



# Gitting there

William E Oswald, PhD

25 February 2022

# Motivation

- Renewed call for greater transparency in research
- Need for reproducibility at all stages, including data cleaning and analysis
- Inefficiencies in analysis workflows
  - Long cleaning and assembly code developed over long periods of time with multiple contributors
  - No clear record of what was done, when, and why
  - No easy mechanism for collaborating and sharing materials
- Clear benefits of open research from recent COVID 19 studies



# Learning outcomes

- What is Git?
- How does Git work?
- What is Github?
- How is all this useful?
- What can I do with this?
- How does Git work with R (and Stata)?



# What is Git?

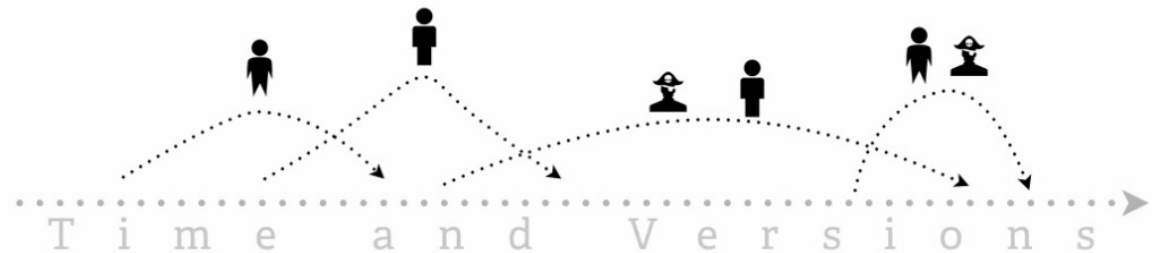
- Git created by Linus Torvalds in 2005
- Free and open-source version-control software for tracking changes in source code
- Designed for coordinated work among programmers
- Can be used to track changes in ANY set of text files
- Goals include speed, data integrity, and support for distributed, non-linear workflows



# How does Git work?

- Track changes
  - When, why, and what
- Commit
  - A record of file changes since last commit (only storing one item versus two versions of file)
  - Allows you to return to the state of a project at any point
  - Include brief description of change
- Specifically designed to facilitate collaboration

## Collaborative History Tracking



# What is GitHub?

- GitHub is not Git
- GitHub is the most popular code hosting platform for version control and collaboration
- Lets you and others work together on projects from anywhere
- Includes project board for assigning tasks and to-dos (i.e. Trello)



# How is all this useful?

- Not just for software designers and R users
- “Code” = any text-based programming script
- Stata .do files are okay too!
- Lots of tutorials available online
  - A bit daunting, jargon-heavy, and R-focussed
- Shout-out to earlier R Users presentations on Git (<https://blogs.lishtm.ac.uk/rusers/resources/>):
  - Roz Eggo, Hunter Blanks, Adam Kucharski, *30 May 2019*
  - Chathika Weerasuriya, *29 Nov 2019 – great reference list*





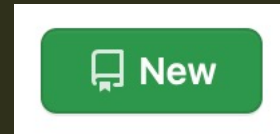
# What can I do with this? To begin

1. Create a Github account (“Pro” is free for academic users)

- <https://github.com/williameoswald?tab=repositories>

2. Create a repository or “repo” for your project or analysis

- Easiest to do via github.com
- Add name, specify public/private (can change later), select to initialize this repository with a README
- Don’t bother with .gitignore or license for now – can create later



3. Download and install Git (<https://git-scm.com/downloads>)

4. Install Github desktop (Git can also be run using “command line” but trickier)

5. Use Github desktop to “Clone repository” to create local copy

- Creates folder on your local drive where you can store code for specified project or analysis
- Dropbox and Onedrive – a warning





# What can I do with this? In practice

## 5. Think of workflow:

- Data cleaning and assembly
- Analysis

## 6. Create separate code for these

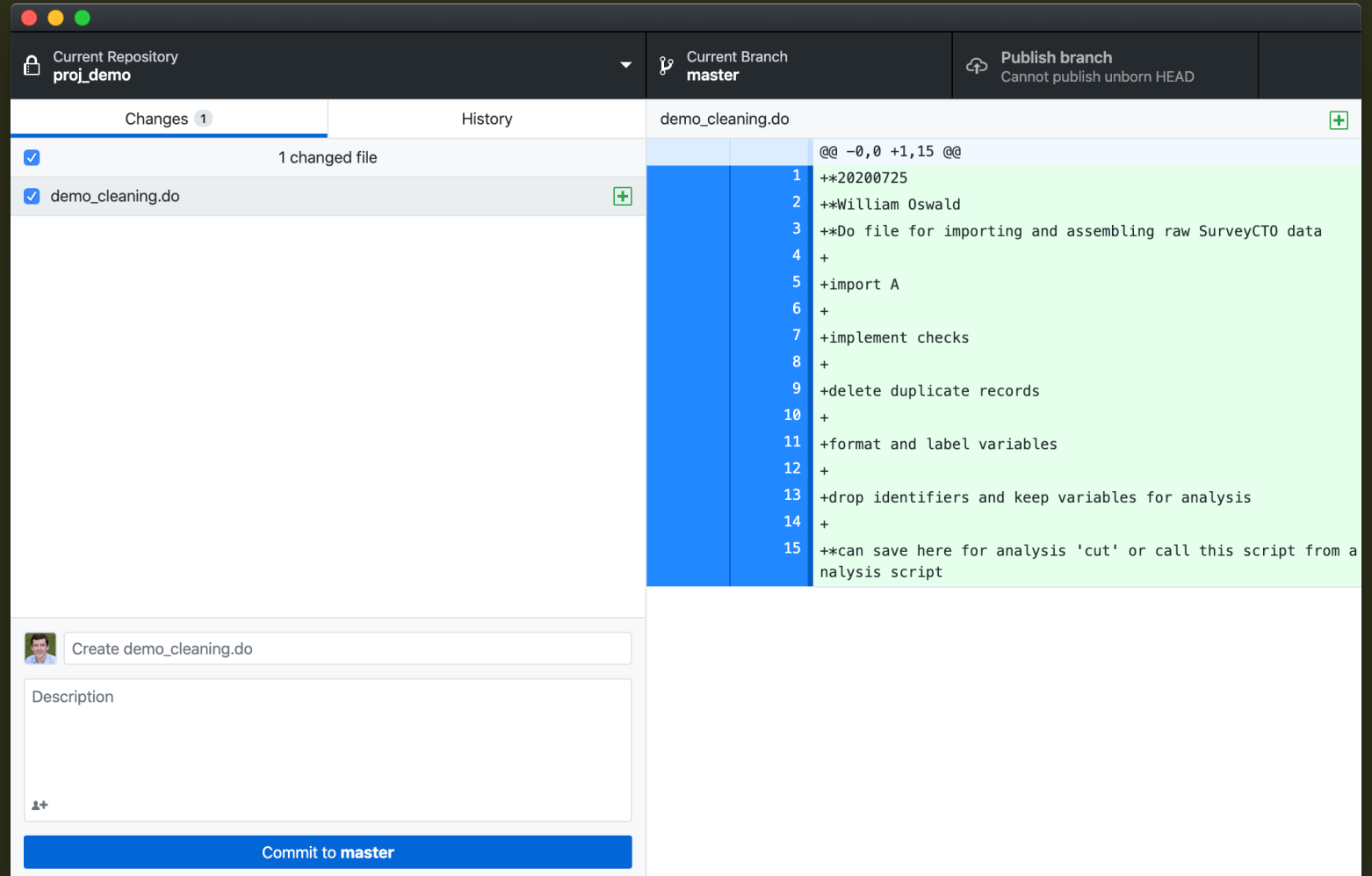
5. Chain - Call previous stage in subsequent steps (ensures latest data in use)
6. Master – Single file that sequentially calls each separate code

## 7. Save code (.R or .do file) to your local repository folder (linked to Github)



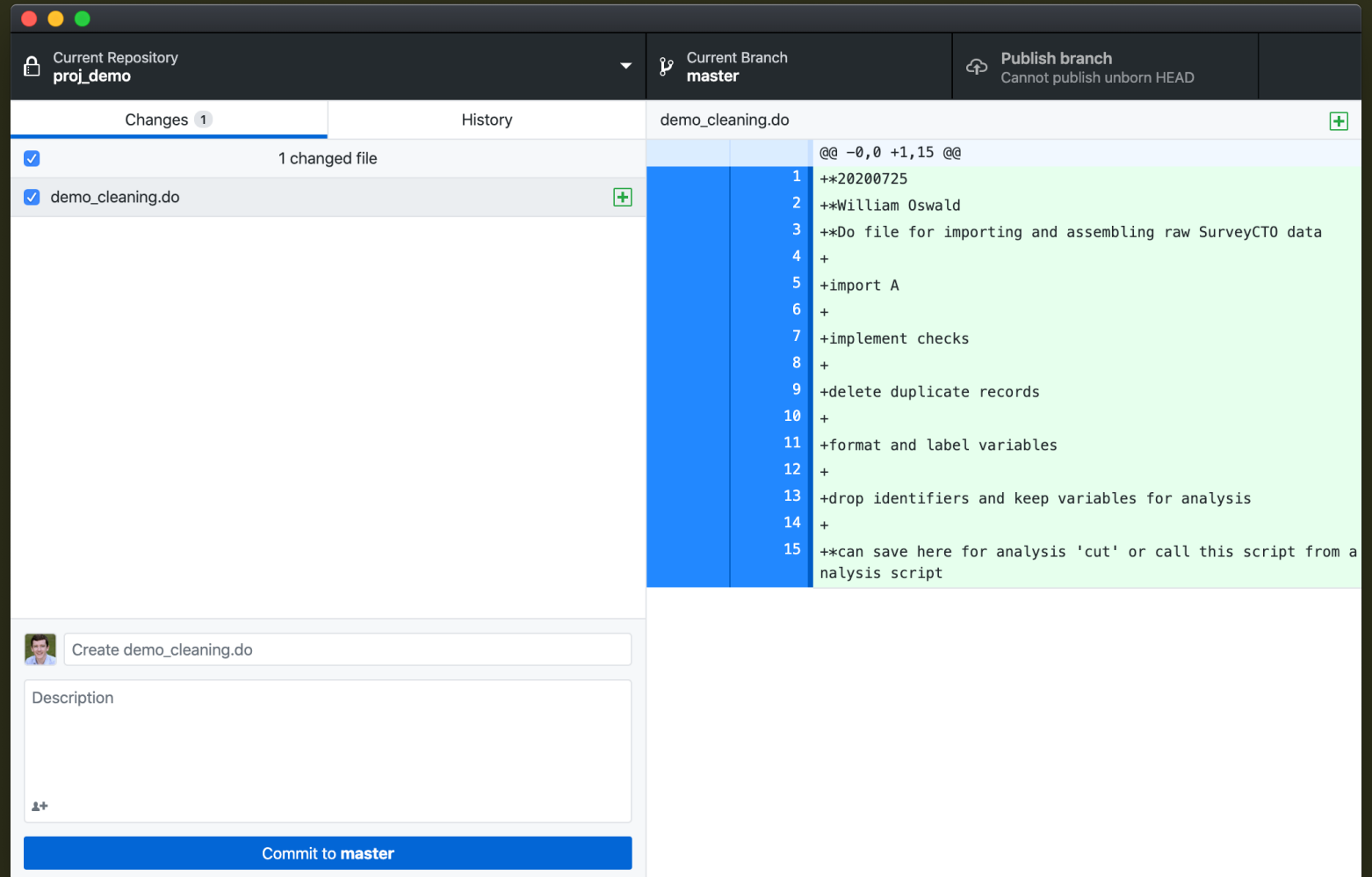
# What can I do with this? In practice

9. Once you save a file to your local folder, Github desktop will detect it
  - Add brief title for commit (or let Github desktop do this for you)
  - Can add more detailed description of what file does
  - Displays content of code all in green to reflect added content
10. “Commit to master” creates the first snapshot of your code



# What can I do with this? In practice

- But...
- Code version and changes are only stored locally
- After committing use “Publish branch” to sync with Github remote repository



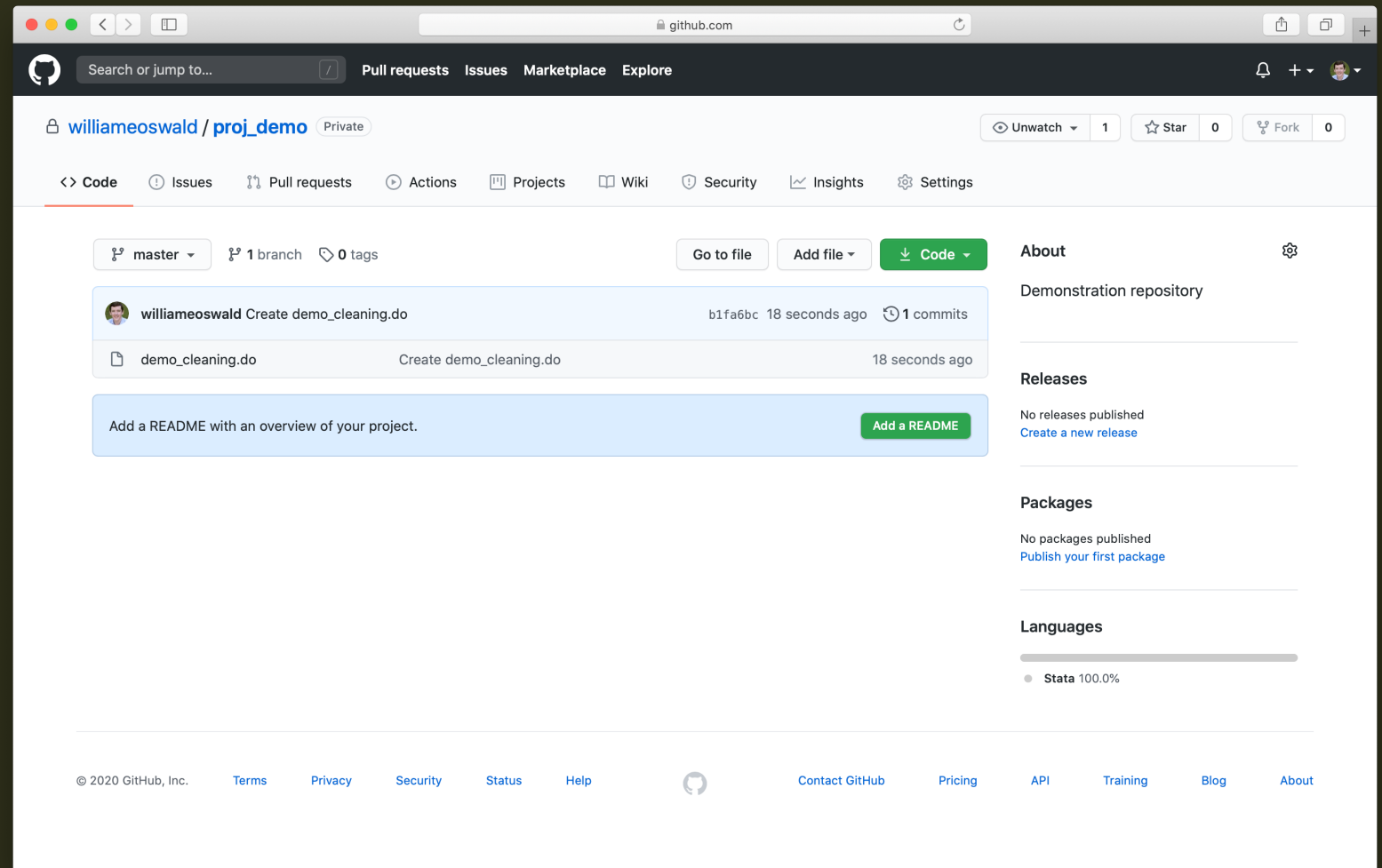
The screenshot shows a Git GUI interface for a repository named 'proj\_demo' on the 'master' branch. The 'Changes' tab is active, showing one changed file: 'demo\_cleaning.do'. The 'History' tab is also visible. The file content is displayed in a code editor, showing a script with 15 lines of code. The code is as follows:

```
@@ -0,0 +1,15 @@
1 ++20200725
2 ++William Oswald
3 ++Do file for importing and assembling raw SurveyCTO data
4 +
5 +import A
6 +
7 +implement checks
8 +
9 +delete duplicate records
10 +
11 +format and label variables
12 +
13 +drop identifiers and keep variables for analysis
14 +
15 ++can save here for analysis 'cut' or call this script from a
    nalysis script
```

At the bottom of the interface, there is a commit message field containing 'Create demo\_cleaning.do', a description field, and a 'Commit to master' button.

# What can I do with this? In practice

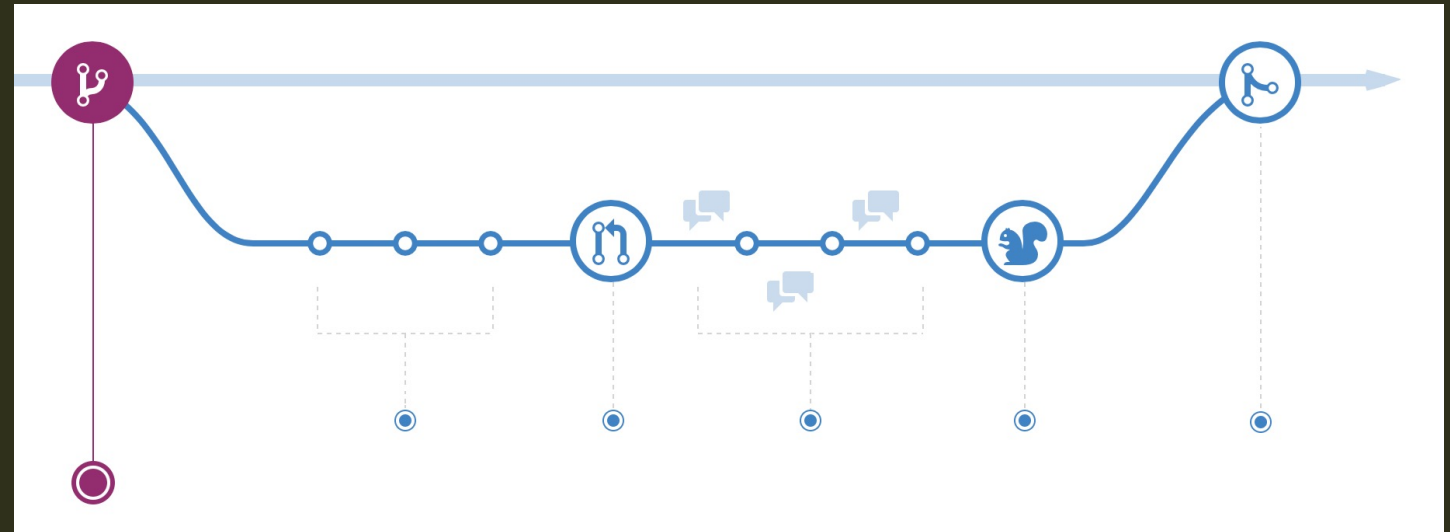
- “Commit” your code early and often (at least daily)
  - Commit to master creates the initial version
  - Can stack local commits in staging area
  - “Publish” to push and archive changes



The screenshot shows a GitHub repository page for 'proj\_demo' by user 'williameoswald'. The repository is private and has 1 star and 0 forks. The main content area shows a commit history with one commit: 'williameoswald Create demo\_cleaning.do' with commit hash 'b1fa6bc' and '18 seconds ago'. Below the commit is a file 'demo\_cleaning.do' with the same commit information. A blue banner prompts to 'Add a README with an overview of your project.' with an 'Add a README' button. The right sidebar contains sections for 'About' (Demonstration repository), 'Releases' (No releases published), 'Packages' (No packages published), and 'Languages' (Stata 100.0%). The footer includes copyright information and various links like Terms, Privacy, Security, Status, Help, Contact GitHub, Pricing, API, Training, Blog, and About.

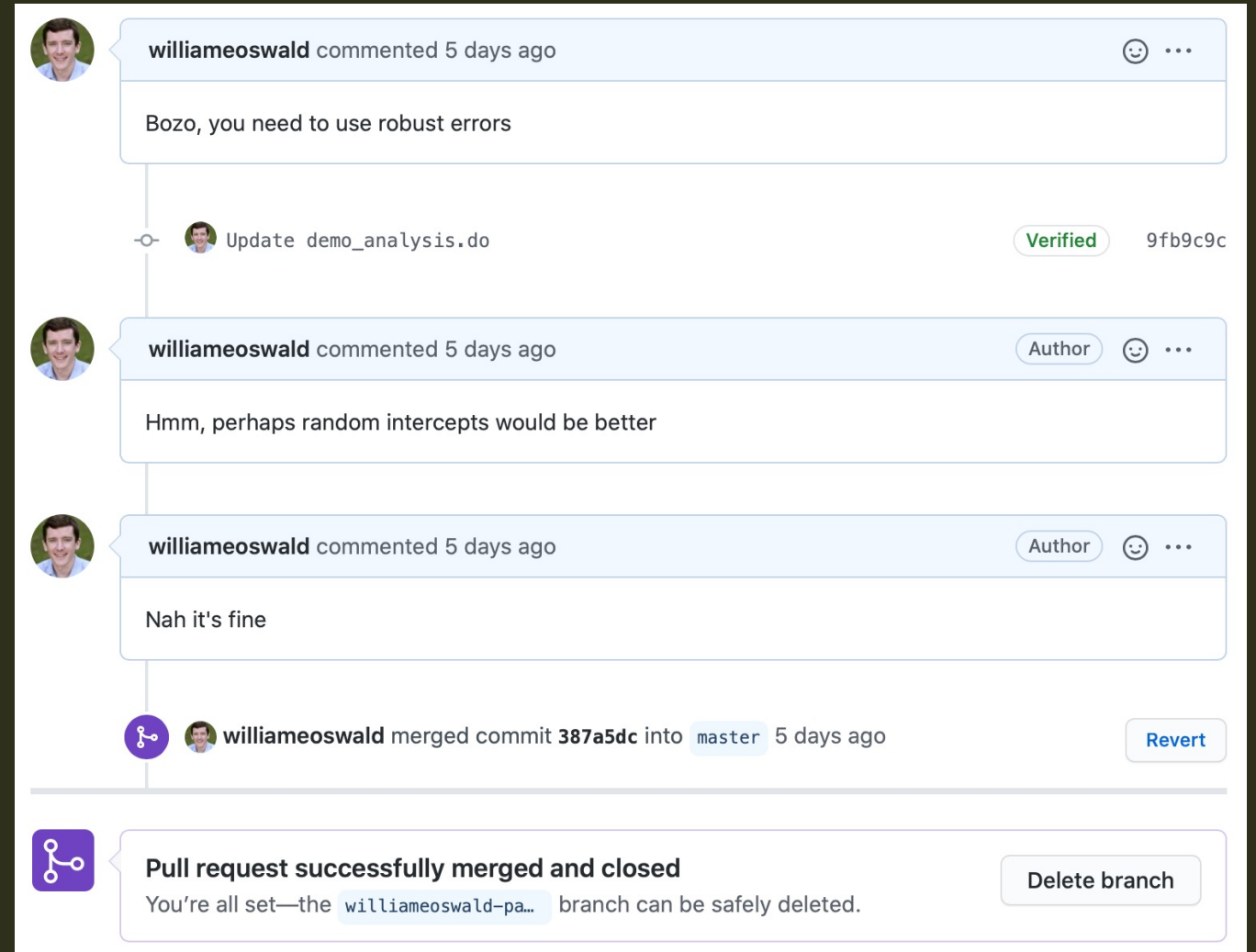
# Branching

- Branching is a core concept in Git
- Branches allow diverging revisions to maintain functionality
  - Use for experimentation
- Advanced workflow
  - Branch, Commit, Pull Request, Review, Deploy, Merge
- One rule:
  - Anything in the main branch is always deployable



# Branching

- Branching facilitates collaboration
- Others can clone your repository and suggest revisions using “pull request”
- Revisions can then be merged into main branch once reviewed and accepted



The screenshot shows a GitHub pull request interface. At the top, a comment from **williameoswald** (commented 5 days ago) says "Bozo, you need to use robust errors". Below this, a commit update is shown: "Update demo\_analysis.do" by **williameoswald** (Verified, 9fb9c9c). A second comment from **williameoswald** (commented 5 days ago) says "Hmm, perhaps random intercepts would be better". A third comment from **williameoswald** (commented 5 days ago) says "Nah it's fine". Below the comments, a merge event is shown: "williameoswald merged commit 387a5dc into master 5 days ago". At the bottom, a notification states "Pull request successfully merged and closed" and "You're all set—the williameoswald-pa... branch can be safely deleted." with a "Delete branch" button.

# Branching

- Branching facilitates collaboration
- Others can clone your repository and suggest revisions using “pull request”
- Revisions can then be merged into main branch once reviewed and accepted

The screenshot shows a GitHub pull request interface. At the top, a comment from user 'williameoswald' says 'Bozo, you need to use robust errors'. Below the comment is a commit card for 'Update demo\_analysis.do' with a 'Verified' badge and commit hash '9fb9c9c'. The pull request title is 'Update demo\_analysis.do #1' and it is marked as 'Merged'. The merge summary states 'williameoswald merged 1 commit into master from williameoswald-patch-1 5 days ago'. The file changes section shows 'demo\_analysis.do' with 2 changes. A diff view is displayed below, comparing the original code with the proposed changes. The original code (left) has a red background for the change on line 8, and the proposed code (right) has a green background for the same line, showing the addition of 'vce(robust)'.

williameoswald commented 5 days ago

Bozo, you need to use robust errors

Update demo\_analysis.do Verified 9fb9c9c

### Update demo\_analysis.do #1

Merged williameoswald merged 1 commit into master from williameoswald-patch-1 5 days ago

Conversation 2 Commits 1 Checks 0 Files changed 1 +1 -1

Changes from all commits File filter... Jump to... Review changes

#### Update demo\_analysis.do

master (#1)

williameoswald committed 5 days ago Verified commit 9fb9c9c4df228d2afdf320bc96795b043f481c14

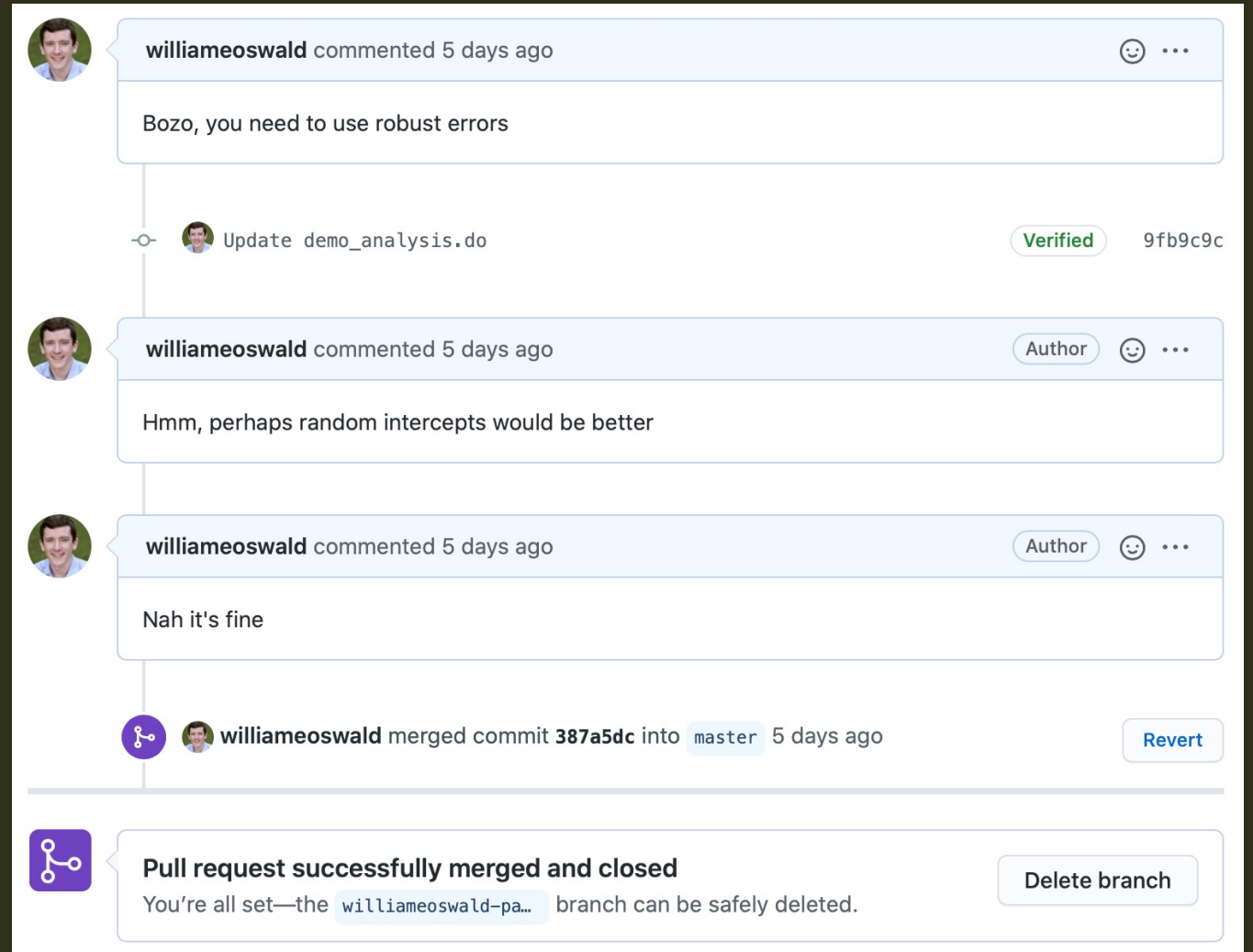
2 demo\_analysis.do

@@ -5,5 +5,5 @@	
5 *Run assembly do file	5 *Run assembly do file
6 do "demo_cleaning.do"	6 do "demo_cleaning.do"
7	7
8 - logistic A B C	8 + logistic A B C, vce(robust)
9	9



# Branching

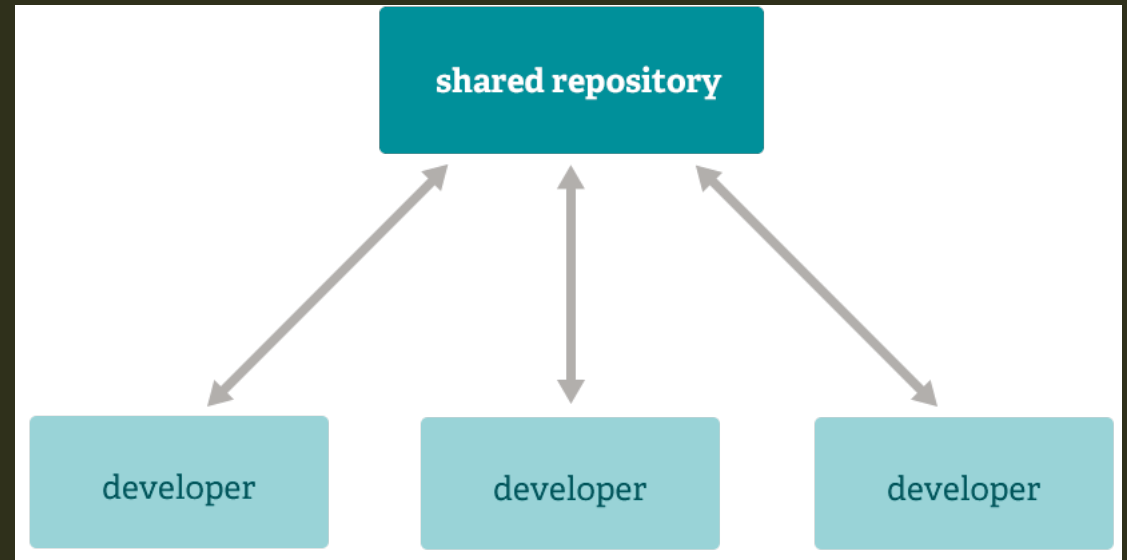
- Branching facilitates collaboration
- Others can clone your repository and suggest revisions using “pull request”
- Revisions can then be merged into main branch once reviewed and accepted



The screenshot shows a GitHub pull request interface. At the top, a comment from **williameoswald** (commented 5 days ago) says "Bozo, you need to use robust errors". Below this, a commit update is shown: "Update demo\_analysis.do" by **williameoswald** (Verified, 9fb9c9c). A second comment from **williameoswald** (commented 5 days ago) says "Hmm, perhaps random intercepts would be better". A third comment from **williameoswald** (commented 5 days ago) says "Nah it's fine". Below the comments, a merge event is shown: "williameoswald merged commit 387a5dc into master 5 days ago". At the bottom, a notification states "Pull request successfully merged and closed" and "You're all set—the williameoswald-pa... branch can be safely deleted." with a "Delete branch" button.

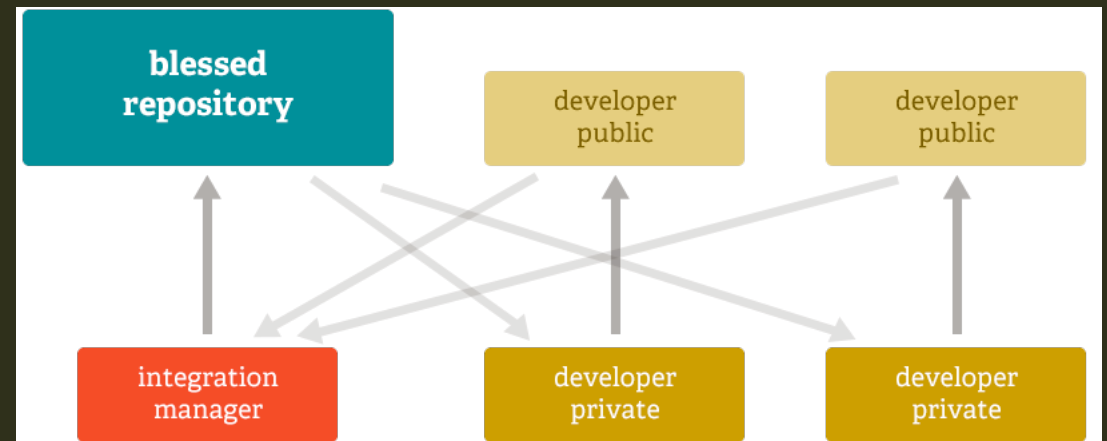
# Workflows

- “Subversion-Style”
  - Centralised model
  - Smaller projects
- Git will not allow you to push if someone has pushed since the last time you fetched



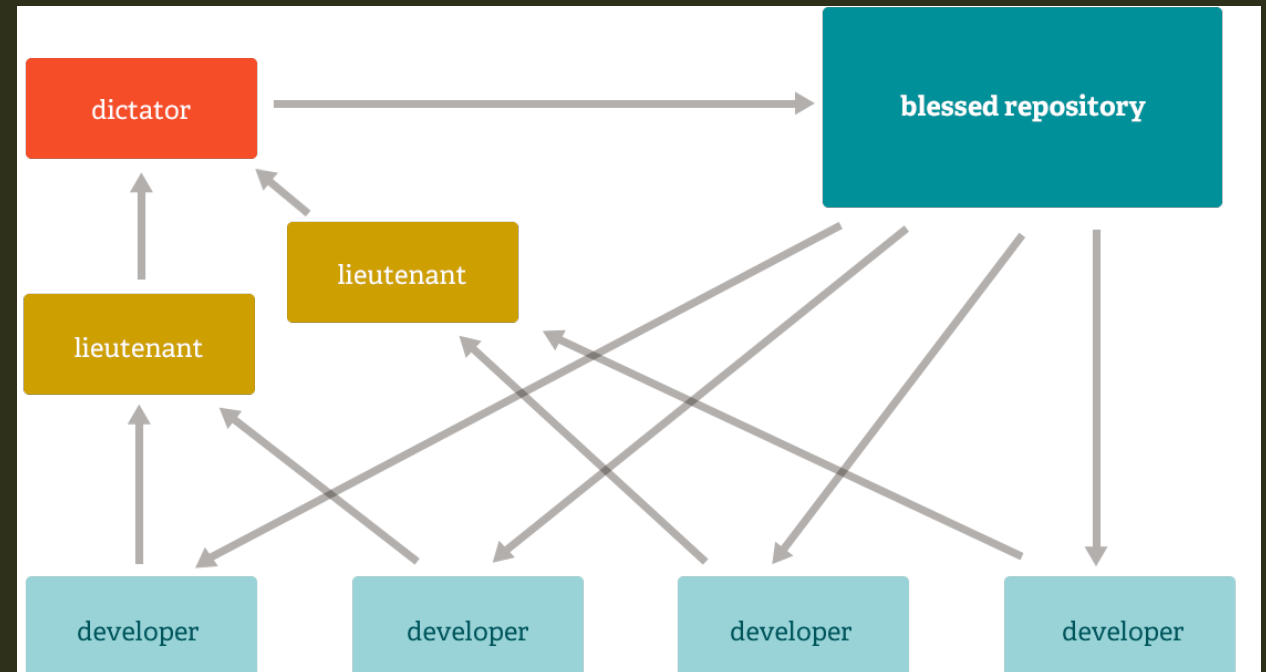
# Workflows

- “Integration Manager Workflow”
- Single person who commits to “blessed repository”
- Most often seen with open source or GitHub repositories



# Workflows

- “Dictator and Lieutenants Workflow”
- Massive projects



# Git for 'typical' research projects

- New project
  - Create Github repository
  - Clone repository to local drive
  - Save scripts and .do files via Stata or R to repository folder and push changes



# Git for 'typical' research projects

- New project

- Create Github repository
- Clone repository to local drive
- Save scripts and .do files via Stata or R to repository folder and push changes

- Existing project (and you're able to convince people of the benefits)

- Create Github repository
- Clone repository to local drive
- Save scripts and .do files via Stata or R to repository folder and push changes

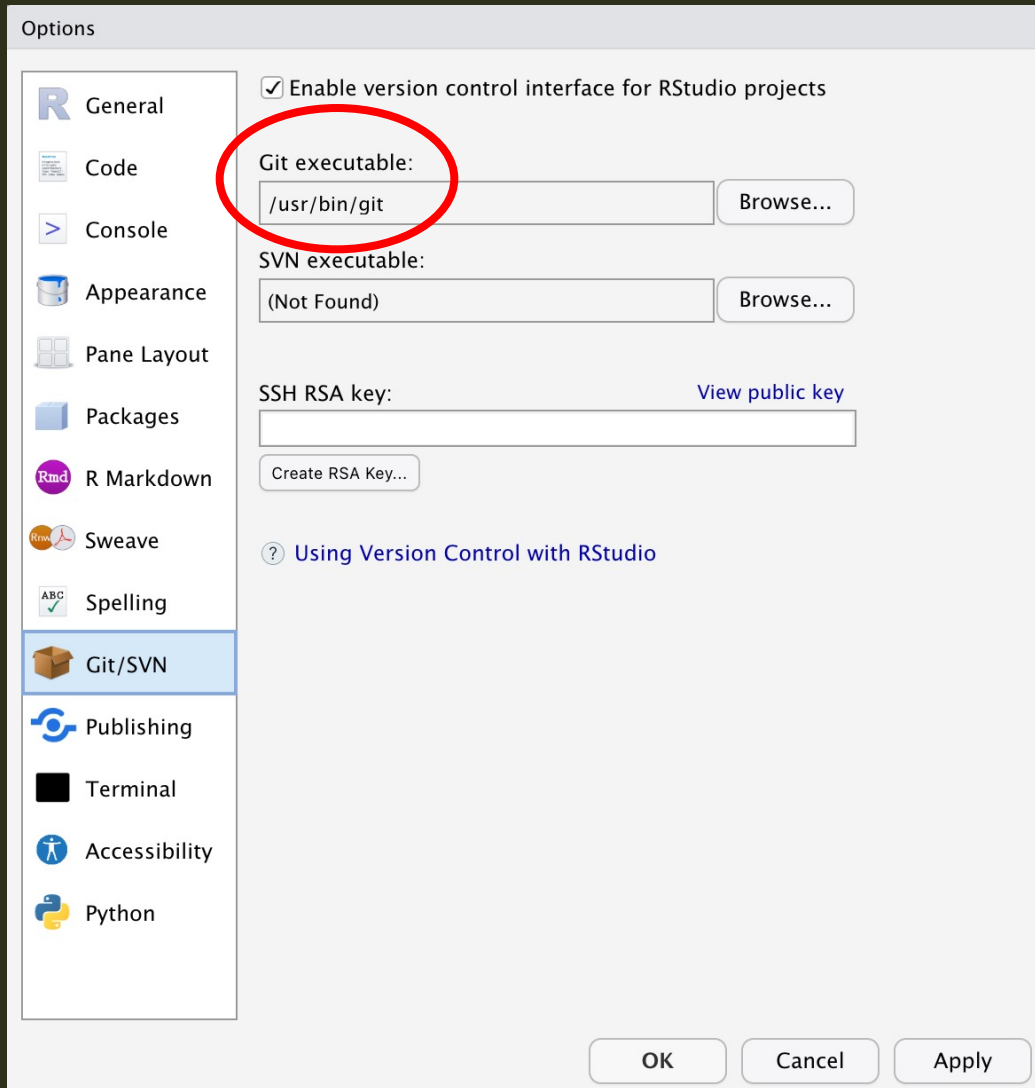
- If time allowed, save earlier starting file in local repository, then overwrite with subsequent ones, committing changes each time. This approach would stack changes and build a version history.

# How does Git work with R (and Stata)?

- For R users
  - R (or more specifically R Studio) includes add-ins for Git and Github.com
  - Changes to scripts can be committed and pushed/pulled via Rstudio or Github Desktop
  - RStudio “project” can be nested within local repository folder
- For Stata users
  - No direct integration with Git or Github
  - Changes to .do files can be pushed/pulled via Github Desktop
- Use .gitignore file to exclude data or other files (e.g. \*.rproj) from Git and just sync scripts
  - Right click file or folder in Github Desktop and select *Ignore file (Add to .gitignore)*

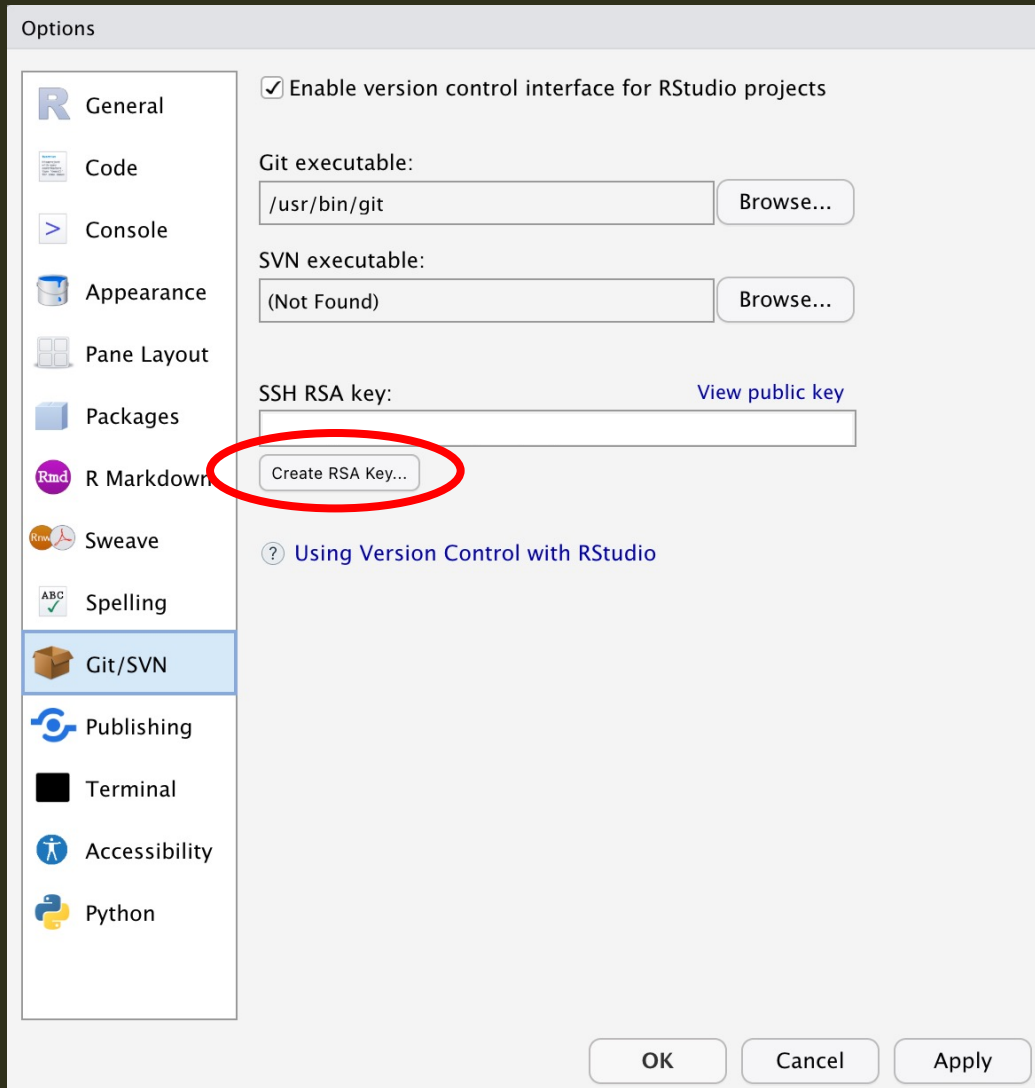


# Configure RStudio to use git



- Open RStudio
- Click *Tools* -> *Global Options* -> *Git/SVN*
- If *Git executable* shows (*none*), click *Browse* and select the git executable installed on your system
  - On a Mac, this will likely be one of
    - /usr/bin/git
    - /usr/local/bin/git
    - /usr/local/git/bin/git
  - On Windows, git.exe will likely be somewhere in Program Files
- Click *OK*

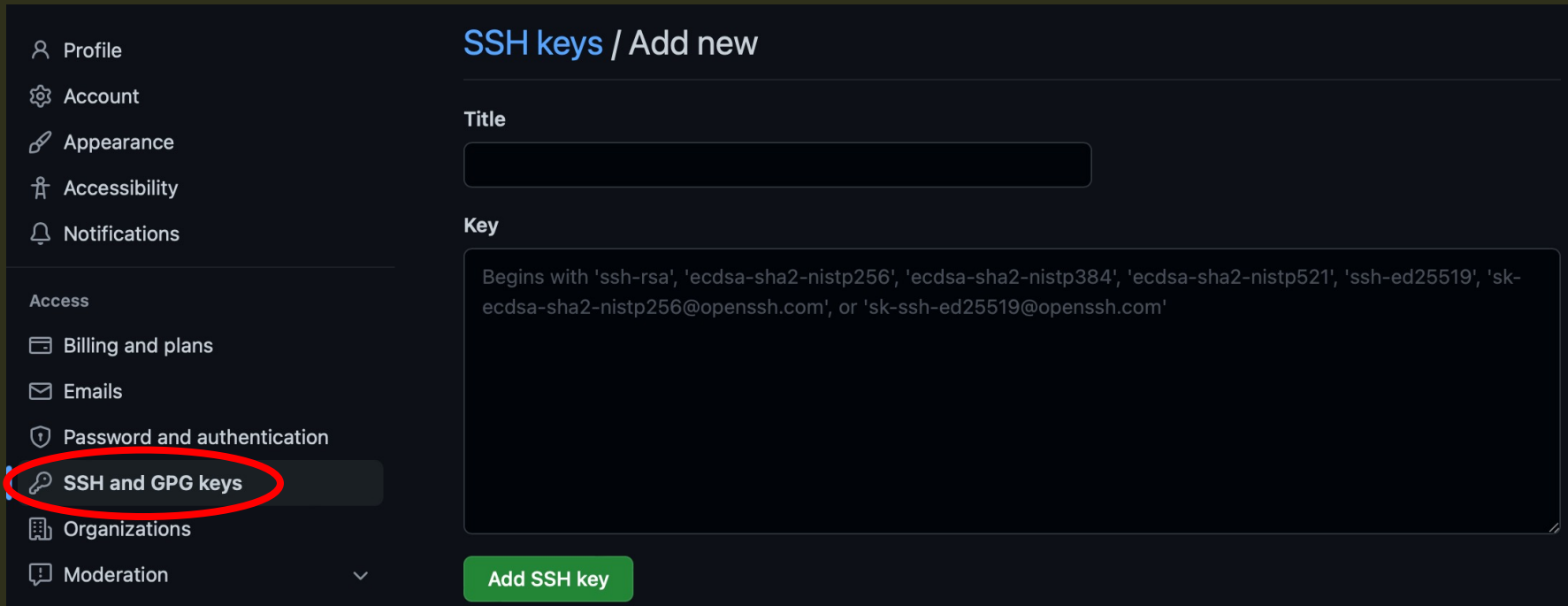
# Configure RStudio to use git



- You may need to add a key to connect to Github via Rstudio as they no longer allow passwords\*
- Click *Tools* -> *Global Options* -> *Git/SVN*
- Click *Create RSA Key*
- Click *View public Key* and copy the key

\*Thanks to Antonio Gasparrini

# Configure RStudio to use git



The screenshot shows the GitHub 'SSH keys / Add new' page. On the left is a navigation sidebar with options: Profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, SSH and GPG keys (highlighted with a red circle), Organizations, and Moderation. The main content area has a 'Title' input field and a 'Key' text area containing a list of key types: 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', and 'sk-ssh-ed25519@openssh.com'. A green 'Add SSH key' button is at the bottom.

- Open GitHub and log into your account
- Go to Settings and then select *SSH and GPG keys*
- Click *New SSH key* and paste the key and add a title
- Allows computer to link directly with GitHub when you pull/push without the need to register your details every time



# Using git for 'typical' research project

- Save script and .do files via Stata or R (Rstudio) to repository folder and push changes
- Edit code directly via Github.com and pull changes
  - Stata 17 will detect changes on disk and can automatically load changes
  - In RStudio use separate 'run' script to run main script



# Demonstration



# Benefits

- Changes are recorded and easily traceable via Github
- Can revert to earlier versions
- Flexible workflows
- New project members and contributors can “clone” repository and have exact same files
- Multiple contributors can edit same code by creating branches to preserve functionality of main code
  - Avoids worries about “breaking” .do files
- Reproducible research
  - Easy to find and share code later
  - Archiving also possible via other non-private platforms with DOI (can be linked to LSHTM Data Compass):
    - <https://zenodo.org>
    - <https://osf.io>
    - <https://datadryad.org/stash>
- Bonus features - free public hosting of one .html per account – allows you to setup a live dashboard



“Humans are fallible; that’s why we need code review.”

EDITOR'S CHOICE

## **Code Review as a Simple Trick to Enhance Reproducibility, Accelerate Learning, and Improve the Quality of Your Team’s Research** FREE

[Anusha M Vable](#) ✉, [Scott F Diehl](#), [M Maria Glymour](#)

*American Journal of Epidemiology*, Volume 190, Issue 10, October 2021, Pages 2172–2177,

<https://doi.org/10.1093/aje/kwab092>





# Conclusions

- Learning curve
- Bypass the jargon
- Use Github desktop
- Upload existing code now, any snapshot better than none!
- Try it!



# Questions?



[william.oswald@lshtm.ac.uk](mailto:william.oswald@lshtm.ac.uk)

[@williameoswald](#)