

Extra 3 classes on Git and Github

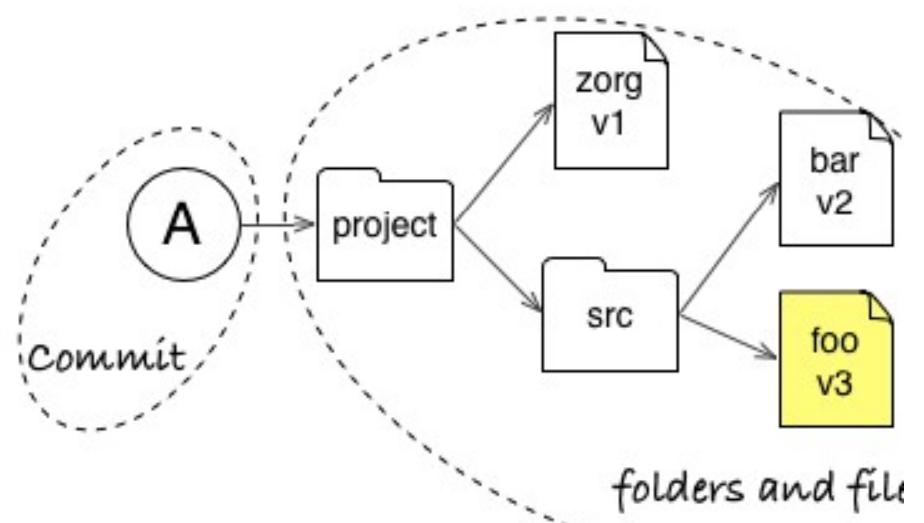
Module 7_extra

Core concepts in Git

Module 7-2

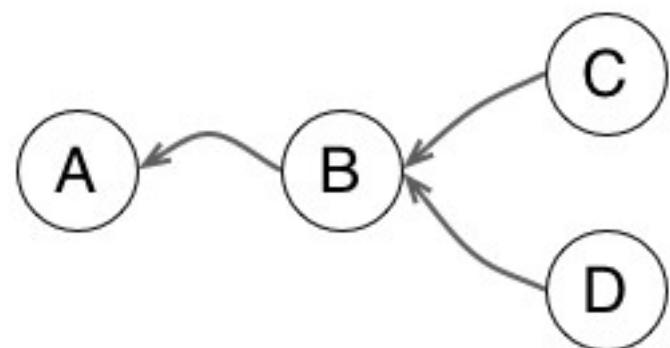
Core Concepts

Git stores snapshots
(commits) of your repository



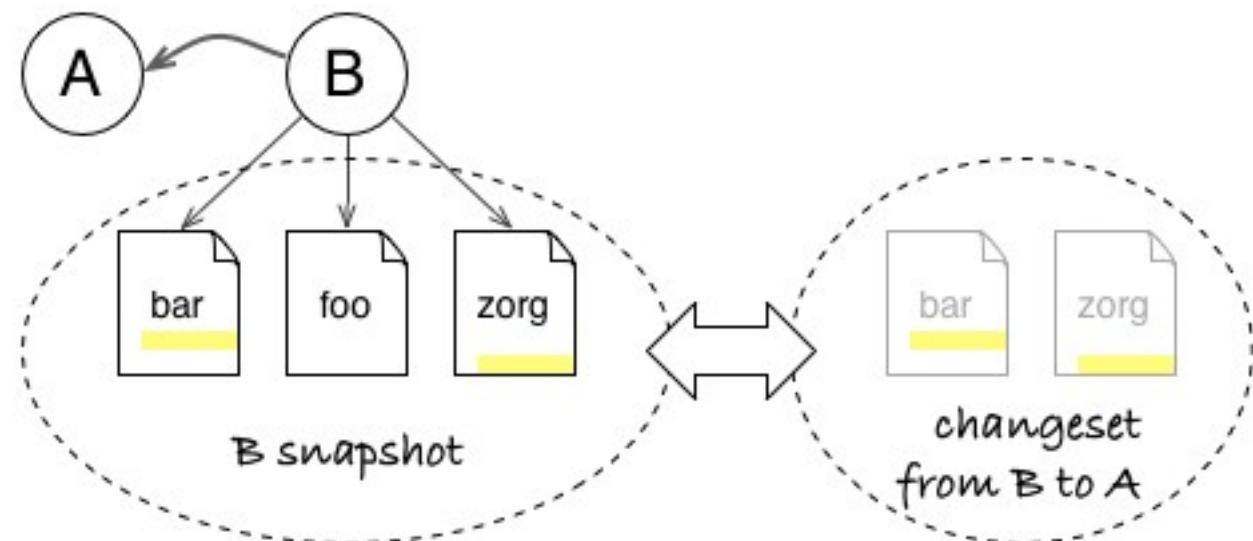
Core Concepts

Git represents relationships between commits as a graph



Core Concepts

Git can compute changesets between any two commits of your project



Core Concepts

Git sees changes at the level of lines in a text file

Fix error in boxplot labelling

master

 robschick committed on Jan 20 1 parent da9140b

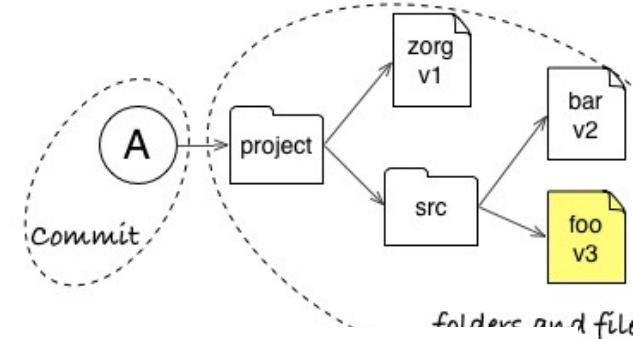
Showing 1 changed file with 1 addition and 1 deletion.

2 R/plotBoxplotHealth.R

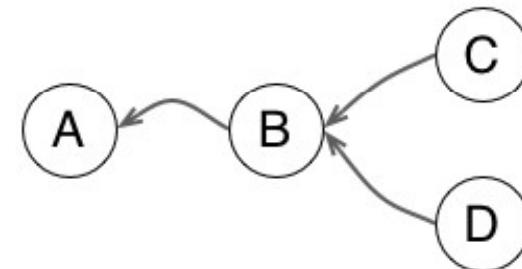
		@@ -35,7 +35,7 @@ plotBoxplotHealth <- function(dfLong, bsize, cval = 4){
35	35	nvals\$Freq[nvals\$Var1 == 'NonRepFem' & nvals\$Var2 == 2],
36	36	nvals\$Freq[nvals\$Var1 == 'RepFem' & nvals\$Var2 == 2],
37	37	nvals\$Freq[nvals\$Var1 == 'NonRepFem' & nvals\$Var2 == 3],
38	-	nvals\$Freq[nvals\$Var1 == 'RepFem' & nvals\$Var2 == 2],
	38	+ nvals\$Freq[nvals\$Var1 == 'RepFem' & nvals\$Var2 == 3],
39	39	nvals\$Freq[nvals\$Var1 == 'NonRepFem' & nvals\$Var2 == 1],
40	40	nvals\$Freq[nvals\$Var1 == 'RepFem' & nvals\$Var2 == 1])
41	41	

3 Core Concepts

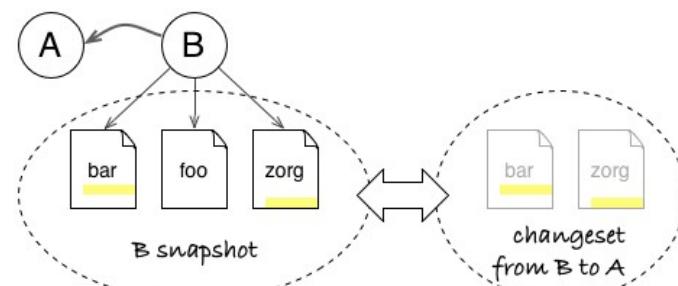
- Snapshot



- Graph



- Changeset



States of a git Repository

The Repository

- Collection of files managed by git
- History (all of it)
- Encompassing file on the Operating System is considered the working directory
 - Can include files managed by git
 - Files ignored by git
 - Files not yet managed by git
- Quasi-hidden **.git** folder
- Since the repo contains all the history, keep the repos narrowly focused

Three Local States

Working Directory

```
m7:src
```



```
if p == None or p.v == 0:
    cond = str(p) + ' is None'
    Break()
else if p.v > 0:
    Comp(p.v)
    if p.v == 12:
        return 12
    else:
        cond = str(p) + ' is not None'
        Break()
else:
    cond = str(p) + ' is None'
    Break()
for C in range(0, 12):
    if C == 0:
        q = q1
    else:
        q = q1 + q2 + q3 + q4
```

Staging Area

```
m7:src
```



```
if p == None or p.v == 0:
    cond = str(p) + ' is None'
    Break()
else if p.v > 0:
    Comp(p.v)
    if p.v == 12:
        return 12
    else:
        cond = str(p) + ' is not None'
        Break()
else:
    cond = str(p) + ' is None'
    Break()
for C in range(0, 12):
    if C == 0:
        q = q1
    else:
        q = q1 + q2 + q3 + q4
```



git add

Repository (.git folder)

```
m7:src
```

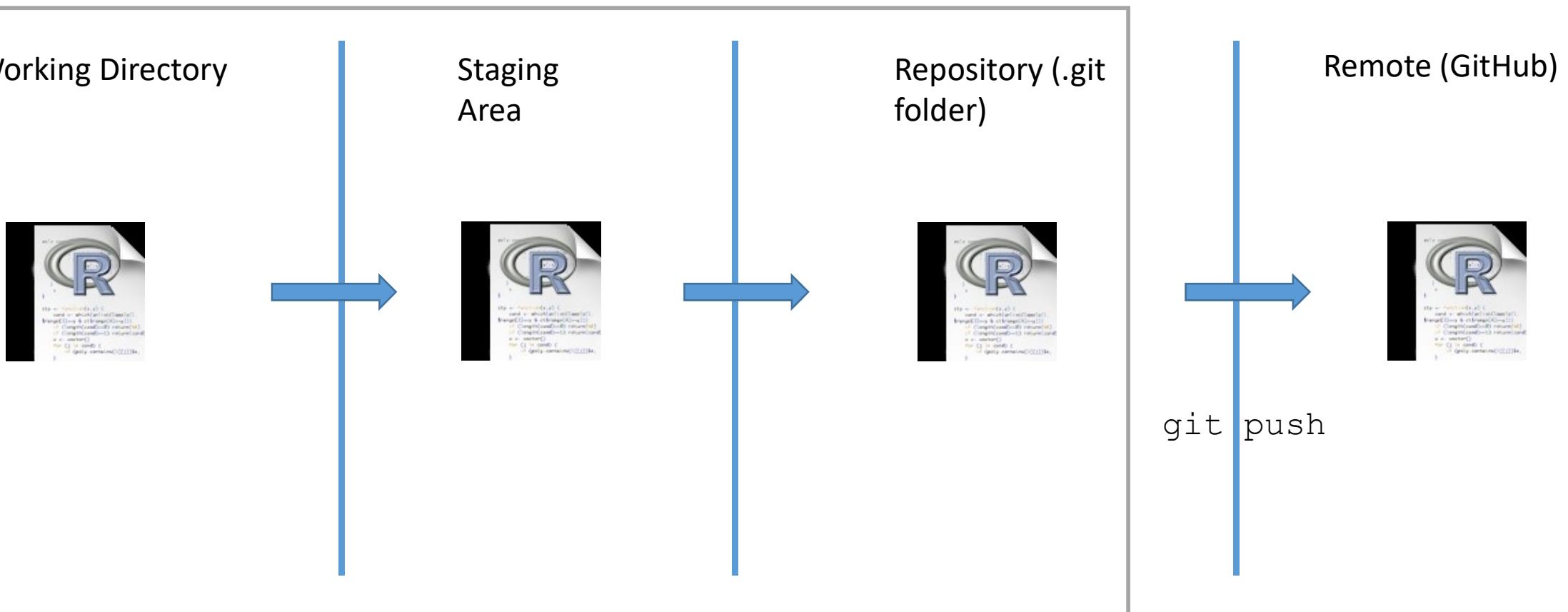


```
if p == None or p.v == 0:
    cond = str(p) + ' is None'
    Break()
else if p.v > 0:
    Comp(p.v)
    if p.v == 12:
        return 12
    else:
        cond = str(p) + ' is not None'
        Break()
else:
    cond = str(p) + ' is None'
    Break()
for C in range(0, 12):
    if C == 0:
        q = q1
    else:
        q = q1 + q2 + q3 + q4
```



git commit

Three Local States with Remote



Basic Commands

Mastering a Basic Workflow

git init – Initialize an Empty Repo

```
MINGW64:/c/Users/Dossa/AFEC-2019
Dossa@Dossa-PC MINGW64 ~
$ mkdir AFEC-2019

Dossa@Dossa-PC MINGW64 ~
$ cd AFEC-2019

Dossa@Dossa-PC MINGW64 ~/AFEC-2019
$ git init
Initialized empty Git repository in C:/Users/Dossa/AFEC-2019/.git/

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$
```

git add – Add a Document to the Staging Area

```
MINGW64:/c/Users/Dossa/AFEC-2019
Dossa@Dossa-PC MINGW64 ~
$ mkdir AFEC-2019

Dossa@Dossa-PC MINGW64 ~
$ cd AFEC-2019

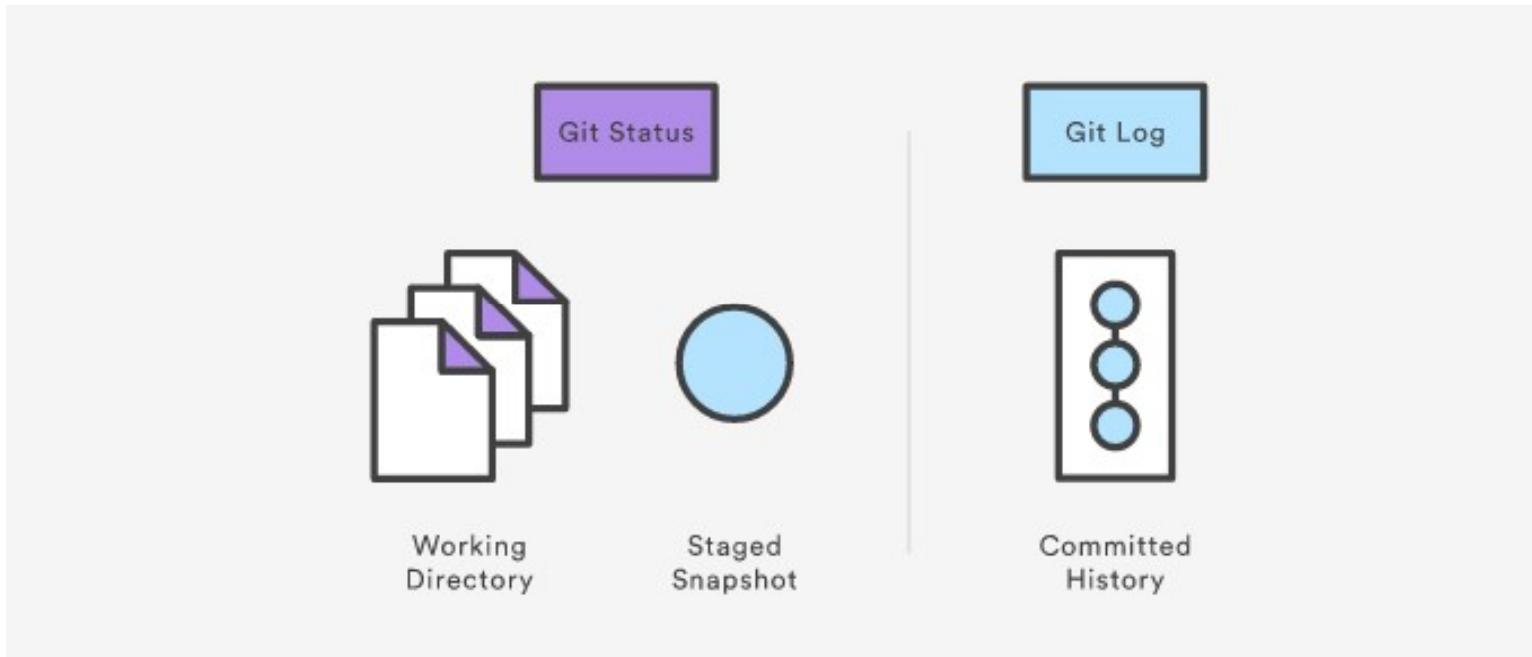
Dossa@Dossa-PC MINGW64 ~/AFEC-2019
$ git init
Initialized empty Git repository in c:/Users/Dossa/AFEC-2019/.git/

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ touch Myrscript.R

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git add Myrscript.R

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ ls -a
```

Viewing the Repo



<https://www.atlassian.com/git/tutorials/inspecting-a-repository>

git status – What's Happening?

```
MINGW64:/c/Users/Dossa/AFEC-2019
Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:  Myrscript.R

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$
```

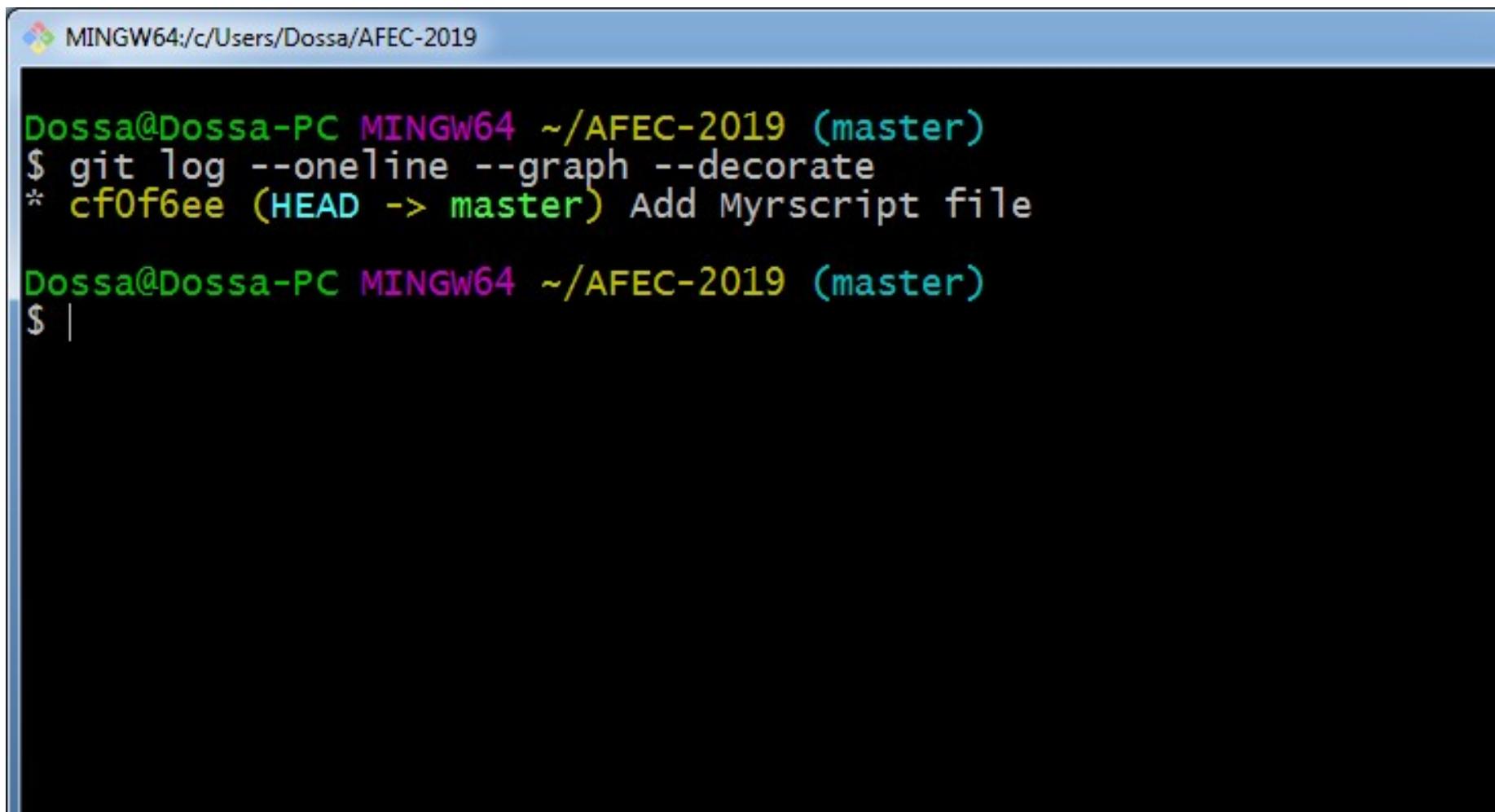
git log – To view the history of Repo

```
MINGW64:/c/Users/Dossa/AFEC-2019
Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git log
fatal: your current branch 'master' does not have any commits yet

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git commit -m "Add Myrscript file"
[master (root-commit) cf0f6ee] Add Myrscript file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Myrscript.R
[REDACTED]
Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git log
commit cf0f6ee9327d757ac438ef3c346605836cf53650
Author: Dossa <dossag@postgrad.unu.edu>
Date:   Thu Oct 24 05:58:35 2019 +0800

    Add Myrscript file
```

git log – With Options



A screenshot of a terminal window titled "MINGW64:/c/Users/Dossa/AFEC-2019". The window displays a command-line session:

```
Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git log --oneline --graph --decorate
* cf0f6ee (HEAD -> master) Add Myrscript file

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ |
```

The terminal shows a single commit with the hash cf0f6ee, labeled as the HEAD of the master branch. The commit message is "Add Myrscript file". The prompt at the bottom is \$ |.

git commit – Records changes in the Repo

- git commit -m "Second check in of my R Script"

```
MINGW64:/c/Users/Dossa/AFEC-2019
Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git log
fatal: your current branch 'master' does not have any commits yet

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git commit -m "Add Myrscript file"
[master (root-commit) cf0f6ee] Add Myrscript file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Myrscript.R

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git log
commit cf0f6ee9327d757ac438ef3c346605836cf53650
Author: Dossa <dossag@postgrad.unu.edu>
Date:   Thu Oct 24 05:58:35 2019 +0800

    Add Myrscript file
```

Ok, What Just happened?

```
MINGW64:/c/Users/Dossa/AFEC-2019
Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git log
fatal: your current branch 'master' does not have any commits yet

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git commit -m "Add Myrscript file"
[master (root-commit) cf0f6ee] Add Myrscript file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Myrscript.R

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git log
commit cf0f6ee9327d757ac438ef3c346605336cf53650
Author: Dossa <dossag@postgrad.unu.edu>
Date:   Thu Oct 24 05:58:35 2019 +0800

    Add Myrscript file
```

sha-1 commit

What's Changed

Git rm

- Don't delete or rename tracked files with the OS; use:
- git rm
- git mv

```
MINGW64:/c/Users/Dossa/AFEC-2019
nothing to commit, working directory clean

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git ls-files
Myrscript.R
Myrscript2.R

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ mkdir Code_folder

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git mv Myrscript2 Code_folder
fatal: bad source, source=Myrscript2, destination=Code_folder

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git mv Myrscript2.R Code_folder

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git ls-files
Myrscript.R
Myrscript2.R

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
$ git rm Myrscript.R
rm 'Myrscript.R'

Dossa@Dossa-PC MINGW64 ~/AFEC-2019 (master)
```

Git mv

- Don't delete or rename tracked files with the OS; use:
- `git mv`

Lather, Rinse, Repeat

Good Commit Messages*

- **Be concise, yet evocative.** At a glance, you should be able to see what a commit does. But there should be enough detail so you can remember (and understand) what was done
- **Describe the why, not the what.** Since you can always retrieve the diff associated with the commit, the message doesn't need to say exactly what changed. Instead it should provide a high-level summary that focuses on the reasons for the change

*<http://r-pkgs.had.co.nz/git.html#commit-best-practices>

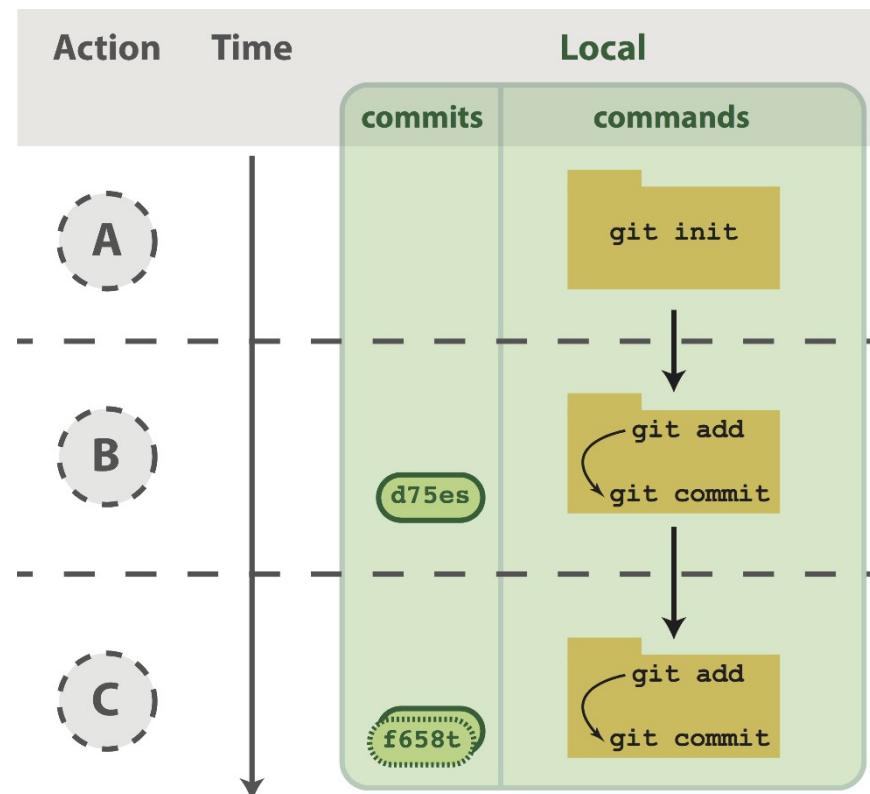
Good Commit Messages

The seven rules of a great Git commit message

Keep in mind: This has all been said before.

1. Separate subject from body with a blank line
2. Limit the subject line to 50 characters
3. Capitalize the subject line
4. Do not end the subject line with a period
5. Use the imperative mood in the subject line
6. Wrap the body at 72 characters
7. Use the body to explain *what* and *why* vs. *how*

Workflow Visualized



Blischak, John D., Emily R. Davenport, and Greg Wilson. 2016.
“A Quick Introduction to Version Control with Git and GitHub.”
PLoS Computational Biology 12 (1): e1004668.

Lifecycle of status

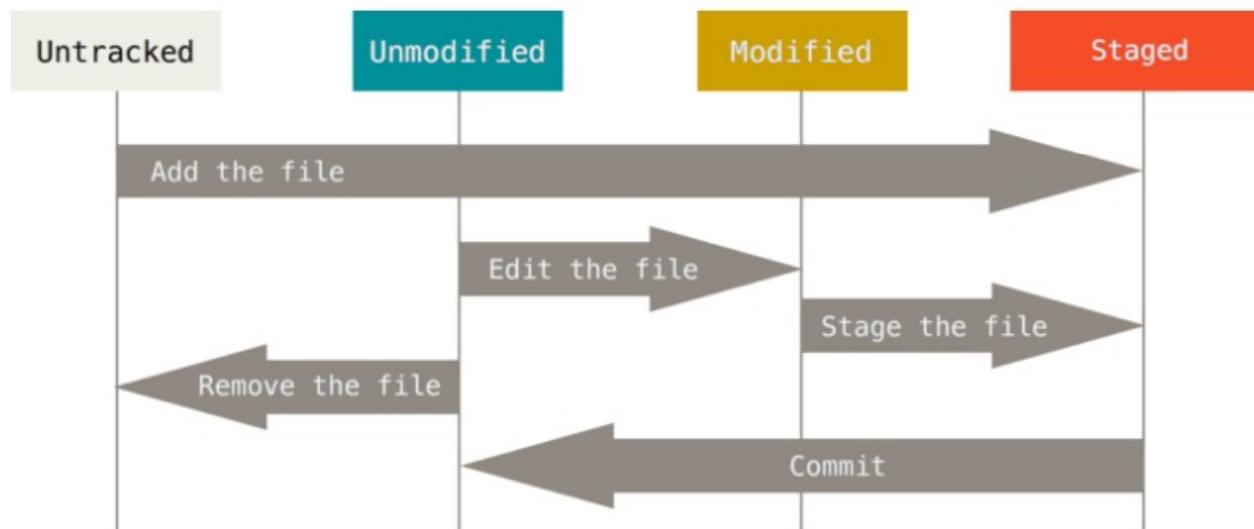
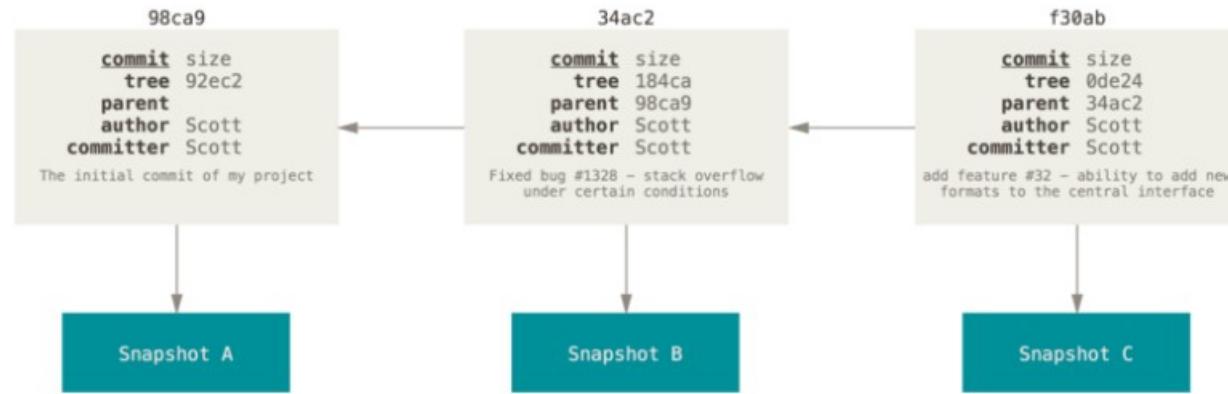


FIGURE 2-1

The lifecycle of the status of your files.

Commit Graph Visualized



A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is `master`. As you start making commits, you're given a `master` branch that points to the last commit you made. Every time you commit, it moves forward automatically.

.gitignore

Branch: master ▾ [gitignore / R.gitignore](#) Find file Copy path

 **jrnold** Add knitr and R markdown patterns to R.gitignore 4956277 on Apr 29, 2016

11 contributors 

34 lines (24 sloc) | 500 Bytes Raw Blame History   

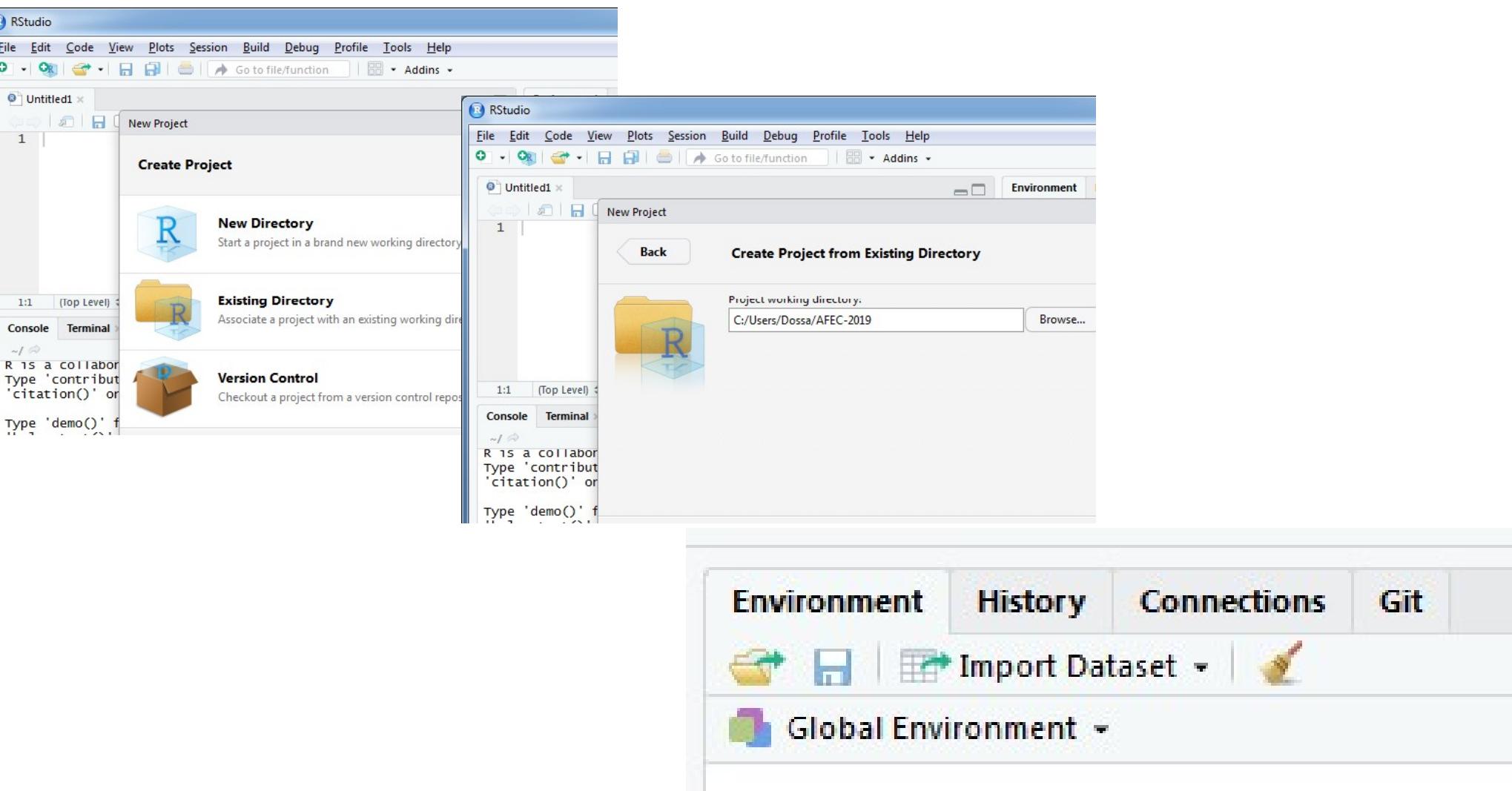
```
1 # History files
2 .Rhistory
3 .Rapp.history
4
5 # Session Data files
6 .RData
7
8 # Example code in package build process
9 *-Ex.R
10
11 # Output files from R CMD build
12 /*.tar.gz
```

.gitignore

- Make the code produce a plot file, say a pdf
- Run the code
- Make a .gitignore file
 - .pdf
 - .Rout
- Add & commit the .gitignore
- Run ls (you should see the pdf and the Rout file)
- Run git ls-files (you should *not* see the pdf and the Rout file)

Break Time?

New R Project With Your Existing Repo



Make the Changes in RStudio

C:/Users/Dossa/AFEC-2019 - master - RStudio

Untitled1*

Source | Source

```
1 #This will do awesome things ... some day|
```

Environment History Connect

Import Dataset | Global Environment

Environment is

Files Plots Packages Help

New Folder Delete

C: > Users > Dossa > AFEC-2019

Name

1:42 (Top Level) R Script

Add / made changes

push/pull greyed out

build Debug Profile Tools Help

to file/function | Addins

Environment History Connections Git

Diff Commit Staged Status Path

- .gitignore
- AFEC-2019.Rproj
- Myrscript.R
- Myrscript2.R -> Code_folder/Myrscript2.R

Files Plots Packages Help Viewer

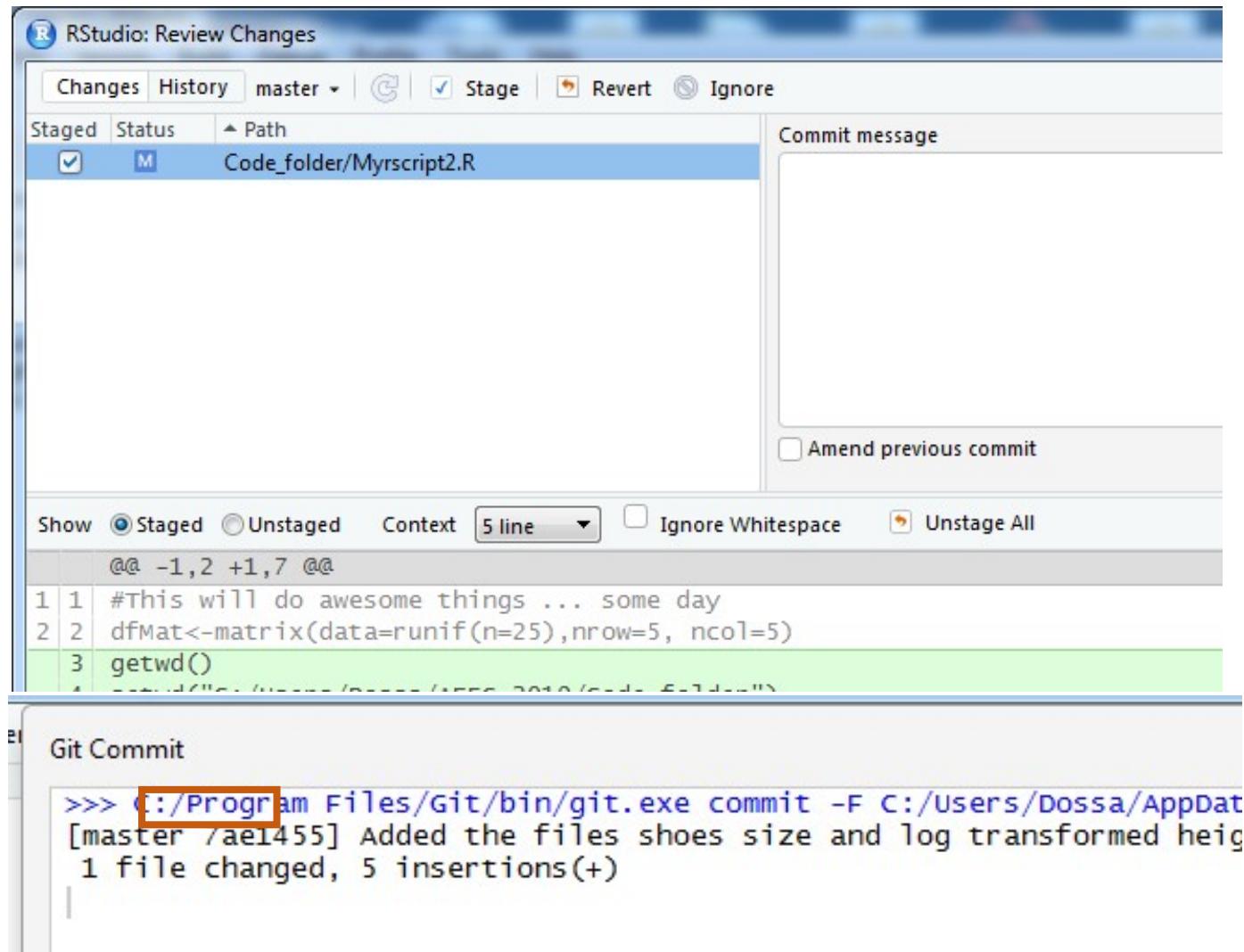
New Folder Delete Rename More

C: > Users > Dossa > AFEC-2019

Name Size

..

Commit the Changeset



Make Some Modifications

The screenshot shows two instances of the RStudio interface, each displaying a 'Git' pane. The left pane shows the state before changes, and the right pane shows the state after changes have been staged.

Left Pane (Initial State):

- File: invertMatrix.R
- Content:

```
# this will do awesome things...some day
dfMat <- matrix(data
```
- Git Status:
 - Diff: No changes
 - Commit: No changes
 - Pull: No changes
 - Push: No changes
 - History: No history

Right Pane (Modified State):

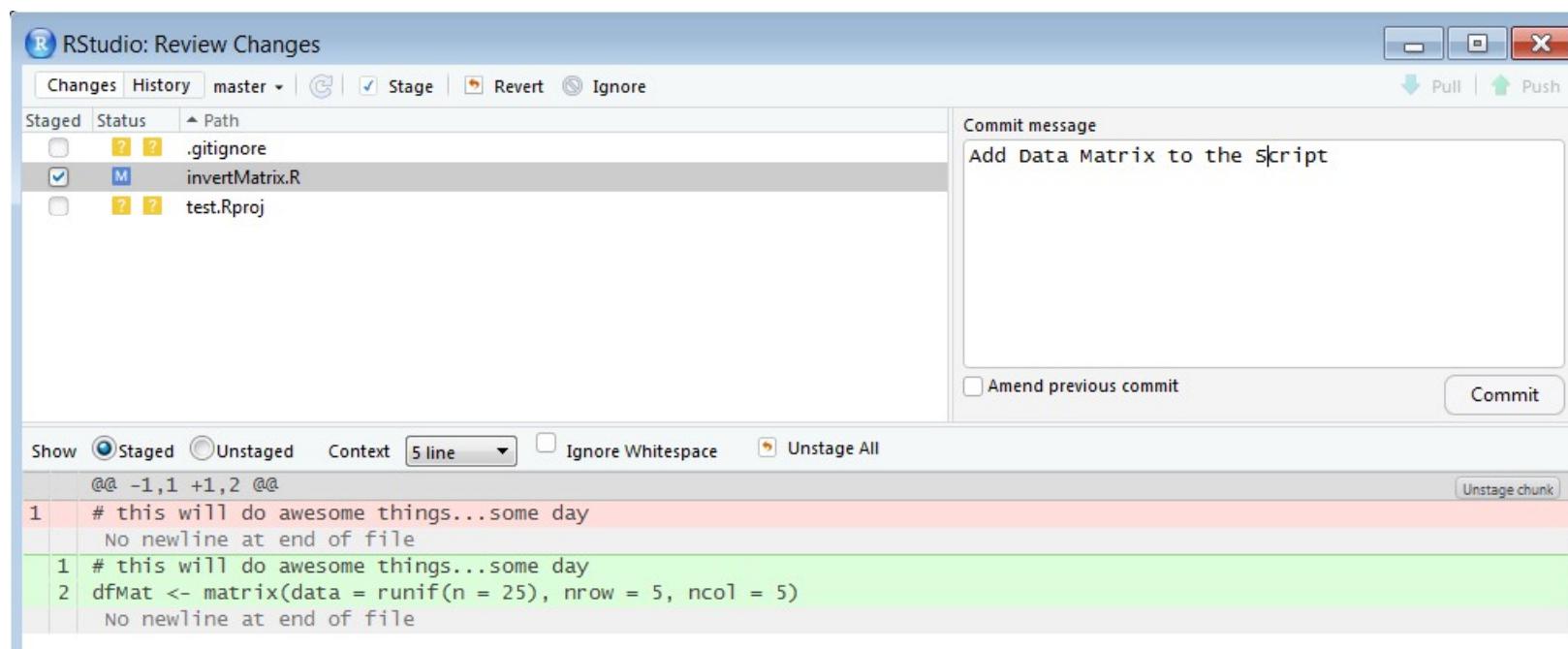
- File: invertMatrix.R
- Content:

```
# this will do awesome things...some day
dfMat <- matrix(data
```
- Git Status:
 - Diff: No changes
 - Commit: No changes
 - Pull: No changes
 - Push: No changes
 - History: No history

Annotations:

- A blue arrow points from the text "invertMatrix.R has been modified" to the 'invertMatrix.R' entry in the left Git pane, which has an 'M' status icon.
- A blue arrow points from the text "invertMatrix.R has been added" to the 'invertMatrix.R' entry in the right Git pane, which has a checked 'Commit' icon.

Commit the Modifications



Git in RStudio Covers *Most* of Your Needs

- If RStudio does all of this, why bother with the command line?
- *Most* of the time you won't need to, but when you need it, you need it
 - Merge conflicts, for example

Code Time

Connecting a Local Repo to a Remote Repo

Module 7-3

Git is a *Distributed* Version Control System

- Peer-to-peer as opposed to server-client
- Common operations (commits, viewing history, etc.) are fast since there is no need to communicate with a central server
- Communication is only necessary with sharing changes among peers
- Each working copy effectively functions as a remote backup of a codebase and its change history – protecting against data loss

How Does My Local Repo Sync with GitHub*

- Several ways to do this:
- Clone an existing repository from GitHub to a local folder
 - Your own, or
 - Another coder's
- Initialize a repository locally and push to GitHub
- Start a new project in RStudio with version control

*or bitbucket, gitlab, etc.

Local First

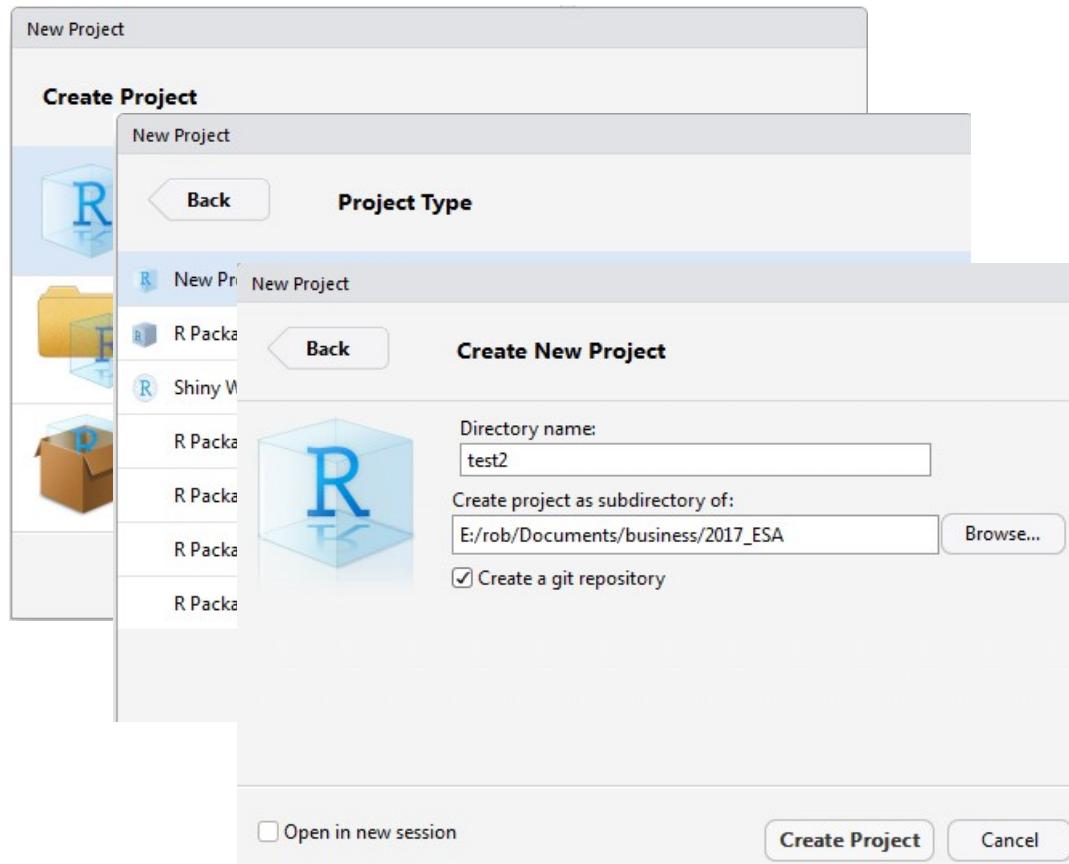
- We've started this with git init in a local repo
- How connect to GitHub?
 - Make an empty repo on GitHub, i.e. no README
 - Copy the url of this repo from GitHub
 - Navigate to your current folder
 - Add the remote
 - Push the repo

GitHub First

- You did a version of this when you worked through the Happy Git with R install tutorial
- How connect to GitHub?
 - Make a repo on GitHub (I typically add both a README and a .gitignore file)
 - Copy the url of this repo from GitHub
 - Navigate to your current folder
 - Clone the repo

From RStudio*

- Make a new project
- Project type
- Choose name & git



*<https://support.rstudio.com/hc/en-us/articles/200532077?version=1.1.322&mode=desktop>

My Preferred Way

- Start on GitHub
- Create an empty repo
 - Include a README file
 - Include a .gitignore file
- Clone it locally on the command line
- Set up a new project in RStudio in the existing directory

Let's Practice

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner Repository name



robschick ▾

/ esatest



Great repository names are short and memorable. Need inspiration? How about [ubiquitous-octo-funicular](#).

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: R ▾

Add a license: None ▾



Create repository

Let's Practice

robschick / esatest

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights ▾

No description, website, or topics provided. Edit

Add topics

1 commit 1 branch 0 releases 1 contributor

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

robschick Initial commit
.gitignore Initial commit
README.md Initial commit

Clone with HTTPS ⓘ Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/robschick/esatest.git>

Open in Desktop Download ZIP

esatest

Let's Practice

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA
$ git clone https://github.com/robschick/esatest.git
```

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA
$ git clone https://github.com/robschick/esatest.git
Cloning into 'esatest'...
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA
$ cd esatest/

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

Let's Practice

New Project

Create Project

New Directories

Start a project

Existing Directories

Associate a project

Version Control

Checkout a project

Back

Create Project from Existing Directory

Project working directory:

E:/rob/Documents/business/2017_ESA/esatest

Browse...

Open in new session

Create Project

Cancel

Environment History Connections Git

Diff Commit Pull Push History More

Staged Status Path

M .gitignore

? ? esatest.Rproj

Files Plots Packages Help Viewer

New Folder Delete Rename More

E: > rob > Documents > business > 2017_ESA > esatest

Name

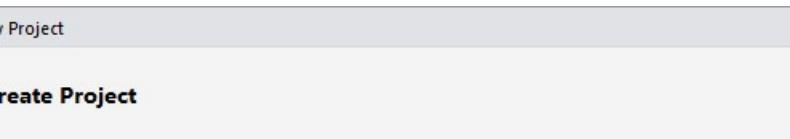
..

.gitignore

esatest.Rproj

README.md

Bypass the Command Line



New Project

Start

Existing

Associated

Version

Check

Git

Clone a project

SVN

Checkout

Subversion

Back

Create Project from Version Control

New Project

Back

Clone Git Repository

Repository URL:

Project directory name:

Create project as subdirectory of:

Open in new session

Environment History Connections Git

Diff Commit Pull Push History More

Staged Status Path

M .gitignore

? ? esatest.Rproj

Files Plots Packages Help Viewer

New Folder Delete Rename More

E: > rob > Documents > business > 2017_ESA > esatest

Name

..

.gitignore

esatest.Rproj

README.md

Pushing to GitHub – How Often?

- Several schools of thought on how often to push
- A prominent one (Hadley Wickham) is to push considerably less often than you commit*
 - “Pushing code means publishing code”
 - “Strive to push code that works”
- I tend to push more often than this, because I code on small teams, and I like the back up

*<http://r-pkgs.had.co.nz/git.html#commit-best-practices>

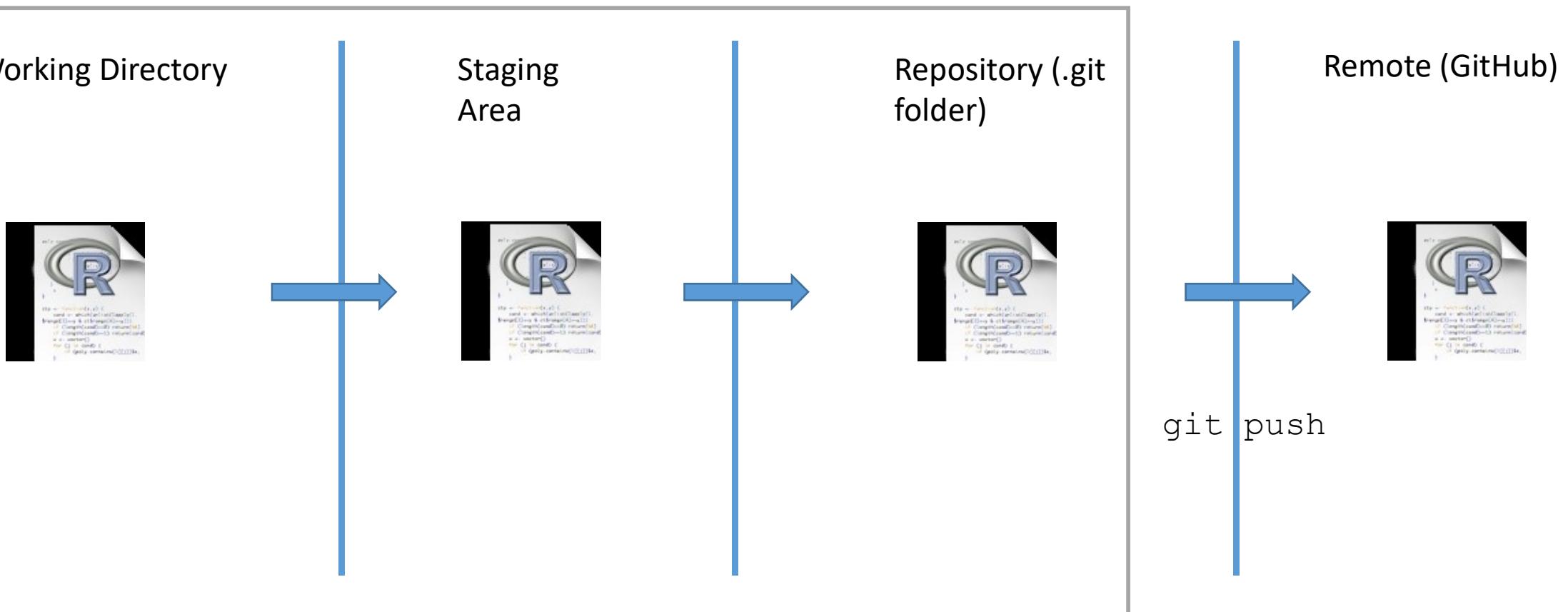
Pushing to GitHub – Good Commit Messages*

- **Be concise, yet evocative.** At a glance, you should be able to see what a commit does. But there should be enough detail so you can remember (and understand) what was done
- **Describe the why, not the what.** Since you can always retrieve the diff associated with the commit, the message doesn't need to say exactly what changed. Instead it should provide a high-level summary that focuses on the reasons for the change

*<http://r-pkgs.had.co.nz/git.html#commit-best-practices>

Practice a Cycle in Local Repo

Three Local States with Remote



Adding GitHub to the Workflow – git push

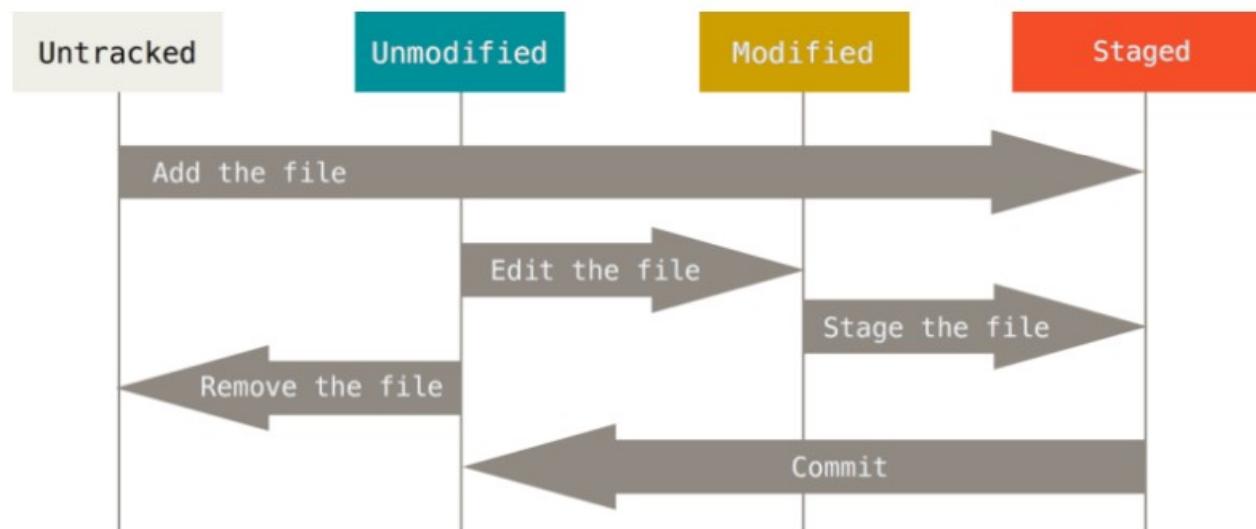
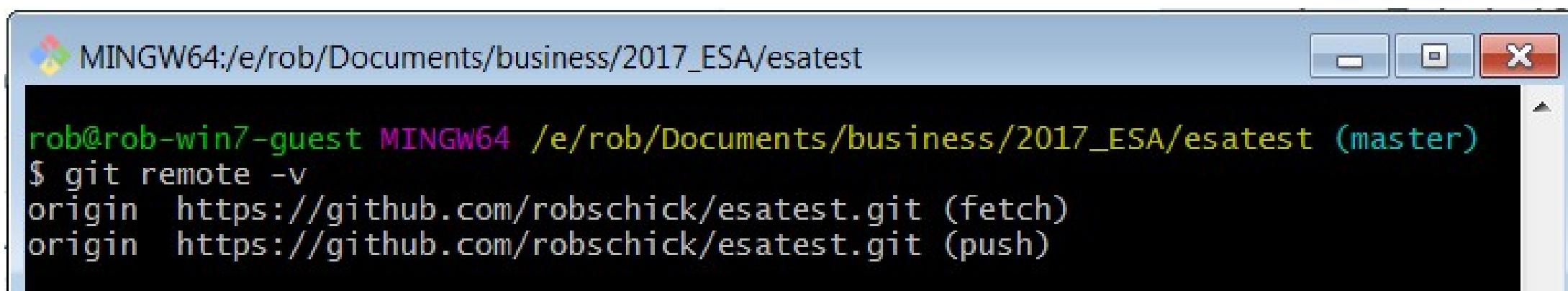


FIGURE 2-1

The lifecycle of the status of your files.

Knowing Your Remotes



A screenshot of a terminal window titled "MINGW64:/e/rob/Documents/business/2017_ESA/esatest". The window shows the command \$ git remote -v and its output, which lists two remotes: origin (https://github.com/robschick/esatest.git) for both fetch and push operations.

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git remote -v
origin https://github.com/robschick/esatest.git (fetch)
origin https://github.com/robschick/esatest.git (push)
```

What is a Merge Conflict?

- A merge operation in git is when you try to blend changes made:
 - To the *same* file
 - On two *different* branches
- Wait, but we haven't talked (much) about branches?
- For now, know that we've been working with one branch – MASTER
- So we can work with the same branch – MASTER – in two (or more repos):
 - Local
 - GitHub
 - Node
- And these can come into conflict – let's try!

Merge Conflicts

- Make sure your working directory shows a clean status
- Do a git pull
- Do a git push
- Modify the file locally
- Commit it, but don't push it yet
- Navigate to the repo on GitHub
- Modify the same file on GitHub
- Commit it
- Go back to local repo and attempt a push

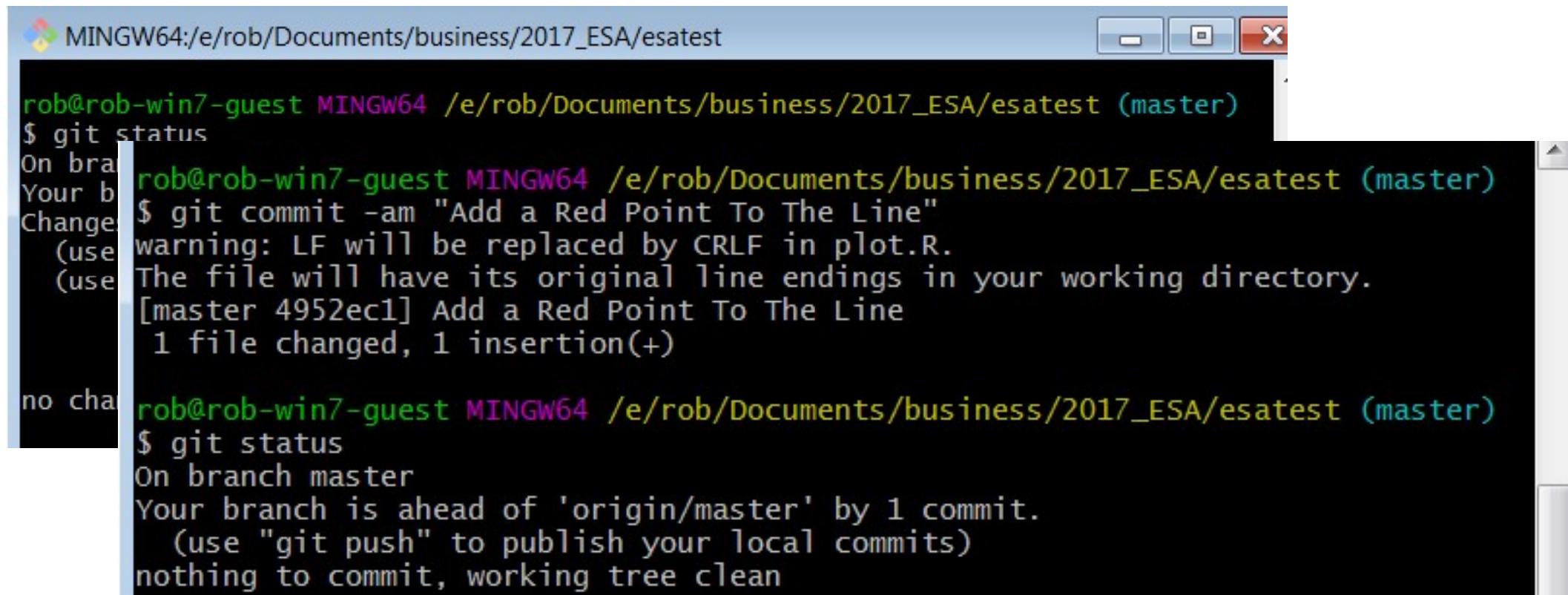
Check Status of Your Local Repo

- Best practice to pull before you push (n.b. if it's just you and GitHub, this isn't as big a concern)

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git pull
Already up-to-date.
```

Make Local Changes



MINGW64:/e/rob/Documents/business/2017_ESA/esatest

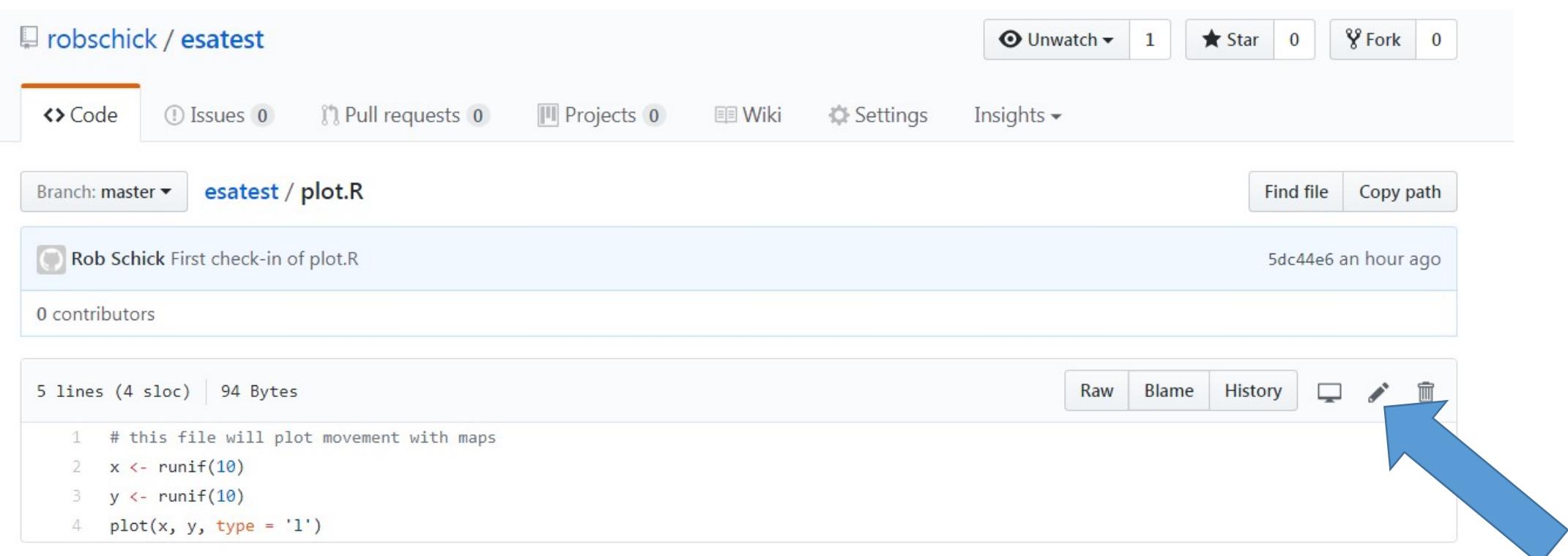
```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working tree clean
```

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
\$ git commit -am "Add a Red Point To The Line"
warning: LF will be replaced by CRLF in plot.R.
The file will have its original line endings in your working directory.
[master 4952ec1] Add a Red Point To The Line
 1 file changed, 1 insertion(+)

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
\$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
 (use "git push" to publish your local commits)
nothing to commit, working tree clean

Make Remote Changes

- Edit on GitHub



robschick / esatest

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Branch: master esatest / plot.R Find file Copy path

Rob Schick First check-in of plot.R 5dc44e6 an hour ago

0 contributors

5 lines (4 sloc) | 94 Bytes Raw Blame History

```
1 # this file will plot movement with maps
2 x <- runif(10)
3 y <- runif(10)
4 plot(x, y, type = 'l')
```

Commit the Changes



Commit changes

Add Title To Plot

Add an optional extended description...

- ➔ Commit directly to the `master` branch.
- ➕ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

View Status on GitHub

robschick / esatest

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Branch: master esatest / plot.R Find file Copy path

robschick Add Title To Plot 7a1d6e8 just now

1 contributor

5 lines (4 sloc) | 140 Bytes Raw Blame History

```
1 # this file will plot movement with maps
2 x <- runif(10)
3 y <- runif(10)
4 plot(x, y, type = 'l', main = 'This is Probably the Best Plot Ever')
```

What's Happened

- We have one repo – esatest (or whatever you've named it)
- We've made and committed changes in two different places
- Now let's try to sync the repos by pushing our local to GitHub

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git push origin master
To https://github.com/robschick/esatest.git
 ! [rejected]      master -> master (fetch first)
error: failed to push some refs to 'https://github.com/robschick/esatest.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```



Pull the Changes

```
MINGW64:/e/rob/Documents/business/2017_ESA/esatest
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/robschick/esatest
  5dc44e6..7a1d6e8  master      -> origin/master
Auto-merging plot.R
CONFLICT (content): Merge conflict in plot.R
Automatic merge failed; fix conflicts and then commit the result.
```

Check Status

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master|MERGING)
$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:  plot.R

no changes added to commit (use "git add" and/or "git commit -a")
```

View the Conflict

unchanged lines

incoming marker

branch Separator

current marker

```
MINGW64:/e/rob/Documents/business/2017_ESA/esatest
# this file will plot movement with maps
x <- runif(10)
y <- runif(10)
<<<<<< HEAD
plot(x, y, type = 'l')
points(x[1], y[1], col = 'red')
=====
plot(x, y, type = 'l', main = 'This is Probably the Best Plot Ever')
>>>>> 7a1d6e8674c8bb342653668d219e1f14eed33f94
```

sha-1 from GitHub

Manually Resolve the Conflict

choose this Block →

```
MINGW64:/e/rob/Documents/business/2017_ESA/esatest
# this file will plot movement with maps
x <- runif(10)
y <- runif(10)
<<<<< HEAD
plot(x, y, type = 'l')
points(x[1], y[1], col = 'red')
```

Or this Block →

```
plot(x, y, type = 'l', main = 'This is Probably the Best Plot Ever')
>>>>> 7a1d6e8674c8bb342653668d219e1f14eed33f94
~
```

You can configure graphical merge tools like p4merge to make
this easier

Add it, Commit it, Push it

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master|MERGING)
$ git add plot.R
```

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master|MERGING)
$ git commit -m "Resolve Merge Conflict on Plot Line"
[master 713ec2a] Resolve Merge Conflict on Plot Line
```

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git pull
Already up-to-date.
```

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git push
Counting objects: 4, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 522 bytes | 0 bytes/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/robschick/esatest.git
  7a1d6e8..713ec2a  master -> master
```

Note status

Merge Res

Merge Commits Look a Bit Different

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git show head
commit 713ec2a36e4e4c6d60c2caeb02ac3de7c23c6eac (HEAD -> master, origin/master, origin/HEAD)
Merge: 4952ec1 7a1d6e8
Author: Rob Schick <robschick@gmail.com>
Date:   wed Aug 2 20:25:24 2017 -0400

    Resolve Merge Conflict on Plot Line

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
$ git log --oneline --decorate --graph
*   713ec2a (HEAD -> master, origin/master, origin/HEAD) Resolve Merge Conflict on Plot Line
|\ 
| * 7a1d6e8 Add Title To Plot
* | 4952ec1 Add a Red Point To The Line
|/
* 5dc44e6 First check-in of plot.R
* 771ea28 Initial commit

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/esatest (master)
```

Just Another Repo!

- Well, yes, but...
- I'll try a live demo to show what happens
- If it fails, following is an example of the normal bs that you need to put up with
- (But it's worth it!)

SSH to a Linux Box

- Cloned my repo from GitHub
- Created a new file
- Added it
- Pushed it:

```
[rss10@tippy2 esatest]$ git push origin master
error: The requested URL returned error: 403 Forbidden while access
ithub.com/robschick/esatest.git/info/refs
fatal: HTTP request failed
```
- ??? Stackoverflow to the rescue

Need to Use SSH not HTTP



I just got the same problem and just figured out what's cause.

618

Github seems only supports ssh way to read&write the repo, although https was 'Read&Write'.



So you need to change your repo config on your PC to ssh way:

1. edit `.git/config` file under your repo directory
2. find `url=` entry under section `[remote "origin"]`
3. change it from `url=https://MichaelDrogalis@github.com/derekermann` to `url=ssh://git@github.com/derekermann/lunch_call.git` . that is, change `@` before `@` symbol to `ssh://git`
4. Save `config` file and quit. now you could use `git push origin master` on GitHub

```
q: command not found.  
shell returned 1  
  
Press ENTER or type command to continue  
[rss10@tippy2 esatest]$ more .git/config  
[core]  
    repositoryformatversion = 0  
    filemode = true  
    bare = false  
    logallrefupdates = true
```

Need to change this

```
rss10@tippy2:esatest  
[core]  
    repositoryformatversion = 0  
    filemode = true  
    bare = false  
    logallrefupdates = true  
[remote "origin"]  
    fetch = +refs/heads/*:refs/remotes/origin/*  
    url = ssh://git@github.com/robschick/esatest.git  
[branch "master"]  
    remote = origin
```

No LUCK!

```
[rss10@tippy2 esatest]$ git push origin master
The authenticity of host 'github.com (192.30.253.113)' can't be
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.253.113' (RSA) to
n hosts.
Permission denied (publickey).
```

Need to pair the SSH keys....

<http://happygitwithr.com/ssh-keys.html>

```
rss10@tippy2:esatest
8a:b4:1d:aa:7c:0a:ad:7d:d5:04:b0:e1:88:8c:27:98 robschick@gmail.
The key's randomart image is:
+--[ RSA 4096]----+
|   o.               |
|+o o o.             |
|E.o o .             |
| o       .           |
|     . .oS          |
|   . . =.o.          |
|   . . +.o           |
| = .o              |
| . =+              |
+-----+
[rss10@tippy2 esatest]$ ssh-add ~/.ssh/id_rsa
Could not open a connection to your authentication agent.
[rss10@tippy2 esatest]$ eval "$(ssh-agent -s)"
Illegal variable name.
[rss10@tippy2 esatest]$ eval $(ssh-agent)
Illegal variable name.
```

```
[rss10@tippy2 esatest]$ git push origin master  
@oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo  
@          WARNING: UNPROTECTED PRIVATE KEY FILE!          @  
@oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo  
Permissions 0750 for '/home/rss10/.ssh/id_rsa' are too open.  
It is recommended that your private key files are NOT accessible  
This private key will be ignored.  
bad permissions: ignore key: /home/rss10/.ssh/id_rsa
```

```
[rss10@tippy2 esatest]$ chmod 600 ~/.ssh/id_rsa
[rss10@tippy2 esatest]$ git push origin master
Warning: Permanently added the RSA host key for IP address '192.30.253.112' to the l
Enter passphrase for key '/home/rss10/.ssh/id_rsa':
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 404 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
```

 [robschick](#) / [esatest](#)

[Unwatch](#) 1

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Settings](#) [Insights](#)

No description, website, or topics provided.

[Add topics](#)

 [2 commits](#)  [1 branch](#)  [0 releases](#)

Branch: [master](#) [New pull request](#) [Create new file](#) [Upload files](#) [Find](#)

 Rob Schick First check-in of plot.R Latest commit

 .gitignore	Initial commit
 README.md	Initial commit
 plot.R	First check-in of plot.R

```
rob@rob-Precision-5510: ~/Documents/business/2017_ESA/esatest
rob@rob-Precision-5510:~/Documents/business/2017_ESA$ git clone https://github.com/robsch
Cloning into 'esatest'...
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done
```

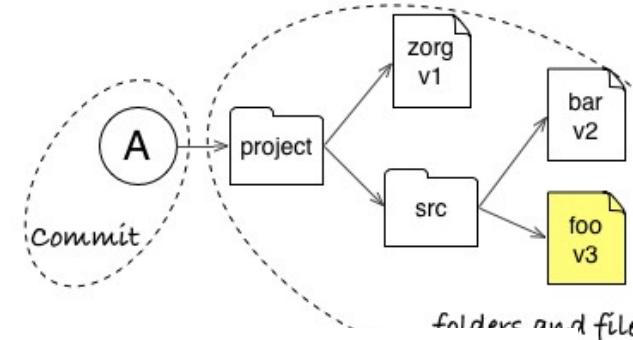
```
rob@rob-Precision-5510:~/Documents/business/2017_ESA/esatest$ .gitignore
README.md
plot.R
rob@rob-Precision-5510:~/Documents/business/2017_ESA/esatest$ On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

How Does the Repo Change?

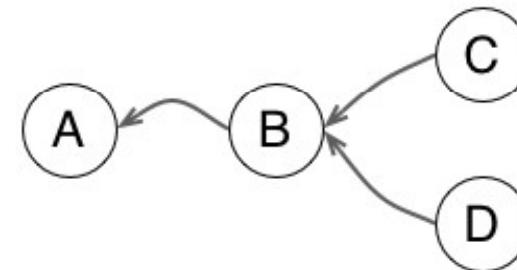
Lesson 7-4

3 Core Concepts

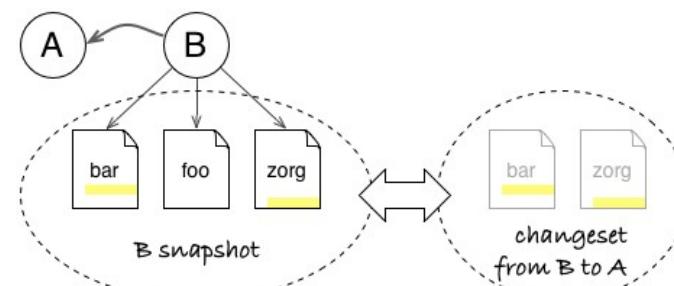
- Snapshot



- Graph



- Changeset

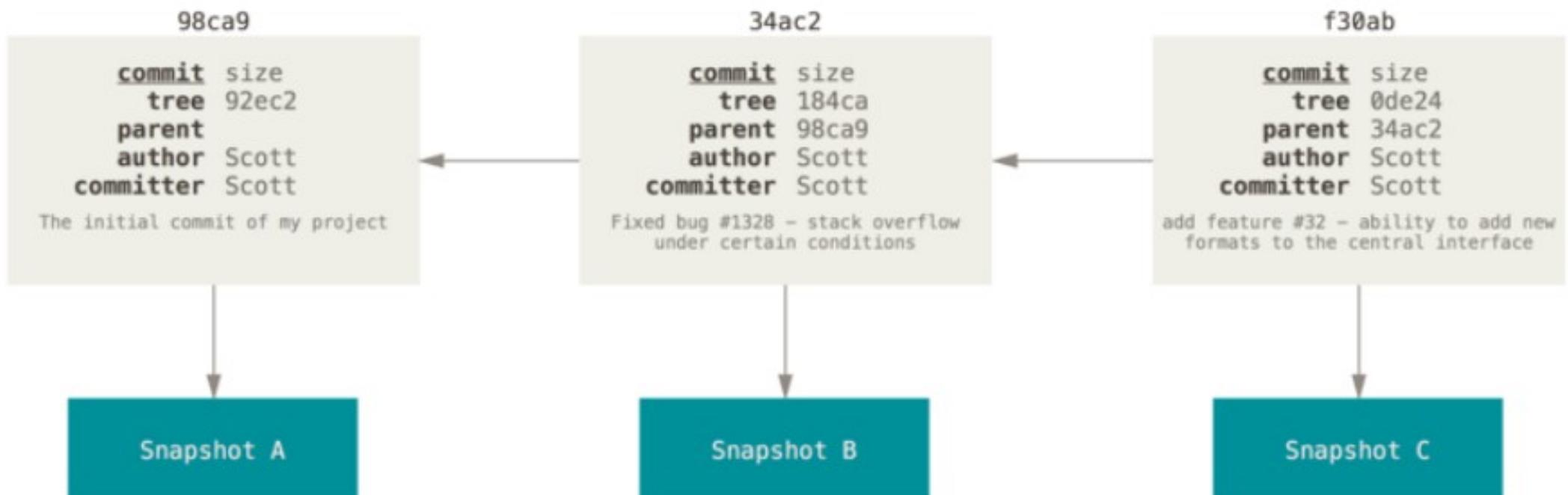


With These Concepts, We Can

- Use the sha-1 unique identifier to view a snapshot
- Compare the differences from one snapshot to a next
- Compare two different snapshots/commits
- Revert to the project at specific points (commits) in time

sha-1 Identifier – These Are Crucial

- These are the nodes along the graph
- We use them to view and/or navigate



Seeing Changes Before a Commit

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git ls-files
invertMatrix.R

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ vi invertMatrix.R

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git diff
diff --git a/invertMatrix.R b/invertMatrix.R
index d84a7d4..736e263 100644
--- a/invertMatrix.R
+++ b/invertMatrix.R
@@ -1,3 +1,4 @@
 # this will do awesome things...some day
 dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
 head(dfMat)
+dfMat[1, ]

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ |
```

alize Changes
re a commit



Seeing Changes Between Two Diffs

1 commit IDs
analyze differences
between 2 commits

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git log --oneline
7e23a0b (HEAD -> master) See First Row of Matrix
c8e3e5e Look at first 6 lines
c572b41 Add Data Matrix to the Script
354062b First Commit of invertMatrix.R

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git diff 7e23a0b c572b41
diff --git a/invertMatrix.R b/invertMatrix.R
index 736e263..fd757e0 100644
--- a/invertMatrix.R
+++ b/invertMatrix.R
@@ -1,4 +1,2 @@
 # this will do awesome things...some day
-dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
-head(dfMat)
-dfMat[1, ]
+dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
\ No newline at end of file

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ |
```

Order Matters Between Two Diffs

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git diff 7e23a0b c572b41
diff --git a/invertMatrix.R b/invertMatrix.R
index 736e263..fd757e0 100644
--- a/invertMatrix.R
+++ b/invertMatrix.R
@@ -1,4 +1,2 @@
 # this will do awesome things...some day
-dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
-head(dfMat)
-dfMat[1, ]
+dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
\\ No newline at end of file
```

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git diff c572b41 7e23a0b
diff --git a/invertMatrix.R b/invertMatrix.R
index fd757e0..736e263 100644
--- a/invertMatrix.R
+++ b/invertMatrix.R
@@ -1,2 +1,4 @@
 # this will do awesome things...some day
-dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
\\ No newline at end of file
+dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
+head(dfMat)
+dfMat[1, ]
```

Seeing Changes Between Two Diffs

- `git diff master~1 master`
- `git diff <sha-1> <sha-1>`
- `git diff` (this just shows changes made but not added)
- Why all the hassle?
 - Diffs tell you what changed
 - Commit messages tell you why
 - With both, you can easily navigate through the history of a repo

Going to a Particular Commit

- git checkout <sha-1>

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git checkout c572b41
Note: checking out 'c572b41'.
```

Going to a Particular Commit

- git checkout c572b41

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git checkout c572b41
Note: checking out 'c572b41'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:

```
git checkout -b <new-branch-name>
```

```
HEAD is now at c572b41... Add Data Matrix to the Script
```

Detached Head???

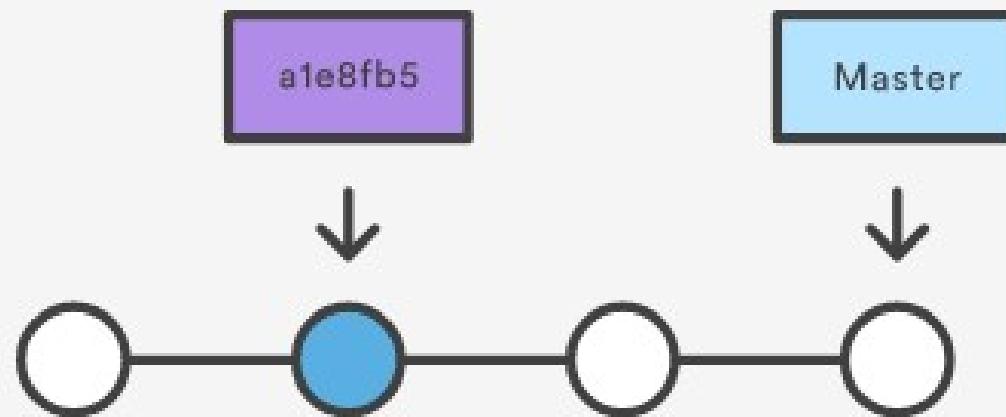
- git log --oneline --decorate

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test ((c572b41...))
$ git log --oneline --decorate
c572b41 (HEAD) Add Data Matrix to the Script
354062b First Commit of invertMatrix.R
```

- Has to do with where the branch is pointing – now it's to a previous commit
- Explore your script and you'll see it from that state:

Where are we?

Checking out a previous commit



Detached Head???

- git log --oneline --decorate

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test ((c572b41...))  
$ git log --oneline --decorate  
c572b41 (HEAD) Add Data Matrix to the Script  
354062b First Commit of invertMatrix.R
```

Where did the other commits go?

The Commits Are Still There - Phew!

- git checkout master

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test ((c572b41...))
$ git checkout master
Previous HEAD position was c572b41... Add Data Matrix to the Script
Switched to branch 'master'

rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test (master)
$ git log --oneline --decorate
7e23a0b (HEAD -> master) See First Row of Matrix
c8e3e5e Look at first 6 lines
c572b41 Add Data Matrix to the Script
354062b First Commit of invertMatrix.R
```

Why Would You Want to Do This?

- Allows you to work with the file as it was at that time:

```
rob@rob-win7-guest MINGW64 /e/rob/Documents/business/2017_ESA/test ((c572b41...)  
$ cat invertMatrix.R  
# this will do awesome things...some day  
dfMat <- matrix(data = runif(n = 25), nrow = 5, ncol = 5)
```

- And then bring those changes into the current commit
- I'll argue for using a branch to do this in the last section

Backing Out Changes

- This can get complicated!
- Three main commands:
 - git reset (hard, soft, mixed)
 - git rebase
 - git revert
 - git cherry-pick
- This diagram helps:

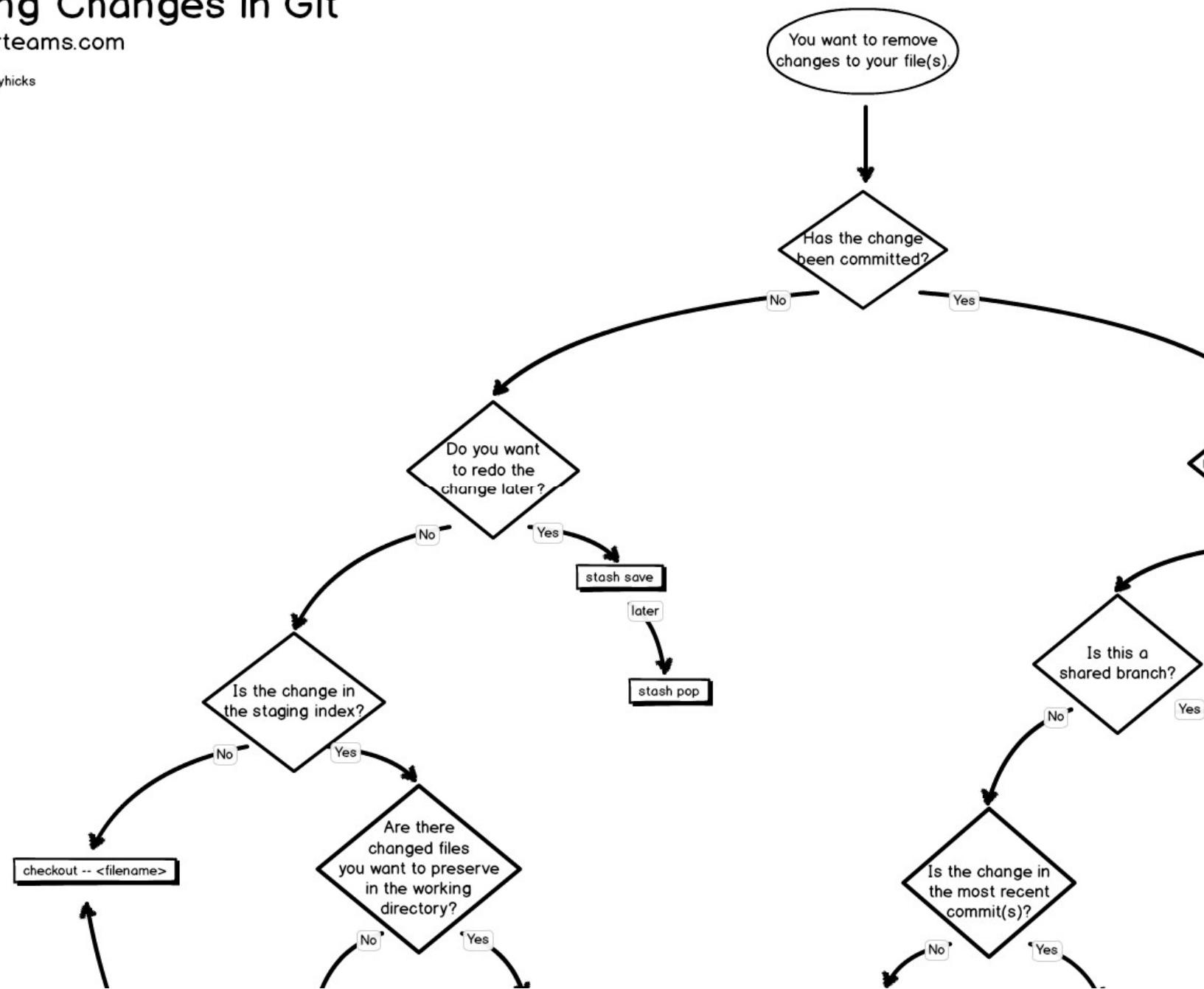
Undoing Changes in Git

www.gitforteams.com

Author: emmajane

Contributors: JameyHicks

License: CC-BY



More Resources

- <https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things>
- <https://www.atlassian.com/git/tutorials/undoing-changes>
- <https://www.atlassian.com/git/tutorials/rewriting-history>