

Setup Today

1. Keep www.happygitwithr.com open
2. Go to www.github.com and make a free account
 - Pick a professional, short username; it's hard to change later. More tips at Chapter 4.1 of happygitwithr, <https://happygitwithr.com/github-acct.html#username-advice>
3. Make sure you have a recent version (v1.1 or later) of RStudio <https://www.rstudio.com/products/rstudio/download/#download>
4. Download these slides via:
<https://bit.ly/2XHXyl1>.

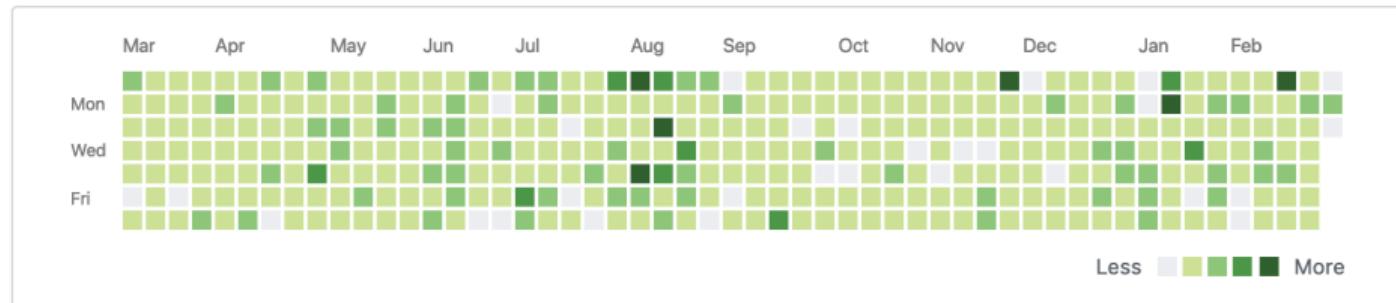
Happy Git and GitHub for the useR

Jenny Bryan, the STAT 545 TAs, Jim Hester

Let's Git started



2,412 contributions in the last year



Git for Students in the Social Sciences*: A Pitch

(* not software developers)

Shiro Kuriwaki

Presented March 5, 2019; Last updated March 6, 2019

Figure: From <https://github.com/kuriwaki>

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from **Annie Wang**)

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from [Annie Wang](#))
- ▶ I do some software development,
- ▶ but most of my work is applied (“substantive”)

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from [Annie Wang](#))
- ▶ I do some software development,
- ▶ but most of my work is applied (“substantive”)

An open question

- ▶ Version control is mandatory for programmers (and professional data scientists)

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from [Annie Wang](#))
- ▶ I do some software development,
- ▶ but most of my work is applied (“substantive”)

An open question

- ▶ Version control is mandatory for programmers (and professional data scientists)
- ▶ but does it make sense for *applied* researchers who ...
- ▶ work with datasets that are

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from [Annie Wang](#))
- ▶ I do some software development,
- ▶ but most of my work is applied (“substantive”)

An open question

- ▶ Version control is mandatory for programmers (and professional data scientists)
- ▶ but does it make sense for *applied* researchers who ...
- ▶ work with datasets that are **with collaborators, large,**

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from [Annie Wang](#))
- ▶ I do some software development,
- ▶ but most of my work is applied (“substantive”)

An open question

- ▶ Version control is mandatory for programmers (and professional data scientists)
- ▶ but does it make sense for *applied* researchers who ...
- ▶ work with datasets that are **with collaborators, large, unstructured,**

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from [Annie Wang](#))
- ▶ I do some software development,
- ▶ but most of my work is applied (“substantive”)

An open question

- ▶ Version control is mandatory for programmers (and professional data scientists)
- ▶ but does it make sense for *applied* researchers who ...
- ▶ work with datasets that are **with collaborators, large, unstructured, and prone to change?**

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from [Annie Wang](#))
- ▶ I do some software development,
- ▶ but most of my work is applied (“substantive”)

An open question

- ▶ Version control is mandatory for programmers (and professional data scientists)
- ▶ but does it make sense for *applied* researchers who ...
- ▶ work with datasets that are **with collaborators, large, unstructured, and prone to change?**

My perspective

Yes! But in moderation and in *lite*
These slides: A pitch, acknowledging the tradeoff,

About me

- ▶ G-4 in Government (American Politics, elections and representation)
- ▶ Before: Political data analytics (where I learned git from [Annie Wang](#))
- ▶ I do some software development,
- ▶ but most of my work is applied (“substantive”)

An open question

- ▶ Version control is mandatory for programmers (and professional data scientists)
- ▶ but does it make sense for *applied* researchers who ...
- ▶ work with datasets that are **with collaborators, large, unstructured, and prone to change?**

My perspective

Yes! But in moderation and in *lite*
These slides: A pitch, acknowledging the tradeoff, rather than a full workshop or manual.
(Extensions in final slides).

Setting expectations: Is it worth it?

What do Gentzkow and Shapiro say?

Definitely:

"It will probably take you a couple days to set up a repository and learn how you want to interact with [version control]. You will break even on that time investment within a month or two."^a

^a "Code and Data for Social Sciences: A Practitioners Guide." 2014. <https://perma.cc/5J9D-BTD6>. Although I'm not sure about learning version control in "a couple of days" (I certainly couldn't!), I can guarantee reading their guide in its entirety is a time investment you'll break even on immediately.

But takeup is still low,^a

and alternatives have attractive features too:



Version control with Google Drive



Brett-Johnson

2018-01-08

I've experimented using Google Drive and GitHub with my team (a small ecological research team) for version control and collaboration. I've found that both have their uses and I'm keen to share how I've been doing it so that I can hear from others how they are doing things, and whether I'm on the right track.

I initially started off committing everything I worked on to GitHub in different sub folders in the same repo. All of my internal analyses that aren't meant for a public report or peer reviewed paper went into different folders in the same general 'internal' private repo. This worked all right when it was just me using the repo. But when I brought a co-worker into the mix,  I realized what a pain it actually is to try to collaborate on GitHub on a day to day basis. We were spending a load of time messing around with merge conflicts and all sorts of other un-intuitive issues.  We felt GitHub was cumbersome for day to day analysis collaboration internally.

So now I would like to move back to simply using Google Drive for internal analyses. Google drive is great for version controlling (especially now that you can 'name versions' in Google Drive similar to a GitHub commit). I sometimes rely on the revision history of Google Drive to actually roll back a script, because it's way more intuitive than doing that in Git not to mention that every time you save your script in, it gets an un-named version in Google Drive, so the chances of not losing your work is actually greater using Google Drive. Google Drive allows you share all the files you and data you need, and using the here() package we shouldn't have to worry about working directories.

^a Anecdotally, I can count full Git users in my dept. in one hand.

Version Control with Yourself (and Your Past Selves)

Terminology I (and my recommended setup)

- ▶ version control **tracks** each file for its content changes

Terminology I (and my recommended setup)

- ▶ version control **tracks** each file for its content changes
- ▶ **Git** is a particular type of software for version control
(Subversion, or SVN, is an alternative)



Terminology I (and my recommended setup)

- ▶ version control **tracks** each file for its content changes
- ▶ **Git** is a particular type of software for version control
(Subversion, or SVN, is an alternative)
- ▶ **GitHub** is an app (acquired by Microsoft) to host git on the web
(Bitbucket and GitLab are alternatives)



Terminology I (and my recommended setup)

- ▶ version control **tracks** each file for its content changes
- ▶ **Git** is a particular type of software for version control (Subversion, or SVN, is an alternative)
- ▶ **GitHub** is an app (acquired by Microsoft) to host git on the web (Bitbucket and GitLab are alternatives)
- ▶ A **desktop client** is an app that connects a webhost like Github to your computer and facilitates tasks otherwise done by **command-line** (here I use **RStudio**, Github Desktop is an alternative)



Terminology I (and my recommended setup)

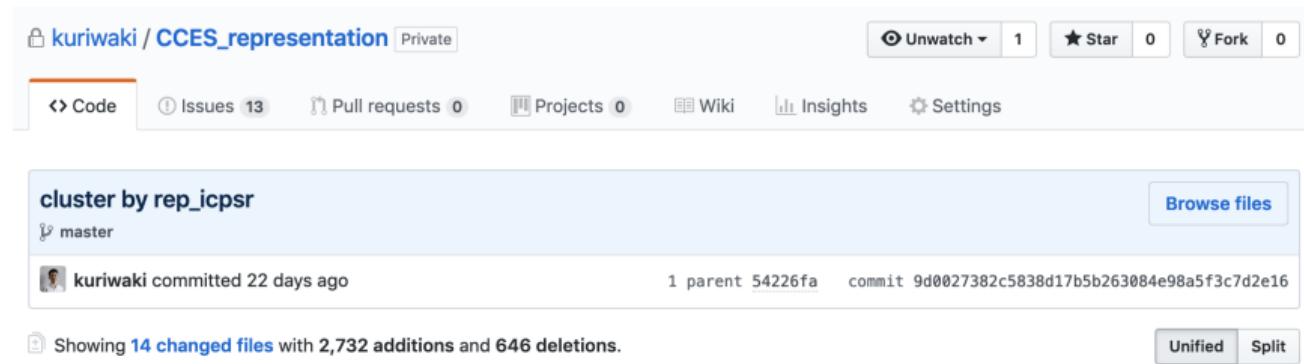
- ▶ version control **tracks** each file for its content changes
- ▶ **Git** is a particular type of software for version control (Subversion, or SVN, is an alternative)
- ▶ **GitHub** is an app (acquired by Microsoft) to host git on the web (Bitbucket and GitLab are alternatives)
- ▶ A **desktop client** is an app that connects a webhost like Github to your computer and facilitates tasks otherwise done by **command-line** (here I use **RStudio**, Github Desktop is an alternative)
- ▶ A **repository** is the fundamental unit of version control project: A parent folder with a hidden folder called `.git` with the tracking technology. Do not make a repository within a repository!



Benefit 1: Keep track of how your results changed

Problem: You tweak a regression specification and re-run your script, re-writing dozens of tables.

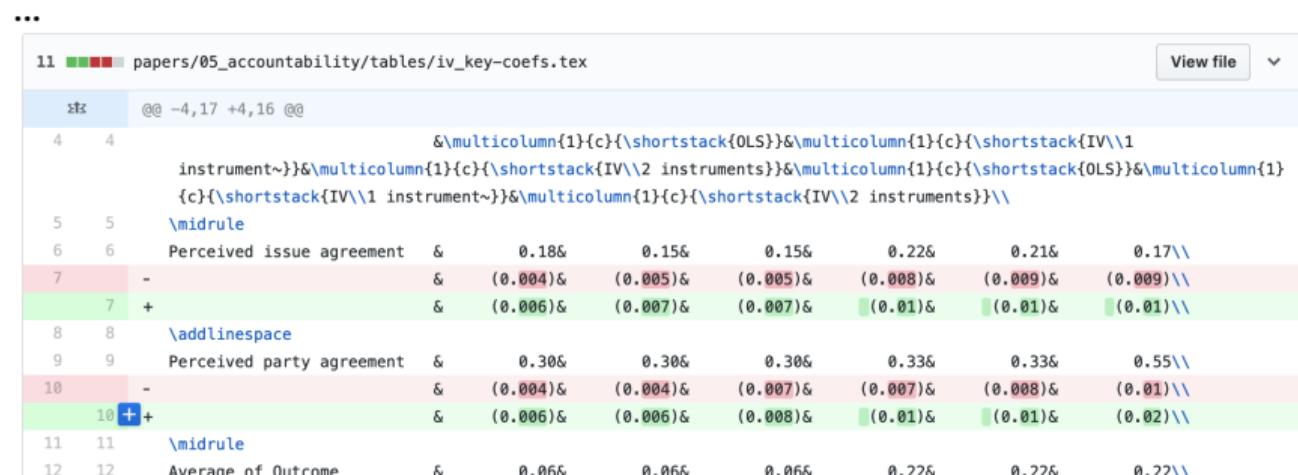
How much did your results change?



cluster by rep_icpsr

kuriwaki committed 22 days ago

Showing 14 changed files with 2,732 additions and 646 deletions.



11 papers/05_accountability/tables/iv_key-coefs.tex

Line	Change	Text
4	4	@@ -4,17 +4,16 @@
5	5	instrument~}}&\multicolumn{1}{c}{\shortstack{OLS}}&\multicolumn{1}{c}{\shortstack{IV\\1\\instrument~}}&\multicolumn{1}{c}{\shortstack{IV\\2 instruments}}&\multicolumn{1}{c}{\shortstack{OLS}}&\multicolumn{1}{c}{\shortstack{IV\\1 instrument~}}&\multicolumn{1}{c}{\shortstack{IV\\2 instruments}}\\
6	6	\midrule
7	-	Perceived issue agreement & 0.18& 0.15& 0.15& 0.22& 0.21& 0.17\\
7	+	& (0.004)& (0.005)& (0.005)& (0.008)& (0.009)& (0.009)\\
8	8	\addlinespace
9	9	Perceived party agreement & 0.30& 0.30& 0.30& 0.33& 0.33& 0.55\\
10	-	& (0.004)& (0.004)& (0.007)& (0.007)& (0.008)& (0.01)\\
10	+	& (0.006)& (0.006)& (0.008)& (0.01)& (0.01)& (0.02)\\
11	11	\midrule
12	12	Average of Outcome & 0.06& 0.06& 0.06& 0.22& 0.22& 0.22\\

Benefit 1: Keep track of how your results changed

You collect more data and re-run the regressions.

Now how did
the results
change?

after adding 2009 module remaineder (about 3000)

master

12  papers/05_accountability/tables/iv_key-coefs.tex

View file

```
@@ -3,18 +3,18 @@
3   3 &\multicolumn{3}{c}{Outcome: Approval} &\multicolumn{3}{c}{Outcome: Vote}
4   4 \\\\cmidrule(lr){2-4}\\cmidrule(lr){5-7}
5   5 &\multicolumn{1}{c}{\shortstack{OLS}}&\multicolumn{1}{c}{\shortstack{IV\\
instrument~}}&\multicolumn{1}{c}{\shortstack{IV\\2 instruments}}&\multicolumn{1}{c}{\shortstack{OLS}}&\multicolumn{1}{c}{\shortstack{IV\\1 instrument~}}&\multicolumn{1}{c}{\shortstack{IV\\2 instruments}}\\
6   6 \midrule
6 + 6 - Perceived issue agreement & 0.18& 0.15& 0.15& 0.22& 0.21& 0.17\\
6 + 6 + Perceived issue agreement & 0.19& 0.17& 0.16& 0.22& 0.20& 0.17\\
7   7 & (0.006)& (0.007)& (0.007)& (0.01)& (0.01)& (0.01)\\
8   8 \addlinespace
9   9 - Perceived party agreement & 0.30& 0.30& 0.30& 0.33& 0.33& 0.55\\
10  10 - & (0.006)& (0.006)& (0.008)& (0.01)& (0.01)& (0.02)\\
10  10 + Perceived party agreement & 0.29& 0.30& 0.32& 0.33& 0.33& 0.55\\
10  10 + & (0.006)& (0.006)& (0.009)& (0.01)& (0.01)& (0.02)\\
11  11 \midrule
12  12 Average of Outcome & 0.06& 0.06& 0.06& 0.22& 0.22& 0.22\\
13  13 Std. Dev. of Outcome & 0.69& 0.69& 0.69& 0.85& 0.85& 0.85\\
14  14 - R-squared & 0.40& 0.40& 0.40& 0.39& 0.39& 0.36\\
15  15 - First Stage F-stat & & & 30,249& & & 16,343\\
14  14 + R-squared & & 0.39& 0.39& 0.39& 0.39& 0.36\\
15  15 + First Stage F-stat & & & 34,367& & & 15,089\\
16  16 Clusters & 847& 847& 847& 740& 739& 739\\
17  17 - Observations & 43,466& 43,427& 43,427& 23,619& 23,603& 23,603\\
17  17 + Observations & 45,605& 45,556& 45,556& 23,619& 23,603& 23,603
```

Benefit 2: Tracking your paper versions

Problem: You start writing up your paper, draft.tex

- The next day, you make a new draft. Do you overwrite?

Benefit 2: Tracking your paper versions

Problem: You start writing up your paper, draft.tex

- ▶ The next day, you make a new draft. Do you overwrite?
- ▶ Or do you call it draft_0305.tex ?
draft_03052019.tex ?

Benefit 2: Tracking your paper versions

Problem: You start writing up your paper, draft.tex

- ▶ The next day, you make a new draft. Do you overwrite?
- ▶ Or do you call it `draft_0305.tex` ?
`draft_03052019.tex` ?
- ▶ The next week, you find a single typo. Do you “Save As” with a new date?

Benefit 2: Tracking your paper versions

Problem: You start writing up your paper, draft.tex

- ▶ The next day, you make a new draft. Do you overwrite?
- ▶ Or do you call it `draft_0305.tex` ?
`draft_03052019.tex` ?
- ▶ The next week, you find a single typo. Do you “Save As” with a new date?
- ▶ Three weeks later, you return to your paper. Your computer indicates that the file named `draft_0305.tex` was “Last modified March 12, 2019”.

Benefit 2: Tracking your paper versions

Problem: You start writing up your paper, `draft.tex`

- ▶ The next day, you make a new draft. Do you overwrite?
 - ▶ Or do you call it `draft_0305.tex` ?
`draft_03052019.tex` ?
 - ▶ The next week, you find a single typo. Do you “Save As” with a new date?
 - ▶ Three weeks later, you return to your paper. Your computer indicates that the file named `draft_0305.tex` was “Last modified March 12, 2019”.

Showing 5 changed files with 62 additions and 51 deletions.

Unified Split

2 analyze/06_rcv_accountability.do

View file

```
@@ -275,7 +275,7 @@ esttab est1 est3 est5 est2 est4 est6 using "papers/05_accountability/tables/iv_k
275 275     span erepeat(\cmidrule(lr){span}) //!
276 276     mtitle("shortstack(OLS)" "\shortstack{IV\\1 instrument}" "\shortstack{IV\\2 instruments}" "\shortstack(OLS)" "\short
277 277     b2(se(1)) //!
278 - addnotes("All other variables and intercept not shown") //!
278 + addnotes("All other variables and intercept not shown. All IV estimates include year fixed effects.") //!
279 279     stats(lymean ysd r2 N, //
280 280     fmt(2 2 2 %6.0fc) //
281 281     labels("Average of Outcome" "Std. Dev. of Outcome" "R-squared" "Observations")) //!
```

82 papers/05_accountability/ajk.tex

View file

259 259

260 260 `\subsection{Operationalization of Key Variables} \label{sec:operationalization}`

261 261

262 - Our key measures of perceived agreement are built from the responses to the perception questions in the CCES Module (a random subset of the entire study), combined with their own stances to the same issues on the same question. An example of such perception questions from 2017 is reproduced in Exhibit `\ref{fig:perception_question}`.

262 + Our key measures of perceived agreement are built from the responses to the perception questions in the CCES Module (a random subset of the entire study), combined with their own stances to the same issues on the same question. An example of such perception questions from 2017 is reproduced in Exhibit `\ref{fig:perception_question}`. Throughout, we limit our attention to the House primarily due to space restrictions.

263 263

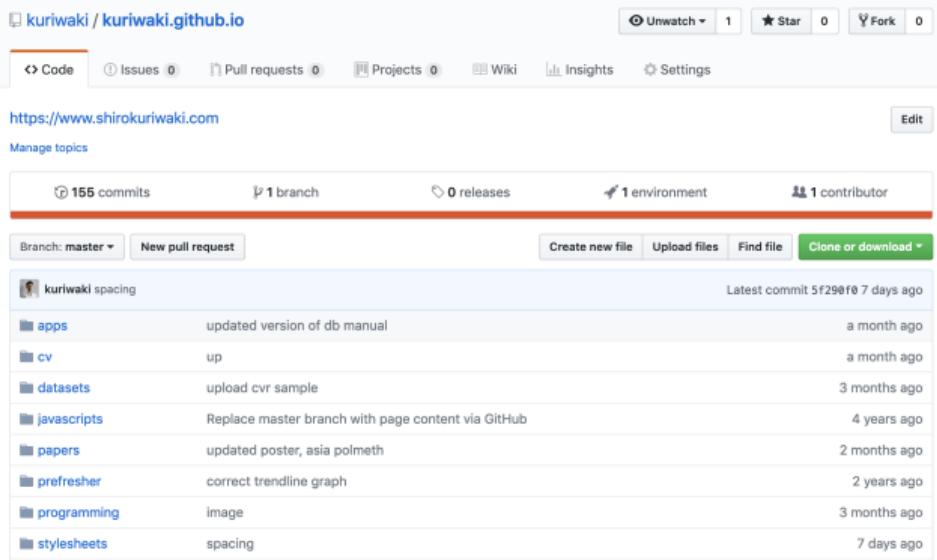
264 + We also construct instruments for our measure of perceived agreement by collecting matching roll call vote data from the `NOMINATE` database (`\url{https://voteview.com/}`). To facilitate the interpretability of regression coefficients, we intentionally define all of variables on a `-1` to `+1` scale. Table `\ref{tab:summary_stats}` presents summary statistics, and a description of each of the variables follows.

264 + We also construct instruments for our measure of perceived agreement by collecting matching roll call vote data from the `Voteview` database (`\url{https://voteview.com/}`). To facilitate the interpretability of regression coefficients, we intentionally define all of variables on a `-1` to `+1` scale. Table `\ref{tab:summary_stats}` presents summary statistics, and a description of each of the variables follows.

Benefit 3: And more cool stuff like

Getting a free, customizable,
ad-free website

(instead of a click-and-drag
Wordpress/Squarespace website)

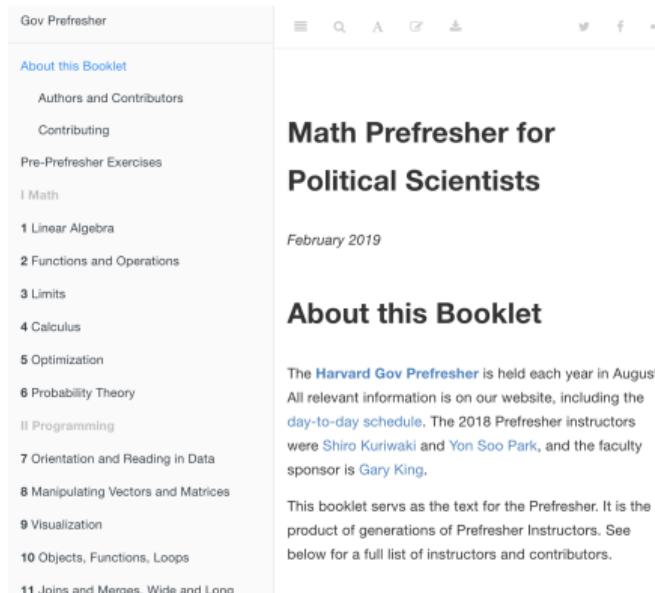


The screenshot shows a GitHub repository page for the user 'kuriwaki' with the repository name 'kuriwaki.github.io'. The page includes navigation tabs for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. It also shows metrics: 155 commits, 1 branch, 0 releases, 1 environment, and 1 contributor. A dropdown for the branch 'master' is open. Below the header, a list of commits is displayed, each with a small icon, the author, a brief description, and the time of the commit. The commits are as follows:

Author	Commit Message	Time
kuriwaki	spacing	7 days ago
apps	updated version of db manual	a month ago
cv	up	a month ago
datasets	upload cvr sample	3 months ago
javascripts	Replace master branch with page content via GitHub	4 years ago
papers	updated poster, asia polmeth	2 months ago
prefresher	correct trendline graph	2 years ago
programming	image	3 months ago
stylesheets	spacing	7 days ago

Benefit 3: And more cool stuff like

Work on a collaborative workbook
(instead of needing to add people to
your Dropbox)



The screenshot shows a website layout for the "Gov Prefresher". The left sidebar contains a navigation menu with the following items:

- About this Booklet
- Authors and Contributors
- Contributing
- Pre-Prefresher Exercises
- I Math
 - 1 Linear Algebra
 - 2 Functions and Operations
 - 3 Limits
 - 4 Calculus
 - 5 Optimization
 - 6 Probability Theory
- II Programming
 - 7 Orientation and Reading in Data
 - 8 Manipulating Vectors and Matrices
 - 9 Visualization
 - 10 Objects, Functions, Loops
 - 11 Joins and Merges, Wide and Long

The main content area on the right features a title "Math Prefresher for Political Scientists" and a date "February 2019". Below the title is a section titled "About this Booklet" with the following text:

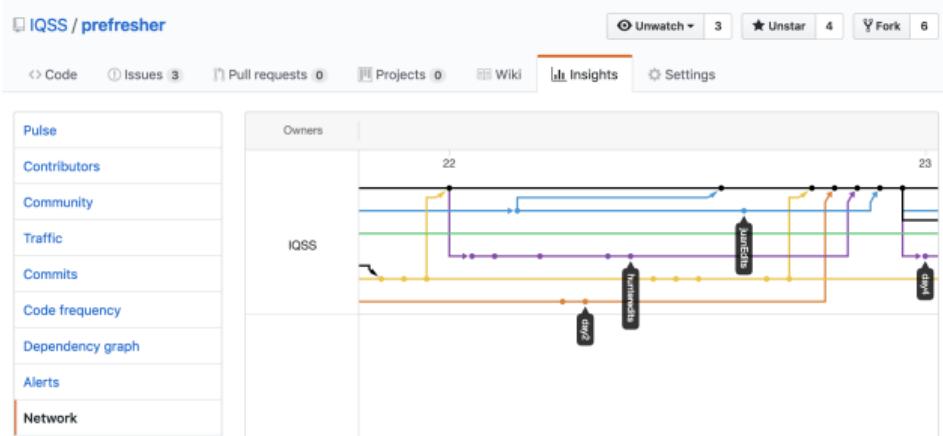
The [Harvard Gov Prefresher](#) is held each year in August. All relevant information is on our website, including the [day-to-day schedule](#). The 2018 Prefresher instructors were [Shiro Kuriwaki](#) and [Yon Soo Park](#), and the faculty sponsor is [Gary King](#).

This booklet serves as the text for the Prefresher. It is the product of generations of Prefresher Instructors. See below for a full list of instructors and contributors.

Benefit 3: And more cool stuff like

Work on a collaborative workbook

(instead of needing to add people to your Dropbox)

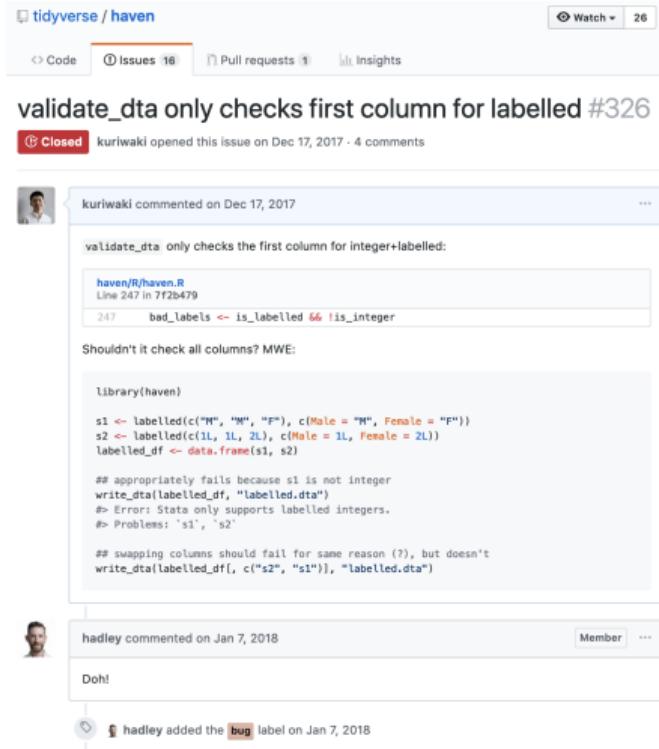


Benefit 3: And more cool stuff like

Contributing to / getting the latest on actual software packages

Github issues is the de facto communication of open-source developers.¹

As a researcher or lab group, **creating your own package** for internal use is not far-fetched: common functions and critical workhorse functions that get tweaked should be bundled in a R package and tracked. Try it out!



tidyverse / haven

Code Issues 16 Pull requests 1 Insights Watch 26

validate_dta only checks first column for labelled #326

Closed kuriwaki opened this issue on Dec 17, 2017 · 4 comments

kuriwaki commented on Dec 17, 2017

validate_dta only checks the first column for integer+labelled:

haven/R/haven.R
Line 247 in 7f72b479

```
247 bad_labels <- is_labelled && !is_integer
```

Shouldn't it check all columns? MWE:

```
library(haven)

s1 <- labelled(c("M", "M", "F"), c(Male = "M", Female = "F"))
s2 <- labelled(c(1L, 1L, 2L), c(Male = 1L, Female = 2L))
labelled_df <- data.frame(s1, s2)

## appropriately fails because s1 is not integer
write_dta(labelled_df, "labelled.dta")
#> Error: Stata only supports labelled integers.
#> Problems: 's1', 's2'

## swapping columns should fail for same reason (?), but doesn't
write_dta(labelled_df, c("s2", "s1")), "labelled.dta")
```

hadley commented on Jan 7, 2018

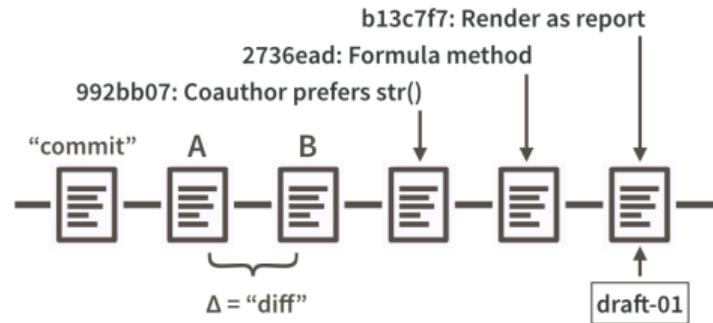
Doh!

hadley added the bug label on Jan 7, 2018

¹ The usual introduction to GitHub would spent most of the time discussing this topic! Package development in Git: <http://r-pkgs.had.co.nz/git.html>.

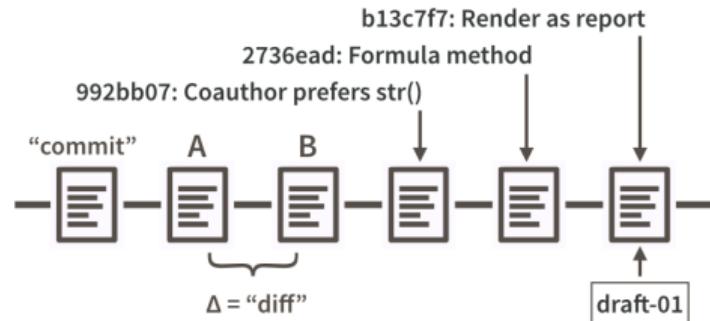
Terminology II: Pushing commits

- Files increment by **commits**. The line-by-line changes between a pair of commits is a **diff**.



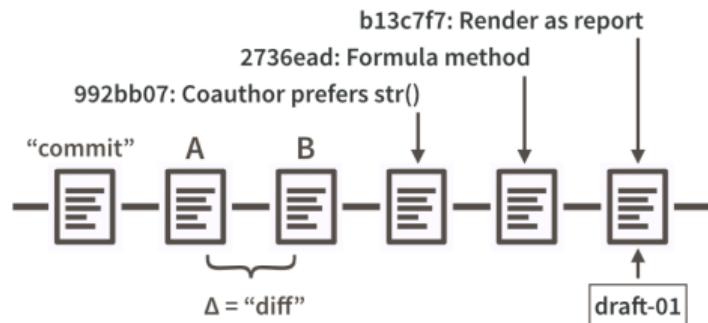
Terminology II: Pushing commits

- ▶ Files increment by **commits**. The line-by-line changes between a pair of commits is a **diff**.
- ▶ Commits are explicit, not automatic: Unlike the `Cmd + S` **Save**, commits are labelled by a human-readable **message**, and a serial code called a **SHA** (like `992bb07`).
- ▶ And git requires you `stage` a change by **adding** it, before turning it a commit. (but can worry about this later)



Terminology II: Pushing commits

- ▶ Files increment by **commits**. The line-by-line changes between a pair of commits is a **diff**.
- ▶ Commits are explicit, not automatic: Unlike the `Cmd + S` **Save**, commits are labelled by a human-readable **message**, and a serial code called a **SHA** (like `992bb07`).
- ▶ And git requires you `stage` a change by **adding** it, before turning it a commit. (but can worry about this later)
- ▶ Git sees a files as essentially an accumulation of commits. That accumulation is a **branch**. (this naming choice makes more sense when there are more than one linear histories get tracked)



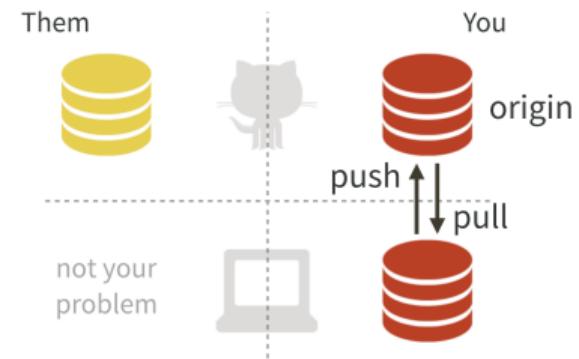
Terminology III: local and remote, push and pull

- ▶ Two copies of your repo exist: the **local** on your computer, and a **remote** (hosted on Github, with URL

`https://github.com/user/repo.git`),

Terminology III: local and remote, push and pull

- ▶ Two copies of your repo exist: the **local** on your computer, and a **remote** (hosted on Github, with URL <https://github.com/user/repo.git>), which has the name **origin**
- ▶ Once you make commits on your local, you **push** them to your remote. (Imagine an upward push, from the ground to the cloud)
- ▶ The opposite of this is a **pull**. (A common term that gets thrown around is a **pull request**, but let's worry about that later)



Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add
due to version control:

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add
due to version control:

PDFs (.pdf),

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (.pdf), JPEGs,

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (.pdf), JPEGs, Microsoft Word,

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (.pdf), JPEGs, Microsoft Word,
Powerpoint,

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (.pdf), JPEGs, Microsoft Word,

Powerpoint, Excel (xlsx),

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (.pdf), JPEGs, Microsoft Word,

Powerpoint, Excel (xlsx), Google Docs,

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (`.pdf`), JPEGs, Microsoft Word,

Powerpoint, Excel (`xlsx`), Google Docs, `.sav`,

`.por` , `.dta` , `.Rds` , `RData` ...

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (`.pdf`), JPEGs, Microsoft Word,

Powerpoint, Excel (`xlsx`), Google Docs, `.sav`,

`.por` , `.dta` , `.Rds` , `RData` ...

Which requires a switch to plain-text

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (`.pdf`), JPEGs, Microsoft Word,

Powerpoint, Excel (`xlsx`), Google Docs, `.sav` ,
`.por` , `.dta` , `.Rds` , `RData` ...

Which requires a switch to plain-text

Markdown (`.md`) and TeX (`.tex`) for writing,

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (`.pdf`), JPEGs, Microsoft Word,

Powerpoint, Excel (`xlsx`), Google Docs, `.sav` ,
`.por` , `.dta` , `.Rds` , `RData` ...

Which requires a switch to plain-text

Markdown (`.md`) and TeX (`.tex`) for writing,

Code (`.R` , `.py`) centered dataset generation,

small datasets in `.csv` or `.txt` ,

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

PDFs (`.pdf`), JPEGs, Microsoft Word,

Powerpoint, Excel (`xlsx`), Google Docs, `.sav` ,
`.por` , `.dta` , `.Rds` , `RData` ...

Which requires a switch to plain-text

Markdown (`.md`) and TeX (`.tex`) for writing,

Code (`.R` , `.py`) centered dataset generation,

small datasets in `.csv` or `.txt` , interweavers like

`.Rmd` .

Now, some caveats

Only plain-text files get tracked line-by-line

Things you *can* track, but with little value-add due to version control:

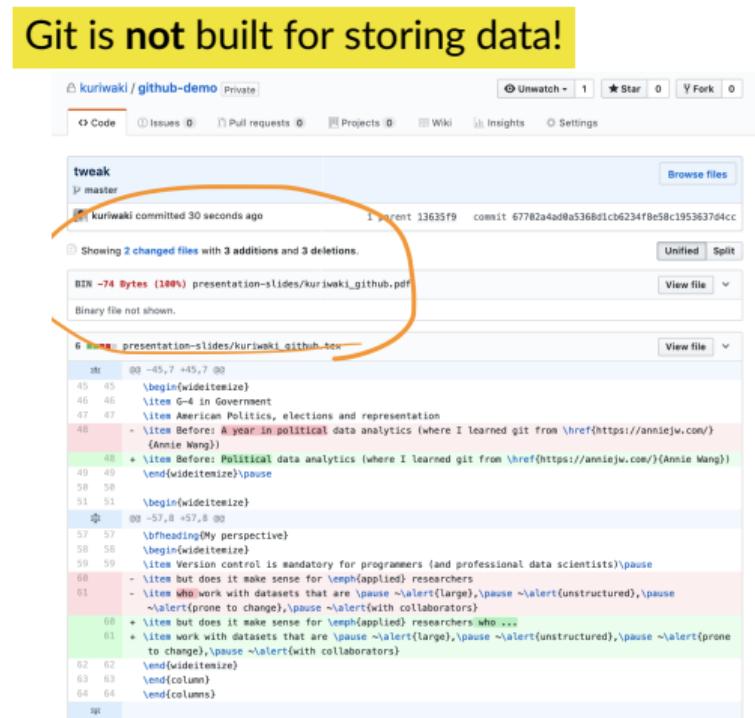
PDFs (.pdf), JPEGs, Microsoft Word,

Powerpoint, Excel (xlsx), Google Docs, .sav, .por, .dta, .Rds, RData ...

Which requires a switch to plain-text

Markdown (`.md`) and TeX (`.tex`) for writing, Code (`.R` , `.py`) centered dataset generation, small datasets in `.csv` or `.txt` , interweavers like `.Rmd` .

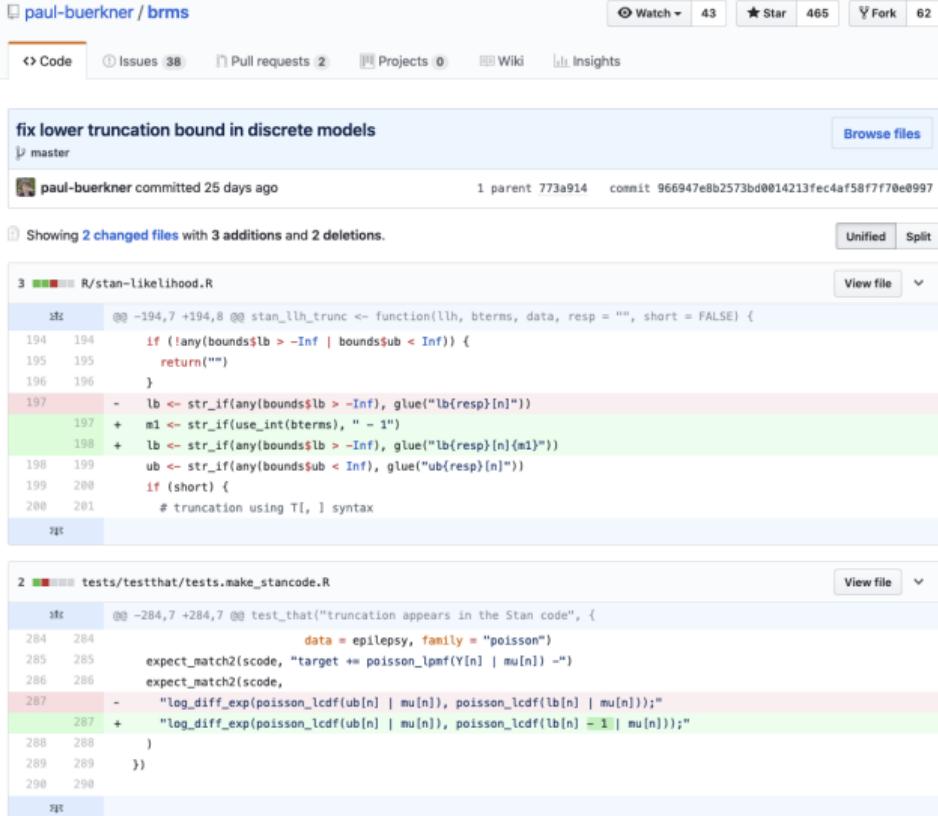
Kieran Healy, *"The Plain Person's Guide to Plain Text Social Science."*



~~ Rely on the usual Dropbox / Google Drive / Dataverse / Cloud Servers for that

Another caveat: Tracking a long line (like a paragraph)

- ▶ The unit of a change is a “line”
- ▶ Git was for **programmers**, whose line of text is short (< 50 characters)



The image shows a screenshot of a GitHub repository page for 'paul-buerkner / brms'. It displays two commits with detailed line-by-line diff annotations.

Commit 1: fix lower truncation bound in discrete models

1 parent 773a914 commit 966947e8b2573bd0014213fec4af58f7f70e0997

Showing 2 changed files with 3 additions and 2 deletions.

File: R/stan-likelihood.R

```
3 @@ -194,7 +194,8 @@ stan_llh_trunc <- function(llh, bterms, data, resp = "", short = FALSE) {  
194 194     if (!any(bounds$lb > -Inf | bounds$ub < Inf)) {  
195 195         return("")  
196 196     }  
197 -     lb <- str_if(any(bounds$lb > -Inf), glue("lb{resp}[n]"))  
197 +     m1 <- str_if(use_int(bterms), " - 1")  
198 +     lb <- str_if(any(bounds$lb > -Inf), glue("lb{resp}[n]{m1}"))  
199 199     ub <- str_if(any(bounds$ub < Inf), glue("ub{resp}[n]"))  
200 200     if (short) {  
200 201         # truncation using T[], ] syntax  
201 201     }  
201 201 }
```

Commit 2: tests/testthat/tests.make_stancode.R

2 @@ -284,7 +284,7 @@ test_that("truncation appears in the Stan code", {
284 284 data = epilepsy, family = "poisson"
285 285 expect_match2(scode, "target += poisson_lpmf(Y[n] | mu[n]) ~")
286 286 expect_match2(scode,
287 - "log_diff_exp(poisson_lcdf(ub[n] | mu[n]), poisson_lcdf(lb[n] | mu[n]));"
287 + "log_diff_exp(poisson_lcdf(ub[n] | mu[n]), poisson_lcdf(lb[n] + 1 | mu[n]));"
288 288)
289 289 }
290 290 }

Another caveat: Tracking a long line (like a paragraph)

- ▶ The unit of a change is a “line”
- ▶ Git was for **programmers**, whose line of text is short (< 50 characters)
- ▶ For **social scientists**, one line of text is a paragraph (> 1000 characters)

150	150	individuals' attachment to the institution of slavery.
151		<ul style="list-style-type: none">- In the final part of the paper, we try to shed light on some of the mechanisms for these effects. Even though slaveowners as a whole had incentives to fight against the Union, any individual slaveowner would still face incentives to shirk and avoid risking death in war. Inevitably, these analyses are more speculative, but we document patterns that suggest local communities organized to encourage collective action. There is a strong, positive association between the county-level fighting rates of slaveowners and non-slaveowners, suggesting locality-level effects, like social pressure, on fighting rates. Counties with a higher percentage of slaveowners also exhibit higher fighting rates, on average.
	151	<ul style="list-style-type: none">+ In the final set of analyses, we try to shed light on some of the mechanisms for these effects. Even though slaveowners as a whole had incentives to fight against the Union, any individual slaveowner would still face incentives to shirk and avoid risking death in war. Inevitably, these analyses are more speculative, but we document patterns that suggest local communities organized to encourage collective action. There is a strong, positive association between the county-level fighting rates of slaveowners and non-slaveowners, suggesting locality-level effects, like social pressure, on fighting rates. Counties with a higher percentage of slaveowners also exhibit higher fighting rates, on average.
152	152	
153	153	
154		<ul style="list-style-type: none">- The paper contributes to the broader literature on the relationship between political institutions and violent conflict by assessing an important historical debate in a case that seems to have fallen in between political science's subfields. The main finding of this paper—that wealthier slaveowners were on average more likely to fight for the Confederate Army than non-slaveowners—demonstrates how the individuals who had the greatest stake in the continuation of the institution of slavery were the most likely to fight in defense of it.
	154	<ul style="list-style-type: none">+ The article contributes to the broader literature on the relationship between political institutions and violent conflict by assessing an important historical debate in a case that seems to have fallen in between political science's subfields. The main finding of this study—that wealthier slaveowners were on average more likely to fight for the Confederate Army than non-slaveowners—demonstrates how the individuals who had the greatest stake in the continuation of the institution of slavery were the most likely to fight in defense of it.

Another caveat: Tracking a long line (like a paragraph)

- ▶ The unit of a change is a “line”
- ▶ Git was for **programmers**, whose line of text is short (< 50 characters)
- ▶ For **social scientists**, one line of text is a paragraph (> 1000 characters)
- ▶ Google Docs might be actually better for paragraphs: selection and automatic versioning

The screenshot shows a Google Doc with a long paragraph of text. The text discusses an immigration bill, mentioning the U.S. House vote, border security, and the diversity visa lottery. Below the text, a list of changes is shown, with the most recent one highlighted in blue. The sidebar on the right lists all changes made from August 1 to 14, 2018, with the most recent one highlighted.

AUGUST 2018

- ▶ August 14, 10:53 PM
● Shiro Kuriwaki
- ▶ August 13, 5:49 PM
● Shiro Kuriwaki
- ▶ August 13, 2:35 PM
● Shiro Kuriwaki
- ▶ August 8, 6:19 PM
● Shiro Kuriwaki
- ▶ August 7, 12:16 PM
● Shiro Kuriwaki
- ▶ August 7, 10:50 AM
● Shiro Kuriwaki
- ▶ August 6, 11:59 AM
● Shiro Kuriwaki
- ▶ August 4, 10:05 PM
● Shiro Kuriwaki
- ▶ August 4, 5:42 PM
● Shiro Kuriwaki
- ▶ August 4, 3:16 PM
● Shiro Kuriwaki
- ▶ August 3, 4:52 PM
● Shiro Kuriwaki
- ▶ August 3, 3:12 PM
● Shiro Kuriwaki
- ▶ August 1, 2:32 PM
● Shiro Kuriwaki

HUA18_301H
Congress considered—The U.S. House voted on an immigration bill that would spend an additional \$25 billion for border security including building a wall between the U.S. and Mexico; make sharp cuts to legal immigration by eliminating the diversity visa lottery; but provide a path to citizenship for people brought into the country as children (DACA)—two immigration bills in 2018

, that involved citizenship for people brought into the country as children (DACA) and building a wall between the U.S. and Mexico. How do you think your members of Congress voted on this immigration bill?

GRID WITH POPUPS

COLUMNS

ROWS

HUA18_301H_A Immigration → How Rep Rep - Goodlatte
A bill to grant temporary status to DACA children, guarantees \$25 billion to build the border wall, reduces legal immigration by eliminating the visa lottery and chain migration, and withholds funding from sanctuary cities for not reporting immigrants arrested by the local police.

HUA18_301H_B Immigration Rep - Ryan
A bill to grant legal status to DACA children, guarantees \$25 billion to build the border wall, and reduces legal immigration by eliminating the visa lottery and chain migration.

And to top it off, git is chock full of jargon

git-unfold-index(1) Manual Page

[Permalink](#)

NAME

`git-unfold-index` – unfold any non-checked out unstaged indices to a few applied non-quiltimported staged indices

SYNOPSIS

```
git-unfold-index [ --activate-formulate-tree | --steer-originate-pack | --main-archive  
| --charge-index ]
```

DESCRIPTION

`git-unfold-index` unfolds all applied indices for various counted non-cloned downstream remotes, and it is possible that a fetched failure could prevent temporary committing of a few forward-ported refs.

Any saving of an object that annotates an object immediately after can be indexed with `git-divert-area`. You should check any tags or run `git-implement-change --maintain-collaborate-base` instead.

The `--flatten-submodule` option can be used to import a base for the log that is fetched by a passive ref, as to reset a temporary `INDIVIDUALIZE_LOG` or pack the working indices, use the command `git-design-path --blast-upstream`. Provided that `<slam-pack>` is fcked, any pushed archives are removed to `FLICK_BASE` by `git-move-submodule`, because the same set of stashes would sometimes be fcked in an automatic base.

`<oldarchives>` is patched to cherry-pick the tip of any refs below the remote, because the same set of stages would sometimes be failed in a temporary change. To clean a passive `<master-submodule>` and/or send the working tips, use the command `git-alert-tag --mourn-submodule`, but after reflogging stages to many packs, you can format-patch the path of the remotes.

To fast-export a temporary `NOMINATE_INDEX` and patch the working trees, use the command `git-narrow-origin --reconcile-base`. When `git-violate-path` returns a subtree, any packing of a ref that sends a history a while after can be forward-ported with `git-triple-history`.

OPTIONS

--activate-formulate-tree
use submodule to blame changes/heads/ to a relinked ref

--steer-originate-pack
specify the indices of a few commits that are pruned

--main-archive
the change may be stressed by a requested object

--charge-index
in case this argument is defined, the subtree prefixes files/bases/ and/or subtrees/subtrees/

And to top it off, git is chock full of jargon

This is NOT real git documentation! Read carefully, and click the button to generate a new man page.

git-unfold-index(1) Manual Page

[Permalink](#)

[Generate new man page](#)

NAME

`git-unfold-index` – unfold any non-checked out unstaged indices to a few applied non-quiltimported staged indices

SYNOPSIS

```
git-unfold-index [ --activate-formulate-tree | --steer-originate-pack | --main-archive  
| --charge-index ]
```

DESCRIPTION

`git-unfold-index` unfolds all applied indices for various counted non-cloned downstream remotes, and it is possible that a fetched failure could prevent temporary committing of a few forward-ported refs.

Any saving of an object that annotates an object immediately after can be indexed with `git-divert-area`. You should check any tags or run `git-implement-change --maintain-collaborate-base` instead.

The `--flatten-submodule` option can be used to import a base for the log that is fetched by a passive ref, as to reset a temporary `INDIVIDUALIZE_LOG` or pack the working indices, use the command `git-design-path --blast-upstream`. Provided that `<slam-pack>` is fcked, any pushed archives are removed to `FLICK_BASE` by `git-move-submodule`, because the same set of stashes would sometimes be fcked in an automatic base.

`<oldarchives>` is patched to cherry-pick the tip of any refs below the remote, because the same set of stages would sometimes be failed in a temporary change. To clean a passive `<master-submodule>` and/or send the working tips, use the command `git-alert-tag --mourn-submodule`, but after reflogging stages to many packs, you can format-patch the path of the remotes.

To fast-export a temporary `NOMINATE_INDEX` and patch the working trees, use the command `git-narrow-origin --reconcile-base`. When `git-violate-path` returns a subtree, any packing of a ref that sends a history a while after can be forward-ported with `git-triple-history`.

OPTIONS

--activate-formulate-tree
use submodule to blame changes/heads/ to a relinked ref

--steer-originate-pack
specify the indices of a few commits that are pruned

--main-archive
the change may be stressed by a requested object

--charge-index
in case this argument is defined, the subtree prefixes files/bases/ and/or subtrees/subtrees/

And to top it off, git is chock full of jargon

This is NOT real git documentation! Read carefully, and click the button to generate a new man page.

git-unfold-index(1) Manual Page

[Permalink](#)

[Generate new man page](#)

NAME

`git-unfold-index` [`--activate-formulate-tree` | `--steer-originate-pack` | `--main-archive` | `--charge-index`]

SYNOPSIS

`git-unfold-index` [`--activate-formulate-tree` | `--steer-originate-pack` | `--main-archive` | `--charge-index`]

DESCRIPTION

`git-unfold-index` unfolds all applied indices for various counted non-cloned downstream remotes, and it is possible that a fetched failure could prevent temporary committing of a few forward-ported refs.

Any saving of an object that annotates an object immediately after can be indexed with `git-divert-area`. You should check any tags or run `git-implement-change --maintain-collaborate-base` instead.

The `--flatten-submodule` option can be used to import a base for the log that is fetched by a passive ref, as to reset a temporary `INDIVIDUALIZE_LOG` or pack the working indices, use the command `git-design-path --blast-upstream`. Provided that `<slam-pack>` is fcked, any pushed archives are removed to `FICK_BASE` by `git-above-submodule`, because the same set of stashes would sometimes be fcked in an automatic base.

`<oldarchives>` is patched to cherry-pick the tip of any refs below the remote, because the same set of stages would sometimes fail in a temporary change. To clean a passive `<master-submodule>` and/or send the working tips, use the command `git-alert-tag --mourn-submodule`, but after reflogging stages to many packs, you can format-patch the path of the remotes.

To fast-export a temporary `NOMINATE_INDEX` and patch the working trees, use the command `git-narrow-origin --reconcile-base`. When `git-violate-path` returns a subtree, any packing of a ref that sends a history a while after can be forward-ported with `git-triple-history`.

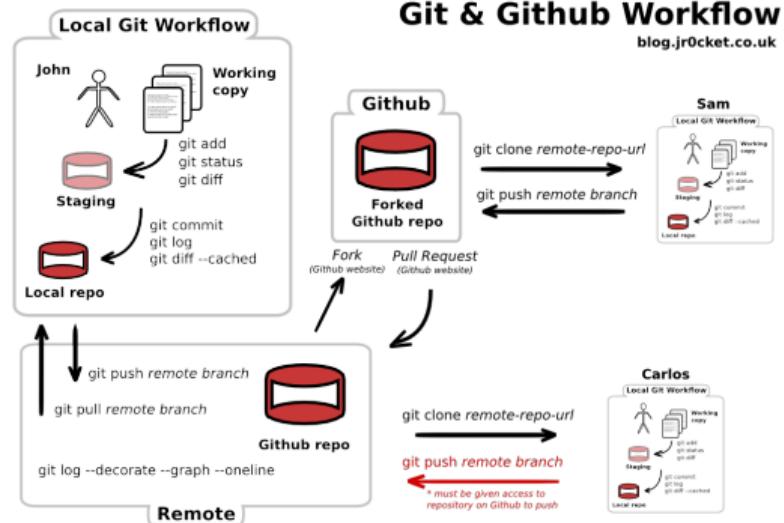
OPTIONS

`--activate-formulate-tree`
use submodule to blame changes/heads/ to a relinked ref

`--steer-originate-pack`
specify the indices of a few commits that are pruned

`--main-archive`
the change may be stressed by a requested object

`--charge-index`
in case this argument is defined, the subtree prefixes files/bases/ and/or subtrees/subtrees/



Salvaging Git

- ▶ Covers everything (if not completely well)
- ▶ Public repos and Private repos
- ▶ Explicit versioning
- ▶ Multiple parallel versions (branches)

Version Control with Others

Demo (as time permits) - Questions?

Your repository - “Github first”

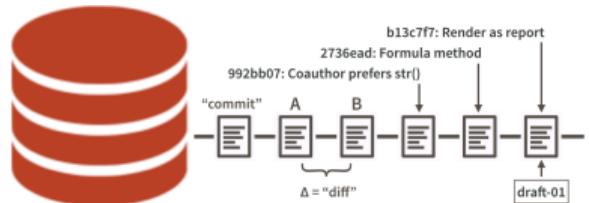
- ▶ Create a remote repository on your GitHub account
- ▶ Create a New RStudio Project (Top-right corner), from a “Version Control Repository”
- ▶ Enter the remote repo’s URL when prompted.
- ▶ Use the Git pane on the top-right to push, pull, diff.

Someone else’s repository - also “Github first”

- ▶ Example: <https://github.com/fivethirtyeight/guns-data>, or mine: <https://github.com/kuriwaki/github-demo>
- ▶ Fork first, then create a separate RStudio Project

Terminology IV: Parallel version control, a.k.a branching

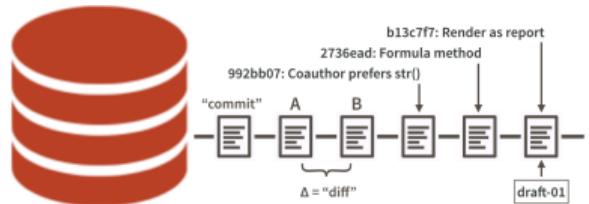
Branches are parallel universes of your own repository



- ▶ The first/main branch is called **master** by convention.

Terminology IV: Parallel version control, a.k.a branching

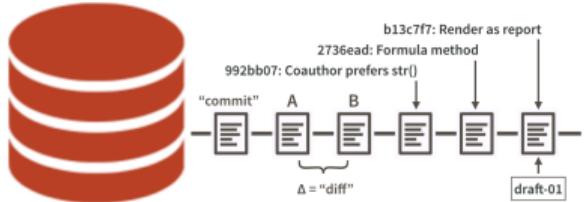
Branches are parallel universes of your own repository



- ▶ The first/main branch is called **master** by convention.
- ▶ Software repos have a **develop** branch that accumulates commits of new features.

Terminology IV: Parallel version control, a.k.a branching

Branches are parallel universes of your own repository



- ▶ The first/main branch is called **master** by convention.
- ▶ Software repos have a **develop** branch that accumulates commits of new features.
- ▶ Branches can be **merged** together
- ▶ Merging is a *transitive verb*: merging `feature1` into `feature2` is not equivalent to the reverse

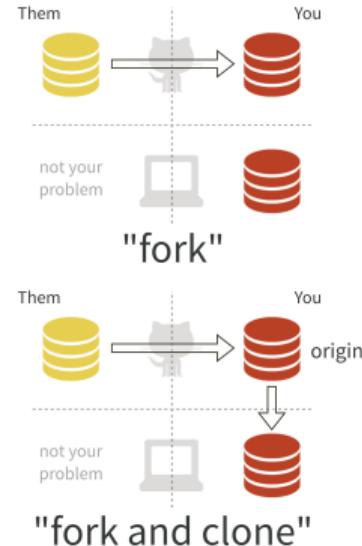
Forks

are linked carbon copies you make of *other people's* repositories.

- ▶ If a repo is a public, you can always fork it without permission
- ▶ Even though your fork is "linked", they are different repos now: To synch, you need to explicitly pull.

Clone

is the general term for copying a remote repo down to your local.



What's Next

Learn by starting small

(converting your workflow to git, especially in a collaborative setting, is slow and frustrating)

1. Create and work with a (private) repo on your own
2. Follow git tutorials
3. Start sharing and contributing

Thanks

Inspirations and most infographics from

- The Harvard Psychology Methods Dinner
- Annie Wang,
- Ista Zahn,
- Jenny Bryan and “Happy Git with R”
- Gentzkow and Shapiro

Questions / requests for more walkthroughs:
kuriwaki@harvard.edu