Tips for Working Reproducibly

Thomas J. Leeper

Senior Visiting Fellow in Methodology
London School of Economics and Political Science

9 May 2019

Tools We'll See Today

- R, RStudio
  - https://cran.r-project.org/
  - https://www.rstudio.com/
- make (and other command line tools)
  - https://cran.r-project.org/bin/windows/Rtools/
- git
  - git (https://git-scm.com/)
  - github (https://github.com/)
  - gitkraken (https://www.gitkraken.com/)
- Collaboration software
  - overleaf (https://www.overleaf.com/)
  - dropbox (https://www.dropbox.com/)
  - google drive (https://drive.google.com/)

Introductions

- Me:
  - Thomas
  - Political Scientist, Methodology Department
  - Experimental and computational methods
  - R
- You:
  - Name
  - Field/Department
  - Methods
  - Tools/Software

Learning Objectives

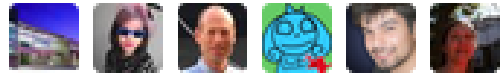
1. Define reproducibility and replicability
2. Understand how to organize a reproducible research project
3. Recognize different approaches to reproducibility and tools for implementing various reproducible workflows
4. Understand how to collaborate reproducibly

Kaitlin Thaney
@kaythaney

Following

"'Reproducible research' is a redundant term. 'Irreproducible research' just used to be known as 'bullshit'." - @fperez_org
::slow clap::

RETWEETS 122 FAVORITES 61



6:11 PM - 8 May 2014

Some Semantics

- **Reproduction**: recreating output from a shared input
- **Replication**: creating new output from new input

`https://thomasleeper.com/2015/05/open-science-language/`

Beyond Scope for Today

- Verification
- Data Transparency
- Replication
- Non-computational reproducibility
  - Software and hardware versioning
  - Deterministic computing
  - Interpretations

Lewis and Clark

Lewis and Clark

Jefferson's instructions:

> *The object of your mission is to explore the Missouri river, & such principal stream of it, as, by its course & communication with the water of the Pacific ocean may offer the most direct & practicable water communication across this continent, for the purposes of commerce*

Beginning at the mouth of the Missouri, you will take observations of latitude and longitude at all remarkable points on the river. . .

Beginning at the mouth of the Missouri, you will take observations of latitude and longitude at all remarkable points on the river. . .

The courses of the river between these points of observation may be supplied by the compass, the logline & by time, corrected by the observations themselves. The variations of the compass too, in different places should be noticed

Your observations are to be taken with great pains & accuracy to be entered distinctly, & intelligibly for others as well as yourself, to comprehend all the elements necessary, with the aid of the usual tables to fix the latitude & longitude of the places at which they were taken, & are to be rendered to the war office, for the purpose of having the calculations made concurrently by proper persons within the U.S. Several copies of these as well as of your other notes, should be made at leisure times, & put into the care of the most trustworthy of your attendants, to guard by multiplying them against the accidental losses to which they will be exposed. A further guard would be that one of these copies be written on the paper of the birch, as less liable to injury from damp than common paper

. . . and finally:

To provide, on the accident of your death, against anarchy, dispersion & the consequent danger to your party, and total failure of the enterprise, you are hereby authorised, by any instrument signed & written in your own hand, to name the person among them who shall succeed to the command

Activity!

How do you organize your files for a project?

Wait, but why do we care?

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

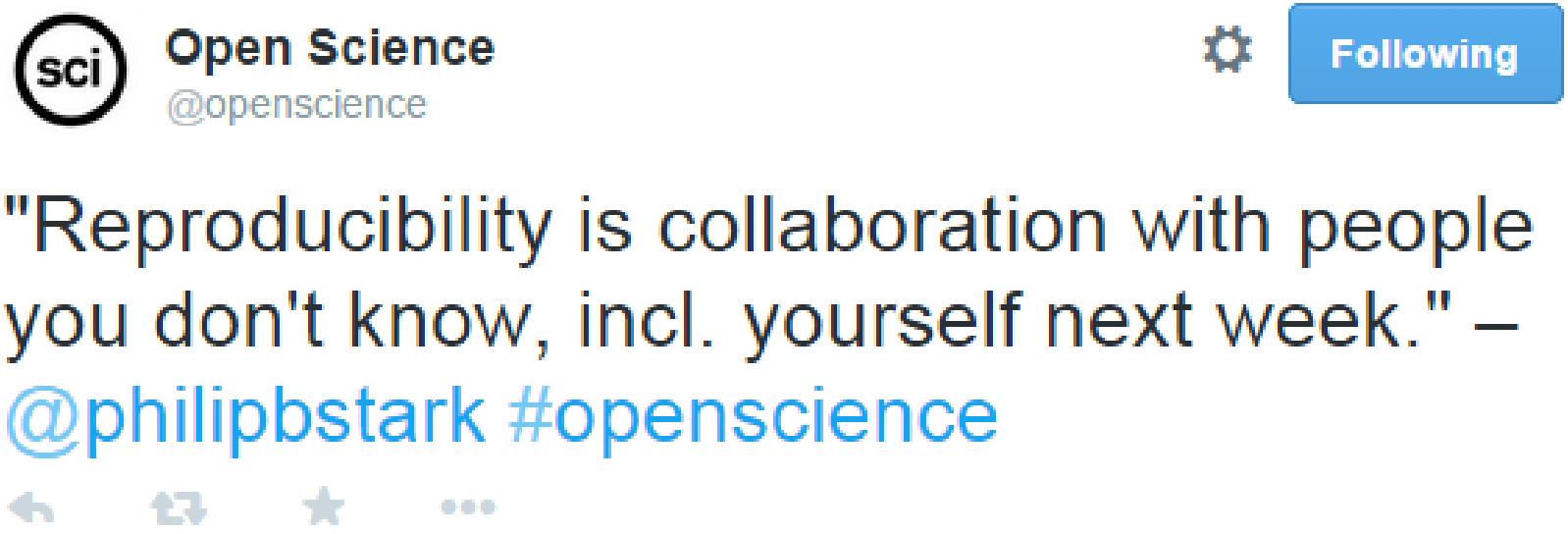
A well-organized, reproducible analysis!

Wait, but why do we care?

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

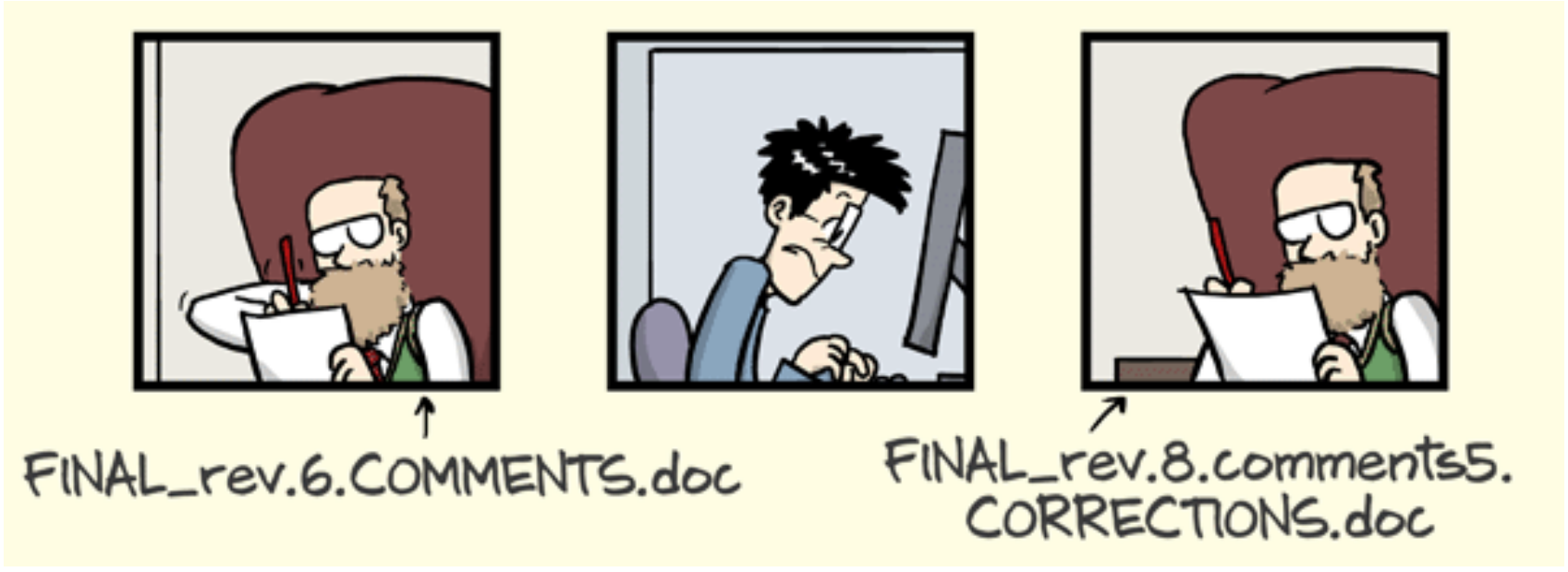
A well-organized, reproducible analysis!

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

The single most important part of reproducibility is naming things!


FINAL.doc!
FINAL_rev.2.doc

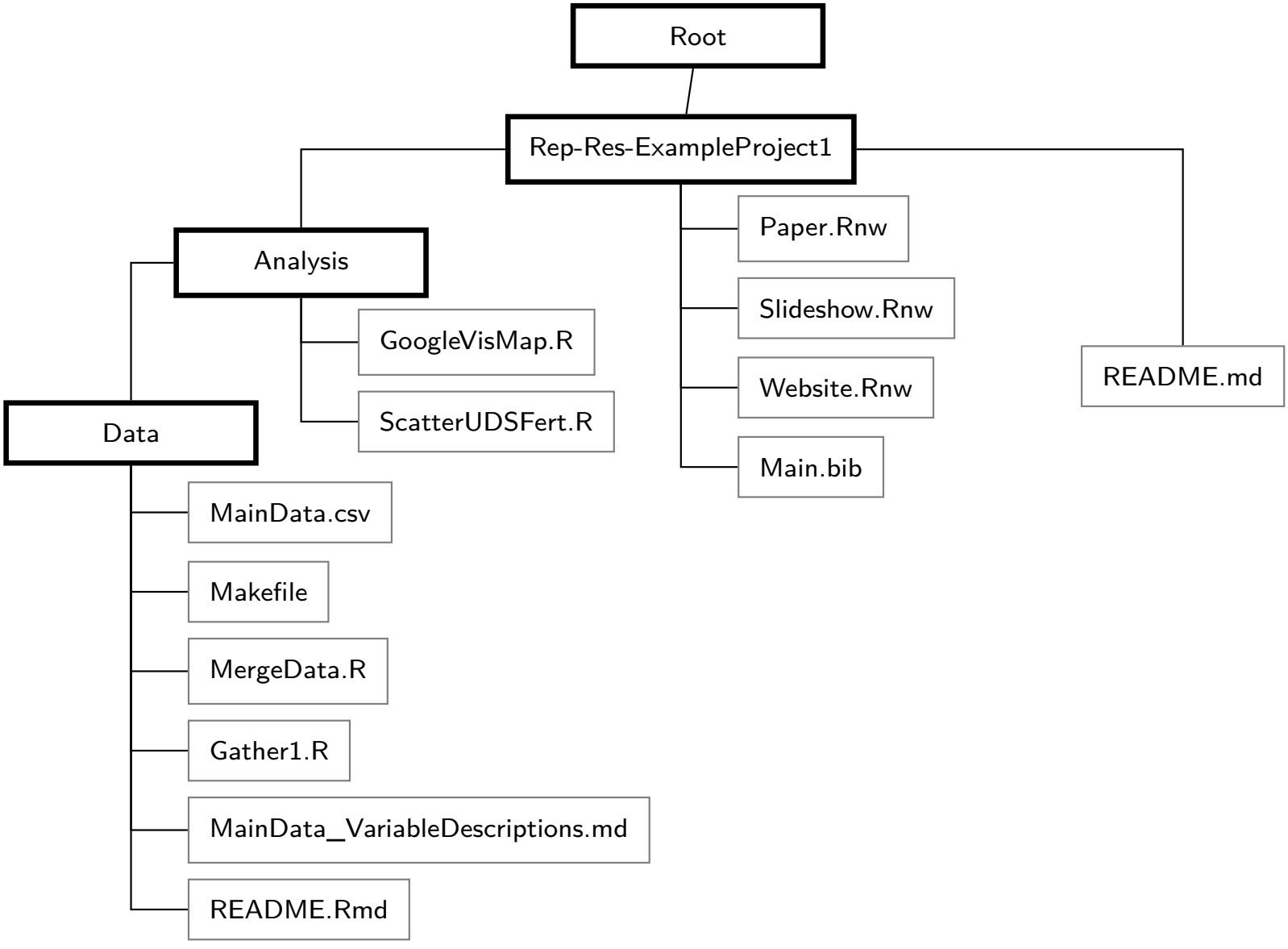

FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc


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

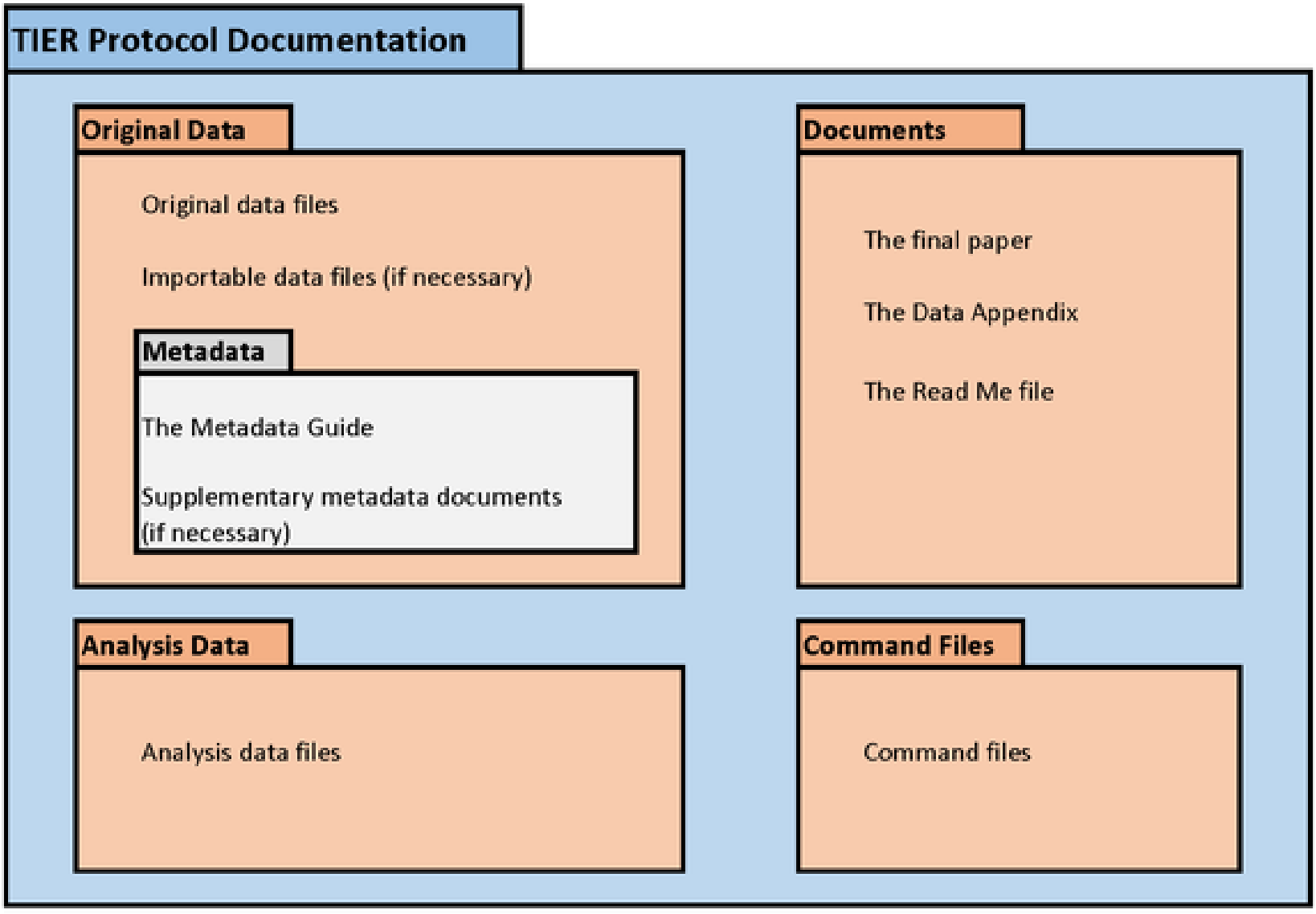
Name	Date modified	Type	Size
Dissertation - Prospectus 0.doc	2015-07-15 09:17	Microsoft Word 9...	18 KB
Dissertation - Prospectus 1.doc	2015-07-15 09:18	Microsoft Word 9...	142 KB
Dissertation - Prospectus 2.doc	2015-07-15 09:18	Microsoft Word 9...	246 KB
Dissertation - Prospectus 3.doc	2015-07-15 09:18	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

What makes up the ideal reproducible research product?

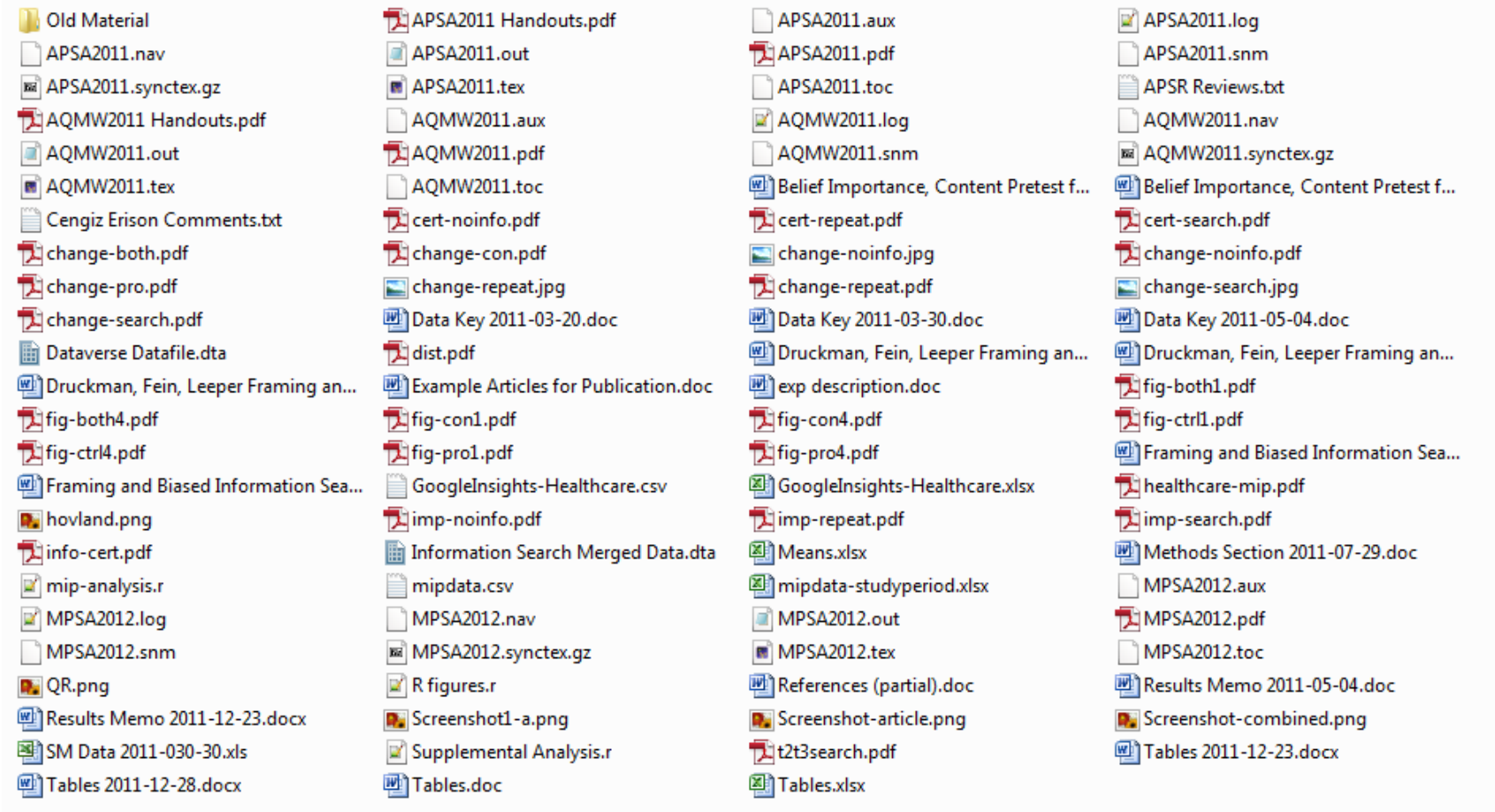
- Gandrud's template
- rOpenSci's "Research Compendium"
- Project TIER
- AJPS Replication/Verification Policy


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

```
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

Don’t be this guy:



AJPS

https://ajps.org/ajps-replication-policy/

```
mkdir code

mkdir data

mkdir figures

echo # My Project > README.md
```

File names

Which of these do we like best?

- PhD Comics style
- Sequential version numbers
- Datestamps

File names

Which of these do we like best?

- PhD Comics style
- Sequential version numbers
- Datestamps
- None of the above (Git!)

Dates

- paper03-05-18.docx
- paper3-5-18.docx
- paper05-03-18.docx
- paper5-3-18.docx
- paper18-3-5.docx
- paper18-5-3.docx
- paper03-05.docx
- paper05-03.docx
- paper3-5.docx
- paper5-3.docx

ISO 8601

There is only one way to write dates:

`YYYY-MM-DD`

Why?

- Unambiguous
- Sortable by most-to-least important digits
- Language agnostic
- Universal

Case

- TitleCase
- camelCase
- snake_case
- Naturalcase
- (Anything with spaces)

Everything you do should be plain text*

Everything you do should be plain text*

* Exceptions to this are images (sometimes)

- **Plain text is always compatible** Every single operating system has a plain text editor and they are all compatible up to the encoding of the text. This means that if you develop your lecture notes on a Mac and I develop them on a PC then we can still easily share - no worrying if you have the right software.
- **Plain text is easy to mix and match** If your lecture materials are in a simple plain text format like markdown you can copy and paste the materials from one lecture into another and when the document is compiled make all the formatting/colors/etc. match. No more looking at hodgepodges of borrowed slides some with one ppt format and some with another.
- **Plain text is easy to maintain** We mostly work on scalable education here at the JHU Data Science Lab. We often think of scalable education in terms of the number of students, but here we have also run into the problem of scaling the number of courses/instructor. I am currently the lead instructor on more than half a dozen classes running all the time. Every time I have to re-record a video it takes set up time, recording time, editing time. If I have an error in a markdown file it is a quick edit to a text file.
- **Plain text is lightweight** Images can be stored online and the lecture notes themselves are small. This might not matter where internet access is good, but in places with limited resources or wifi, this can be the difference from easily accessible lecture notes and bad ones.
- **Plain text is always forward compatible** Regardless of the next platform, if we have all of our knowledge/lecture notes stored in plain text it will be easy to extact them. When you switch platform, or compiling software, or style, there isn't a worry about the files not working appropriately.

https://simplystatistics.org/2017/06/13/the-future-of-education-is-plain-text/

Additionally. . .

- Easy to use in version control
- Easy to dynamically update as part of an analysis “pipeline”

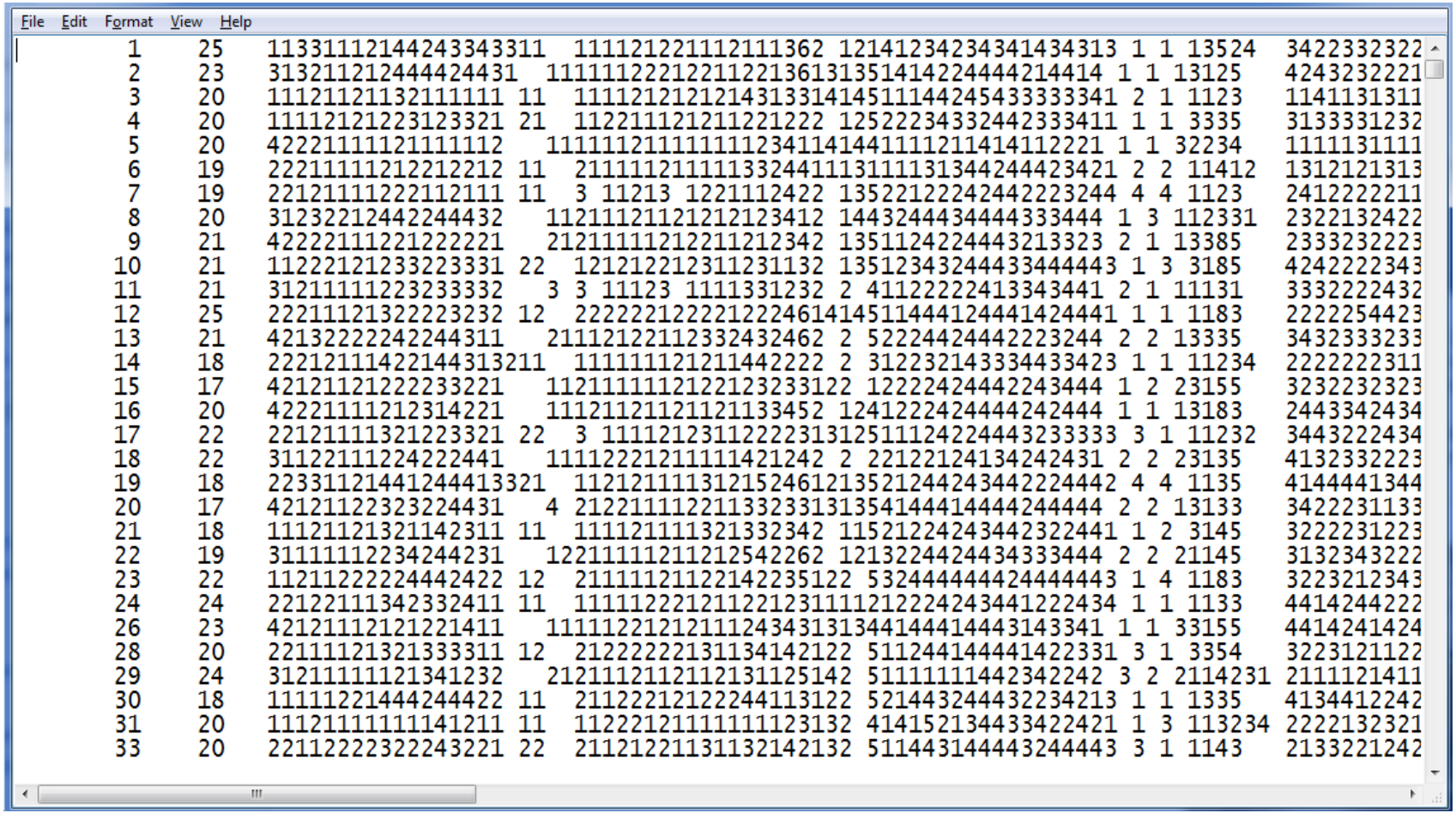
File	Good format(s)
Document	.md, .tex, .Rmd, .Rnw
Presentation	.tex, .Rmd, .Rnw
Code	.R, .Rmd, .py, .do, .ado
Data	.tsv, .csv
Codebook	.txt
Citations	.bib
Images	.svg, .pdf, .png
References	.bib

Reference Management

- Never type/format citations manually!
- Use reference management software
  - Zotero (`https://www.zotero.org/`)
  - Jabref (`http://www.jabref.org/`)
- Develop a workflow where these are automatically generated

```
@Article{Berinsky2017,
  author    = {Berinsky, Adam J.},
  title     = {Measuring Public Opinion with Surveys},
  journal   = {Annual Review of Political Science},
  year      = {2017},
  volume    = {20},
  number    = {1},
  pages     = {309--329},
  month     = {may},
  doi       = {10.1146/annurev-polisci-101513-113724}
}
```

Is it possible to take the plain text ideology too far?



File Edit Format View Help

1	25	113311121442433433 11	111121221112111362	12141234234341434313	1 1	13524	3422332322
2	23	313211212444424431	11111122212211221361313514142244444214414	1 1	13125	4243232221	
3	20	111211211321111111 11	11112121212143133141451114424543333334 1	2 1	1123	1141131311	
4	20	111121212231233 21	11221112121122122	12522234332442333411	1 1	3335	3133331232
5	20	422211111211111112	11111121111111123411414411112114141122 21	1 1	32234	1111131111	
6	19	222111112122122 11	211111211111332441113111131344244423421	2 2	11412	1312121313	
7	19	221211112221121 11	3 11213 1221112422	13522122242442223244	4 4	1123	2412222211
8	20	312322124422444 32	11211121121212123412	14432444344444333444	1 3	112331	2322132422
9	21	422221112212222 21	21211111212211212342	13511242244433213323	2 1	13385	2333232223
10	21	112221212332233 31 22	121212212311231132	13512343244433444443	1 3	3185	4242222343
11	21	312111112232333 32	3 3 11123 1111331232	2 41122222413343441	2 1	11131	3332222432
12	25	222111213222232 12	22222212222122246141451144412444142444 1	1 1	1183	2222254423	
13	21	421322222422443 11	21112122112332432462	2 52224424442223244	2 2	13335	3432333233
14	18	222121114221443132 11	111111121211442222	2 312232143334433423	1 1	11234	2222222311
15	17	42121121222233221	11211111122122123233122	12222424442243444	1 2	23155	3232232323
16	20	422211112123142 21	11121121121121133452	12412224244444242444	1 1	13183	2443342434
17	22	221211113212233 21 22	3 11112123112222313125111242244433233333	3 1	11232	3443222434	
18	22	311221112242224 41	11112222121111142124 2	221221241342424 31	2 2	23135	4132332223
19	18	223311214412444133 21	11212111131215246121352124424344222444 2	4 4	1135	4144441344	
20	17	42121122323224431	4 21221111221133233131354144414444244444	2 2	13133	3422231133	
21	18	111211213211423 11 11	11112111132133234 2	11521224243442322441	1 2	3145	3222231223
22	19	31111112234244231	12211111211212542262	12132244244343334 44	2 2	21145	3132343222
23	22	112112222244422 12	21111121122142235122	532444444424444443	1 4	1183	3223212343
24	24	221221113423324 11 11	11111222121122123111121222424344122243 4	1 1	1133	4414244222	
26	23	421211121212214 11	11111221212111243431313441444144434143341	1 1	33155	4414241424	
28	20	221111213213333 11 12	21222222131134142122	5112441444414223 31	3 1	3354	3223121122
29	24	312111111213412 32	21211121121121311251 42	51111111442342242	3 2	2114231	2111121411
30	18	111112214442444 22 11	21122212122244113122	521443244432234213	1 1	1335	4134412242
31	20	111211111111412 11 11	11222121111111123132	414152134433422421	1 3	113234	2222132321
33	20	221122223222432 21 22	21121221131132142132	511443144443244443	3 1	1143	2133221242

you describe your opinion of **(INSERT ITEM: ROTATE ITEMS a.-i. AND g.-i.)** as very favorable, mostly favorable, mostly UNfavorable, or very unfavorable? **(INTERVIEWERS: PROBE TO DISTINGUISH BETWEEN "NEVER HEARD OF" AND "CAN'T RATE")**

			Very Favor-able	Mostly Favor-able	Mostly Unfavor-able	Very Unfavor-able	Never Heard of	Can't Rate
(115)	a.	Network television news (1-96)	1	2	3	4	5	6
(116)	b.	Local TV news (1-96)	1	2	3	4	5	5
(117)	c.	The daily newspaper you are most familiar with (1-96)	1	2	3	4	5	6
(118)	d.	Congress (1-96)	1	2	3	4	5	6
(119)	e.	Tobacco companies (7-94)	1	2	3	4	5	6
(120)	f.	Labor unions (2-96)	1	2	3	4	5	6
(121)	g.	Bill Clinton (2-96)	1	2	3	4	5	6
(122)	h.	Hillary Clinton (2-96)	1	2	3	4	5	5
(123)	i.	Bob Dole (2-96)	1	2	3	4	5	6

Questions?

Activity!

What's your analytic workflow? How do you get results into a paper, poster, or presentation?

My First Workflow

My First Workflow

1. Make figure/table/analysis in R

My First Workflow

1. Make figure/table/analysis in R
2. Copy/paste into Word document

My First Workflow

1. Make figure/table/analysis in R
2. Copy/paste into Word document
3. Adjust figure/table numbering

My First Workflow

1. Make figure/table/analysis in R
2. Copy/paste into Word document
3. Adjust figure/table numbering
4. Double check references

My First Workflow

1. Make figure/table/analysis in R
2. Copy/paste into Word document
3. Adjust figure/table numbering
4. Double check references
5. Save as PDF
6. Change something in 1, repeat 2-5

My First Workflow

1. Make figure/table/analysis in R
2. Copy/paste into Word document
3. Adjust figure/table numbering
4. Double check references
5. Save as PDF
6. Change something in 1, repeat 2-5
7. Get feedback (f*ck!!), repeat 1-5

My First Workflow

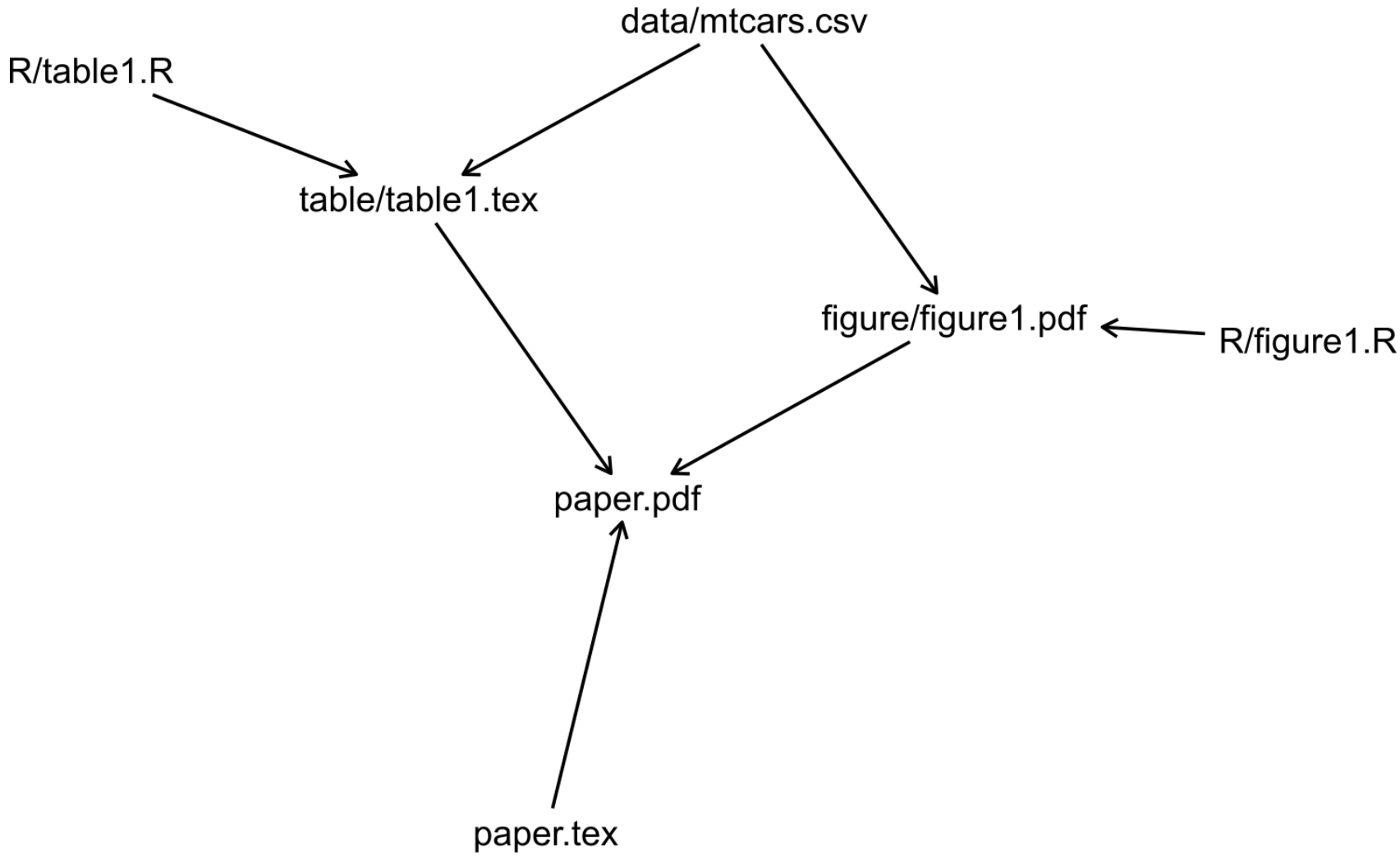
1. Make figure/table/analysis in R
2. Copy/paste into Word document
3. Adjust figure/table numbering
4. Double check references
5. Save as PDF
6. Change something in 1, repeat 2-5
7. Get feedback (f*ck!!), repeat 1-5
8. Get reviews (f*ck!!!!!), repeat 1-5

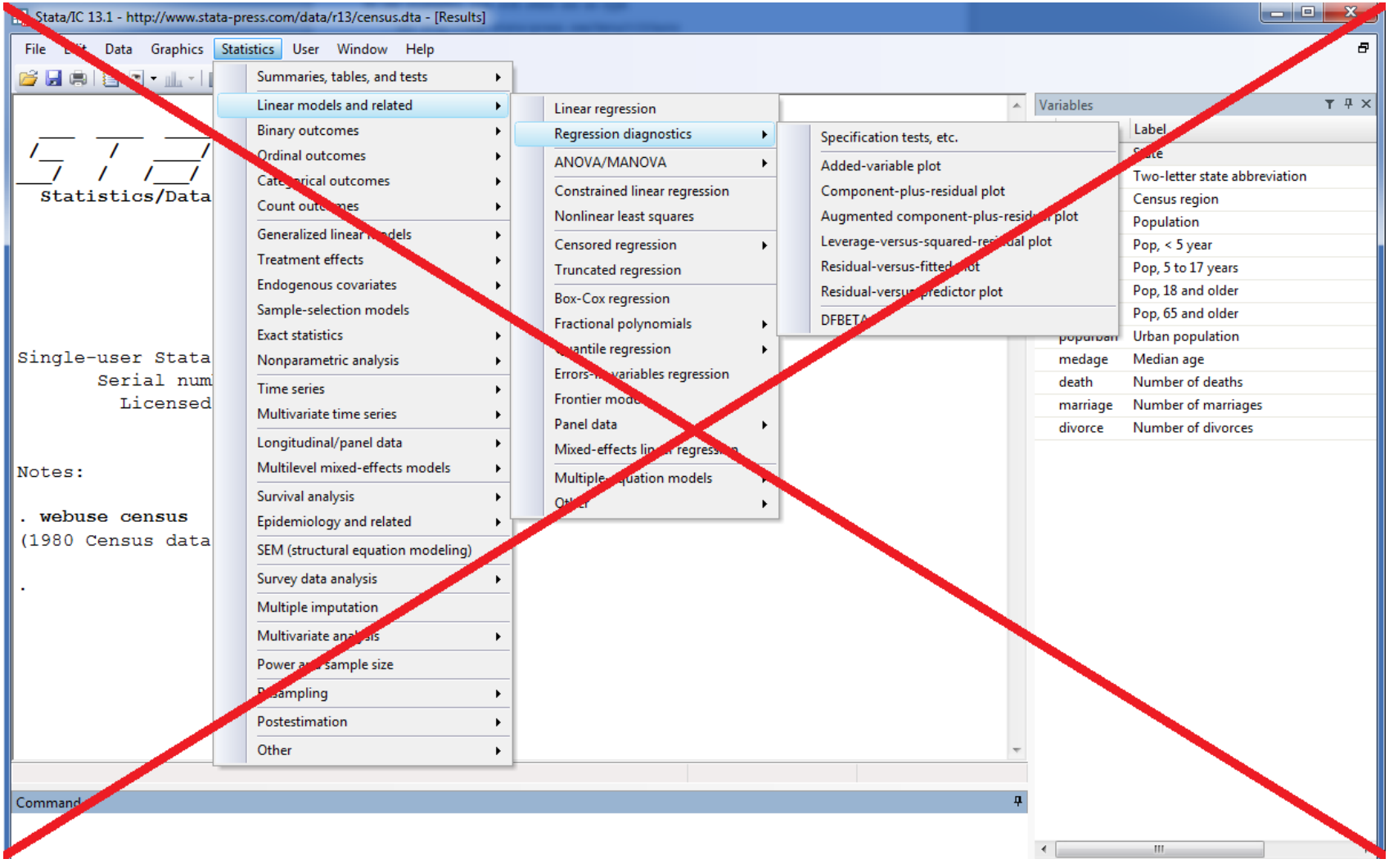
My First Workflow

1. Make figure/table/analysis in R
2. Copy/paste into Word document
3. Adjust figure/table numbering
4. Double check references
5. Save as PDF
6. Change something in 1, repeat 2-5
7. Get feedback (f*ck!!), repeat 1-5
8. Get reviews (f*ck!!!!!), repeat 1-5
9. Repeat 7 (f*ck!!!!!!!!!!!!!!!), repeat 1-5

Workflows as DAGs

- Reproducibility means executing a DAG
- DAG
  - Directed
  - Acyclic
  - Graph
- Files are *nodes*; workflows are *arrows*
- Example: `https://github.com/leeper/make-example`


data/mtcars.csv
R/table1.R
table/table1.tex
figure/figure1.pdf
R/figure1.R
paper.pdf
paper.tex


Stata/IC 13.1 - http://www.stata-press.com/data/r13/census.dta - [Results]
File Edit Data Graphics Statistics User Window Help
Statistics/Data
Single-user Stata
Serial num
Licensed
Notes:
. webuse census
(1980 Census data
.
Summaries, tables, and tests
Linear models and related
Binary outcomes
Ordinal outcomes
Categorical outcomes
Count outcomes
Generalized linear models
Treatment effects
Endogenous covariates
Sample-selection models
Exact statistics
Nonparametric analysis
Time series
Multivariate time series
Longitudinal/panel data
Multilevel mixed-effects models
Survival analysis
Epidemiology and related
SEM (structural equation modeling)
Survey data analysis
Multiple imputation
Multivariate analysis
Power and sample size
Resampling
Postestimation
Other
Linear regression
Regression diagnostics
ANOVA/MANOVA
Constrained linear regression
Nonlinear least squares
Censored regression
Truncated regression
Box-Cox regression
Fractional polynomials
Quantile regression
Errors-in-variables regression
Frontier models
Panel data
Mixed-effects linear regression
Multiple-equation models
Other
Specification tests, etc.
Added-variable plot
Component-plus-residual plot
Augmented component-plus-residual plot
Leverage-versus-squared-residual plot
Residual-versus-fitted plot
Residual-versus-predictor plot
DFBETA
Variables
Label
Two-letter state abbreviation
Census region
Population
Pop, < 5 year
Pop, 5 to 17 years
Pop, 18 and older
Pop, 65 and older
Urban population
medage Median age
death Number of deaths
marriage Number of marriages
divorce Number of divorces
Command

What's wrong with point-and-click?

What's wrong with point-and-click?

- Lose track of the DAG

What's wrong with point-and-click?

- Lose track of the DAG
- Won't comply with DA-RT verification policies

What's wrong with point-and-click?

- Lose track of the DAG
- Won't comply with DA-RT verification policies
- You will make mistakes!

What's wrong with point-and-click?

- Lose track of the DAG
- Won't comply with DA-RT verification policies
- You will make mistakes!
- Eventually, you will have wasted your entire life manually fixing references, figure/table cross-references, and making sure that all of your numbers are correctly rounded and p-values have the correct number of stars next to them!

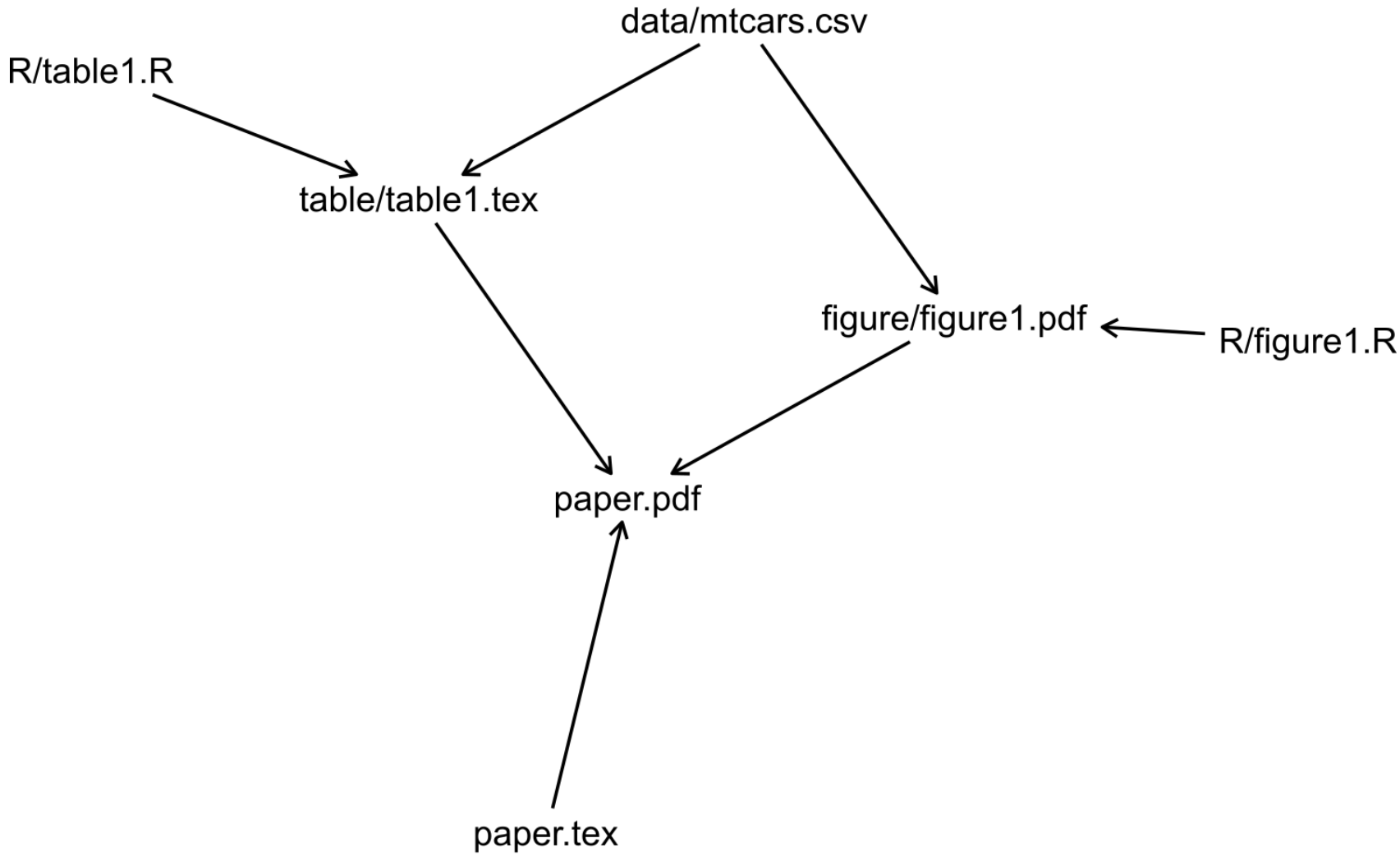
The prevalence of statistical reporting errors in psychology (1985–2013)

Michèle B. Nuijten[1] · **Chris H. J. Hartgerink**[1] · **Marcel A. L. M. van Assen**[1] · **Sacha Epskamp**[2] · **Jelte M. Wicherts**[1]

Published online: 23 October 2015

Abstract This study documents reporting errors in a sample of over 250,000 *p*-values reported in eight major psychology journals from 1985 until 2013, using the new R package "statcheck." statcheck retrieved null-hypothesis significance testing (NHST) results from over half of the articles from this period. In line with earlier research, we found that half of all published psychology papers that use NHST contained at least one *p*-value that was inconsistent with its test statistic and degrees of freedom. One in eight papers contained a grossly inconsistent *p*-value that may have affected the statistical conclusion. In contrast to earlier findings, we found that the average prevalence of inconsistent *p*-values has been stable over the years or has declined. The prevalence of gross inconsistencies was higher in *p*-values reported as significant than in *p*-values reported as nonsignificant. This could indicate a systematic bias in favor of significant results. Possible solutions for the high prevalence of reporting inconsistencies could be to encour-

Most conclusions in psychology are based on the results of null hypothesis significance testing (NHST; Cumming et al., 2007; Hubbard & Ryan, 2000; Sterling, 1959; Sterling, Rosenbaum, & Weinkam, 1995). Therefore, it is important that NHST is performed correctly and that NHST results are reported accurately. However, there is evidence that many reported *p*-values do not match their accompanying test statistic and degrees of freedom (Bakker & Wicherts, 2011; Bakker & Wicherts, 2014; Berle & Starcevic, 2007; Caperos & Pardo, 2013; Garcia-Berthou & Alcaraz, 2004; Veldkamp, Nuijten, Dominguez-Alvarez, Van Assen, & Wicherts, 2014; Wicherts, Bakker, & Molenaar, 2011). These studies highlighted that roughly half of all published empirical psychology articles using NHST contained at least one inconsistent *p*-value and that around one in seven articles contained a gross inconsistency, in which the reported *p*-value was significant and the computed


data/mtcars.csv
R/table1.R
table/table1.tex
figure/figure1.pdf
R/figure1.R
paper.pdf
paper.tex

Four Basic Workflows

Four Basic Workflows

1. Do everything in one file

Four Basic Workflows

1. Do everything in one file

2. Master file calls code for one-file-per-output

Four Basic Workflows

1. Do everything in one file
2. Master file calls code for one-file-per-output
3. make ("code within workflow")

Four Basic Workflows

1. Do everything in one file
2. Master file calls code for one-file-per-output
3. make ("code within workflow")
4. knitr/rmarkdown ("workflow within code")

Everything in One File

```
# Brexit Deservingnes Experiment Analysis
# setwd("c:/users/thomas/dropbox/brexitdeservingness/")

# load data
dat <- rio::import("data/LSE_Hobolt_May18_Client.sav")
stopifnot(identical(dim(dat), c(3273L, 62L)))

# Regression analysis: perceived deservingness
stargazer::stargazer(
  # reduced model (only leavers and remainers) with interaction
  lm(opinion ~ identity * condition, data = subset(dat, identity %in% c("A Leav
  type = "tex",
  out = "figures/results-deservingness.tex",
  star.char = c("*"),
  star.cutoffs = c(0.05),
  notes = c("* $p<0.05$"),
  notes.append = FALSE,
  model.numbers = FALSE,
  float = FALSE,
  digits = 2,
  align = TRUE
)
```

One-File-Per-Output

```
# Preference Trial Experiment Analysis
# Thomas J. Leeper
# 2018-06-25
#setwd("C:/Users/Thomas/Dropbox/KnowledgeGaps")

# code
library("car")
library("xtable")
library("GK2011")
source("Analysis/functions.R")

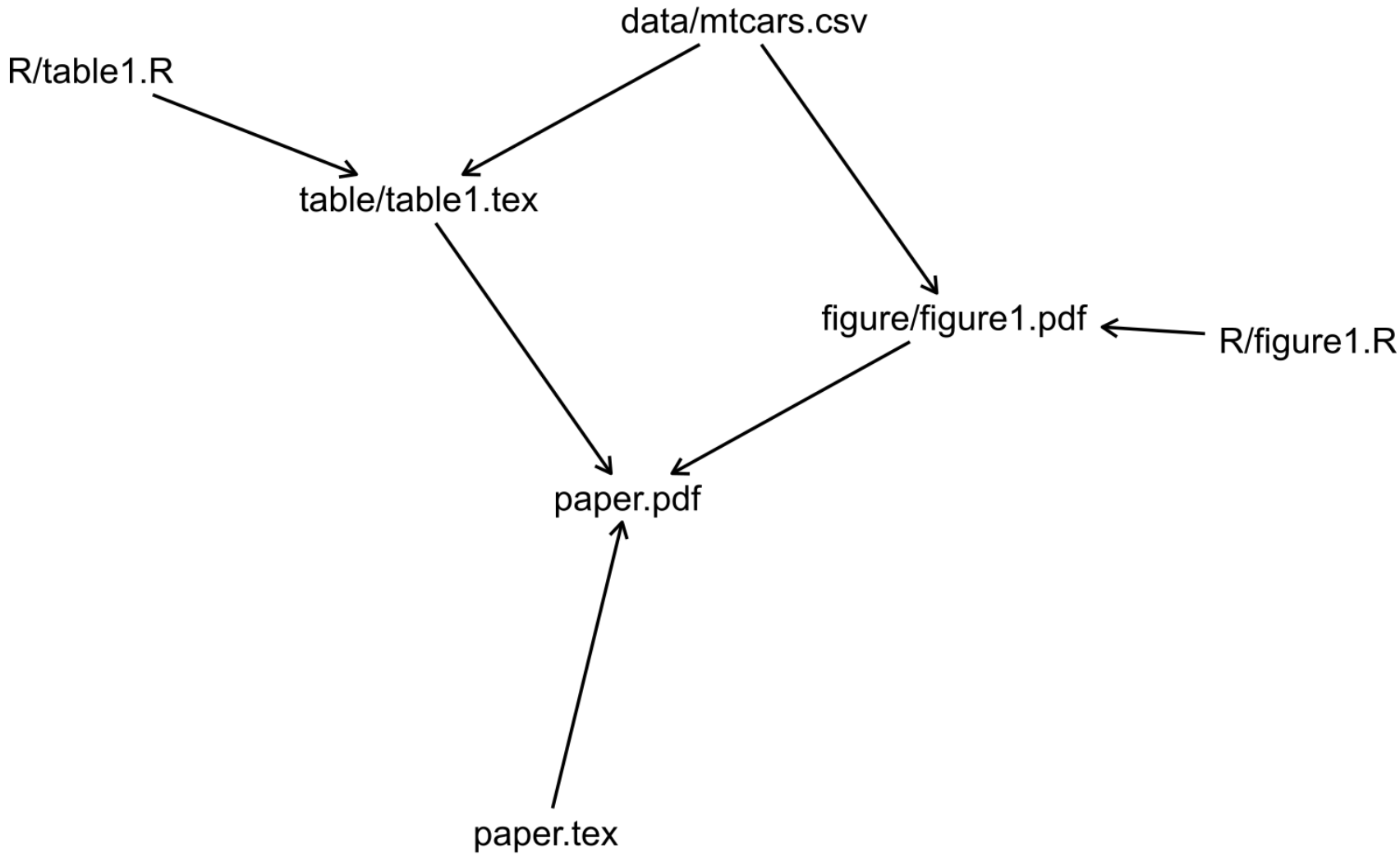
# recoding
source("Analysis/experiment_cleaning.R")

# demographics
source("Analysis/experiment_demographics.R", echo = TRUE)

## Main analysis
source("Analysis/experiment_knowledge.R")

## Appendix
source("Analysis/experiment_appendix.R")
```

What's missing from these workflows?


data/mtcars.csv
R/table1.R
table/table1.tex
figure/figure1.pdf
R/figure1.R
paper.pdf
paper.tex

make with a makefile

```
all: paper.pdf

figure/figure1.pdf: R/figure1.R data/mtcars.csv
    Rscript R/figure1.R

table/table1.tex: R/table1.R data/mtcars.csv
    Rscript R/table1.R

paper.pdf: paper.tex figure/figure1.pdf table/table1.tex
    pdflatex $<
    pdflatex $<
    bibtex $<
    pdflatex $<
```

Dynamic Documents (Rmarkdown)

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

Dynamic Documents (Rmarkdown)

- 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

Rmarkdown

1. YAML metadata header

```
---
title: My Manuscript
author: Thomas J. Leeper
---
```

2. Document contents in **markdown**

```
# A header
## A subhead
This is my manuscript, **bold** and *italic*.
```

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

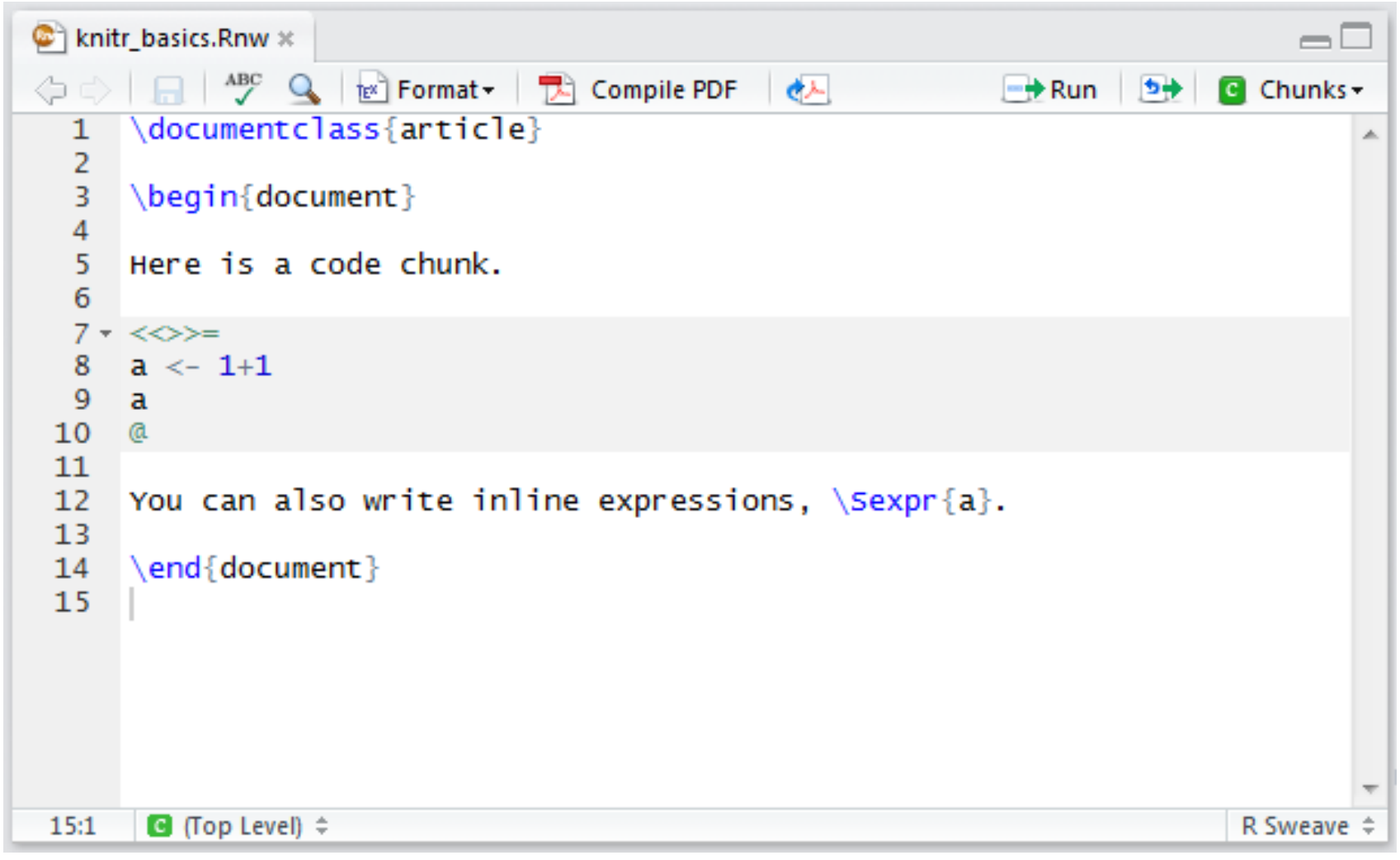
Chunk Options

```
```{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()
```
```

You can work in LaTeX, too!



You can work in LaTeX, too!

```
\begin{document}

Here is a code chunk.

\begin{knitrout}
\definecolor{shadecolor}{rgb}{0.969, 0.969, 0.969}\color{fgcolor}\begin{kframe}
\begin{alltt}
\hlstd{a} \hlkwb{<-} \hlnum{1}\hlopt{+}\hlnum{1}
\hlstd{a}
\end{alltt}
\begin{verbatim}
## [1] 2
\end{verbatim}
\end{kframe}
\end{knitrout}

You can also write inline expressions, 2.

\end{document}
```

You can work in LaTeX, too!

Here is a code chunk.

```
a <- 1+1
a

## [1] 2
```

You can also write inline expressions, 2.

How do you pick a workflow?

- There is no one-size-fits-all workflow!
- Decide what works for you for a given project with particular collaborators
- I use multiple workflows on different projects

Questions?

Activity!

What tools do you use to store, share, and/or archive your research materials?

Keeping things

Three ways of thinking about how you keep and store your research materials:

Keeping things

Three ways of thinking about how you keep and store your research materials:

1. Collaborating with yourself or others in the future
    - Going back in time for long-lived projects
    - Verification at publication stage

Keeping things

Three ways of thinking about how you keep and store your research materials:

1. Collaborating with yourself or others in the future
   - Going back in time for long-lived projects
   - Verification at publication stage
2. Collaborating with others now
   - Collaborating simultaneously
   - Collaborating asynchronously

Keeping things

Three ways of thinking about how you keep and store your research materials:

1. Collaborating with yourself or others in the future
   - Going back in time for long-lived projects
   - Verification at publication stage
2. Collaborating with others now
   - Collaborating simultaneously
   - Collaborating asynchronously
3. Collaborating with others after you die
   - Future reproducibility requests

Keeping things

Live Collaboration

Other Collaboration

Keeping things

Live Collaboration

- Google Docs
- Overleaf
- Dropbox/Box/etc.
- Email?

Other Collaboration

Keeping things

Live Collaboration

- Google Docs
- Overleaf
- Dropbox/Box/etc.
- Email?

Other Collaboration

- Active project: Version control (git)
- Backup: Dropbox, GDrive, S3, Github

Keeping things

Live Collaboration

- Google Docs
- Overleaf
- Dropbox/Box/etc.
- Email?

Other Collaboration

- Active project: Version control (git)
- Backup: Dropbox, GDrive, S3, Github
- Archiving: Dataverse, Zenodo, Figshare, OSF

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 reproducibility?

Why use Git for reproducibility?

- 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 reproducibility?

- 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 reproducibility?

- 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

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!

Using Git

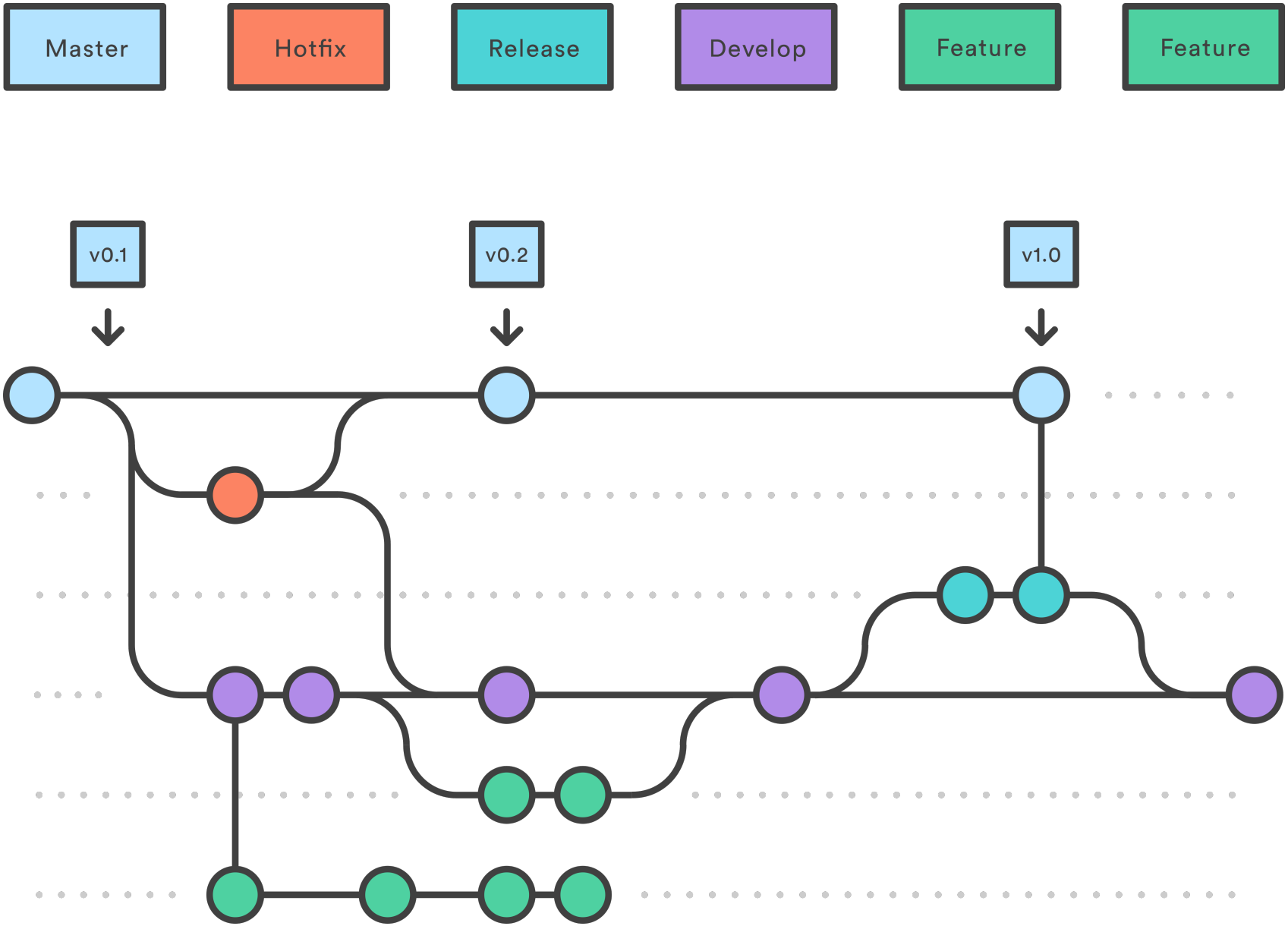
- Git create a "local repository" file that you can interact with using a number of tools
  - Command-line `git`
  - Git Bash
  - Git GUI
  - GitHub Desktop
  - RStudio (via "Projects")
  - GitHub/Bitbucket/GitLab web interfaces
  - Gitkraken
  - git2r (R package)
  - ...

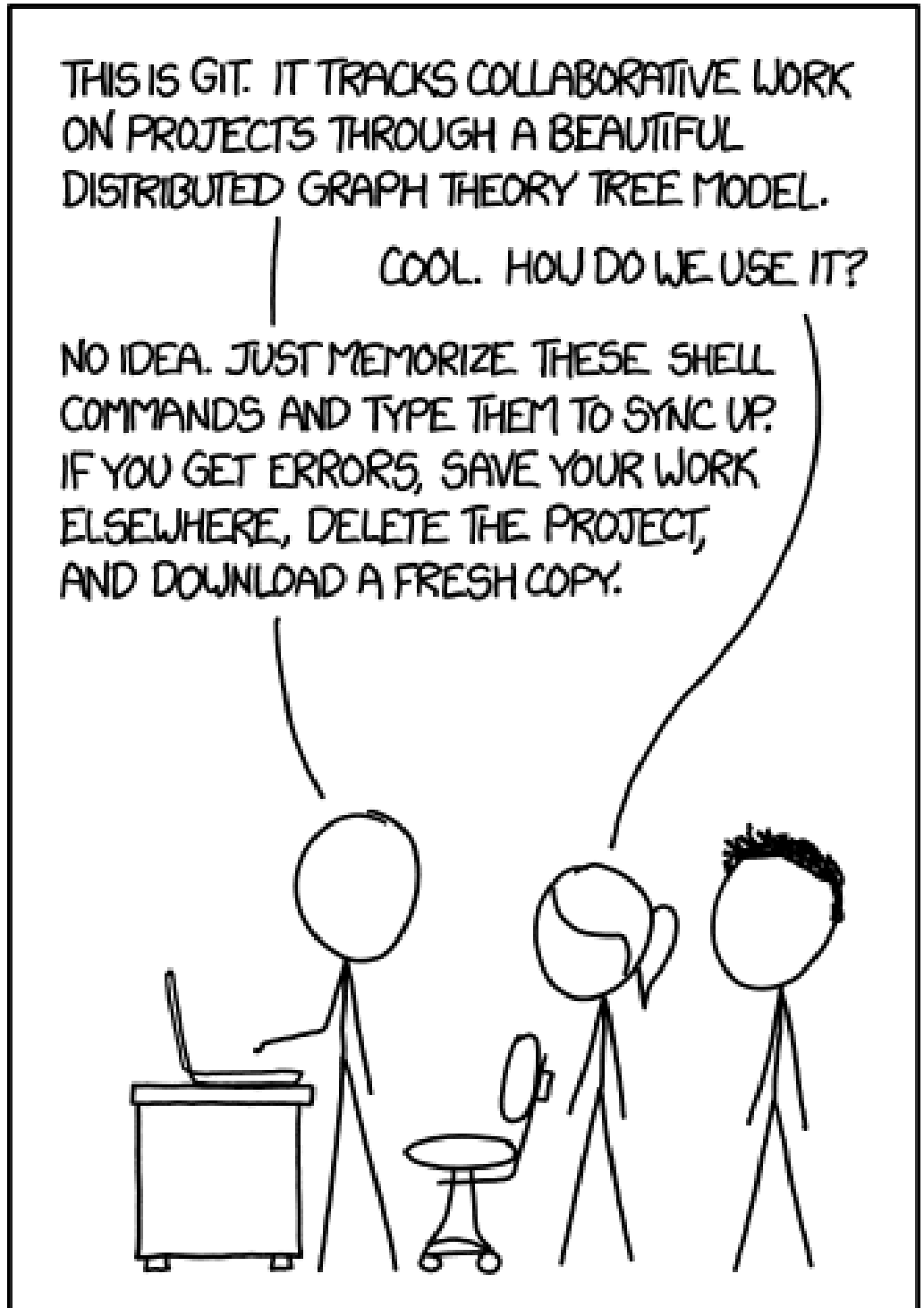
```
git --version
git
git config --global user.name "My Name"
git config --global user.email "me@example.com"
git config --list
```

```
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
git log
```

90% of What You Need

- `git add` (stage) or `git rm` (unstage)
- `git commit`
- `git status`, `git log`
- `git remote`
  - `git push`
  - `git pull`
- `git branch`
  - `git merge`


Master
Hotfix
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0


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.

Questions?

Hands-On Practice

1. Work together on migrating a workflow
2. Dig through replication archives
3. Work individually or in pairs on making workflow more reproducible

Let's vote: What should we do?

Questions?

Learning Objectives

1. Define reproducibility and replicability
2. Understand how to organize a reproducible research project
3. Recognize different approaches to reproducibility and tools for implementing various reproducible workflows
4. Understand how to collaborate reproducibly

Other stuff we didn't get to

- Project management and collaboration
- Reproducible data gathering and protocols
- Licensing and copyright
- Data privacy and anonymization
- Writing and distributing packages

Key Takeaways

- Once you work reproducibly, you'll never want to go back to your old workflow

Key Takeaways

- Once you work reproducibly, you'll never want to go back to your old workflow
- "Advanced" workflows (e.g., make, git) get complicated — StackOverflow is your friend

Key Takeaways

- Once you work reproducibly, you'll never want to go back to your old workflow
- "Advanced" workflows (e.g., make, git) get complicated — StackOverflow is your friend
- Collaborators probably don't know how to (or want to) use these tools

Key Takeaways

- Once you work reproducibly, you'll never want to go back to your old workflow
- "Advanced" workflows (e.g., make, git) get complicated — StackOverflow is your friend
- Collaborators probably don't know how to (or want to) use these tools
- Reproducibility is selfish first and for science second!

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!

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!

```
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 !!

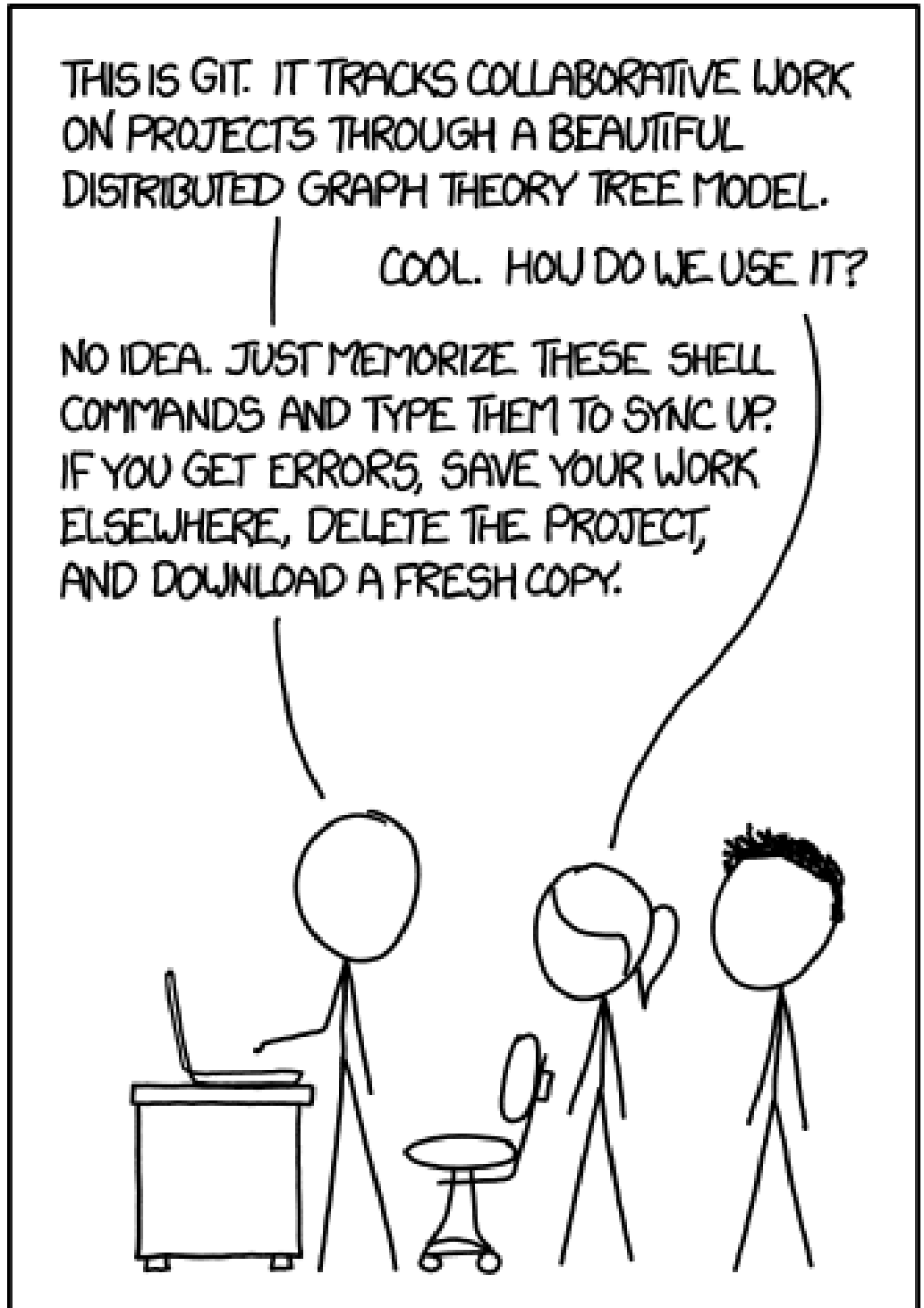
The StackOverflow Question

```
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

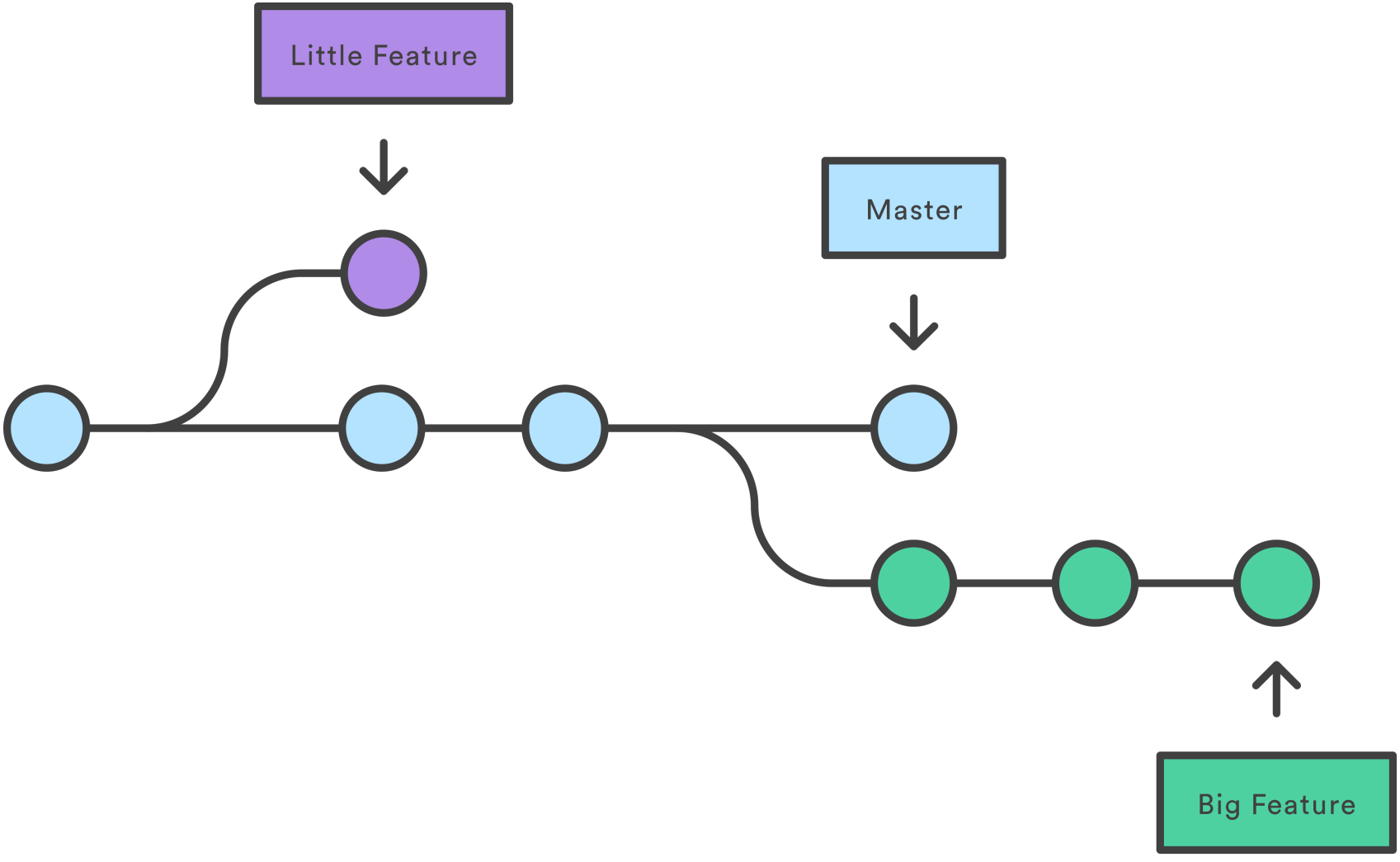
```
git status
git log
git checkout <commit hash>
git status
ls
cat README.md
git checkout master
```

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

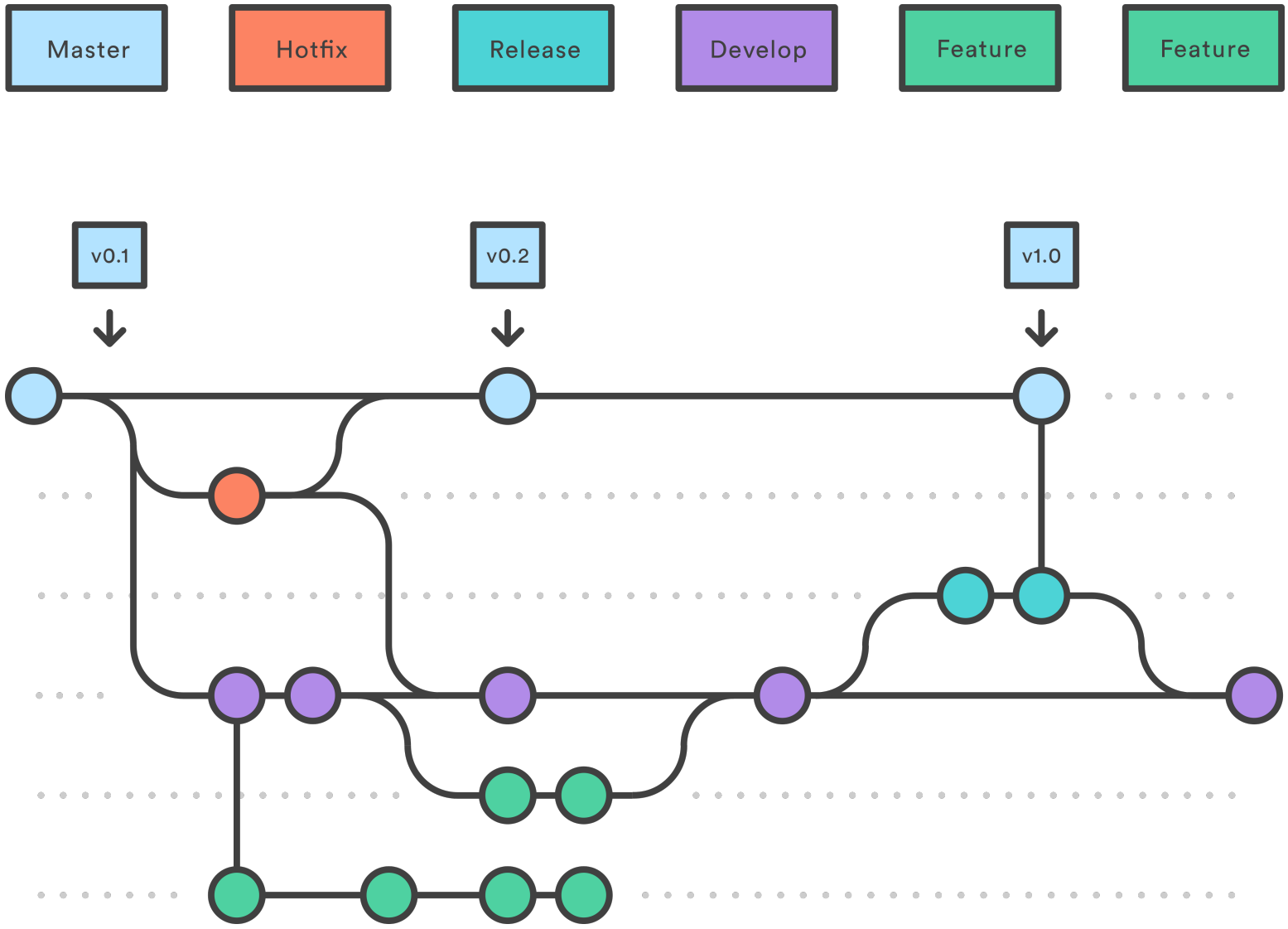

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

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
```

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
```

```
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