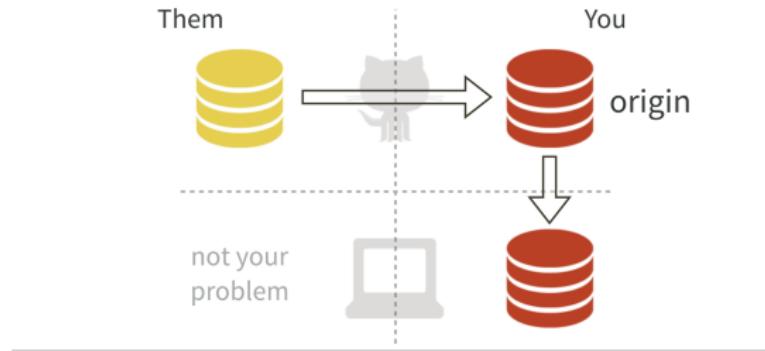


Figure 5: Fork and Clone

# Branches

- Developer B makes a new **branch** to develop a new feature
- Allows Developer A to continue working on the main stream of development
- Developers A and B can work in parallel



Figure 6: Branches

# Pull Requests

- Developer B initiates a **pull request (PR)** asking Developer A to incorporate changes
- Developer A reviews the **PR**
- Developer A makes or requests changes
- Developer A merges the **PR**



Figure 7: Pull Request Workflow

# In Practice

Contributor:

- Fork and clone: `usethis::create_from_github("OWNER/REPO")`
- Create a branch: `usethis::pr_init(branch)`
- Push changes: `usethis::pr_push()`

Developer:

- Download a PR: `usethis::pr_fetch()`
- Push changes to branch: `usethis::pr_push()`
- Merge the PR on Github
- Delete the branch: `usethis::pr_finish()`

## Section 5

Exercise: Create a Pull Request

# Exercise: Create a Pull Request

- Fork and clone this repo:

```
usethis::create_from_github("arisp99/r-with-git")
```

- Create a branch: `usethis::pr_init("USERNAME-fun-fact")`
- Add a geography fun fact and picture to the presentation!
- Push your changes: `usethis::pr_push()`

## Section 6

Geography Fun Facts!

# Greece

- The world's oldest weather station is at the base of the Acropolis



Figure 8: The Acropolis

# Liberia

- Liberia does not use the metric system

