

# Git Tutorial

Sandra Viknander  
January 2021



# Overview

- version control
- basic git (command line)
- GitHub (Remote repository)
- exercises!
- Ask whenever confused

# Perfect Reproducibility

I have:

$$\text{results} = \text{program}(\text{data}, \text{parameters})$$

And data never changes.

**Question:** What do you need from me to get the same results?

# Perfect Reproducibility

`results = program(data, parameters)`

A given set of **results** is **determined uniquely** by

- the program **code**
- parameter **values**

Source **data** should never be altered.

# Real World

- code is very fluid
- results reflect intermediate (“slightly different”) code version
- data not available
- param values not given
- software versions not given

# Show of hands

1. If you use Google Drive, Dropbox, or Box
2. If you use any backup software/method for your personal computer

# Version Control

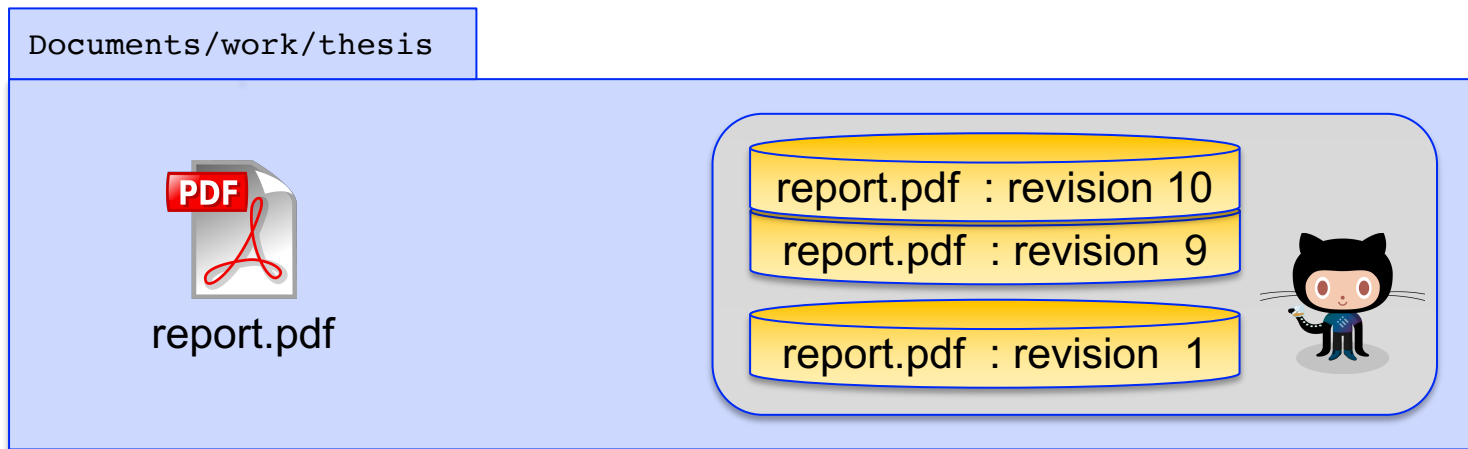
- **Issue:** Files with long, complicated history.  
Want to keep different versions:

```
Report_v3_comments_2018_01_05.docx  
experiment_pipeline_10_2017_11_05.sh
```

- **Compound issue:** Other people work on them too
- Programs like **git** (version control systems) keep track of changes made by different people

# Git Concepts

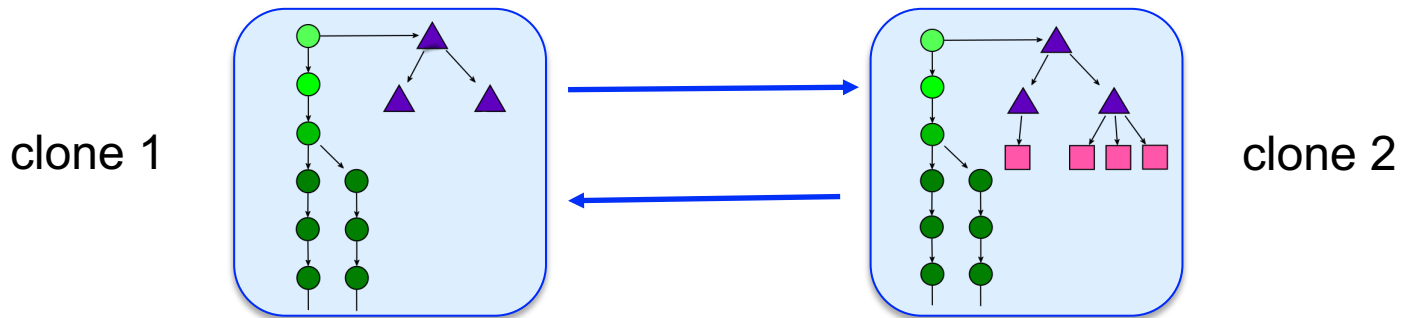
- a git project is called a **repository** or **repo** = directory with history
- a repo contains a collection of snapshots (called **revisions**) of the directory:





# Git Concepts

- revisions are connected in **branches**, reflecting file evolution



Original image © [Bunyk](#) / [Wikimedia Commons](#) / [CC-BY-SA-4.0](#)

- repos are *decentralized*
  - Each **clone** contains everything (all revisions + history)
  - Changes can be passed between clones

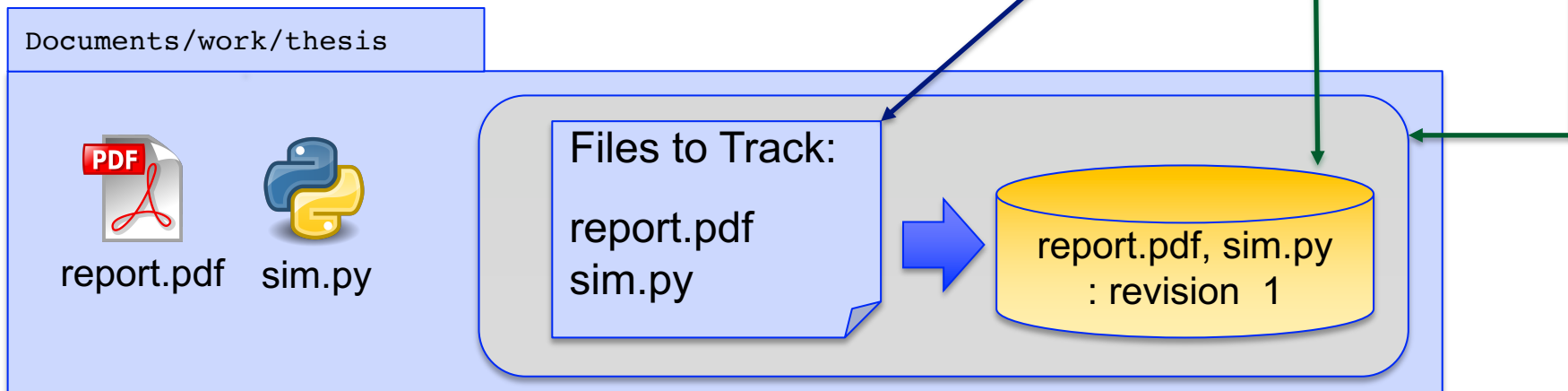
# Creating a Repo and Recording Changes

- 0) Initializing the repo inside your project directory
- 1) Instruct git to start tracking files
- 2) Commit list of files-to-track into a revision

`git init`

`git add`

`git commit`



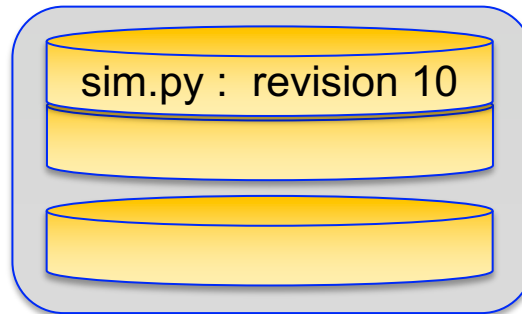
# Exercise 1

- **10-15 minutes**
- Go to `https://mpbio-bbt015.github.io/`
- If you need to, read “How to connect to remote accounts”
- Notes are good-to-know info only

# Making and Committing Changes

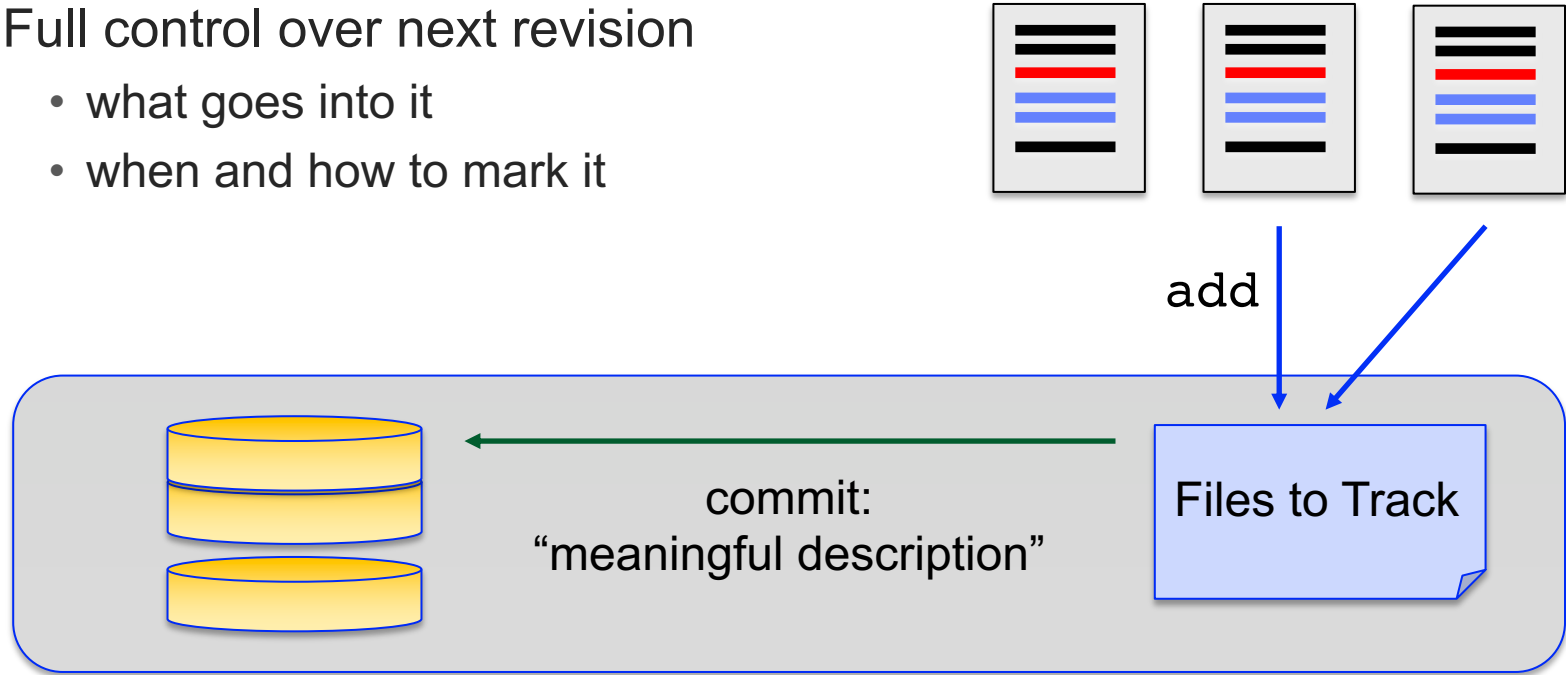
- Git reports what changed since latest revision: `git status`
- Differences can be inspected: `git diff`

(current file)



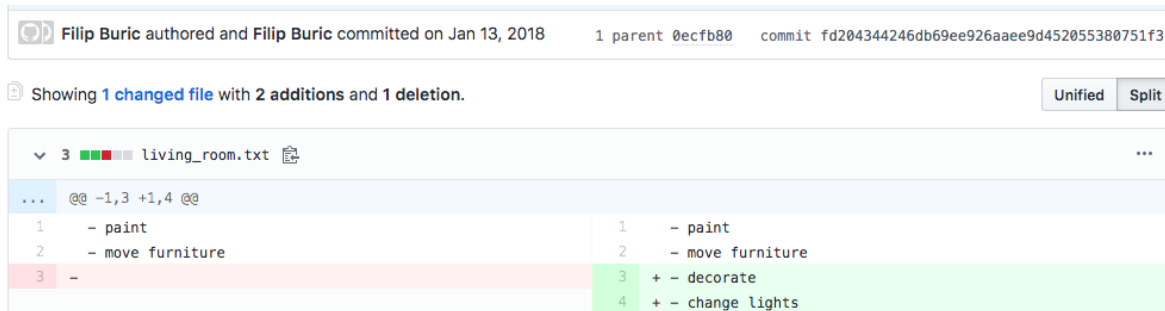
# Making and Committing Changes

- Full control over next revision
  - what goes into it
  - when and how to mark it



# Viewing Differences

- Web (GitHub) or GitHub Desktop (= graphical frontends to the git program)
  - Easier to use
  - Limited functionality
- Command line:
  - More cumbersome
  - Far more flexible
  - Always available



```
buric@C17LQHT [15:39] : apartment_2018 $ git diff fd2043~ fd2043 living_room.txt
diff --git a/living_room.txt b/living_room.txt
index 6b76c07..6efbd9d 100644
--- a/living_room.txt
+++ b/living_room.txt
@@ -1,3 +1,4 @@
- paint
- move furniture
-
+ - decorate
+ - change lights
```

# Git is meant for text files

- Tool for tracking source code (= text / “low level”)
- Can track any type of file BUT can’t see diffs (without extra plugins)

report.md

Markdown: text, readable by any program



report.pdf

PDF: “binary” format, encodes graphics, needs dedicated decoder (Adobe, etc)

```
33
40
41 - ## Number of peptides per protein identifications
42
43
44 - ```{r}
45 percolator_concat_hi_conf %>%
46   dplyr::rename(protein_id = `protein id`) %>%
47   dplyr::group_by(protein_id) %>% |
48   dplyr::distinct(`sequence`) %>%
49   mutate(n_pep = n()) %>% ungroup() %>%
50
51   ggplot(aes(x = n_pep)) +
52     geom_histogram(binwidth = 1) + xlim(0, 20) +
53     xlab('peptides / proteins')
54   ```
```

(only need to track this)

```

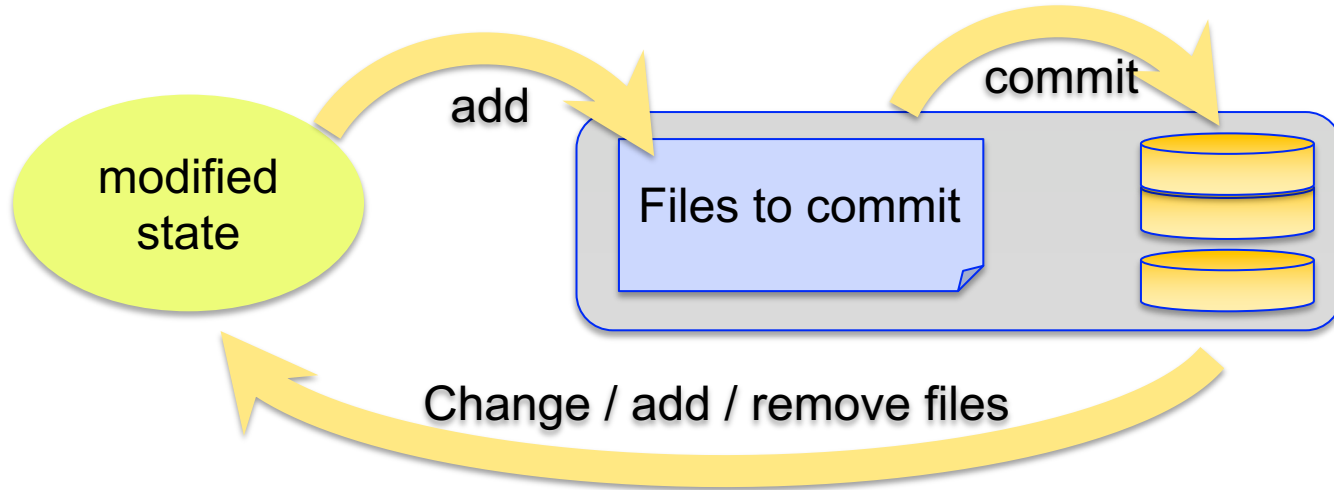
DEd<0x00><0x00><0x00>Æ0b~üyV0ç~%t'Äbhüü67L0*tÉR555aâ-0x01><0x00><0x00><0x00>e
0Eð<0x16>ââ<0x0f>Lzû=<0x8f>0cH$<0x12>6<0x1e><0x00><0x00><0x00>F_||üðE-YVÜ-EzÄ"öööeC
<0x0f>{zN<0x9d><0x00><0x13><0x1f>63<0x15>6<0x1e><0x00><0x00><0x00>F5V|Xx8#cKqñ0<0x9c
Ä11im<0x19><0x19>Yµñ-<0x19>Z'..0x7f-ÖNGY-üé"0a2<0x01><0x00><0x00><0x00><0x00><0x00>pwQ"
ââ+WR<0x0b><0x00><0x00>€<0x01>60022i0j0mY<0x7f>PEQ<0x19>-1<0x00><0x00><0x00><0x00>
ZZAAd<0x03><0x00><0x00>A0000"µyë00.<0x00>y1éââðâ<0x8f>705SpbbAd<0x03><0x00><0x00>

<0x00><0x00><0x00>âAwKv00=WyN0"gx0000ésÿ-ðÉ'j]l.É-0x0f><0x00><0x00><0x00><0x00>€±C
<0x0d>:â<0x17>"6=fff+L6<0x00><0x00><0x00><0x00>Ä0•JNN000U00Y"PU~L"0.o.<0x03>ü"vyy333u&h
<0x1d><0x1d>µ<0x9d><0x00>ðj0":<0x02>Éfll$<0x12>YXEär&D"ñ±',"É0<0x12><0x00><0x00><0x00>µ
E0E00+±=
_0i=EC0000p&jkëimInoWµv.µð"ñv.ñ0hµ#h3<0x1b><0x1b><0x1b><0x1b>///goi<0x13>6L"0â
<0x00><0x00><0x00>cSV<no"lk+ok+oo"hk+i+Ctt00j0w1"L6•É$*üp'A001É'"70147?{A²a<0x18>
<0x00><0x00><0x00>E-~"ie"lm-mm-mk-kk+kk+ie"Ne"<0x02>+*HAvv-yv-ZZV}<0x0f><0x1c><0x00>
UMMGccgMVG}g}<0x9d>±ññE±±±ZNV<0x11>0000<0x0f><0x00><0x00><0x00><0x00><0x00>

â0<0x1e><0x14><0x14>D{<0x05><0x00><0x00><0x00><0x18>"(É<0x02><0x00><0x00><0x00>âÉ¹y0|M=
D<0x01><0x00><0x00>Eñ<0x8d>ç<0x0c><0x00><0x00>Äz)..~ÿ+•m0000e'IA#vE8SDC{~'
-
<0x00><0x00><0x00>#<0x81>FEjjEon.hjêojëoj*hn.ifu<0x1a>éxIV<0x12>wwlWw;Ww[77;77;<0x1
hü(<0x16>{i>/âüçâ<0x12><0x12>|„It±A"x0+++U*+U*+U*+U*UEEU*U"É~#YR"læiü<0x1f><0x16
16jü"}|<0x12>„I<0x00><0x00><0x00>€>EYR45â74â45â75â56æµµ<0x19>c<0x18>'Hällä1äinää8w
<0x00><0x00><0x00><0x00>ëIU66æ64ÜhlikhEnliko"<0x1c>08"-É<0x1c><0x1c><0x1c><0x1c>|âet7'x
âYKKWaqK8WQ0s0QKQ0sK80.g0<0x9d><0x9d><0x9d>####6b<0x8d>Ixü{<0x00><0x00><0x00><0x00><0x00>
4#<0x10>KBC-ÄÄ=Iâ55E1<0x00><0x00><0x00><0x10>^<0x1a><0x00><0x00>±1$0.±0.€±pzwëpYÜE1

```

# Typical Work Loop





# Pop Quiz!

(yaaay...)

- Go to **socrative.com** > **Student Login**
- Room number: **BBT045**

# Exercise 2

- **15 minutes**

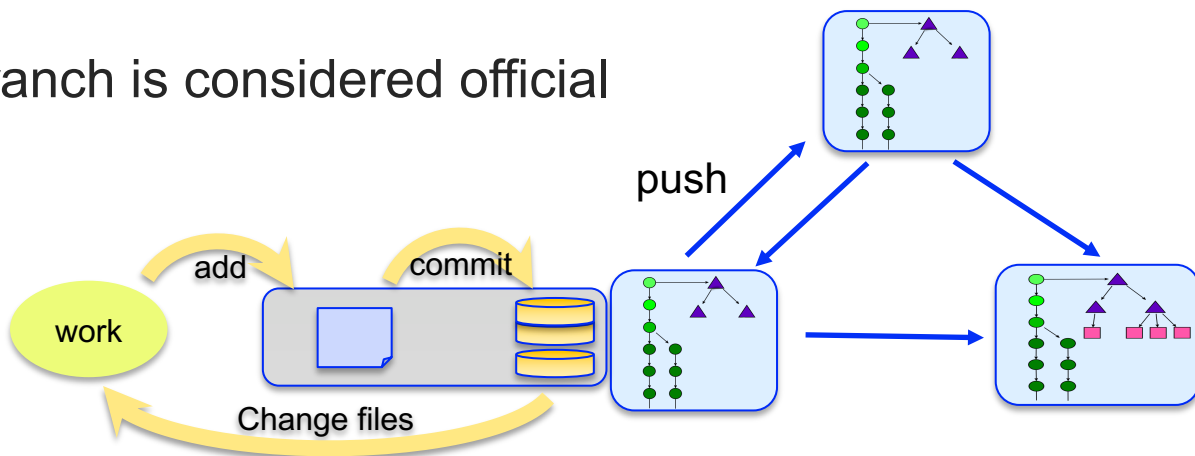
# Some real examples

## Using GitHub for your homework

- Set up your directories
  - HW1, HW2, HW3, HW4
  - Init repo such that all homework folders are in it.
- Set remote
- `push -u origin master (main)`

# Collaborating

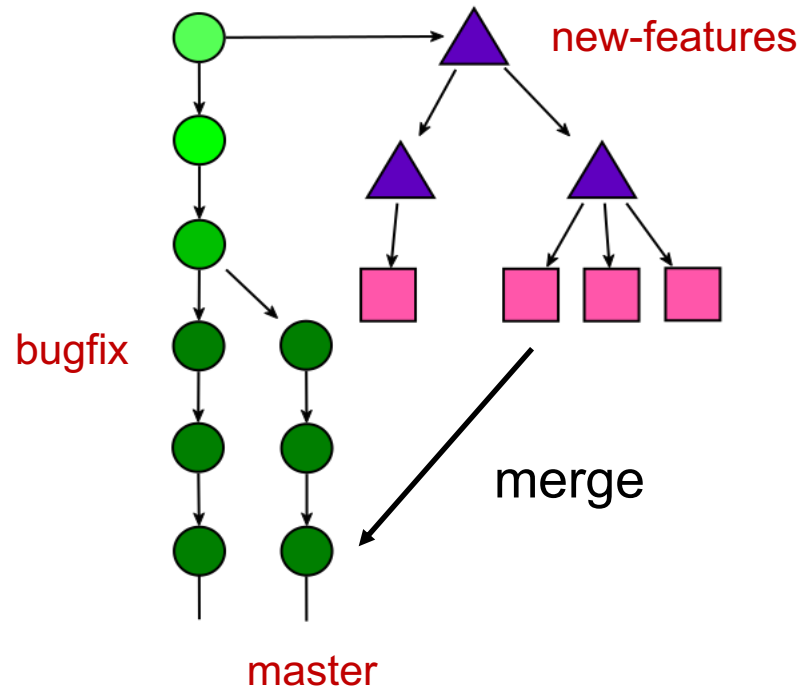
- Convention:  
one repo and one branch is considered official



- Collaborators:
  - clone from this repo
  - work
  - **push** their contributions to it

# Collaborating

- Work usually done on branches:
  - maintain separation of interest  
(e.g. "development" vs "bug fixing")
  - isolate changes  
(e.g. "experimental" branch)



# Reproducibility with Version Tracking

`results = program(data, parameters)`

## You must track

- source code
- a list of package versions
- parameter values \*

## You must share

- source data
- repo version for each result set
- parameter values

\* Note: Tracking parameter values in a publication repo is more stringent. It gives a complete “snapshot” of the conditions in which results were generated. It also gives a more comprehensive history of the project

May be omitted if the repo is a generic (multi-use) software packages but **must** be reported in any study.

# Wrap-up

- Do use git to track your work – even if working alone
- Don't be afraid to break things! Almost always possible to recover.
- Complex tool but daily routine involves only a handful of commands

# Thank you!

