Data Management for Reproducable Research

Thomas R. Cook

23-OCT 2015

Long-term reproducability and Mysterious Data:

- Common Scenario:
- Get novel data file
    - Make some changes to it
    - Save over original file

Long-term reproducability and Mysterious Data:

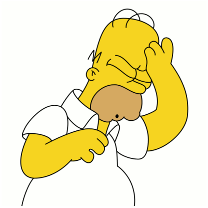
- Common Scenario:
- Get novel data file
  - Make some changes to it
  - Save over original file

Six months later…

- Return to project…
  - What does `log_inerv_1234.b` mean? How did I get it? Why is it driving my results?
- Even worse if someone asks for your replication data
  - You need to be able to explain how you arrived at a given variable/model/etc
- Moar worse if your co-author created `log_inerv_1234.b`

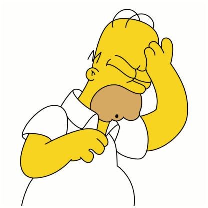
Six months later. . .

- Return to project. . .
  - What does `log_inerv_1234.b` mean? How did I get it? Why is it driving my results?
- Even worse if someone asks for your replication data
  - You need to be able to explain how you arrived at a given variable/model/etc
- Moar worse if your co-author created `log_inerv_1234.b`



Six months later. . .

- Return to project. . .
  - What does `log_inerv_1234.b` mean? How did I get it? Why is it driving my results?
- Even worse if someone asks for your replication data
  - You need to be able to explain how you arrived at a given variable/model/etc
- Moar worse if your co-author created `log_inerv_1234.b`



- Make R script, DO file or other script that generates data

Two big challenges with managing data

1. Track file changes over tiem
    - Long-term reproducibility
    - Version management
2. Collaboration with others

Common Solutions:

5. Edit data in-place (!)
6. Dropbox
7. Track Changes/time-machine
8. Email
9. New folder per version

That's a start. . .

But what about these other nightmare scenarios:

- Someone asks for old version of replication data

That's a start. . .

But what about these other nightmare scenarios:

- Someone asks for old version of replication data
- Coauthor deletes files in his/her dropbox folder

That's a start...

But what about these other nightmare scenarios:

- Someone asks for old version of replication data
- Coauthor deletes files in his/her dropbox folder
- You and co-author try to work on data at the same time (the dreaded `conflited copy`)

That's a start. . .

But what about these other nightmare scenarios:

- Someone asks for old version of replication data
- Coauthor deletes files in his/her dropbox folder
- You and co-author try to work on data at the same time (the dreaded `conflited copy`)
- Need to identify how project today differs from version 6 months ago

That's a start. . .

But what about these other nightmare scenarios:

- Someone asks for old version of replication data
- Coauthor deletes files in his/her dropbox folder
- You and co-author try to work on data at the same time (the dreaded `conflited copy`)
- Need to identify how project today differs from version 6 months ago
- Need to identify prior/abandoned approaches to analysis

That's a start. . .

But what about these other nightmare scenarios:

- Someone asks for old version of replication data
- Coauthor deletes files in his/her dropbox folder
- You and co-author try to work on data at the same time (the dreaded `conflited copy`)
- Need to identify how project today differs from version 6 months ago
- Need to identify prior/abandoned approaches to analysis
- The list goes on. . .

That's a start...

But what about these other nightmare scenarios:

- Someone asks for old version of replication data
- Coauthor deletes files in his/her dropbox folder
- You and co-author try to work on data at the same time (the dreaded `conflited copy`)
- Need to identify how project today differs from version 6 months ago
- Need to identify prior/abandoned approaches to analysis
- The list goes on...
- Git can help resolve all of these

Git is. . .

- Distributed Version Control System

Git is. . .

- Distributed Version Control System
    - Distributed $\rightarrow$ decentralized, many users

Git is. . .

- Distributed Version Control System
  - Distributed → decentralized, many users
  - Version Control → track changes, enable rollbacks,

Git is. . .

- Distributed Version Control System
    - Distributed → decentralized, many users
    - Version Control → track changes, enable rollbacks,
    - System → System

Git is. . .

- Distributed Version Control System
    - Distributed → decentralized, many users
    - Version Control → track changes, enable rollbacks,
    - System → System
- Think: “Track changes” on steroids

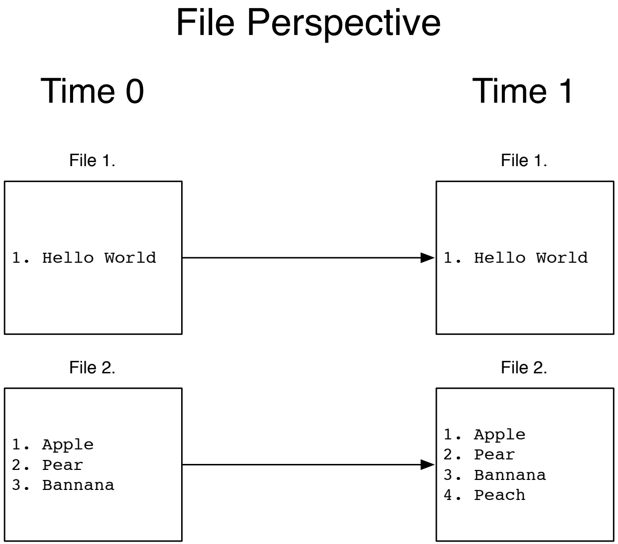
Repos

- Git tracks sets of files – multiple files at once
- Folder with set of files tracked by Git: Repository
    - Generally, a Git repo looks and works just like a folder
- Think: Repo project

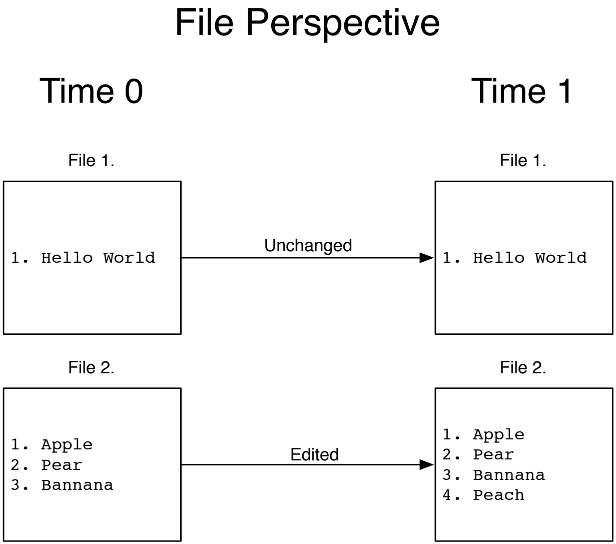
Commits

- A snapshot of (specified) files tracked by Git
    - Captures *changes* in specified files (since last commit)

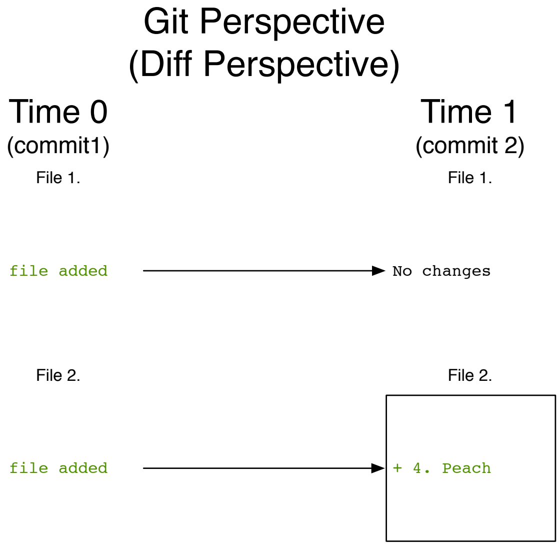
Commits


File Perspective
Time 0
Time 1
File 1.
1. Hello World
File 1.
1. Hello World
File 2.
1. Apple
2. Pear
3. Bannana
File 2.
1. Apple
2. Pear
3. Bannana
4. Peach

Commits


File Perspective
Time 0
Time 1
File 1.
1. Hello World
Unchanged
File 1.
1. Hello World
File 2.
1. Apple
2. Pear
3. Bannana
Edited
File 2.
1. Apple
2. Pear
3. Bannana
4. Peach

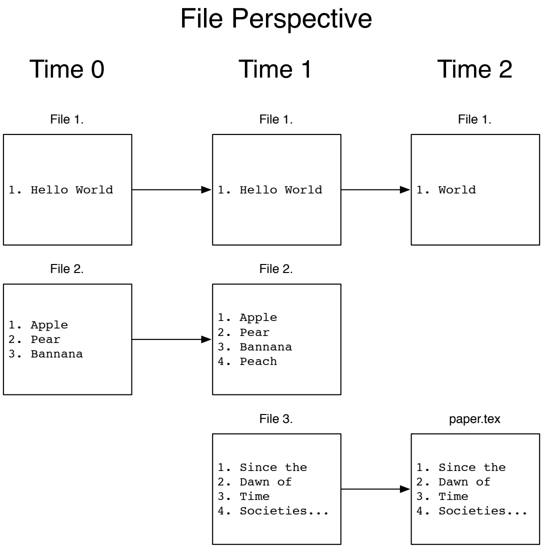
Commits


Git Perspective
(Diff Perspective)
Time 0
(commit1)
Time 1
(commit 2)
File 1.
file added
No changes
File 1.
File 2.
file added
File 2.
+ 4. Peach

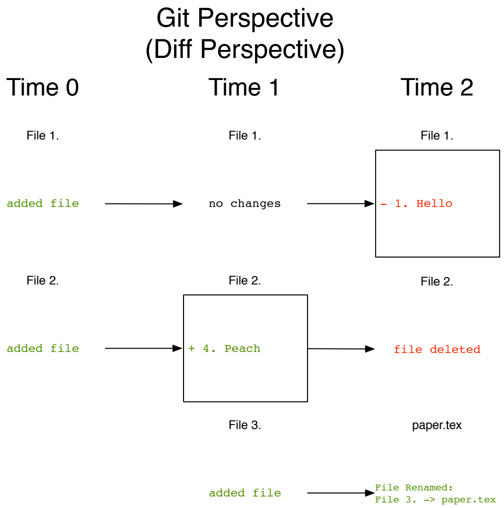
Commits

- A snapshot of (specified) files tracked by Git
    - Captures *changes* in specified files (since last commit)
    - Captures Files Added/Removed/Moved

Commits


File Perspective
Time 0
Time 1
Time 2
File 1.
1. Hello World
File 1.
1. Hello World
File 1.
1. World
File 2.
1. Apple
2. Pear
3. Bannana
File 2.
1. Apple
2. Pear
3. Bannana
4. Peach
File 3.
1. Since the
2. Dawn of
3. Time
4. Societies...
paper.tex
1. Since the
2. Dawn of
3. Time
4. Societies...

Commits


Git Perspective
(Diff Perspective)
Time 0
Time 1
Time 2
File 1.
added file
File 1.
no changes
File 1.
- 1. Hello
File 2.
added file
File 2.
+ 4. Peach
File 2.
file deleted
File 3.
added file
paper.tex
File Renamed:
File 3. -> paper.tex

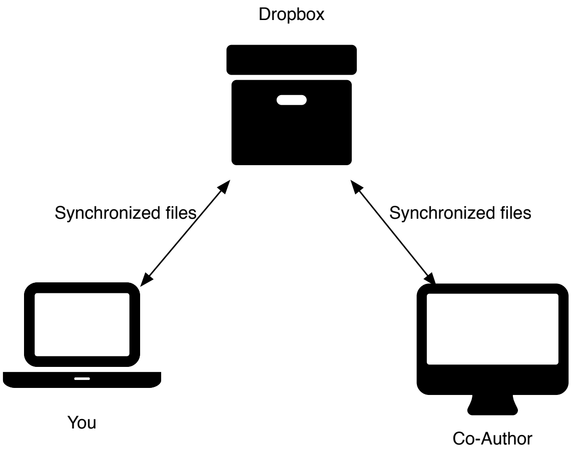
Commits

- A snapshot of (specified) files tracked by Git
  - Upshot: Can track file changes very closely over time

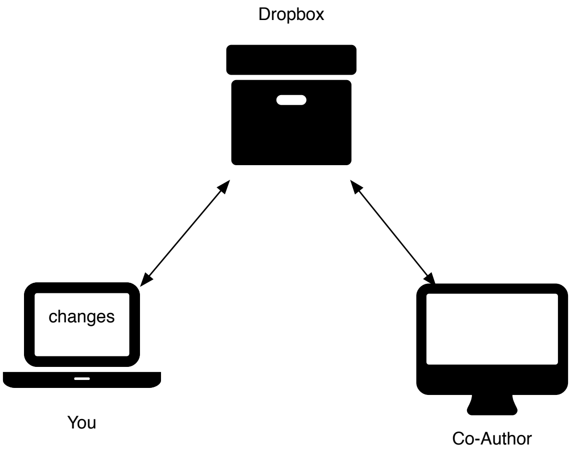
Distribution/Collaboration

- Git enables Collaboration – it is a distributed system.
    - Contrast to Dropbox

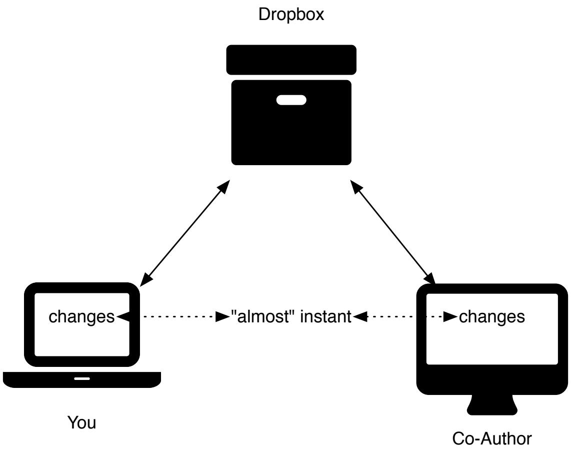
Distribution/Collaboration



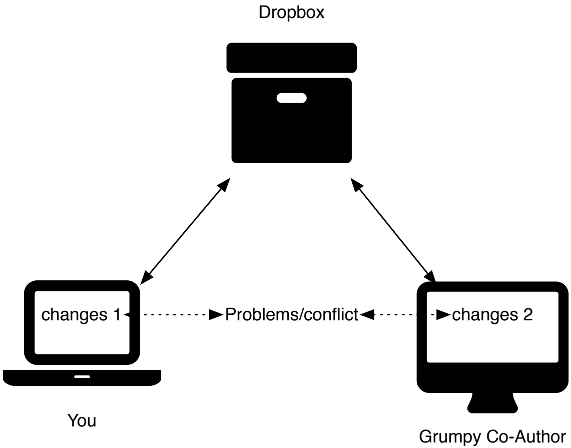
Distribution/Collaboration



Distribution/Collaboration



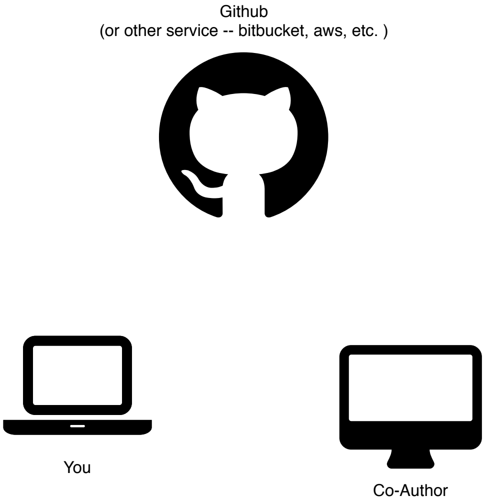
Distribution/Collaboration



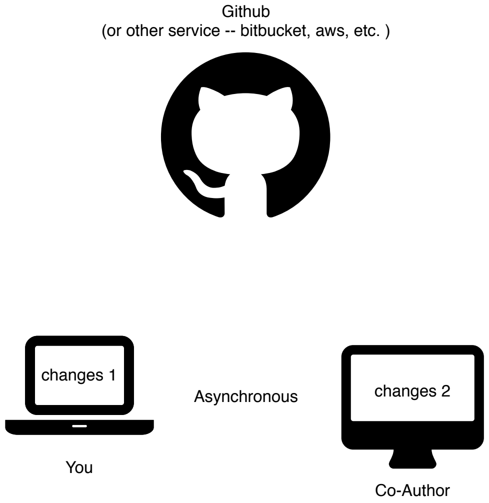
Distribution/Collaboration

- Git enables Collaboration – it is a distributed system.
    - Download repo to local computer
    - Make changes and commit
    - Push changes to server when ready
    - Pull changes from server when ready

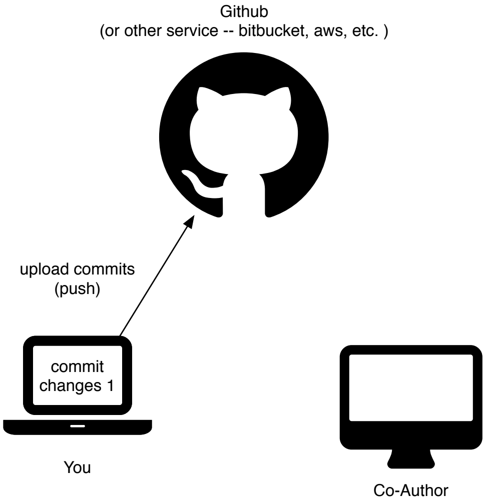
Distribution/Collaboration



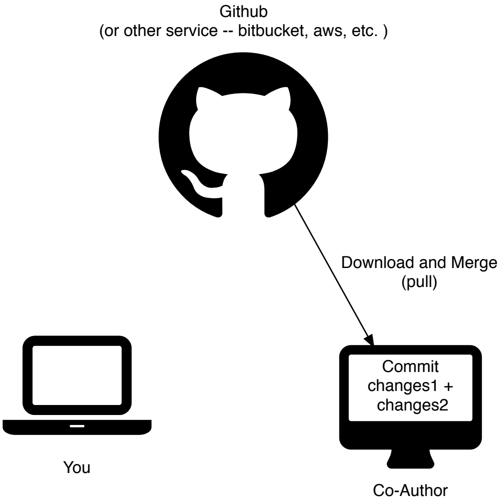
Distribution/Collaboration



Distribution/Collaboration



Distribution/Collaboration



Other things Git does

- Roll-back to previous versions
- Branch development/management
- Integration in to lots of software
- Best way to explore: start using git

Today

- Sourcetree
    - Setup Repo
    - Clone Repo
    - Checkout
    - Commit
    - Pull

Where to get help

- Easy help
  - Lots of places
  - Stackoverflow.com
  - Google
  - Github youtube channel
  - Sourcetree help
- Punching deck and interactive learning:
  - `try.github.io`
  - www.codeschool.com/courses/git-real
  - A great course at lynda.com www.lynda.com/Git-tutorials/Git-Essential-Training/100222-2.html
- Deep Dive
  - pro-git book by Scott Chacon and Ben Straub. – Free online `http://git-scm.com/book/en/v2`

Things Git is bad at

- Tracking binary files – word files, images, etc. It will track them, but it's not ideal

Merge Conflicts

- Git is good at fixing conflicts
- When it can't you need to fix them
- Diff, resolve using 'mine'/'theirs'

Problem:

- Software versions change over time

That’s actually sort of a hard question to answer

- Virtualization software, but not totally
- But totally if on pc/mac