

What They Forgot to Teach You About R

wifi

network: WESTIN-MEETING

password: rstudio

This work is licensed under a **Creative Commons Attribution-ShareAlike 4.0 International License**.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by-sa/4.0/>

Thurs Oct 4 & Fri Oct 5, 2018

- 8am setup & breakfast
- 9-10:30am **work session 1**
- 10:30-11am break
- 11-12:30pm **work session 2**
- 12:30-1:30pm lunch
- 1:30-3pm **work session 3**
- 3-3:30pm break
- 3:30-5pm **work session 4**
- 5:20-6ish office "hours" (Thurs only)

Jennifer (Jenny) Bryan
RStudio (& UBC)

 @jennybc
 @JennyBryan

Jim Hester
RStudio

 @jimhester
 @jimhester_

Alison Hill
OHSU (& special friend of RStudio)

 apreshill
 apreshill

rstd.io/wtf-seattle

office "hours" sign up

<http://bit.ly/wtf-seattle-2018>

Everyone is encouraged to open issues here:

 [rstudio.io/wtf-seattle](https://github.com/jennybc/whattheyforgot-seattle)
<https://github.com/jennybc/whattheyforgot-seattle>

Record glitches, gotchas,
good sidebar discussions, etc.
to address now or later.

Day 1, afternoon

Jennifer (Jenny) Bryan
RStudio (& UBC)

 @jennybc

 @JennyBryan

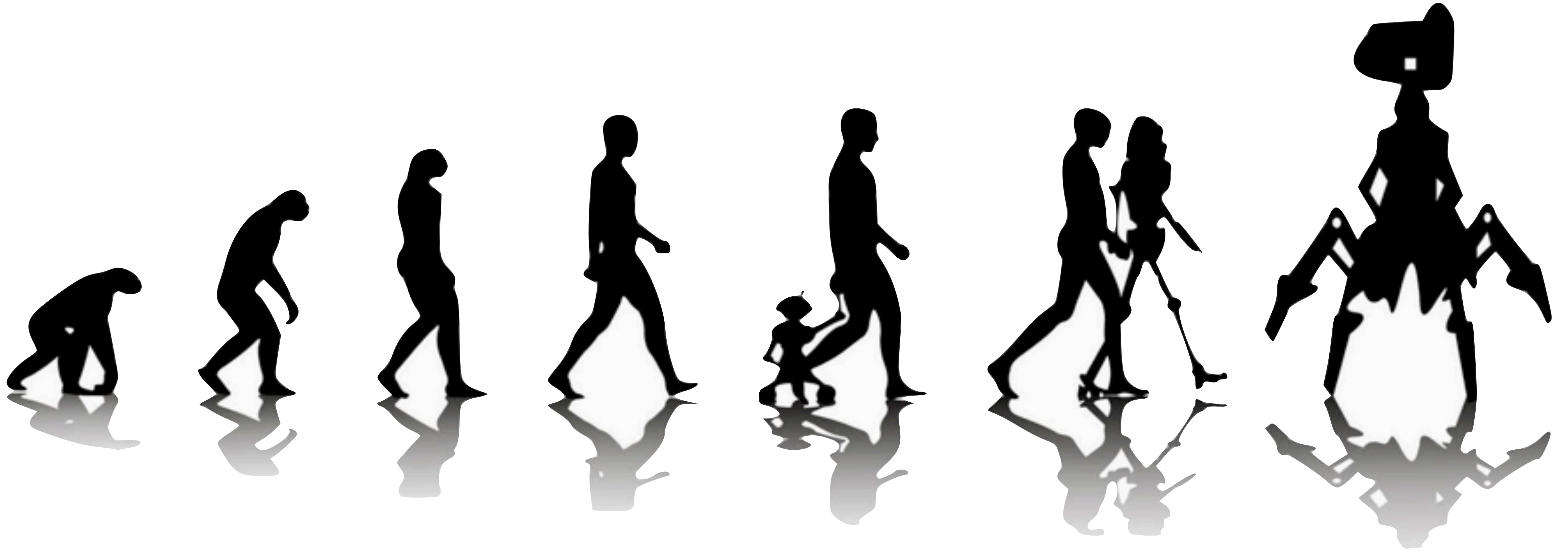
Jim Hester
RStudio

 @jimhester

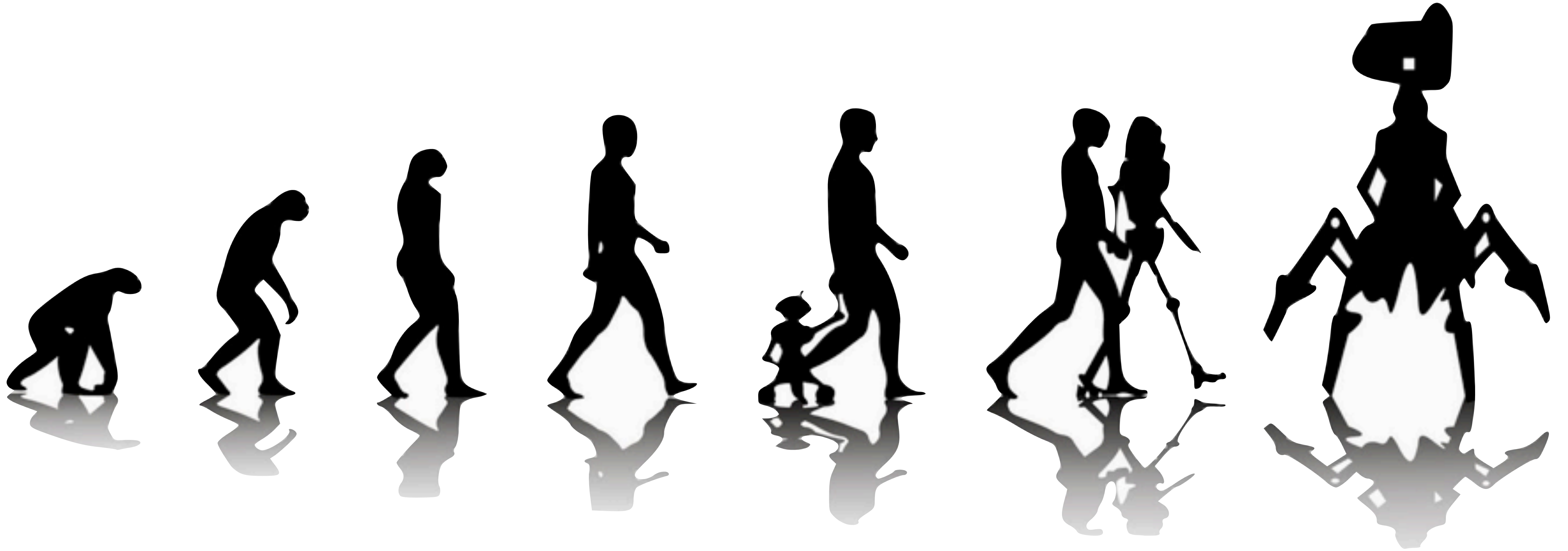
 @jimhester_



Deep Thoughts



use version control

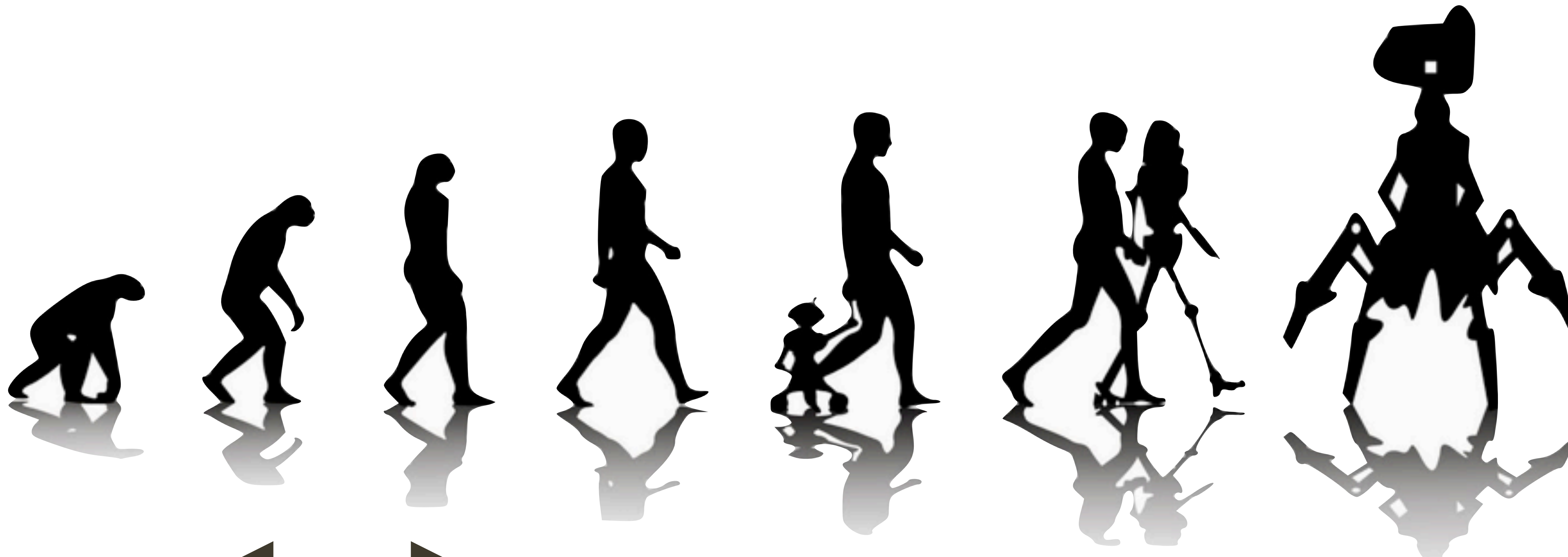


we teach Git + GitHub



"commit"

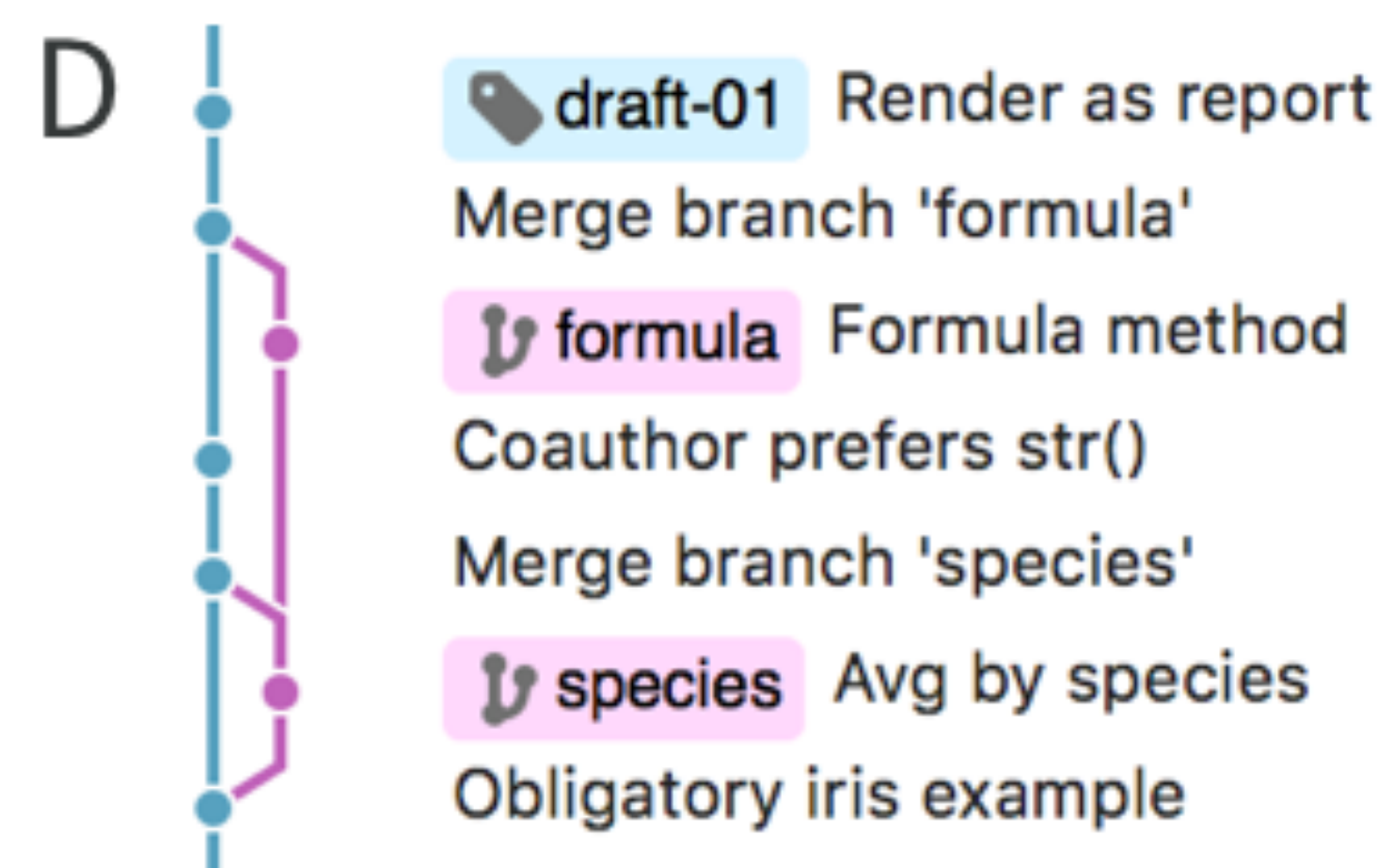
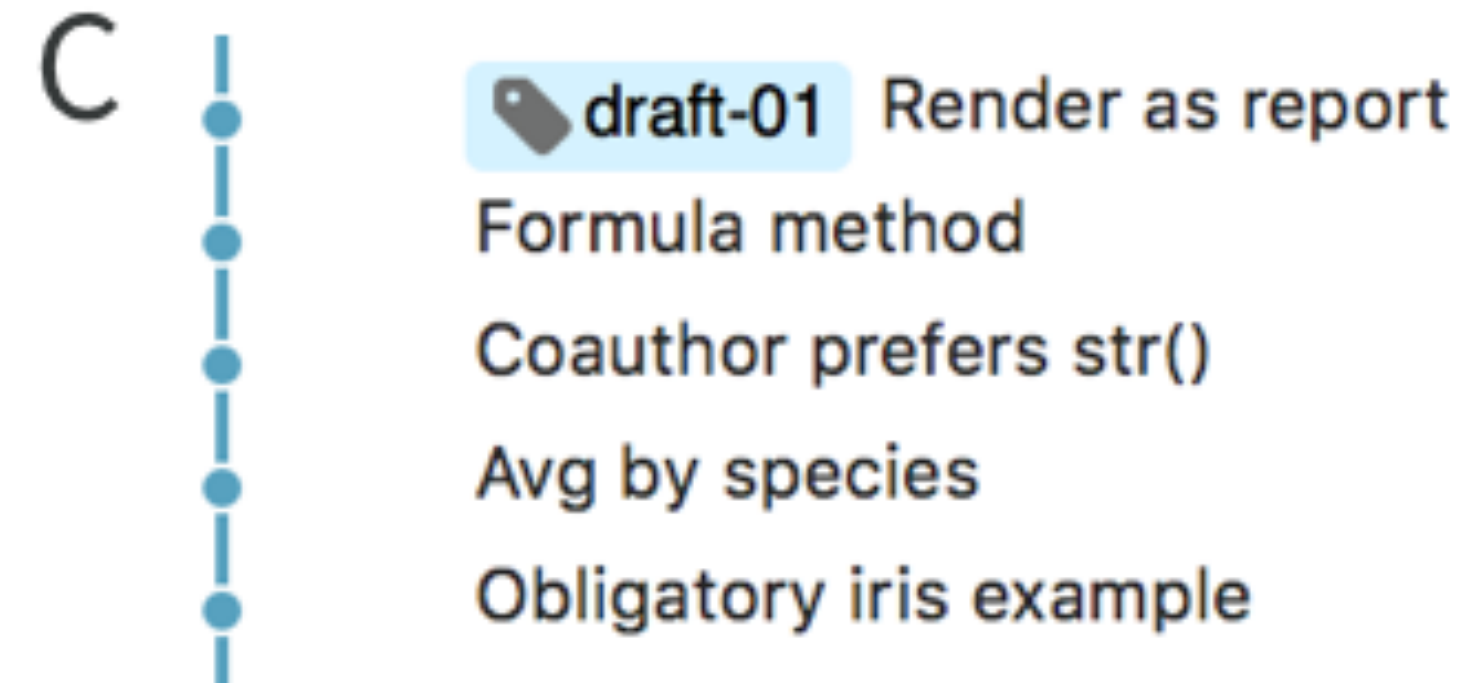
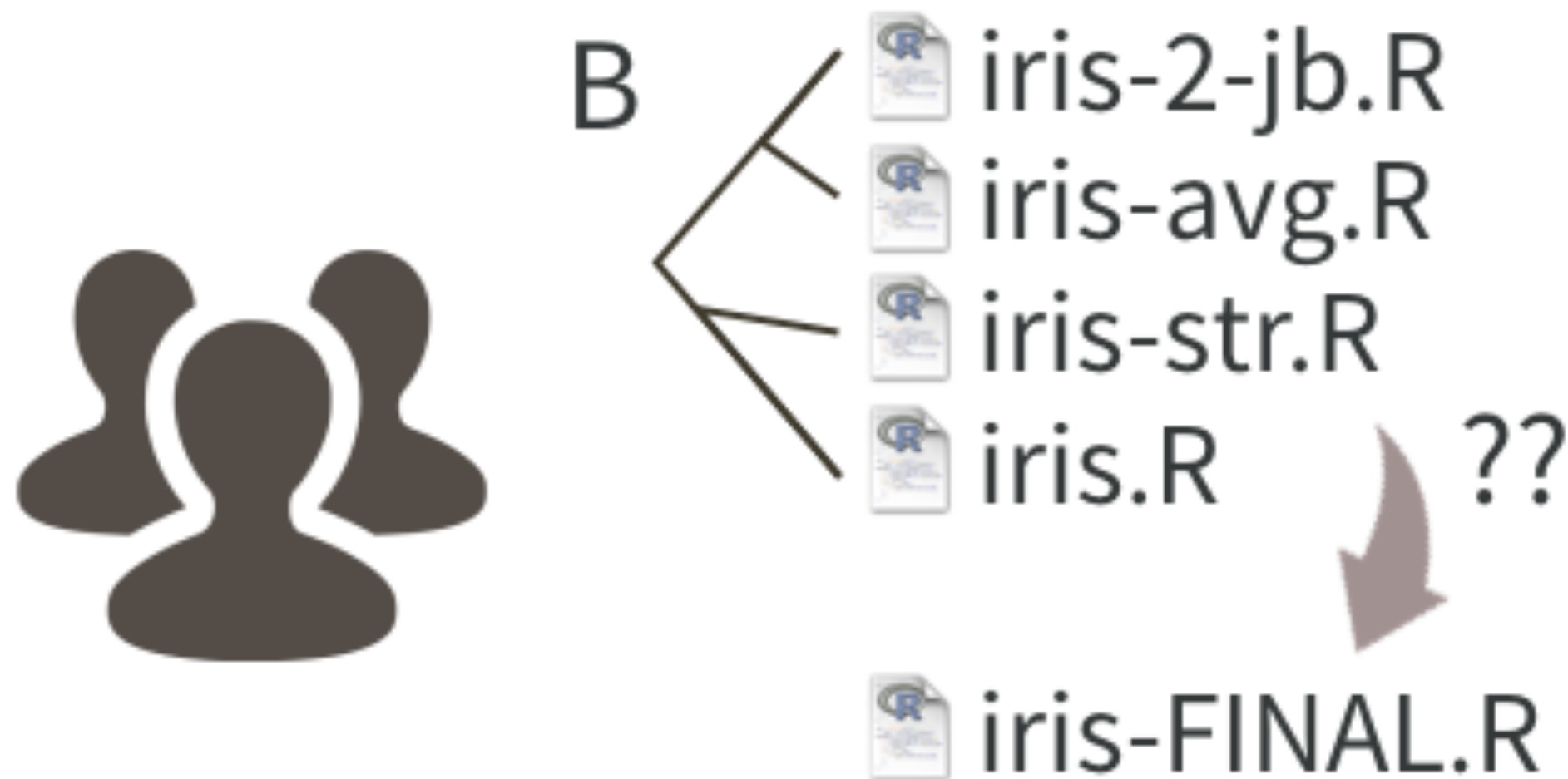
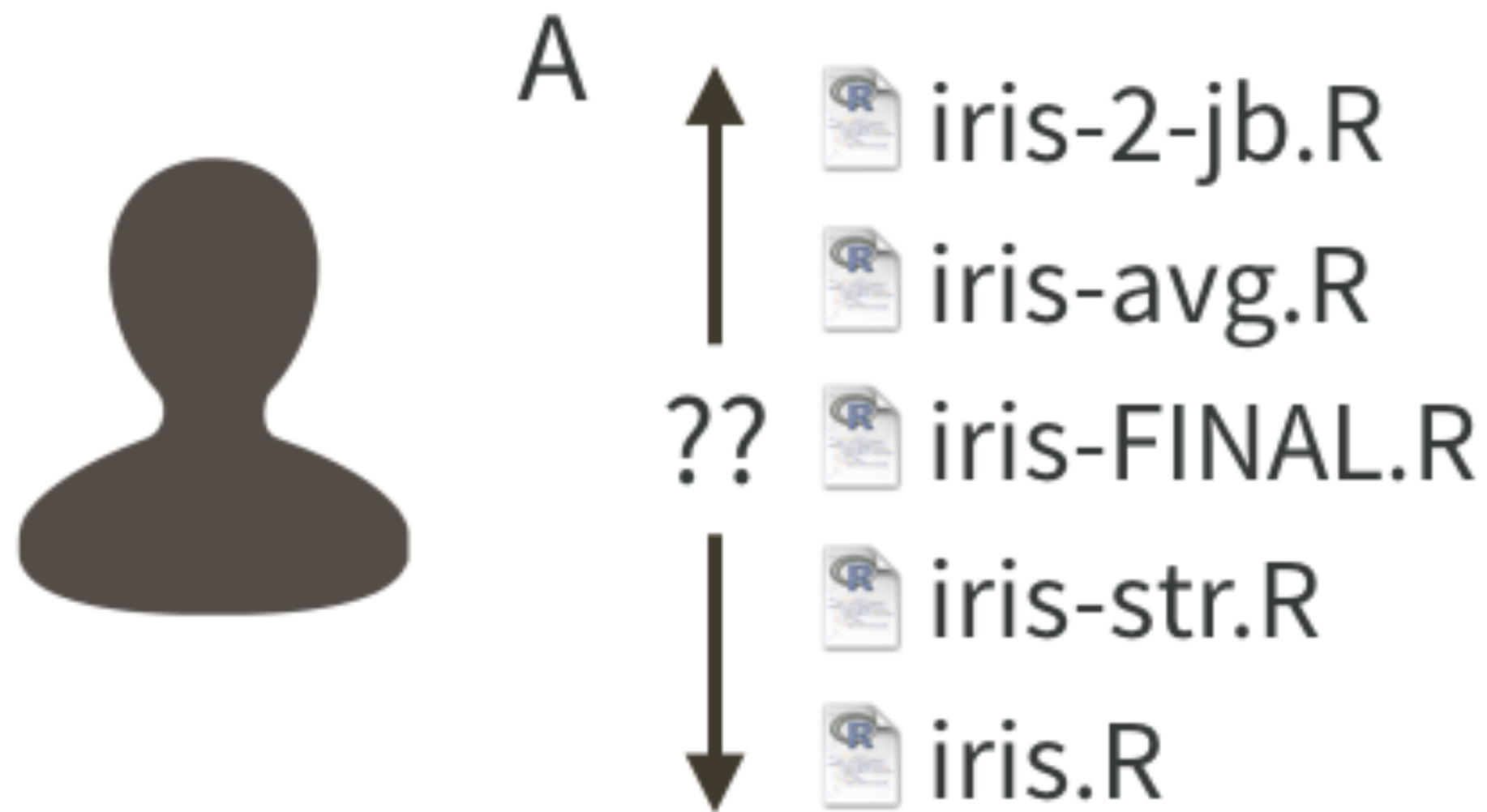
a file or project state that is **meaningful to you**
for inspection, comparison, restoration

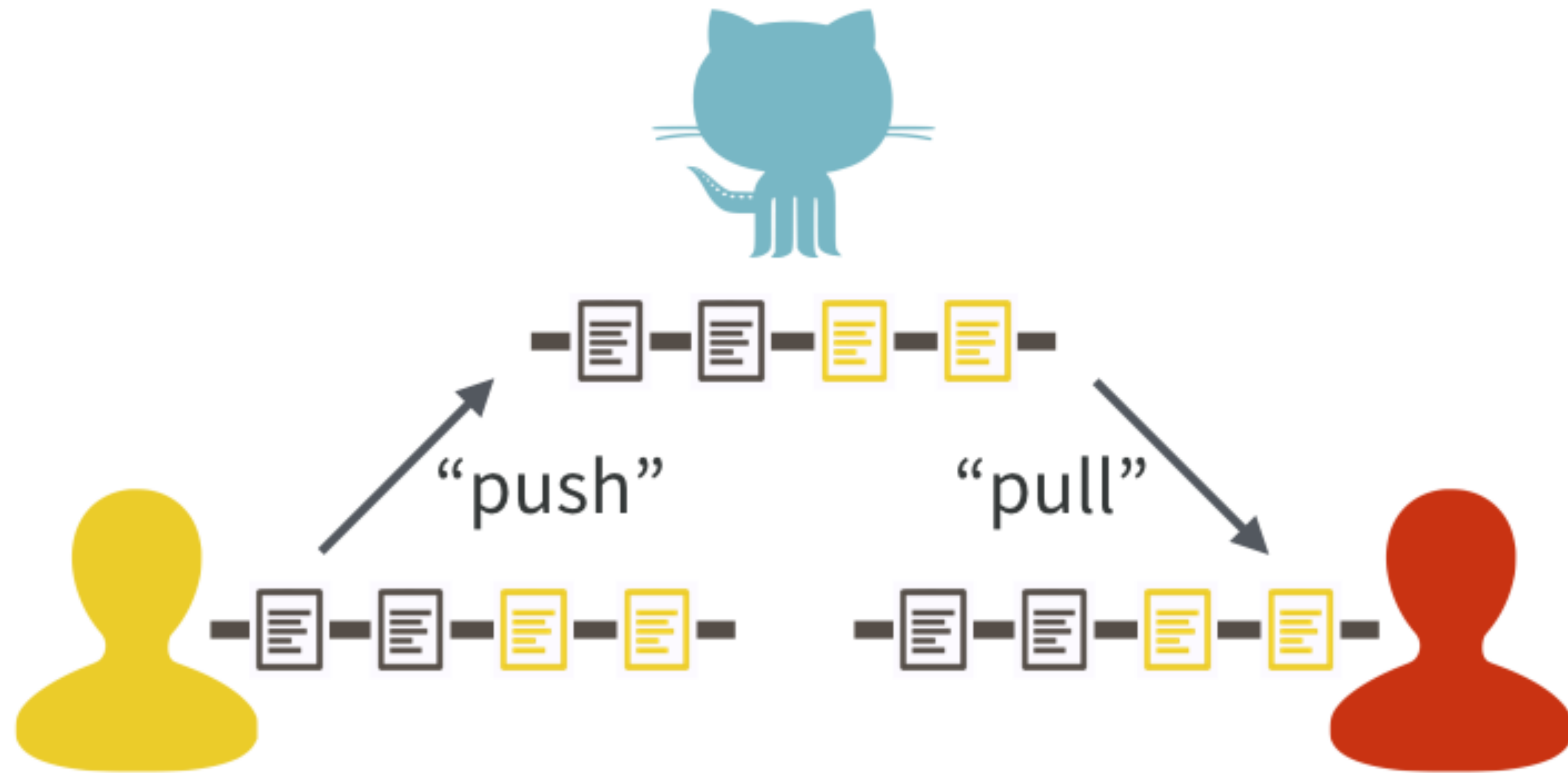


Δ

"diff"

What changed here?
Why?

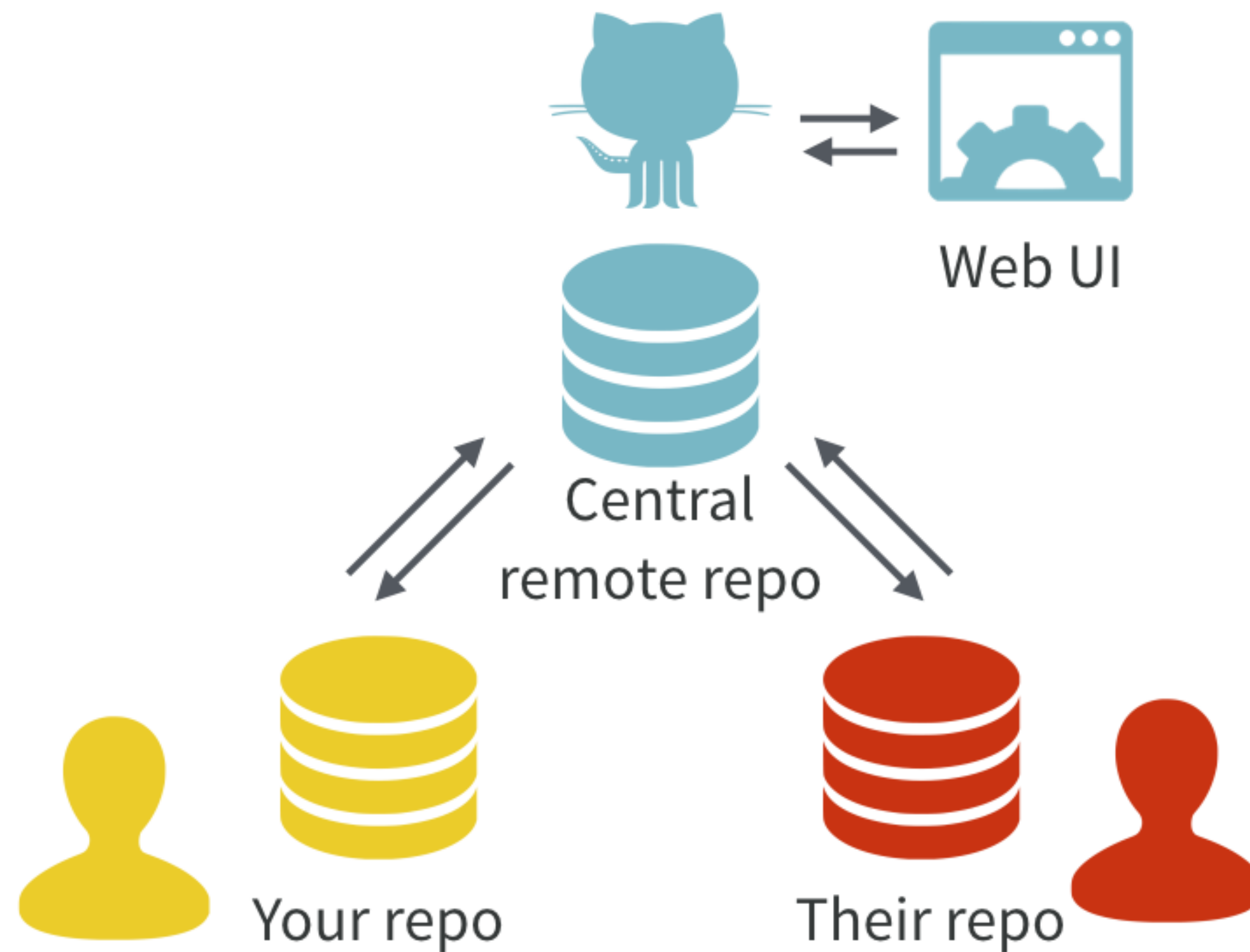




collaboration

Excuse me, do you have a moment to talk about version control?

<https://doi.org/10.7287/peerj.preprints.3159v2>



happygitwithr.com



Why version control?

- experiment without fear
- explore cause and effect
- embrace incrementalism
- collaborate
- expose your work



how



git

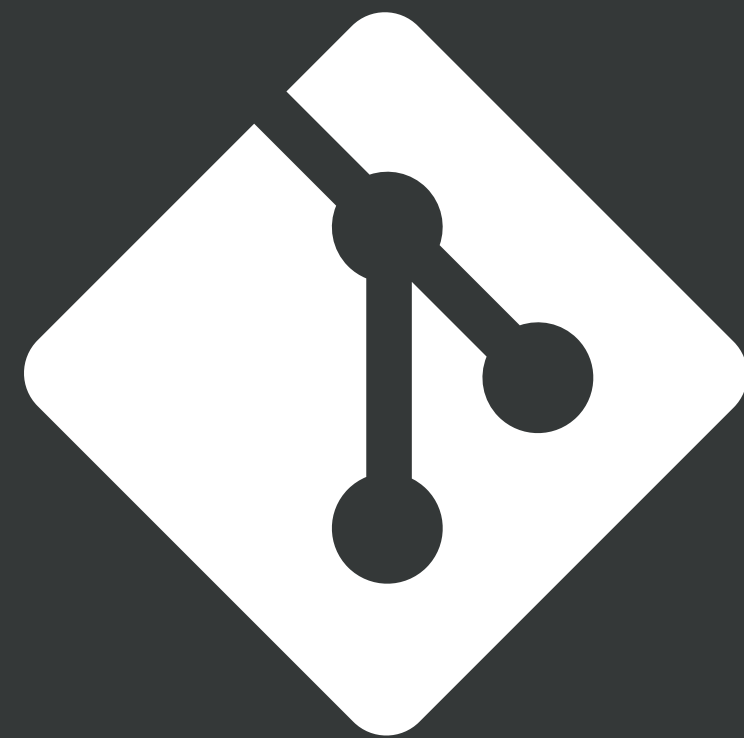
feels

get off the beach!



agony : flow

agony : flow



agony reduction

Use a Git client, if you like

No one is giving out Hard-core Git Nerd Badges

I like RStudio + SourceTree

<http://happygitwithr.com/git-client.html>

Project initiation: the remote case

Make Your remote copy of Their remote repo = "fork"

Make a local Project from a remote repo, Yours or Theirs = "clone"

Make Your remote repo from a local Project ... a bit fiddly

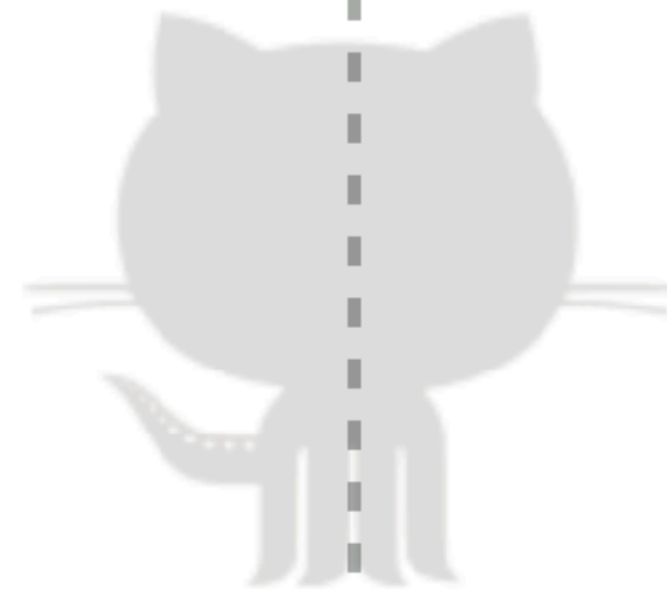
Project initiation strategies, the remote case

Make Your remote copy of Their remote repo = "fork"

Make a local Project from a remote repo, Yours ~~or Theirs~~ = "clone"

Make Your remote repo from a local Project ... a bit fiddly

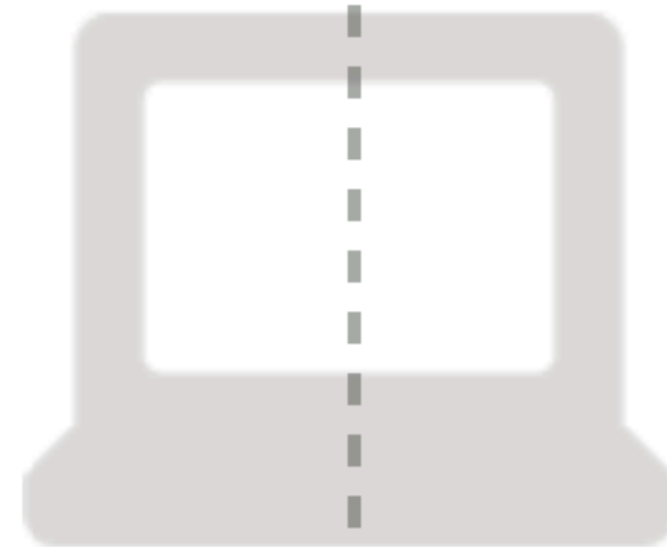
Them



You

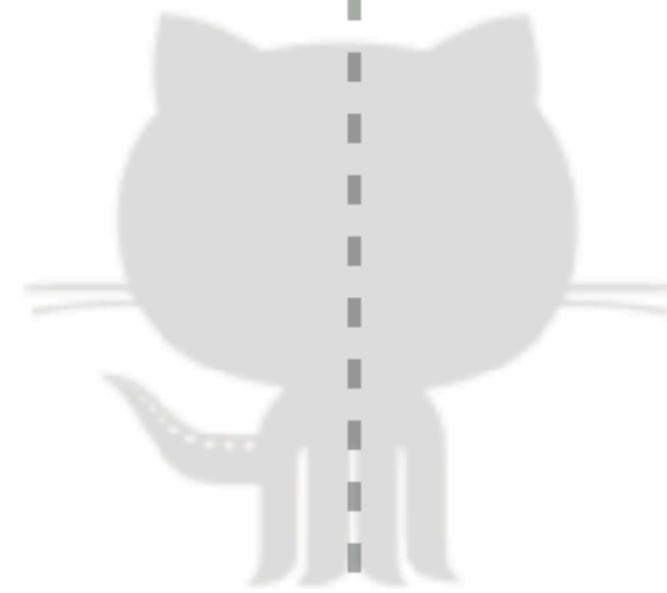


not your
problem



"clone"

Them

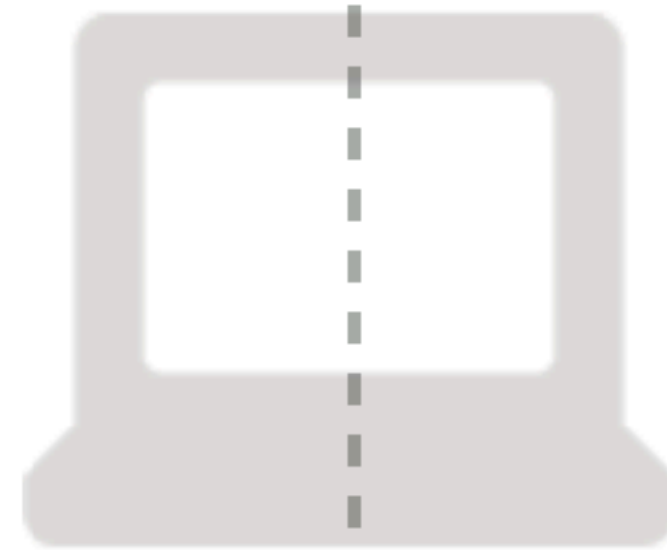


You



push ↑
↓ pull

not your
problem

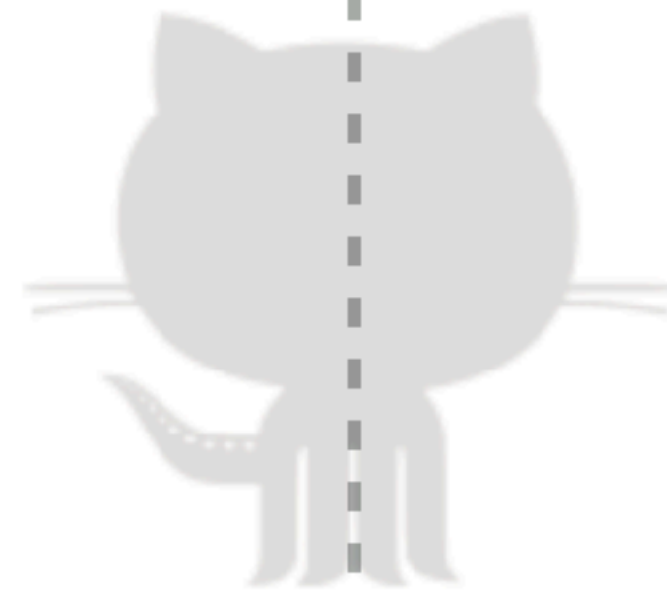


daily work, your stuff

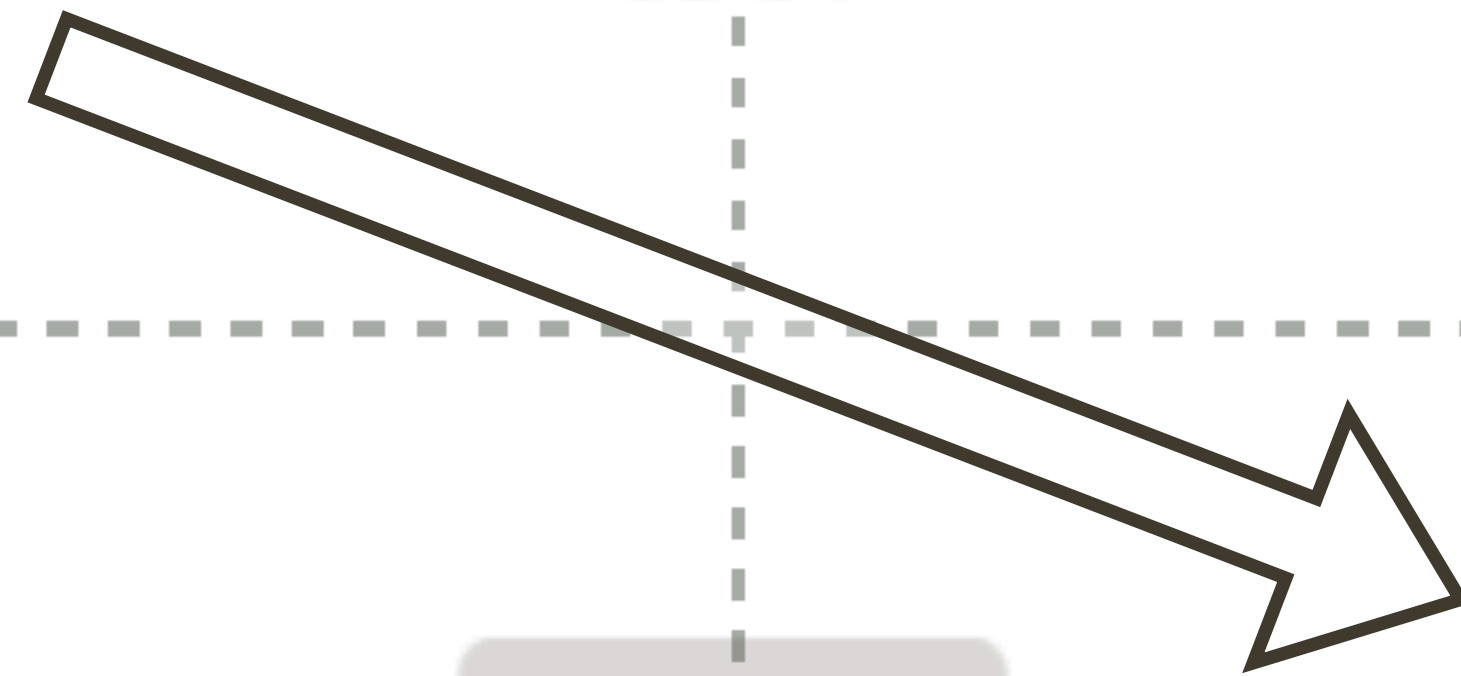
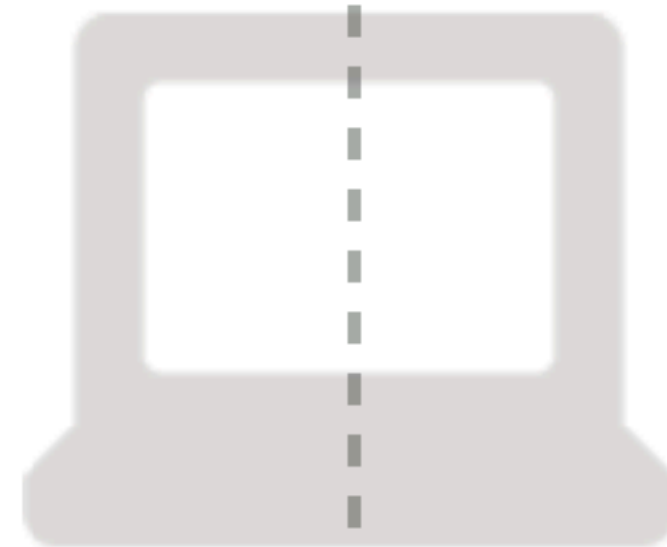
Them



You



not your
problem

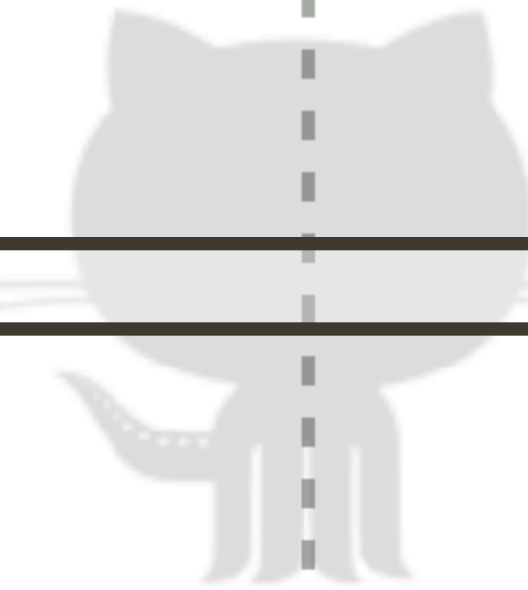


"clone"

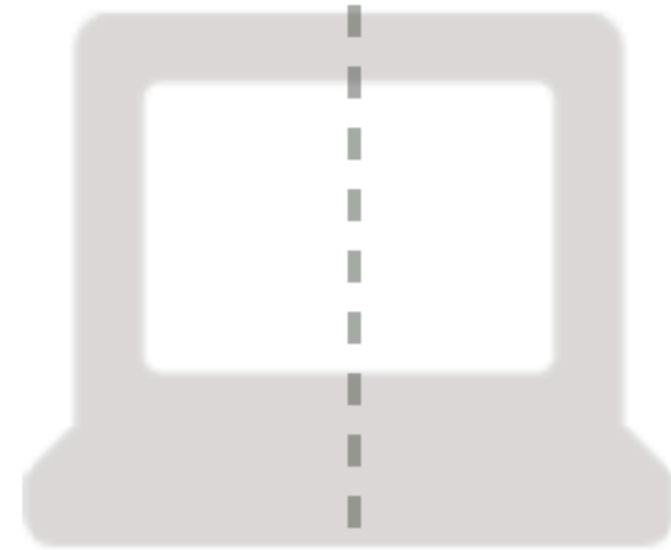
*not as useful as you might think

Them

You



not your
problem



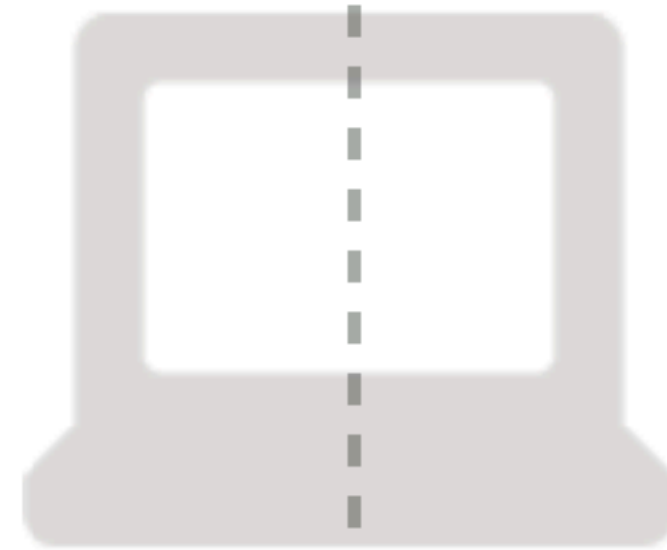
"fork"

Them

You



not your
problem



"fork and clone"

Them

You



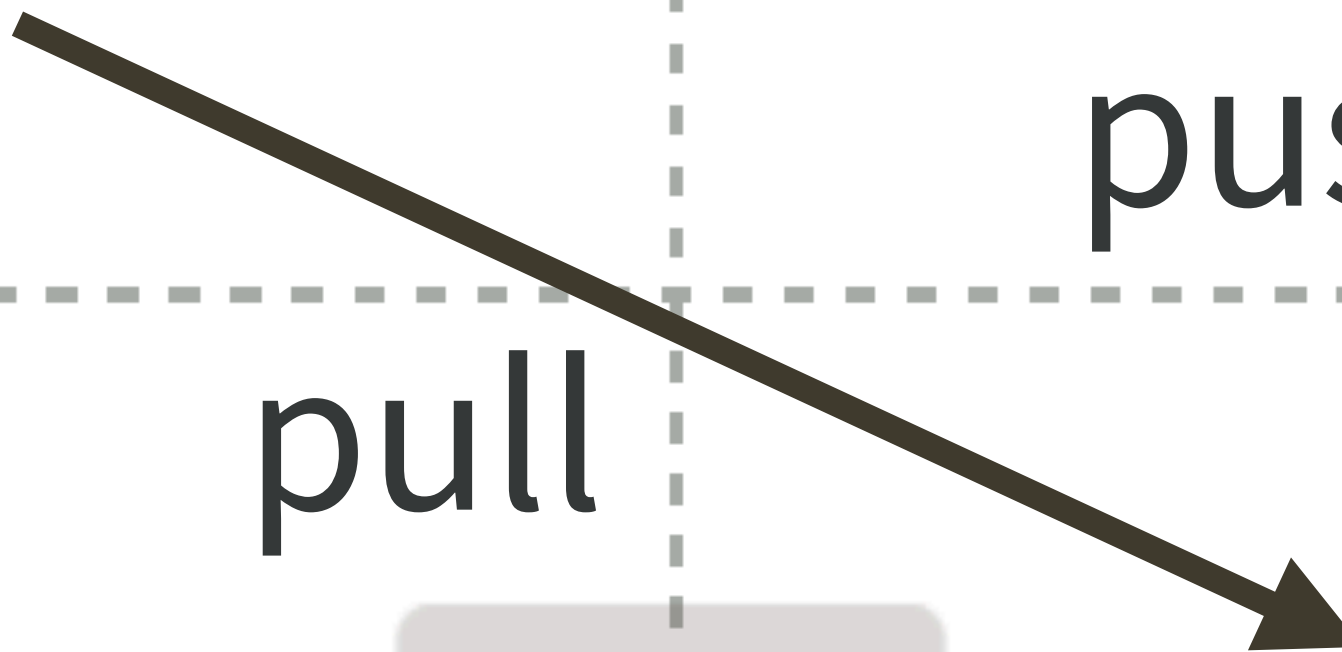
pull request



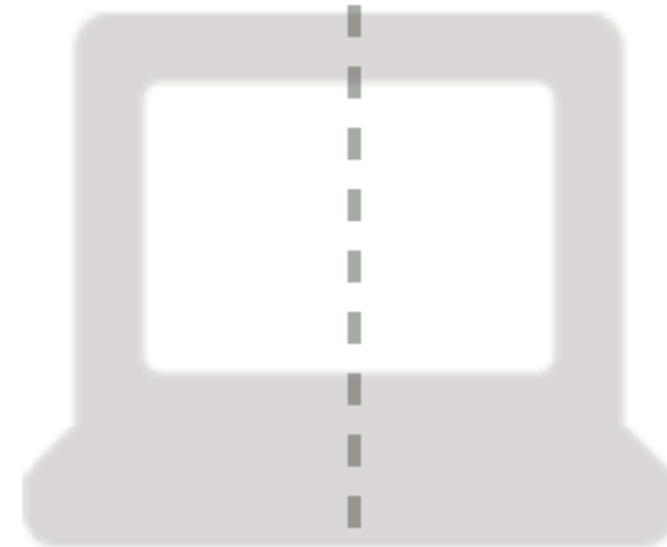
push



pull

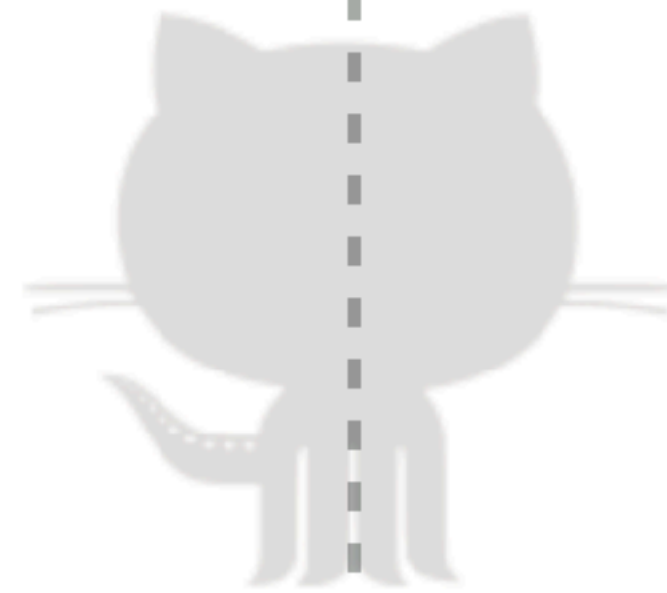


not your
problem



contribute to other people's stuff

Them



You



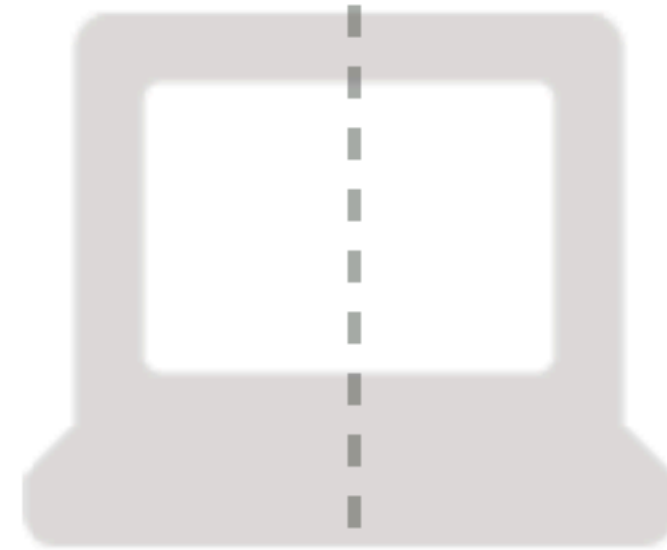
push



pull



not your
problem



daily work, your stuff

"GitHub first, then RStudio"

Why do I emphasize this?

My diagrams omit two big technical points:

- remotes
- branches

I want you to **get off the beach** before fannying around with remotes and branches.

New Project:

"GitHub first, then RStudio" workflow

<http://happygitwithr.com/new-github-first.html>

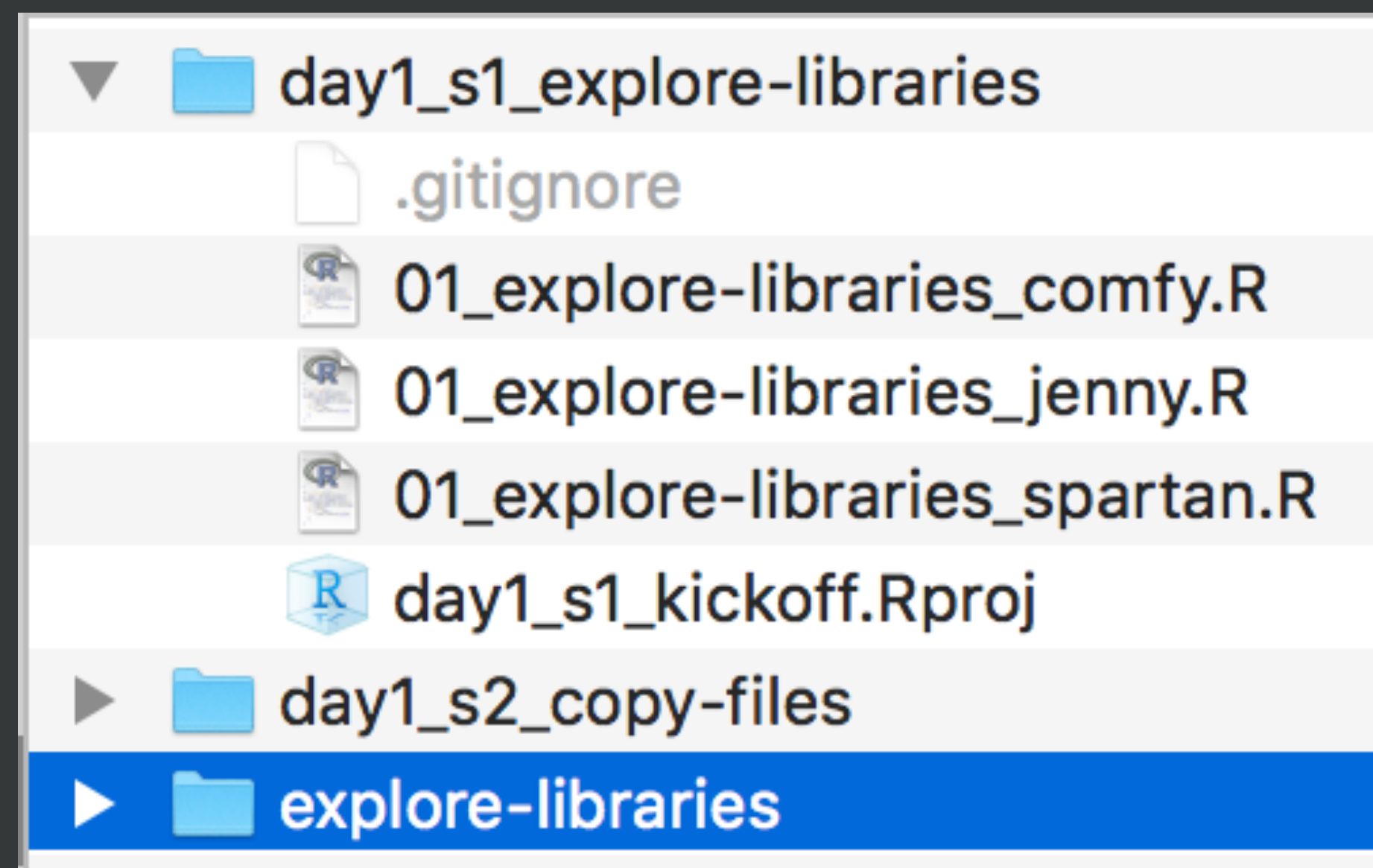
I suggest:

- repo / Project / folder name = "explore-libraries"
- locate as sibling to folders/Projects created earlier

coordinated work through this:

<http://happygitwithr.com/new-github-first.html>

Copy a .R from earlier work today into the explore-libraries directory/Project/repo.
Your choice re: which file or Project.



What changed in Git pane?

Inspect the diff.

Stage.

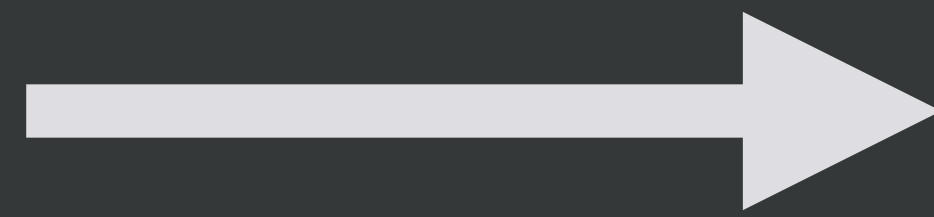
Commit.

Push.

Verify the .R file is now on GitHub.

Wait ... is a .R file all I want to share?

what you
need to write



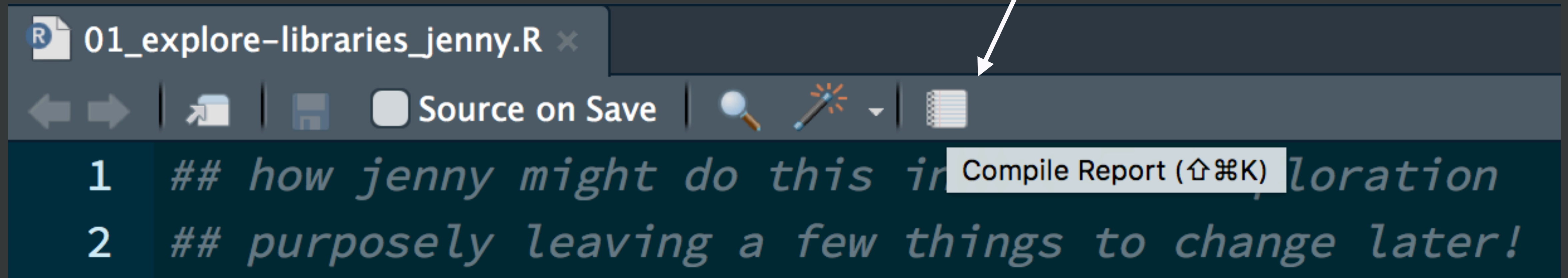
what people
like to read

foo.R
foo.Rmd



foo.md
foo.html

Compile Report



≈ `rmarkdown::render("whatever.R")`

Compile Report from R Script

Create a standalone report that contains the code and output from your R script.

For more information on compiling reports, see the documentation at [Compiling Reports from R Scripts](#)

Report output format:

HTML

Compile

Cancel

Sure, HTML is fine ... for now.

What changed in Git pane?

Inspect the diff. Or not.

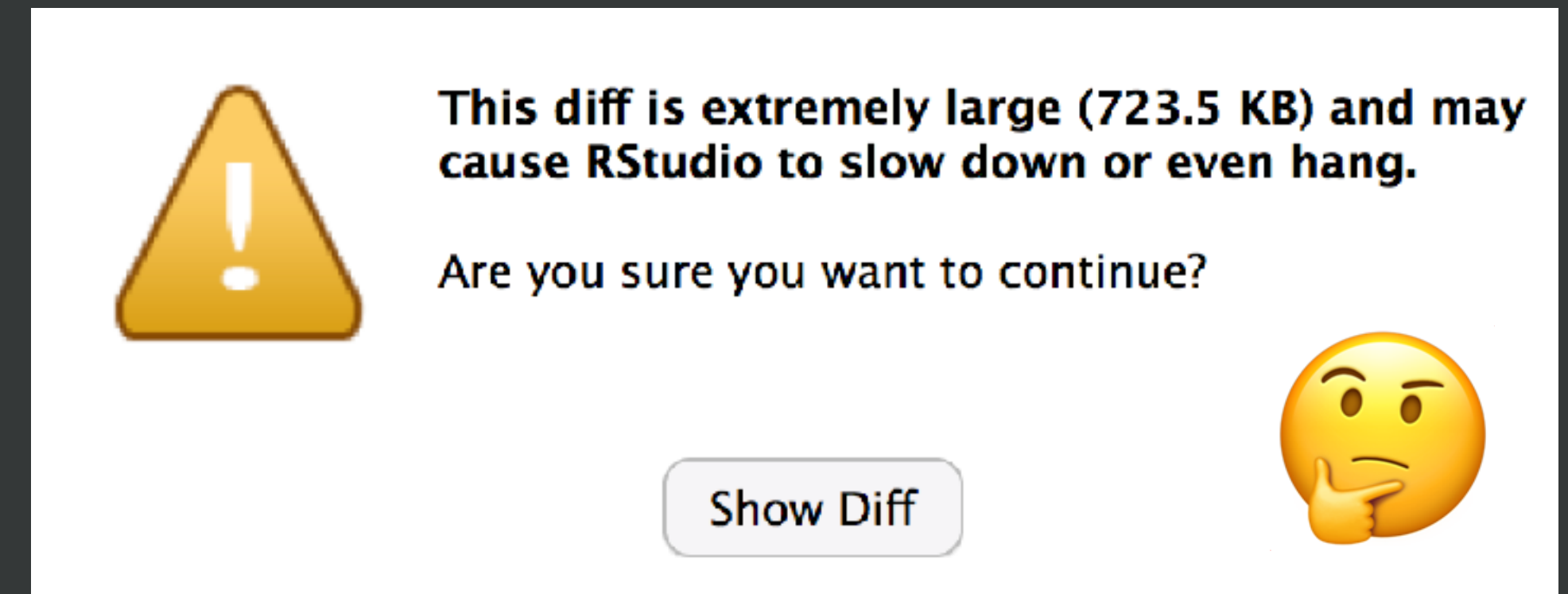
Stage.

Commit.

Push.

Verify the `.html` file is now on GitHub.

Wait... is `.html` immediately useful on GitHub?



Raw Blame History   

```

1  <!DOCTYPE html>
2
3  <html xmlns="http://www.w3.org/1999/xhtml">
4
5  <head>
6
7  <meta charset="utf-8" />
8  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
9  <meta name="generator" content="pandoc" />
10
11
12 <meta name="author" content="jenny" />
13
14
15 <title>01_explore-libraries_jenny.R</title>
16
17
18 <meta name="viewport" content="width=device-width, initial-scale=1" />
19
20 <script src="data:application/x-javascript;base64,LyohCiAqIEJvb3RzdHJhcCB2My4zLjUgKGh0dHA6Ly9nZXRib290c3RyYXAuY29tKQogKiBDdb3B!
21 <script src="data:application/x-javascript;base64,LyoqCiogQHByZXNlc3RzdHJhcCB2My4zLjUgKGh0dHA6Ly9nZXRib290c3RyYXAuY29tKQogKiBDdb3B!
22 <script src="data:application/x-javascript;base64,LyohIFJlc3BvbWQuanMgdjEuNC4yOiBtaW4vbWF4LXdpZHRoIG1lZGlnIHhf1ZXJ5IHbvbHlmaWx!
23 <script src="data:application/x-javascript;base64,CgovKioKICogalF1ZXJ5IFBsdWdpbjogU3RyY2t5IFRhYnMKICoKICogQGF1dGhvc3BvbWQuanMgdjEuNC4yOiBtaW4vbWF4LXdpZHRoIG1lZGlnIHhf1ZXJ5IHbvbHlmaWx!
24 <link href="data:text/css;charset=utf-8,%2Ehljs%2Dliteral%20%7B%0Acolor%3A%20%23990073%3B%0A%7D%0A%2Ehljs%2Dnumber%20%7B%0Aco

```



NO, raw `.html` is NOT immediately useful* on GitHub.

But Markdown = `.md` is useful.

Let's render `.R` to `.md` instead of `.html`!

* it CAN BE useful in actual web publishing workflows

foo.R → foo.html

```
#' ---  
#'| title: "Untitled"  
#'| output: html_document  
#'| ---
```

foo.R → foo.md → foo.html

```
#'| ---  
#'| title: "Untitled"  
#'| output:  
#'|   html_document:  
#'|     keep_md: yes  
#'| ---
```

foo.R → foo.md

```
#'| ---  
#'| output: md_document  
#'| ---
```

foo.R → foo.md

```
#'| ---  
#'| output: github_document  
#'| ---
```


foo.R → foo.html

```
#' ---  
#'| title: "Untitled"  
#'| output: html_document  
#'| ---
```

foo.R → foo.md → foo.html

```
#'| ---  
#'| title: "Untitled"  
#'| output:  
#'|   html_document:  
#'|     keep_md: yes  
#'| ---
```

foo.R → foo.md

```
#'| ---  
#'| output: md_document  
#'| ---
```

foo.R → foo.md

```
#'| ---  
#'| output: github_document  
#'| ---
```

Add this YAML frontmatter

Re-Compile Notebook

```
#' ---  
#' output: github_document  
#' ---
```

What changed?

This is what I mean by "explore cause and effect"
and "experiment without fear".

What changed in Git pane?

Inspect the diff.

Stage.

Commit.

Push.

Verify the .md file is now on GitHub.

Revel in how nice the .md looks!

01_explore-libraries_jenny.R

jenny Sat Jan 27 22:46:07 2018



```
## how jenny might do this in a first exploration  
## purposely leaving a few things to change later!
```

Which libraries does R search for packages?

```
.libPaths()
```

```
## [1] "/Users/jenny/resources/R/library"  
## [2] "/Library/Frameworks/R.framework/Versions/3.4/Resources/library"
```

```
## let's confirm the second element is, in fact, the default library  
.Library
```

```
## [1] "/Library/Frameworks/R.framework/Resources/library"
```

This is what I mean by "expose your work".

Take away #1:

Consider putting rendered products on GitHub.

Just because someone can fork, clone, install all necessary packages, then run your code, it doesn't mean they want to or will.

Be kind. Be realistic.

Take away #2:

For consumption on GitHub, Markdown (.md) is vastly more useful than .html, .docx, .pdf, etc.

Binary formats like .docx and .pdf are also a reliable source of merge conflicts. Think carefully before you track them with Git.

Resources re: which files to commit & how to make
your repo browsable

Excuse Me, ... section re: "Which files to commit"

Make a GitHub repo browsable

<http://happygitwithr.com/repo-browsability.html>

Refine your "explore libraries" R script.

After each meaningful change, re-render.

What changed? Look at the diffs.

Stage. Commit. Push. Check result on GitHub.

This is what I mean by "embrace incrementalism".

independent work on challenge

ideas:

- Bring your whole my-packages project over, gradually, making lots of commits. Play with rendering to .md.
- Tweak the code if you like.
- `add devtools::session_info()` at the end or `sessionInfo()` if no devtools

What now? Depends on what time it is!

Please open issues for questions that you 've raised and we discussed! Useful to me for planning tomorrow's final Git/GitHub session.

What follows are slides I can imagine us referring to today or tomorrow.

What now? Game time decision! Possibilities:

Equivalence between R and Rmd.

Use the secret README in the my-packages project.

GitHub Pages, the Simple Version.

Equivalence between .Rmd and .R

A

```
1 ---
2 title: "Report from R/Rmd"
3 author: "Jenny Bryan"
4 date: "`r format(Sys.Date())`"
5 output: github_document
6 ---
7
8 The iris data is boring, but it won't distract
9 from the Git content.
10
11 ```{r}
12 aggregate(. ~ Species, data = iris, median)
13 ```
```

B

```
1 #' ---
2 #' title: "Report from R/Rmd"
3 #' author: "Jenny Bryan"
4 #' date: "`r format(Sys.Date())`"
5 #' output: github_document
6 #' ---
7
8 #' The iris data is boring, but it won't distract
9 #' from the Git content.
10
11 aggregate(. ~ Species, data = iris, median)
```

C

Report from R/Rmd

Jenny Bryan
2017-06-29

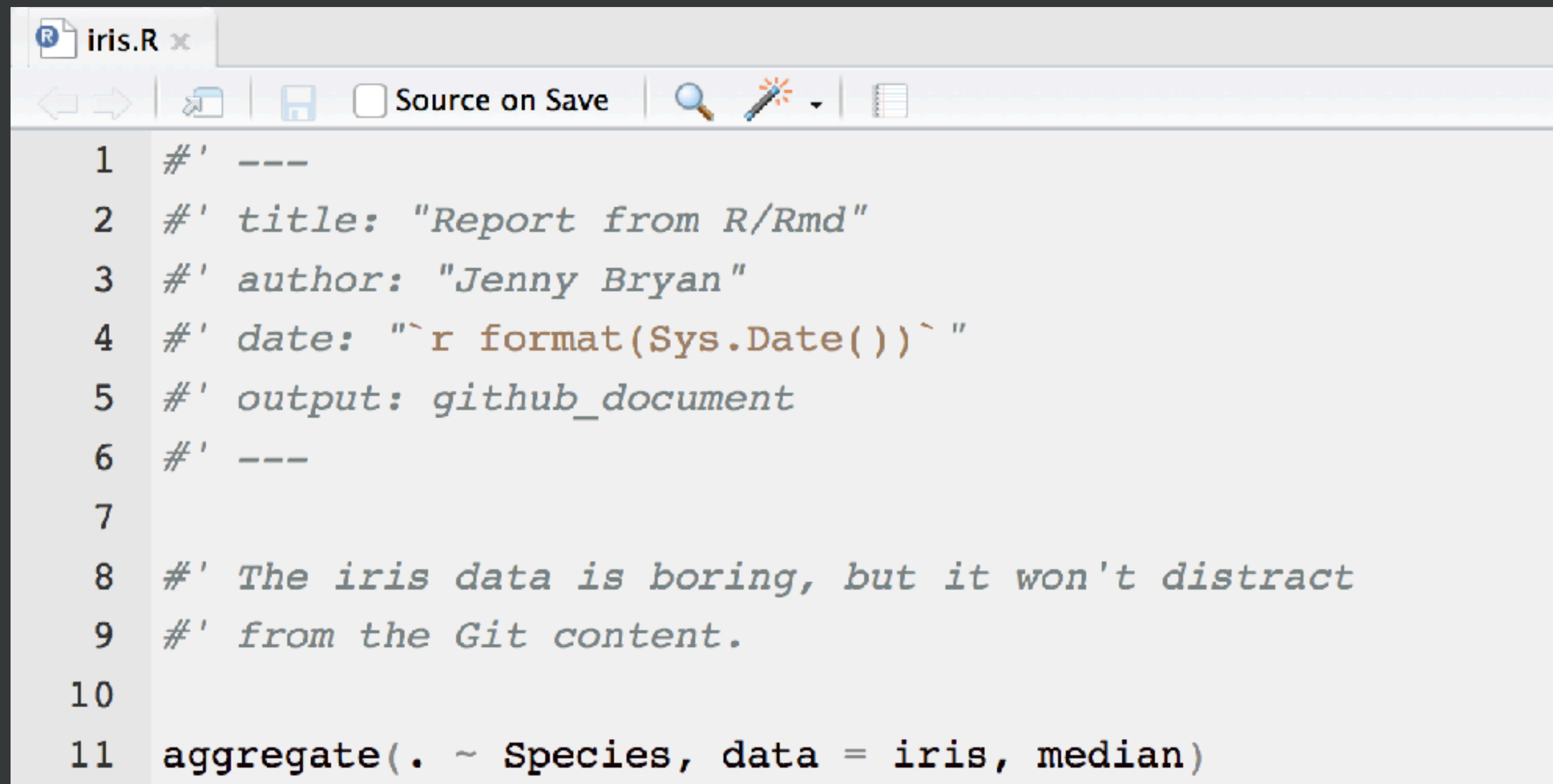
The iris data is boring, but it won't distract from the Git content.

```
aggregate(. ~ Species, data = iris, median)
```

	Species	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
## 1	setosa	5.0	3.4	1.50	0.2
## 2	versicolor	5.9	2.8	4.35	1.3
## 3	virginica	6.5	3.0	5.55	2.0

From "Excuse Me, ..." article

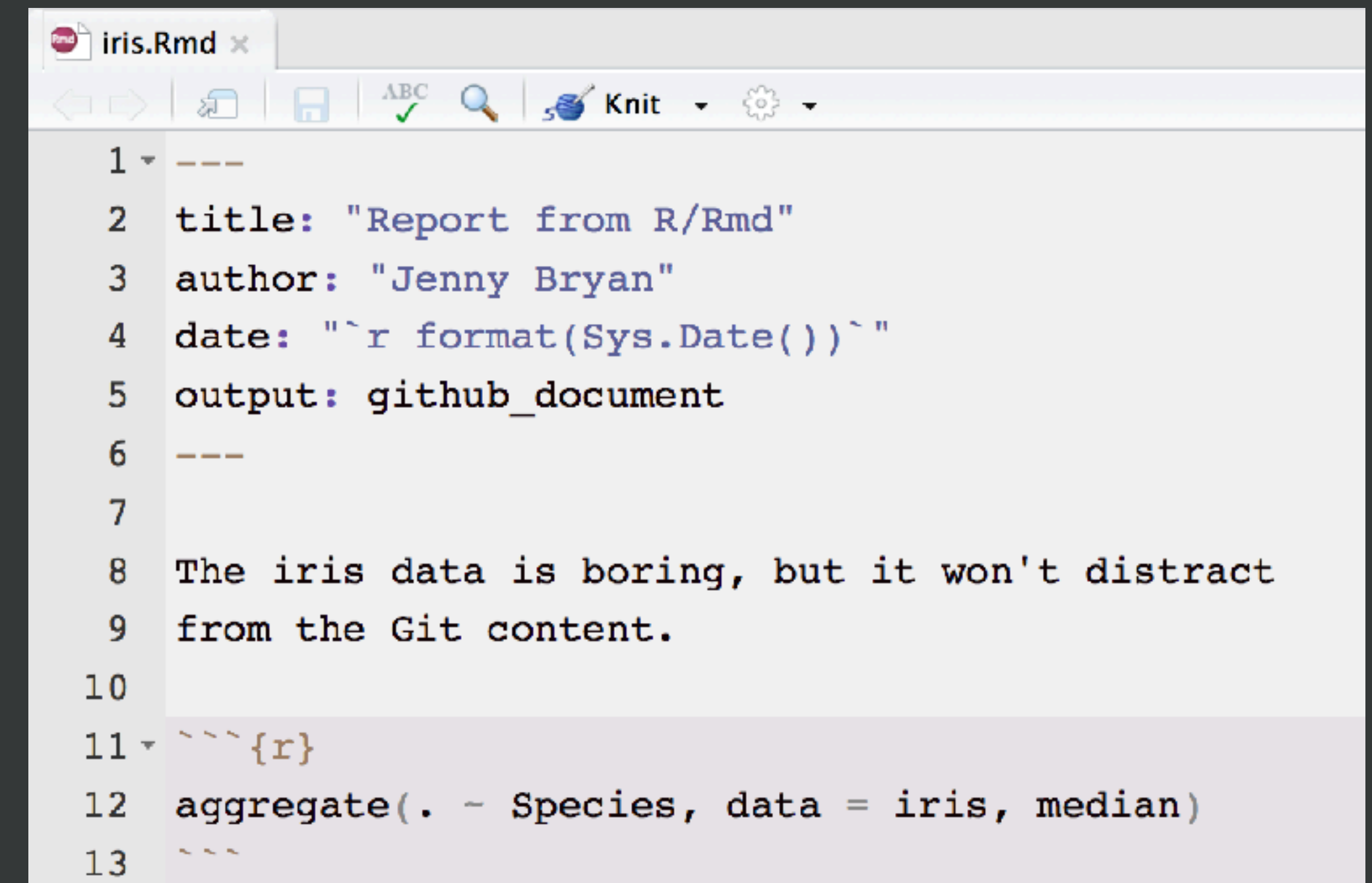
.R



```
1 #' ---
2 #' title: "Report from R/Rmd"
3 #' author: "Jenny Bryan"
4 #' date: "~r format(Sys.Date())~"
5 #' output: github_document
6 #' ---
7
8 #' The iris data is boring, but it won't distract
9 #' from the Git content.
10
11 aggregate(. ~ Species, data = iris, median)
```

R code is top-level
Use #' comment for prose
#+ for chunk header

.Rmd



```
1 ---
2 title: "Report from R/Rmd"
3 author: "Jenny Bryan"
4 date: "~r format(Sys.Date())~"
5 output: github_document
6 ---
7
8 The iris data is boring, but it won't distract
9 from the Git content.
10
11 ~~~{r}
12 aggregate(. ~ Species, data = iris, median)
13 ~~~
```

Prose is top-level
Put R code in chunks

Nice defaults for global chunk options

```
knitr::opts_chunk$set(  
  collapse = TRUE,  
  comment = "#>",  
  out.width = "100%"  
)
```

foo.Rmd → foo.html

```
---  
title: "Untitled"  
output: html_document  
---
```

foo.Rmd → foo.md → foo.html

```
---  
title: "Untitled"  
output:  
  html_document:  
    keep_md: yes  
---
```

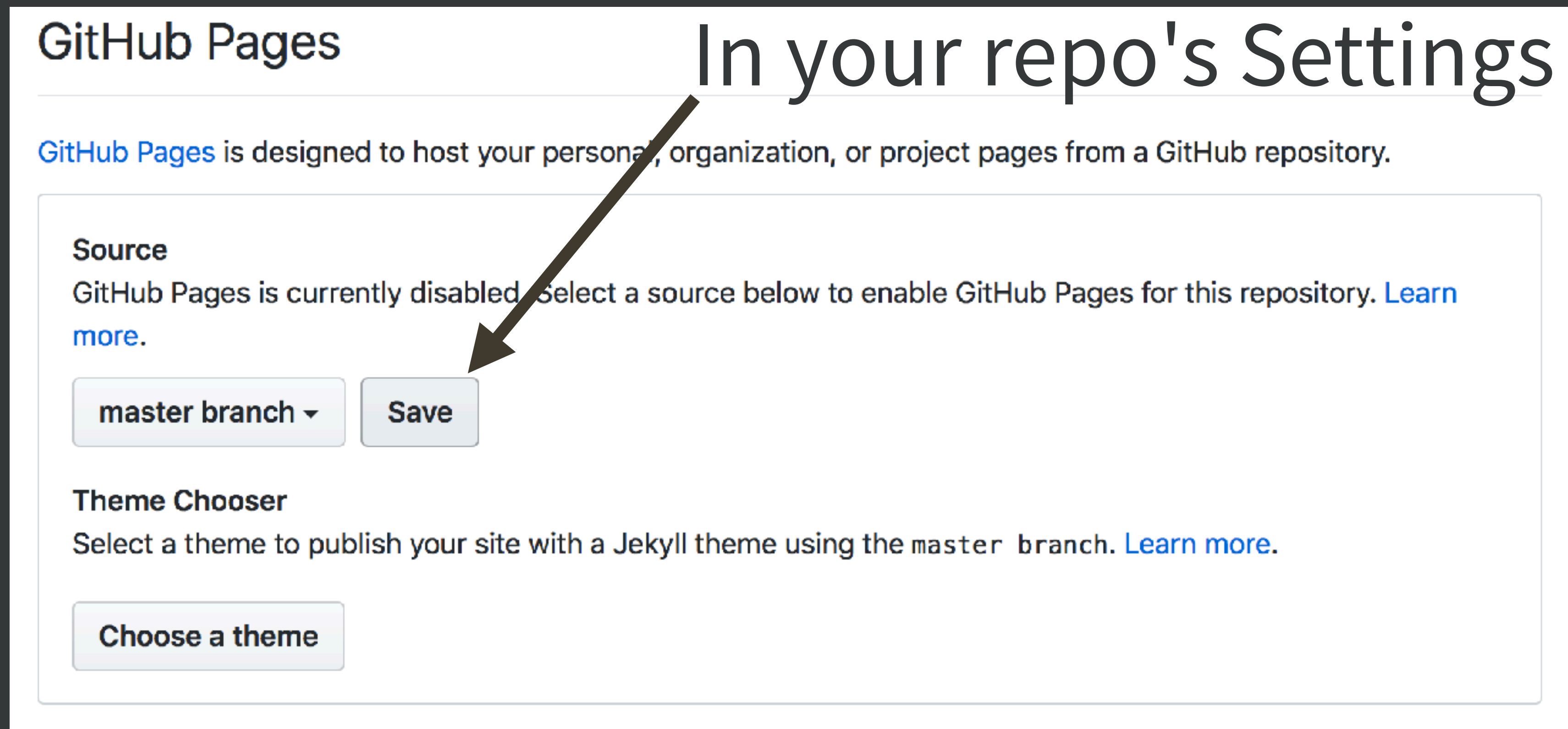
foo.Rmd → foo.md

```
---  
output:  
  md_document  
---
```

foo.Rmd → foo.md

```
---  
output:  
  github_document  
---
```

Simplest use of GitHub Pages = Project webpage



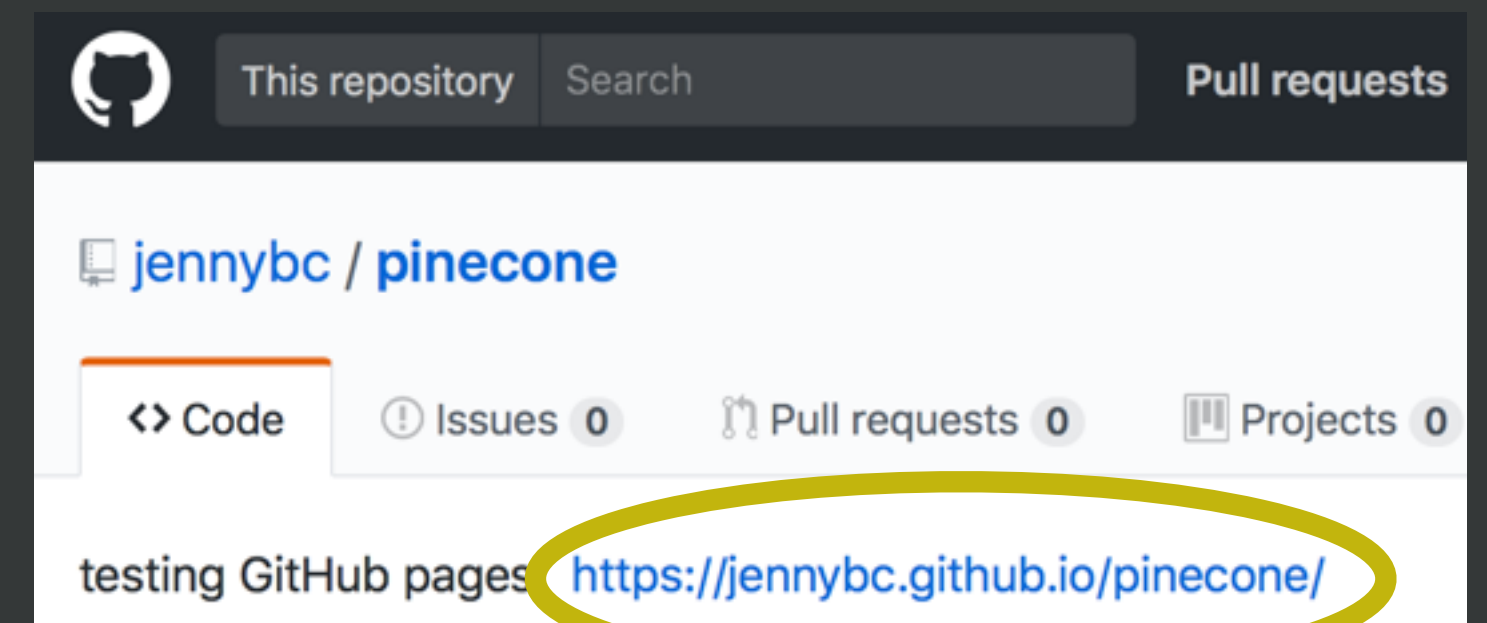
<https://github.com/blog/2289-publishing-with-github-pages-now-as-easy-as-1-2-3>

Hot tips for simple GitHub Pages

README.md becomes index.html, by default

Given that foo.md exists, these internal links work (trial & error):
👍 foo, [foo](foo.md), [foo](foo.html)

Record your site URL as your repo's website



transition to Jim's lesson on
branches and remotes around here

we will likely cover remainder of
this on day 2, if he does not

Recovering from Git(Hub) failure

Scenario: You try to push and cannot

What's the problem?

There are changes on GitHub that you don't have.

Pull. If the gods smile upon you, merge works. Now push.

Let's create this situation.

Make sure local Git pane is clear.

Make sure local and remote are synced (push, pull).

Edit & commit to **file A** locally.

Edit & commit to **file B** remotely.

Try to push. You will fail.



```
jenny@2015-mbp bunny-scarf $ git push
To github.com:jennybc/bunny-scarf.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:jennybc/bunny-scarf.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

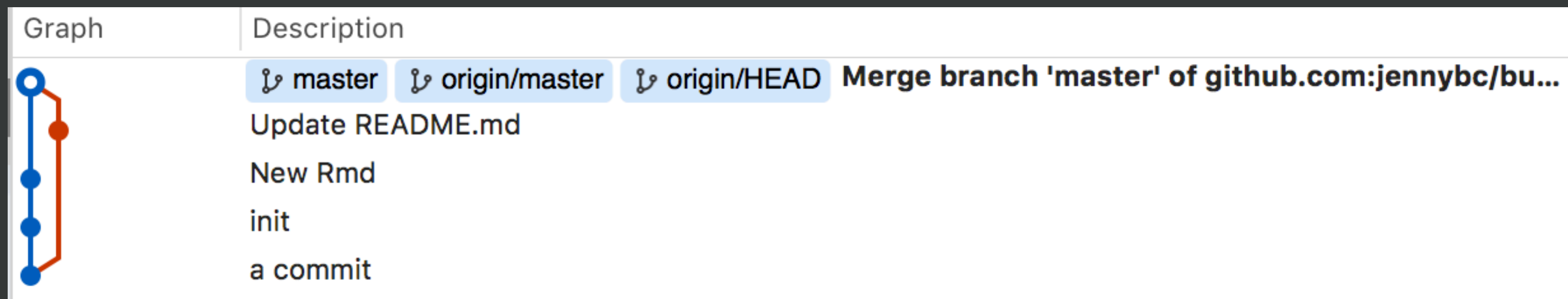
Remedy? Do what it says!

pull, then push ... **pull, then push** ... pull, then push

Look at your Git history.

You will see a merge commit, where the local and remote changes were reconciled.

This is best case scenario and is likely with good Git habits (lots of small frequent commits and merges, no binary files in repo).



Recovering from Git(Hub) failure

Scenario: You pull and get a merge conflict.

What's the problem?

GitHub can't figure out how to reconcile diffs.

Resolve the conflicts.

Or abort ... and come back later.

Let's create this situation.

Make sure local Git pane is clear.

Make sure local and remote are synced (push, pull).

Edit & commit to **file A** locally.

Make conflicting edit & commit to **file A** remotely.

Try to push. You will fail. Try to pull. You will fail. All is fail.

From github.com:jennybc/bunny-scarf

958548f..3357952 master -> origin/master

Auto-merging README.md

CONFLICT (content): Merge conflict in README.md

Automatic merge failed; fix conflicts and then commit the result.

<<<<<< HEAD

Wingardium Leviosaaaaaaa

=====

Wing-GAR-dium Levi-0-sa

>>>>>> 33579525d88af071268b0a0c64c54f357712589a

<<<<<< HEAD

Wingardium Leviosaaaaaaa

=====

Wing-GAR-dium Levi-0-sa

>>>>>> 33579525d88af071268b0a0c64c54f357712589a

Git inserts **markers** at each locus of conflict and shows you both versions.

You must form a consensus version and delete the markers, at each locus. Commit. Push. Carry on.

```
<<<<<< HEAD
```

```
Wingardium Leviosaaaaaaa
```

```
=====
```

```
Wing-GAR-dium Levi-0-sa
```

```
>>>>>> 33579525d88af071268b0a0c64c54f357712589a
```

If you're just not up for this right now, do
`git merge --abort` to back out.

You can keep working locally. But you must deal with this problem before you can resume syncing with GitHub.

Recovering from Git(Hub) failure

Scenario: You have a huge mess you cannot fix.

Official answer: git reset.

Unofficial answer: burn it all down 🔥

So I can face Jim Hester when he sees this:

`git reset (mixed and hard)` is genuinely worth learning.

SourceTree, for example, makes it easy to do hard or mixed resets to previous states.

After you reset to a non-broken state, have another go at whatever you were doing.

**The amount of energy
necessary to refute
bullshit is an order of
magnitude bigger
than to produce it**

- Alberto Brandolini

The amount of Git skillz
necessary to fix a borked up
repo is an order of magnitude
bigger than to bork it.

- Me



BURN IT ALL DOWN

🔥 requires you have a remote repo in a decent state!

Commit early, commit often! And push! It's your safety net.

Rename local repo to, e.g. "foo-borked".

Re-clone to a new, clean local repo, "foo".

Copy any files that are better locally from "foo-borked" to "foo".
Commit. Push. Carry on.

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.



Sidebar re: tracking branches
Hopefully won't come up

typical .git/config with
"Github first, then RStudio"
= git clone

```
[core]
  repositoryformatversion = 0
  filemode = true
  bare = false
  logallrefupdates = true
  ignorecase = true
  precomposeunicode = true
[remote "origin"]
  url = git@github.com:jennybc/bunny-hat.git
  fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
  remote = origin
  merge = refs/heads/master
```

YAASSS

typical .git/config after

```
[core]
```

```
repositoryformatversion = 0
```

```
filemode = true
```

```
bare = false
```

```
logallrefupdates = true
```

```
ignorecase = true
```

```
precomposeunicode = true
```

```
[remote "origin"]
```

```
url = git@github.com:jennybc/bunny-scarf.git
```

```
fetch = +refs/heads/*:refs/remotes/origin/*
```

```
git init  
git remote add
```



typical .git/config after

```
[core]
```

```
repositoryformatversion = 0
```

```
filemode = true
```

```
bare = false
```

```
logallrefupdates = true
```

```
ignorecase = true
```

```
precomposeunicode = true
```

```
[remote "origin"]
```

```
url = git@github.com:jennybc/bunny-scarf.git
```

```
fetch = +refs/heads/*:refs/remotes/origin/*
```

```
[branch "master"]
```

```
remote = origin
```

```
merge = refs/heads/master
```

```
git init
```

```
git remote add
```

```
git push --set-upstream origin master
```

