Welcome to the Wonderful World of `git`

Simon Goring

23/10/2020

Working a Little `git` at a Time

- `git` is version control
- Each change is recorded, all past versions are retained
- One version always works (more or less)

What People Really Think

> "`git` really changed the way I write code. I stopped making a whole bunch of huge changes, and started working on one thing at a time. It's made fixing my code a lot easier, and I don't run into problems as often."

Simon Goring (when I'm feeling optimistic)

Version Control

Basics of Version Control

- Changes to files are important and done with purpose
- We record changing files and the reason for these changes
- Changes may not always be wanted; we want to undo unwanted changes
- We want a clear way to collaborate and understand when changes conflict.

Basics of Version Control - Vocabulary

- A project that is managed with version control is called a **repository**
- Each snapshot of the repository at any one point is called a **commit**

- A **repository** can have multiple versions of the same code, called **branches**
- There is generally a **main** branch (see GitHub's explanation for deprecating *master*)
- New features often get their own **branch** until they are ready to be added to **main**

Basics of Version Control - Tools

- `git` is the most common version control system
- `git` is a command line utility
  - A number of GUIs work with `git`
- `git` watches files in your **repository**, it can also ignore certain files
  - `.gitignore` files are language/project specific; a long list of `.gitignore` files

---

Tool for Working with `git`

- GitKraken (free/paid, Proprietary)
- GitHub Desktop (free, MIT)
- `git` GUI (comes with `git`)
- The `git` list of platform specific tools
- RStudio, Visual Studio Code functions.

Using `git`

Starting from Scratch (with RStudio)

Check your version.

```
simon@partyLaptop:~$ git --version
git version 2.25.1
simon@partyLaptop:~$
```

Create New Project (RStudio)

```
simon@partyLaptop:~$ git init
```

`git` Workflow

- Initialize (set up the infrastructure)
- Edit
- Stage the changes before we commit the changes
- Commit the changes to `git`

Local `git` Workflow

- `git init` - Initialize a repository
- Make changes (atomic changes)
- `git add` to **stage** changes
- `git commit` to **commit** changes

Using RStudio

GitHub & Online Repositories

Public Code Repositories

- Support greater collaboration
- Acts as cloud backup
- Provides secondary services (GitHub Pages, Project boards, Issue trackers)

Range of Options

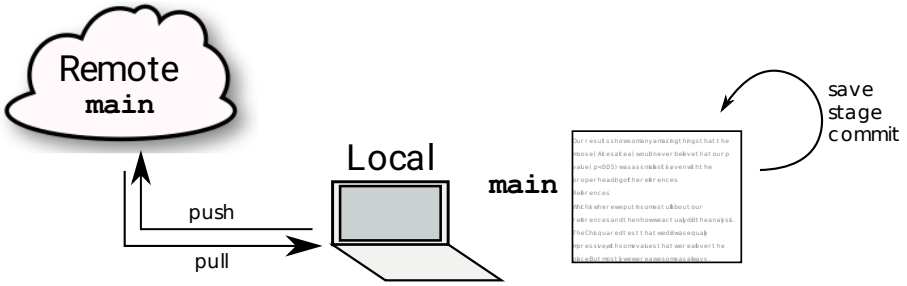
- A large number of options exist across service levels & cost
- Wikipedia provides a comparison table of Source Code Hosting platforms
- We will focus on Github because of SJG's familiarity with the platform

---

A Repository for Home and Away - Naming It

- Your **remote** repository is the `main` version and the `origin`
- **remote** is the backup, the collaboration hub & the authority

- **local** is where you work on things



A Repository for Home and Away - Getting It

- You can **fork** someone else's repository to your account
- You can **clone** someone else's repository (or your own) to your local computer
- You can **create** a new repository

Fork and Clone

- http://github.com/throughput-ec/ThesisIsCode

Fork and Clone (Active)

- Create new Project in RStudio "From Version Control"
- All files copied locally. Fun times!!

What Makes a Good Repository

- Active work
- Explore Github
- Fill in a couple lines here: http://bit.ly/githubrepos

What Makes a Good Repository (Discussion)

- Active discussion