Version Control and GitHub

Marco Morales
mam2519@columbia.edu

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2018
Columbia University

RECAP: A Data Science Project

- Three **aims** of a data science project
  a) **reproducibility**
     - anyone should be able to arrive to your **same results**
  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine
- crucial tenets for **collaborative work**
  c) **scalability**
     - your project should also work for **larger data sets** and/or be on the path of **automation**

Version control (and Git)

though this be madness...

- **version control** allows you to keep track of changes/progress in your code
  - keeps "**snapshots**" of your code over time
  - helpful to **debug**, and to enhance **reproducibility**
  - also great for **team collaboration** (everyone can see who changed what!)
- **Git** is a version control software
- **GitHub** is an online Git repository (on steroids)
  - widely used by data scientists (and in academia)
  - not (strictly) a "software development" tool

Version control (and Git)

...yet there is method in't!

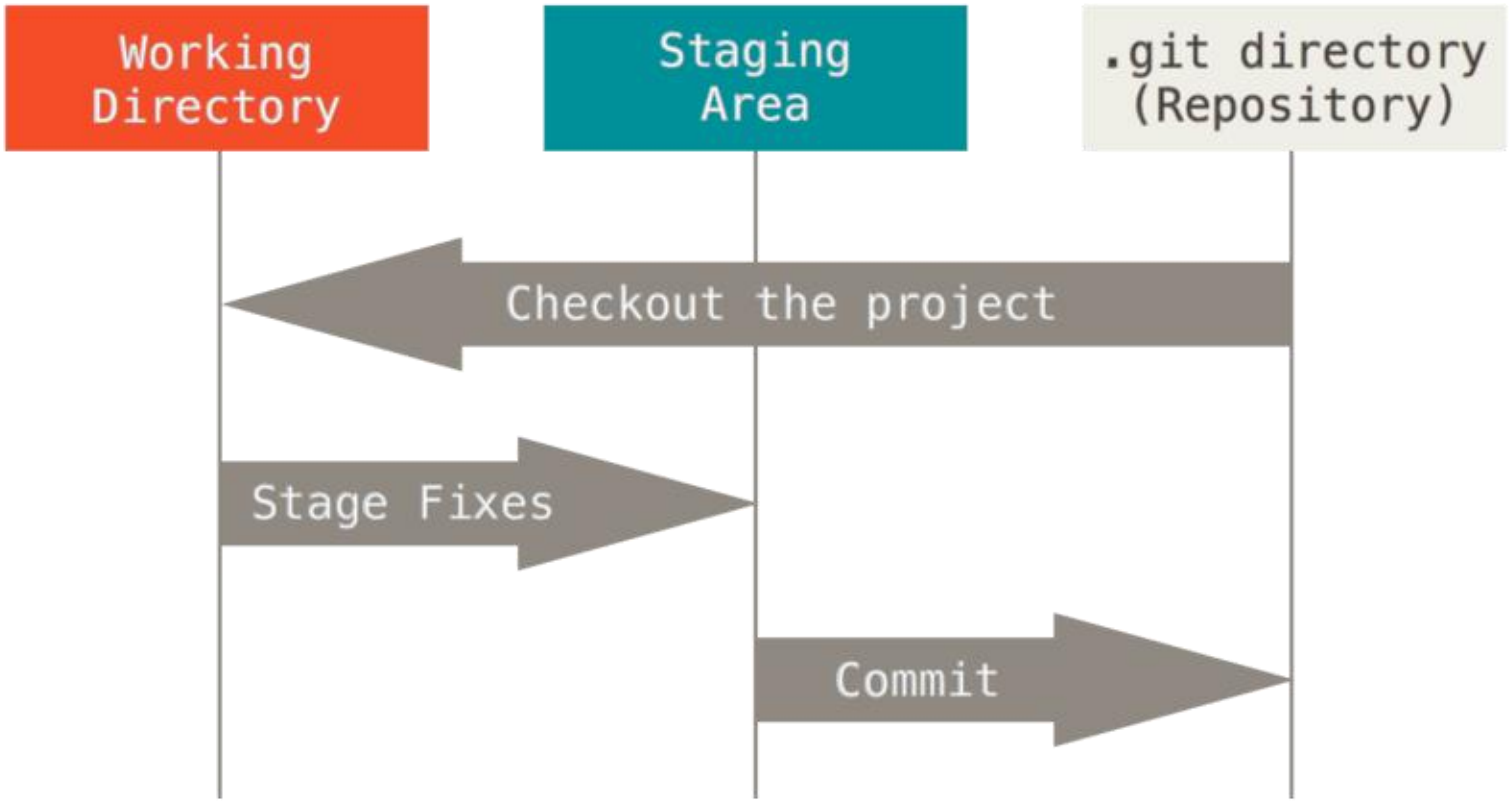

Figure: Pro Git, 2nd Edition

Version control (and Git)

...once you've installed Git

- set your **user name** and **email address**

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

- verify that information was successfully entered

```
$ git config --list
```

- every Git commit uses this information, and it's baked into your commits

Version control (and Git)

...now create a GitHub repo!

Create a new repository

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

Owner: marco-morales / Repository name: testrepo

Great repository names are short and memorable. Need inspiration? How about **friendly-octo-guide**.

Description (optional)

a test repository

- 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: None | Add a license: None

Create repository

...and remember the location!

Version control (and Git)

preamble to initialize a repo...

- there are **files you never want tracked** by Git (e.g. log files, access keys), even by mistake
- go to the root of your local repository and create a `.gitignore` file

```
$ touch .gitignore
```

- add files you want ignored (e.g. `.DS_Store` files) to the `.gitignore` file

```
$ echo ".DS_Store" » .gitignore
```

- add `.gitignore` to staging

```
$ git add .gitignore
```

- further info/templates: https://github.com/github/gitignore

Version control (and Git)

initializing a repo...

- go to the **root** of your project and **initialize** the repo

```
$ git init
```

- set the remote master repo (your new GitHub repo)

```
$ git remote add origin
https://github.com/marco-morales/testrepo.git
```

Version control (and Git)

making version control work for your repo...

- indicate a file to be tracked by Git

```
$ git add samplefile.R
```

- verify what's being tracked

```
$ git status
```

- commit your tracked files

```
$ git commit -m "initial files committed"
```

- send files to your master repo

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

Version control (and Git)

a few confusing things...

- a file will be committed *exactly* as it was when you `git add`-ed it
- if you change it *after* you `git add` it, and want to commit the new changes, you need to `git add` again before the `git commit`
- use `git status` to assess what's being staged and will be commited

Version control (and Git)

...yet there is method in't! (RECAP)

- basic actions to master in Git
  - `git init`: initializes Git, and indicates that the folder should be tracked
  - `git add`: brings new files to the attention of Git to be tracked as well
  - `git commit`: takes a snapshot of alerted files
  - `git push`: sends changes in your local file to the GitHub repository

Version control (and Git)

...project collaboration

- essential Git concepts to keep in mind
  - **clone;** a local copy of a repository that can be updated as changes happen
  - **fork**; a fork is a thread a repository.
  - **pull;** brings changes into master repository
  - **branch**; a local mirror copy of a repository at a given point in time

Version control (and Git)

...basic mode of project collaboration

- add **collaborators** to your repo
  - as a repo **owner** you have control over what gets changed
  - **collaborators** will be able to `push` to the repo

a) **Collaborators:**

- work on a branch on the repo and create code
- send a `pull` request to add that code to the master repo

b) **Owner:**

- comment on the `pull` request
- accept the `pull` request and/or `merge` the code

Version control (and Git)

...a quick demo exercise

- clone somebody else's remote repo

```
$ git clone
https://github.com/marco-morales/testrepo.git
```

- (checkout and) create a branch

```
$ git checkout -b moretesting
```

- make a change in your code file
- go on, verify that git is tracking the change

```
$ git status
```

Version control (and Git)

...a quick demo exercise

- do your usual git routine

```
$ git add testfile.R
```

```
$ git commit -m "adding hubris to the code"
```

- now, you'll create a pull request

```
$ git push origin moretesting
```

- may the magic begin!

Version Control and GitHub

Marco Morales
mam2519@columbia.edu

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2018
Columbia University