Reproducibility with git and rmarkdown

Thomas J. Leeper

Department of Government
London School of Economics and Political Science

30 January 2018

Version Control as Organization

- Version control helps you stay organized

Version Control as Organization

- Version control helps you stay organized
  1. What's important to keep around?

Version Control as Organization

- Version control helps you stay organized
  1. What's important to keep around?
  2. What's not important to keep around?

Version Control as Organization

- Version control helps you stay organized
  1. What's important to keep around?
  2. What's not important to keep around?
  3. What is all this crap?

Version Control as Organization

- Version control helps you stay organized
  1. What's important to keep around?
  2. What's not important to keep around?
  3. What is all this crap?
- Think "tracked changes" for all of your files

Version Control as Organization

- Version control helps you stay organized
  1. What's important to keep around?
  2. What's not important to keep around?
  3. What is all this crap?
- Think "tracked changes" for all of your files
  - Save history of changes/versions

Version Control as Organization

- Version control helps you stay organized
  1. What's important to keep around?
  2. What's not important to keep around?
  3. What is all this crap?
- Think "tracked changes" for all of your files
  - Save history of changes/versions
  - Experiment non-destructively

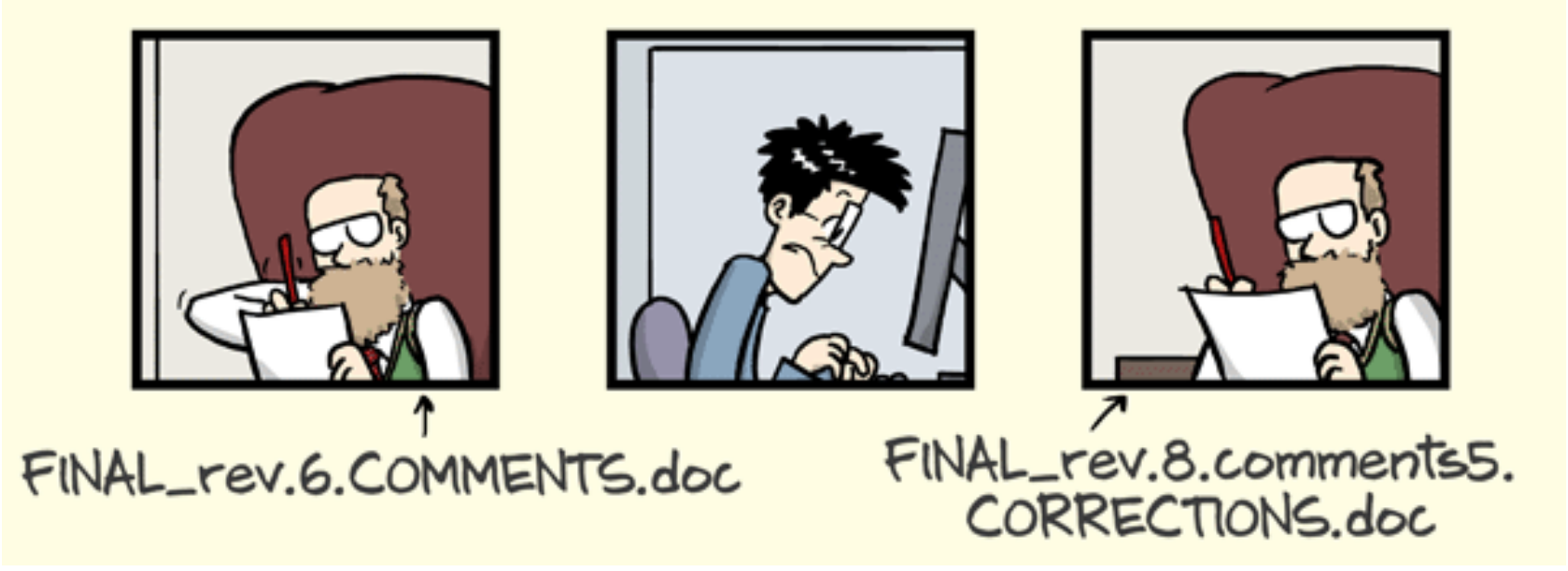
Version Control as Organization

- Version control helps you stay organized
  1. What's important to keep around?
  2. What's not important to keep around?
  3. What is all this crap?
- Think "tracked changes" for all of your files
  - Save history of changes/versions
  - Experiment non-destructively
  - Collaborate

Version Control as Organization

- Version control helps you stay organized
  1. What's important to keep around?
  2. What's not important to keep around?
  3. What is all this crap?
- Think "tracked changes" for all of your files
  - Save history of changes/versions
  - Experiment non-destructively
  - Collaborate
- You're probably already version controlling informally!


FINAL.doc!
FINAL_rev.2.doc


FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc


track changes
JORGE CHAM © 2012
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc

Name	Date modified	Type	Size
Dissertation - Prospectus 0.doc	2015-07-15 09:17	Microsoft Word 9...	18 KB
Dissertation - Prospectus 1.doc	2015-07-15 09:18	Microsoft Word 9...	142 KB
Dissertation - Prospectus 2.doc	2015-07-15 09:18	Microsoft Word 9...	246 KB
Dissertation - Prospectus 3.doc	2015-07-15 09:18	Microsoft Word 9...	250 KB
Dissertation - Prospectus 4.doc	2015-07-15 09:19	Microsoft Word 9...	250 KB
Dissertation - Prospectus 5.doc	2015-07-15 09:19	Microsoft Word 9...	263 KB
Dissertation - Prospectus 6.doc	2015-07-15 09:19	Microsoft Word 9...	287 KB
Dissertation - Prospectus 7.doc	2015-07-15 09:19	Microsoft Word 9...	291 KB
Dissertation - Prospectus 8.doc	2015-07-15 09:19	Microsoft Word 9...	300 KB
Dissertation - Prospectus 9 (For Jamie).pdf	2015-07-15 09:19	PDF File	328 KB
Dissertation - Prospectus 9.doc	2015-07-15 09:19	Microsoft Word 9...	340 KB
Dissertation - Prospectus 10 (Big Question).doc	2015-07-15 09:18	Microsoft Word 9...	19 KB
Dissertation - Prospectus 10 (New) hml.doc	2015-07-15 09:18	Microsoft Word 9...	179 KB
Dissertation - Prospectus 10 (New).doc	2015-07-15 09:18	Microsoft Word 9...	280 KB
Dissertation - Prospectus 10a (Big Question).doc	2015-07-15 09:18	Microsoft Word 9...	19 KB
Dissertation - Prospectus 10b (Big Question).doc	2015-07-15 09:18	Microsoft Word 9...	32 KB
Dissertation - Prospectus 10c (Big Question).doc	2015-07-15 09:18	Microsoft Word 9...	34 KB
Dissertation - Prospectus 11 (Outline).doc	2015-07-15 09:18	Microsoft Word 9...	25 KB
Dissertation - Prospectus 11.doc	2015-07-15 09:18	Microsoft Word 9...	287 KB
Dissertation - Prospectus 12.doc	2015-07-15 09:18	Microsoft Word 9...	175 KB
Dissertation - Prospectus 12a (Outline).doc	2015-07-15 09:18	Microsoft Word 9...	40 KB
Dissertation - Prospectus 12b (Outline).doc	2015-07-15 09:18	Microsoft Word 9...	50 KB
Dissertation - Prospectus 13 (Outline).doc	2015-07-15 09:18	Microsoft Word 9...	159 KB
Dissertation - Prospectus 13a (Outline).doc	2015-07-15 09:18	Microsoft Word 9...	228 KB
Dissertation - Prospectus 13b (Outline).doc	2015-07-15 09:18	Microsoft Word 9...	266 KB
Dissertation - Prospectus 13c.doc	2015-07-15 09:18	Microsoft Word 9...	320 KB
Dissertation - Prospectus 14 (Methods Draft for Jami...	2015-07-15 09:18	Microsoft Word 9...	44 KB

Wait, but why do we care?

If we're going to be transparent *in the end* (e.g., at replication or data archiving stage), what do we need to provide?

Wait, but why do we care?

If we're going to be transparent *in the end* (e.g., at replication or data archiving stage), what do we need to provide?

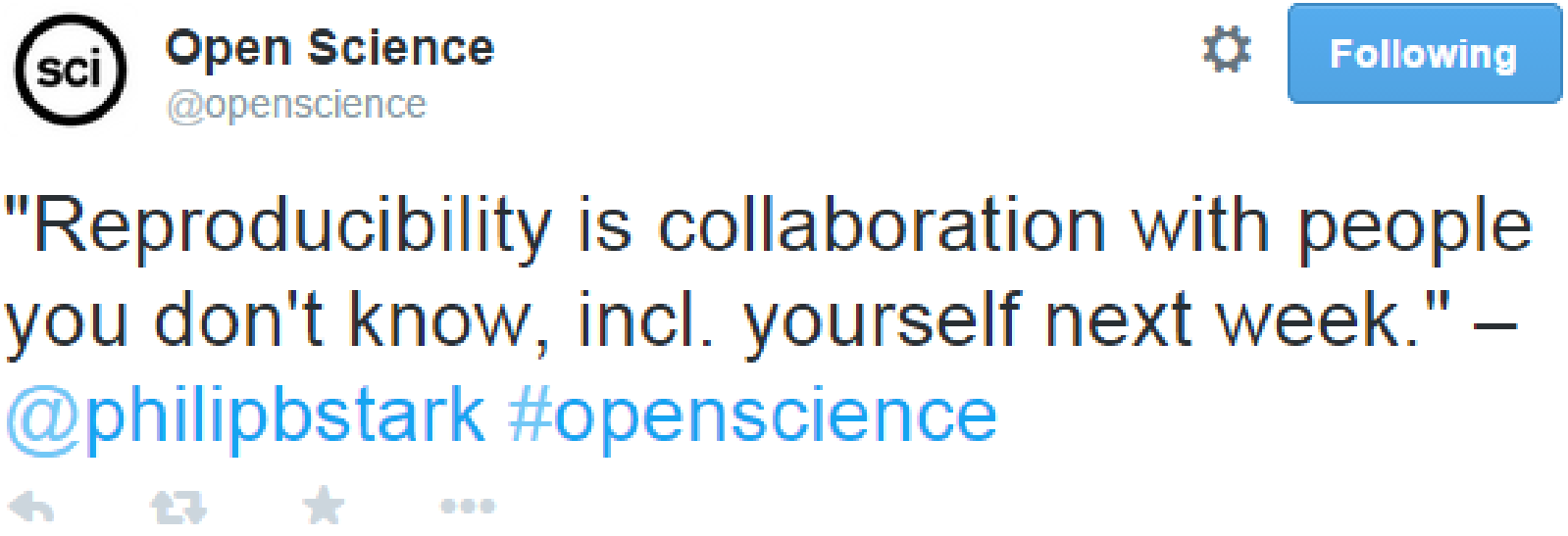
- A well-organized, reproducible analysis!

Wait, but why do we care?

If we're going to be transparent *in the end* (e.g., at replication or data archiving stage), what do we need to provide?

- A well-organized, reproducible analysis!

So rather than make that an annoying, post-hoc exercise related to publication, try to get organized and stay organized throughout your project from the very beginning.


sci
Open Science
@openscience
Following
"Reproducibility is collaboration with people you don't know, incl. yourself next week." – @philipbstark #openscience

Git

- Git is “an open-source distributed version control system”
- Developed in 2005 by Linus Torvalds
- Widely used in software development world

Why use Git for open science?

Why use Git for open science?

- Helps you keep and *annotate* snapshots of your project over time
  - Better than renaming your files all the time
  - Better than using within-file VCS (e.g., Word)
  - Better than single-stream sharing (e.g., Dropbox)

Why use Git for open science?

- Helps you keep and *annotate* snapshots of your project over time
  - Better than renaming your files all the time
  - Better than using within-file VCS (e.g., Word)
  - Better than single-stream sharing (e.g., Dropbox)
- Facilitates collaboration (incl. with future you)

Why use Git for open science?

- Helps you keep and *annotate* snapshots of your project over time
  - Better than renaming your files all the time
  - Better than using within-file VCS (e.g., Word)
  - Better than single-stream sharing (e.g., Dropbox)
- Facilitates collaboration (incl. with future you)
- It's FOSS with lots of clients, tools, and community support
  - Widely used in software development world

LSE

Using Git

- Git create a "local repository" file that you can interact with using a number of tools
  - Command-line `git`
  - Git Bash
  - Git GUI
  - GitHub Desktop
  - RStudio (via "Projects")
  - GitHub/Bitbucket/GitLab web interfaces
  - Gitkraken
  - git2r (R package)
  - ...

Git Essentials

1. `stage`
2. `commit`
3. `branch`
4. `merge`
5. `push and pull`

Git Essentials

1. `stage`
   - **stage**: select files to be recorded in a "snapshot" of the project
   - **unstage**: remove files from the snapshot (but not from your computer)
2. `commit`
3. `branch`
4. `merge`
5. `push` and `pull`

Git Essentials

1. `stage`
2. `commit`
   - **commit**: record a permanent snapshot of the staged files, labelled with a "commit message"
   - **amend**: modify (typically the most recent) commit with new changes or commit message
3. `branch`
4. `merge`
5. `push` and `pull`

Git Essentials

1. `stage`
2. `commit`
3. `branch`
   - produce a complete *local* copy of the project where changes can be made independently of the "master" branch
4. `merge`
5. `push` and `pull`

Git Essentials

1. `stage`
2. `commit`
3. `branch`
4. `merge`
   - update a branch with changes from another local branch (or a remote); you can change multiple branches independently.
5. `push` and `pull`

Git Essentials

1. `stage`
2. `commit`
3. `branch`
4. `merge`
5. `push` and `pull`
   - **push**: send the project (any new commits) to a remote server (like GitHub)
   - **pull**: grab new commits from a remote server

Git Essentials

1. `stage`
2. `commit`
3. `branch`
4. `merge`
5. `push and pull`

Hands-on practice!

```
git --version
git
git config --global user.name "My Name"
git config --global user.email "me@example.com"
git config --list
```

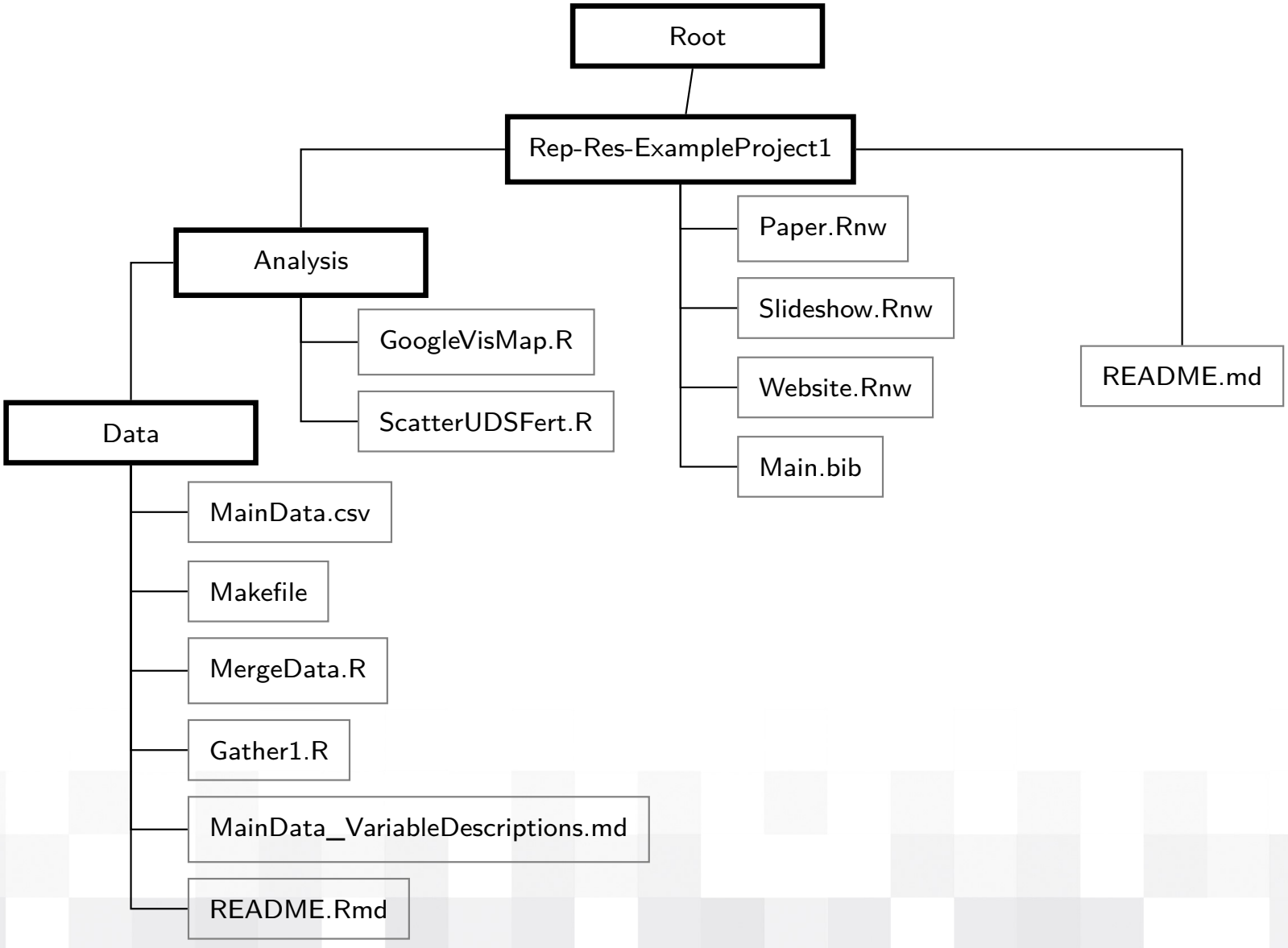
```
git init
git status
echo Hello world! > README.md
git add README.md
git status
git rm --cached README.md
git status
git add --all
git commit -m "my first commit!"
git status
```

Initializing a Project Structure

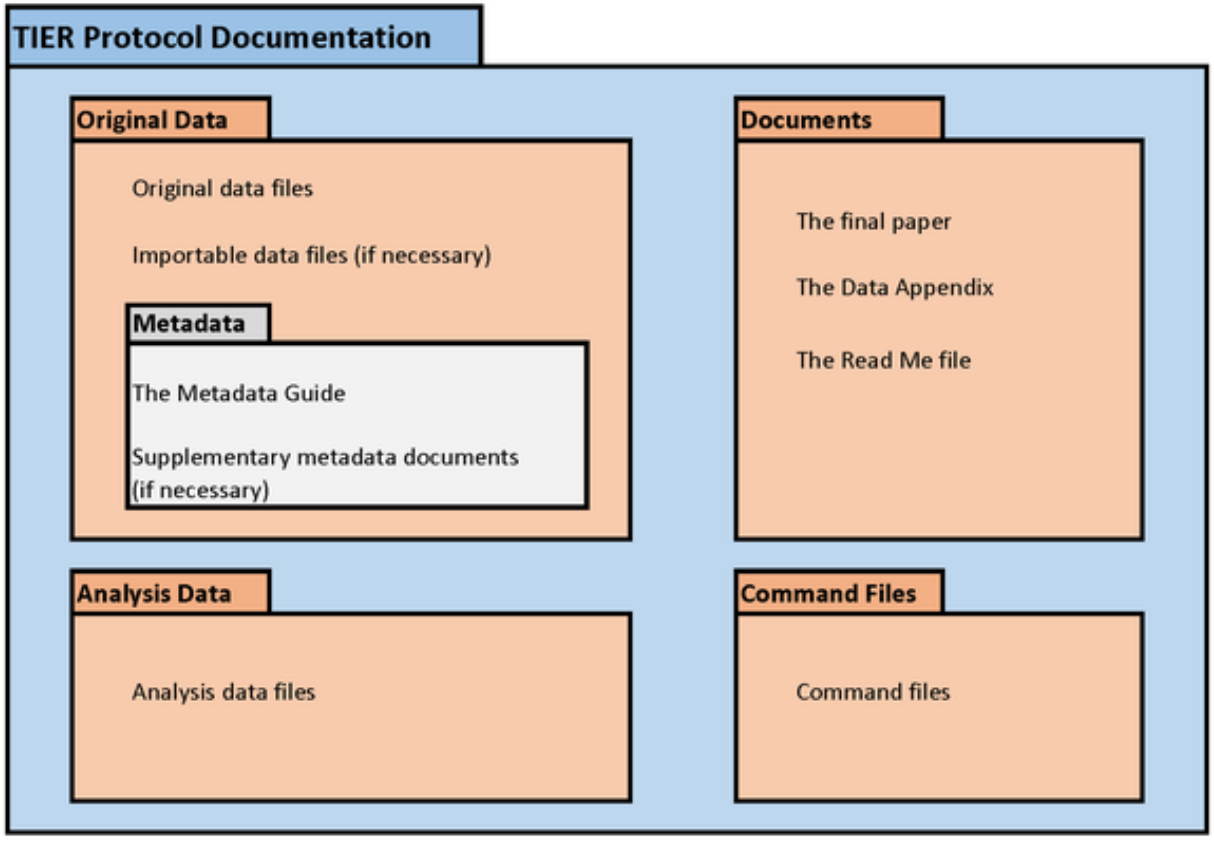
- There's no single best way to organize a project
- But, some words of wisdom:
  - Put like with like
  - Avoid excessive hierarchy
  - Not everything needs to go into git
  - Steal others' structures!

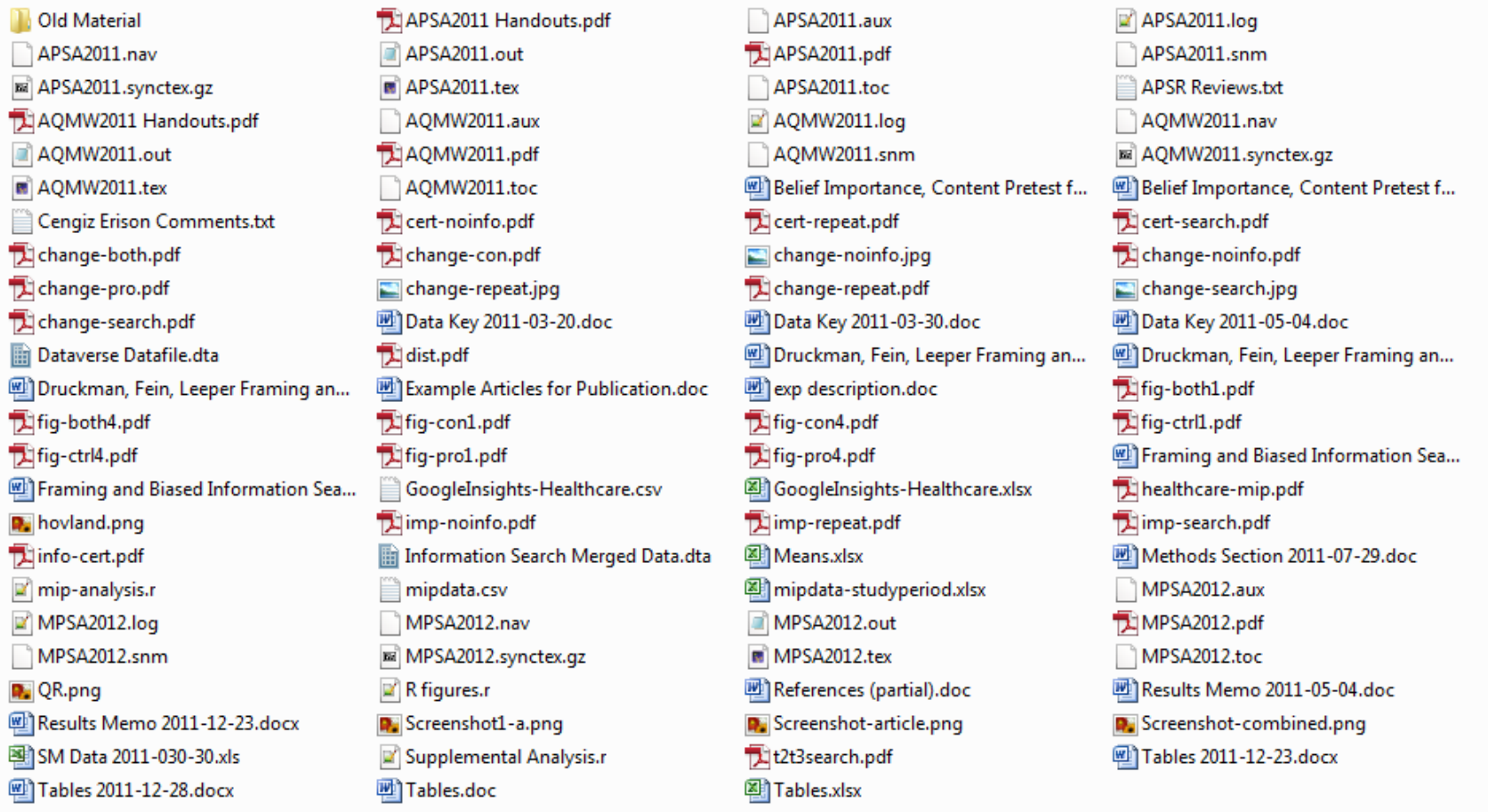
What makes up the ideal reproducible research product?

- Gandrud's template
- rOpenSci's "Research Compendium"
- Project TIER


Root
Rep-Res-ExampleProject1
Paper.Rnw
Slideshow.Rnw
Website.Rnw
Main.bib
README.md
Analysis
GoogleVisMap.R
ScatterUDSFert.R
Data
MainData.csv
Makefile
MergeData.R
Gather1.R
MainData_VariableDescriptions.md
README.Rmd

```
project
|- DESCRIPTION       # project metadata and dependencies
|- README.md         # top-level description of content
|
|- data/             # raw data, not changed once created
|  +- my_data.csv    # data files in open formats
|
|- analysis/         # any programmatic code
|  +- my_scripts.R   # R code used to analyse data
```


TIER Protocol Documentation
Original Data
Original data files
Importable data files (if necessary)
Metadata
The Metadata Guide
Supplementary metadata documents (if necessary)
Documents
The final paper
The Data Appendix
The Read Me file
Analysis Data
Analysis data files
Command Files
Command files


Old Material
APSA2011 Handouts.pdf
APSA2011.aux
APSA2011.log
APSA2011.nav
APSA2011.out
APSA2011.pdf
APSA2011.snm
APSA2011.synctex.gz
APSA2011.tex
APSA2011.toc
APSR Reviews.txt
AQMW2011 Handouts.pdf
AQMW2011.aux
AQMW2011.log
AQMW2011.nav
AQMW2011.out
AQMW2011.pdf
AQMW2011.snm
AQMW2011.synctex.gz
AQMW2011.tex
AQMW2011.toc
Belief Importance, Content Pretest f...
Belief Importance, Content Pretest f...
Cengiz Erison Comments.txt
cert-noinfo.pdf
cert-repeat.pdf
cert-search.pdf
change-both.pdf
change-con.pdf
change-noinfo.jpg
change-noinfo.pdf
change-pro.pdf
change-repeat.jpg
change-repeat.pdf
change-search.jpg
change-search.pdf
Data Key 2011-03-20.doc
Data Key 2011-03-30.doc
Data Key 2011-05-04.doc
Dataverse Datafile.dta
dist.pdf
Druckman, Fein, Leeper Framing an...
Druckman, Fein, Leeper Framing an...
Druckman, Fein, Leeper Framing an...
Example Articles for Publication.doc
exp description.doc
fig-both1.pdf
fig-both4.pdf
fig-con1.pdf
fig-con4.pdf
fig-ctrl1.pdf
fig-ctrl4.pdf
fig-pro1.pdf
fig-pro4.pdf
Framing and Biased Information Sea...
Framing and Biased Information Sea...
GoogleInsights-Healthcare.csv
GoogleInsights-Healthcare.xlsx
healthcare-mip.pdf
hovland.png
imp-noinfo.pdf
imp-repeat.pdf
imp-search.pdf
info-cert.pdf
Information Search Merged Data.dta
Means.xlsx
Methods Section 2011-07-29.doc
mip-analysis.r
mipdata.csv
mipdata-studyperiod.xlsx
MPSA2012.aux
MPSA2012.log
MPSA2012.nav
MPSA2012.out
MPSA2012.pdf
MPSA2012.snm
MPSA2012.synctex.gz
MPSA2012.tex
MPSA2012.toc
QR.png
R figures.r
References (partial).doc
Results Memo 2011-05-04.doc
Results Memo 2011-12-23.docx
Screenshot1-a.png
Screenshot-article.png
Screenshot-combined.png
SM Data 2011-030-30.xls
Supplemental Analysis.r
t2t3search.pdf
Tables 2011-12-23.docx
Tables 2011-12-28.docx
Tables.doc
Tables.xlsx

```
mkdir code
mkdir data
mkdir figures
git status
```

```
git status
cat README.md
# do something to README.md
git diff
git add README.md
git commit -m "second commit"
git status
git log
git log --oneline
git log --oneline -1
git log --oneline --stat
```


LSE

```
git status
git diff README.md
git diff HEAD README.md
git diff HEAD~1 README.md
git diff HEAD~2 README.md
git diff HEAD~3 README.md
git diff HEAD~20 README.md
git diff <commit hash> README.md
git diff <commit hash>
```

!! DANGER: Amend Commit !!

```
git status
git log --oneline
# maybe add/rm files
git amend
# enter the hell of vim
```

```
git config --global core.editor
"<executable> <options>"
```

Safe reversion

```
git status
git log --oneline
git revert <commit hash>
# enter the hell of vim
# or something else terrible
git revert --abort
```

!! DANGER: Unsafe reversion !!

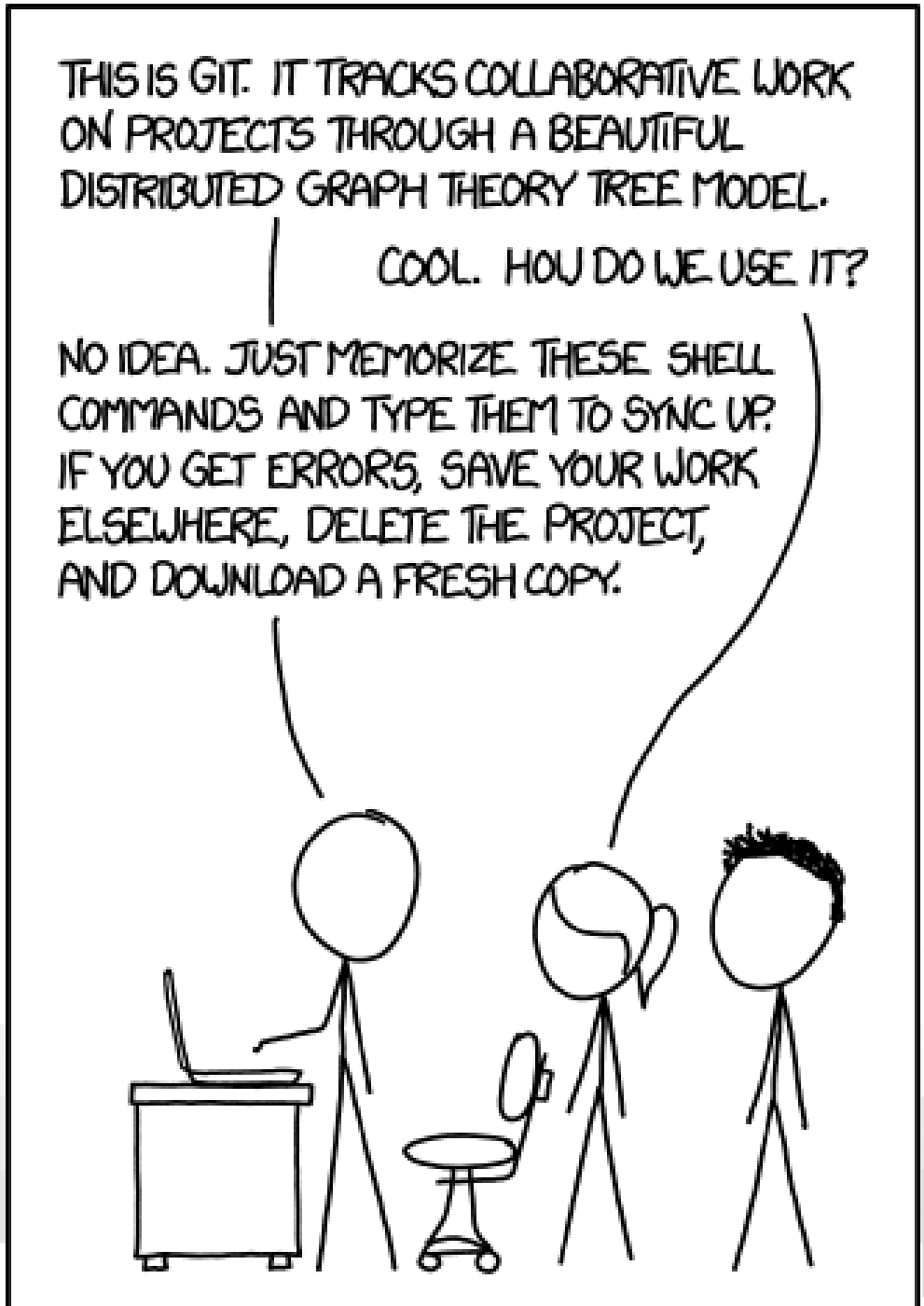
The StackOverflow Question

```
git status
echo "bad bad bad" > bad.txt
git status
echo bad.txt > .gitignore
git status
echo bad bad bad > bad1.txt
echo bad bad bad > bad2.txt
echo bad* > .gitignore
git status
git add bad1.txt -f
git status
```

Navigating History

```
git status
git log
git checkout <commit hash>
git status
ls
cat README.md
git checkout master
```

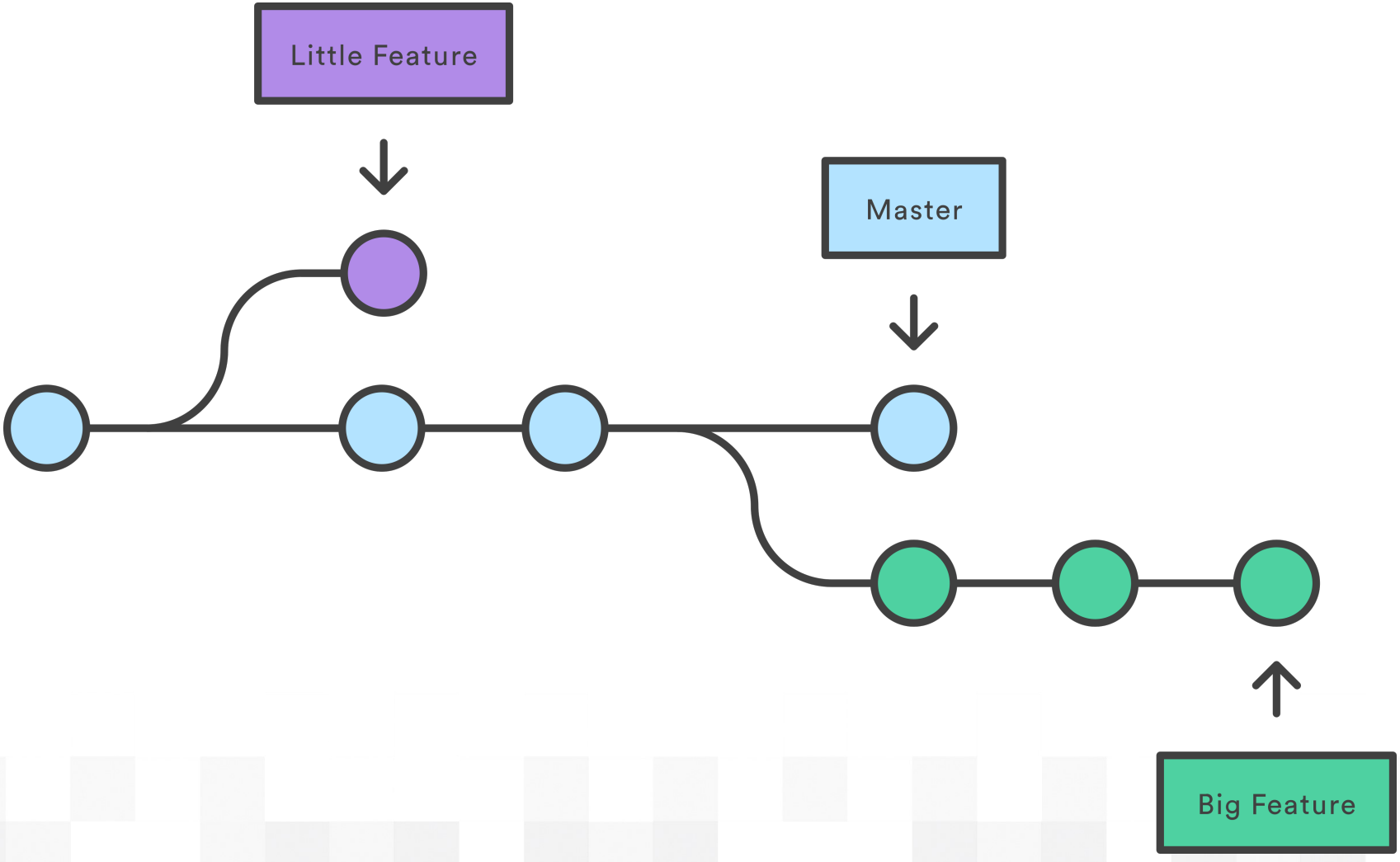
```
git status
git log
git checkout <commit hash>
git status
ls
echo aaaaaah!>manuscript.txt
git checkout master
```


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

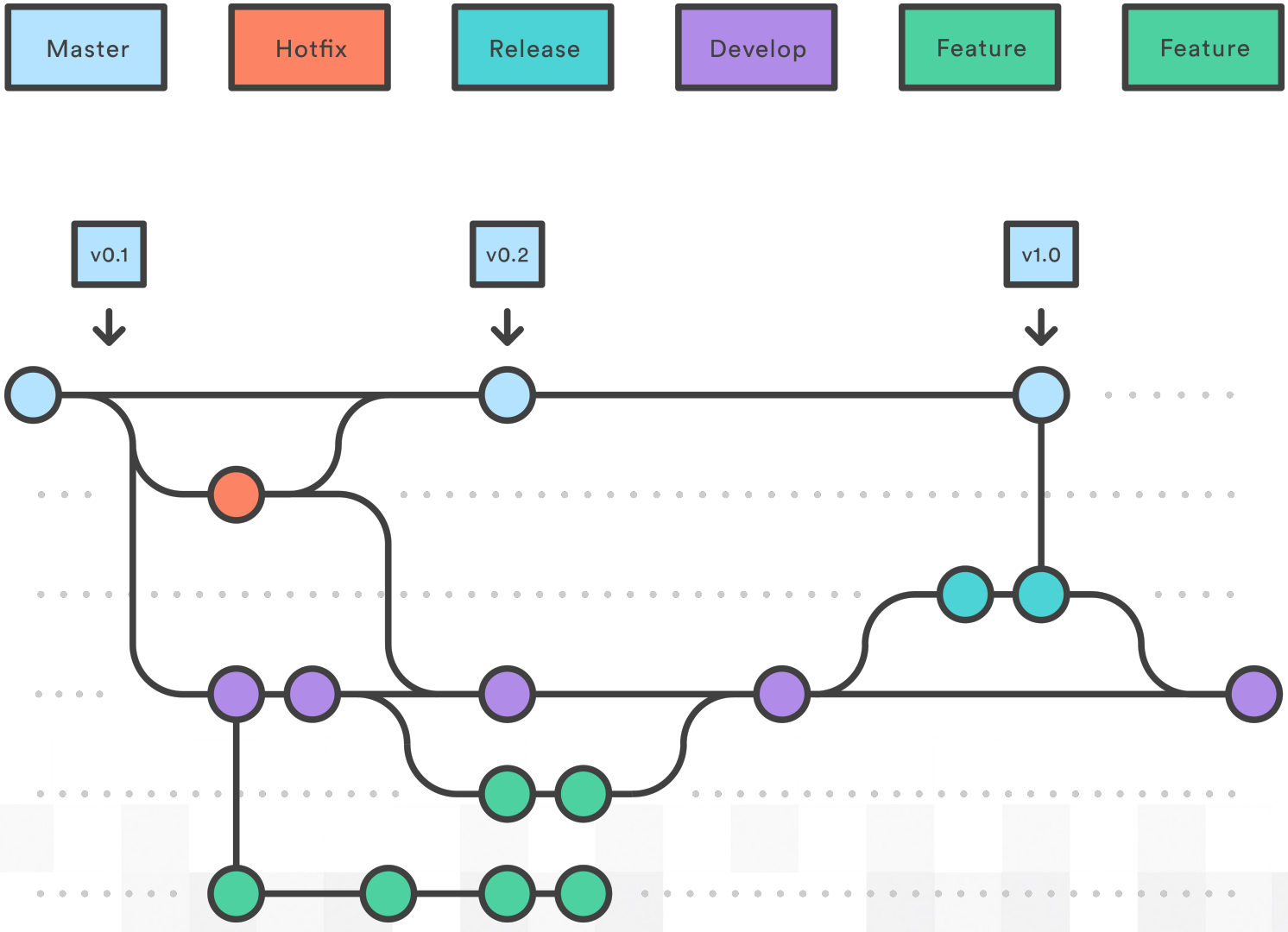

LSE

Branches

- Branches are local, parallel versions of your entire project
- Useful for multiple things:
  - Experimentation
  - Manuscript submissions
  - Collaboration



Source: https://www.atlassian.com/git/tutorials



Source: https://www.atlassian.com/git/tutorials

Simple branch and merge

```
git status
git checkout -b thomas
git status
# do something
git add --all
git commit -m "thomas's commit"
git checkout master
git branch
git log --graph --oneline
git merge thomas
```

GUIs

- You can do everything in Git on the command line
- GUIs can be helpful for:
  - Exploring history
  - Visualizing branches
  - Confirming what you're doing

Merge conflicts

```
git checkout -b thomas
git status
# do something to README.md
git add --all
git commit -m "change on thomas"
git checkout master
# do something to README.md
git add --all
git commit -m "change on master"
git merge thomas
git log
```


LSE

Remotes

- A server ("cloud") instance of the Git repository
- Useful for multiple things:
  - Collaboration
  - Transparency
  - Archiving/backups
  - Using web-based Git interfaces

Remotes

- Three major players in cloud Git
  - GitHub
  - Atlassian Bitbucket
  - GitLab
- Why choose one or the other?
  - Cost
  - Collaborators
  - Private repositories

```
git status
git remote add github
https://github.com/leeper/rt2
git remote
git remote set-url
git remote rename
git remote remove
```

```
git status
git push github master -u
git fetch github
git fetch github master
git checkout -b new-idea
git push github new-idea
git checkout master
git pull github master
git pull
```


LSE

```
git status
git tag -a v0.0.1 -m "v0.0.1"
git push --tags

git tag -d v0.0.1
```

Tags versus Branches

- *Branches* are for working versions of project
  - Collaborator-specific branches
  - Submission-specific branches
  - Experimental or “bug fix” branches
- *Tags* are for marking particular snapshots
  - Significant moments in project history
  - Journal submission or conference version
  - Formal “releases”

Collaboration

- Technical aspects
  - Give collaborators access on GitHub (or wherever)
  - Work on separate branches
  - Merge agreed changes into **master**
- Human factors aspects
  - Requires agreeing on workflow
  - Communication about what goes in “master”
  - Can feel awkward if moving from a Dropbox- or email-based collaboration style

Try it with a partner!

1. Partner A create a GitHub repo; give Partner B access
2. Partner B should `git fetch`/`git pull` the repo
3. Partner B should create a local branch and `git push`
4. Partner A should `git fetch` the branch
5. Partner A should `git merge` the branch to **master** and `git push`
6. Partner B should `git pull` from **master**
7. Both use `git log` to compare


LSE

Conclusion

- Once you use Git, you'll never want to go back to your old workflow

Conclusion

- Once you use Git, you'll never want to go back to your old workflow
- But, collaborators probably don't know or want to use Git!

Conclusion

- Once you use Git, you'll never want to go back to your old workflow
- But, collaborators probably don't know or want to use Git!
- Git is crazy complicated -– StackOverflow is your friend


LSE


LSE

How do you typically get figures, tables, and other material out of analytic software and into papers or presentations?

Dynamic Documents in R

- The dynamic documents landscape is evolving very, very rapidly:
  - Early 2000s: Sweave
  - 2010's: knitr
  - Ongoing: Rmarkdown

Dynamic Documents in R

- The dynamic documents landscape is evolving very, very rapidly:
  - Early 2000s: Sweave
  - 2010's: knitr
  - Ongoing: Rmarkdown
- Embed code (R or otherwise) inside a manuscript that outputs:
  - Word (.docx)
  - HTML
  - LaTeX/PDF
  - HTML or PPT slides

```
# My Manuscript
Thomas J. Leeper
This is my manuscript.
```

Rmarkdown

1. YAML metadata header
2. Document contents in **markdown**
3. Code in "code chunks":

````
```{r chunk1}
# R code
hist(rnorm(1000))
```
````

```
---
- title: My Manuscript
- author: Thomas J. Leeper
- date: 2017-09-21
- output: pdf_document
---

This is my manuscript.

```{r chunk1}
# R code
hist(rnorm(1000))
```
```

Markdown Basics

Markdown is a very simple markup language for formatting simple texts:

`*italics*`	*italics*
`*bold*`	**bold**
`‘preformatted‘`	`preformatted`
`# Heading`	Heading Level 1
`## Heading`	Heading Level 2
`### Heading`	Heading Level 3
`[link](https://google.com)`	link


LSE

Chunk Options

```
```{r chunk1, eval=TRUE, echo=TRUE}
2 + 2
```
```

```
```{r chunk2, eval=TRUE, echo=FALSE}
2 + 2
```
```

```
```{r chunk3, echo=FALSE, results="hide"}
2 + 2
```
```

Global Chunk Options

```
```{r options, eval = TRUE, echo = FALSE}
library("knitr")
opts_chunk$set(echo = FALSE,
               cache = TRUE,
               message = FALSE)
```
```

Basic Tables

````
```{r table1, results = "asis"}
xtable::xtable(table(mtcars$cyl, mtcars$gear))

knitr::kable(head(mtcars))
```
````

Regression Results Tables

````
```{r table2, results = "asis"}
library("stargazer")
stargazer(
  x1 <- lm(mpg ~ disp + wt,
           data = mtcars),
  x2 <- lm(mpg ~ disp + wt + vs,
           data = mtcars),
  header = FALSE
)
```
````

Figures

````
```{r fig1,
    fig.cap = "Fuel Economy by Weight",
    fig.height = 4,
    fig.width = 6}
library("ggplot2")
ggplot(mtcars,
    aes(x = wt,
        y = mpg,
        colour = factor(cyl))) +
  geom_point()
```
````