
What They Forgot to Teach You About R
rstudio::conf
SAN FRANCISCO // JANUARY 27 - 30, 2020
from RStudio

rstd.io/wtf-2020-rsc

Day 1, afternoon
Let's Git it On.

Kara Woo
 @karawoo
 @kara_woo

Jenny Bryan
RStudio
 @jennybc
 @JennyBryan

Everyone is encouraged to open issues here:

rstd.io/wtf-2020-rsc

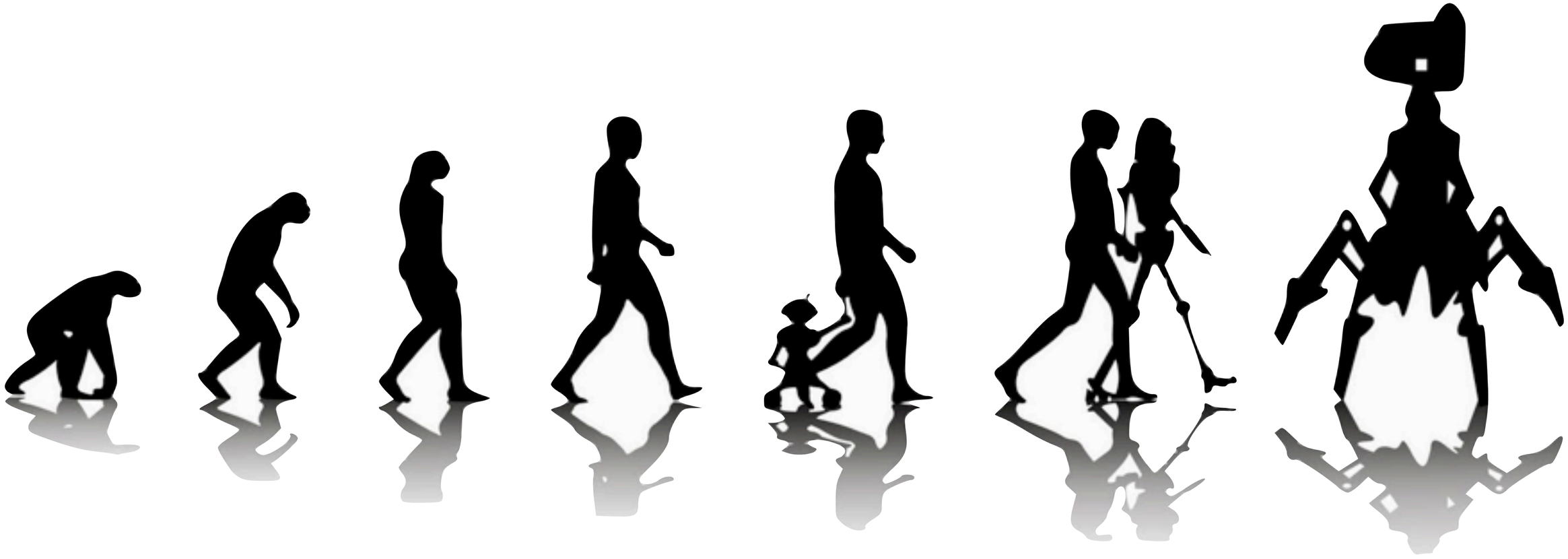
github.com/rstudio-conf-2020/what-they-forgot/issues

Record glitches, gotchas,

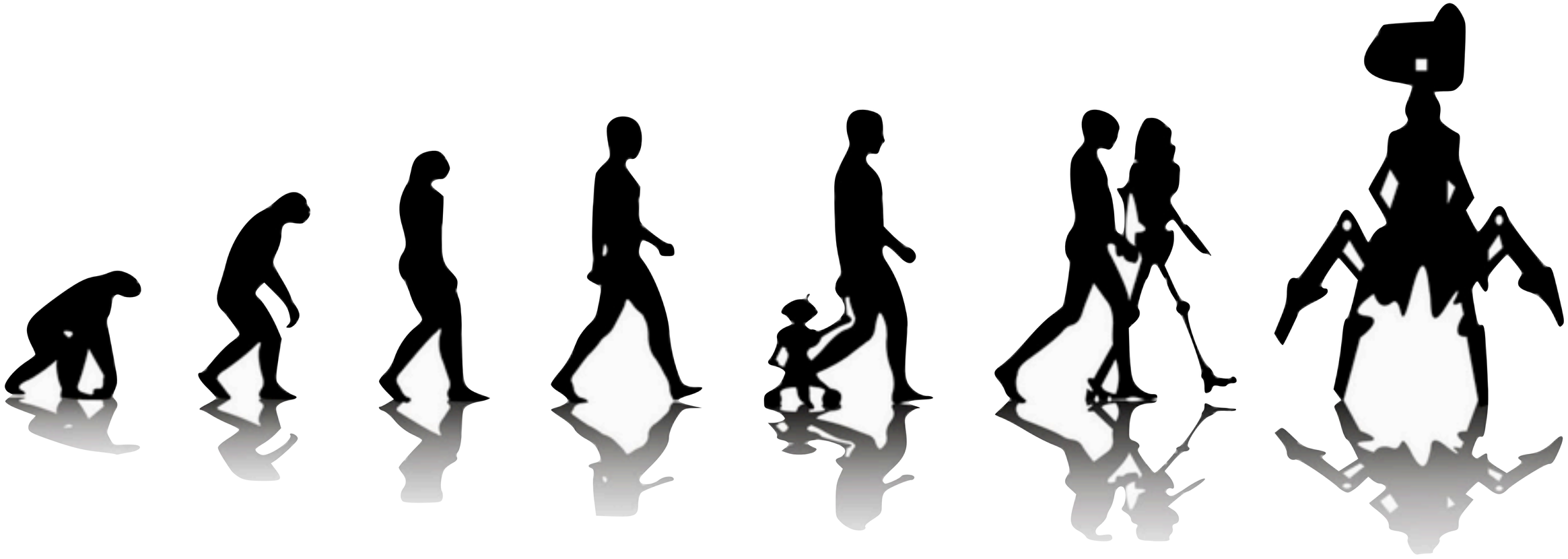
good sidebar discussions, etc.

to address now or later.

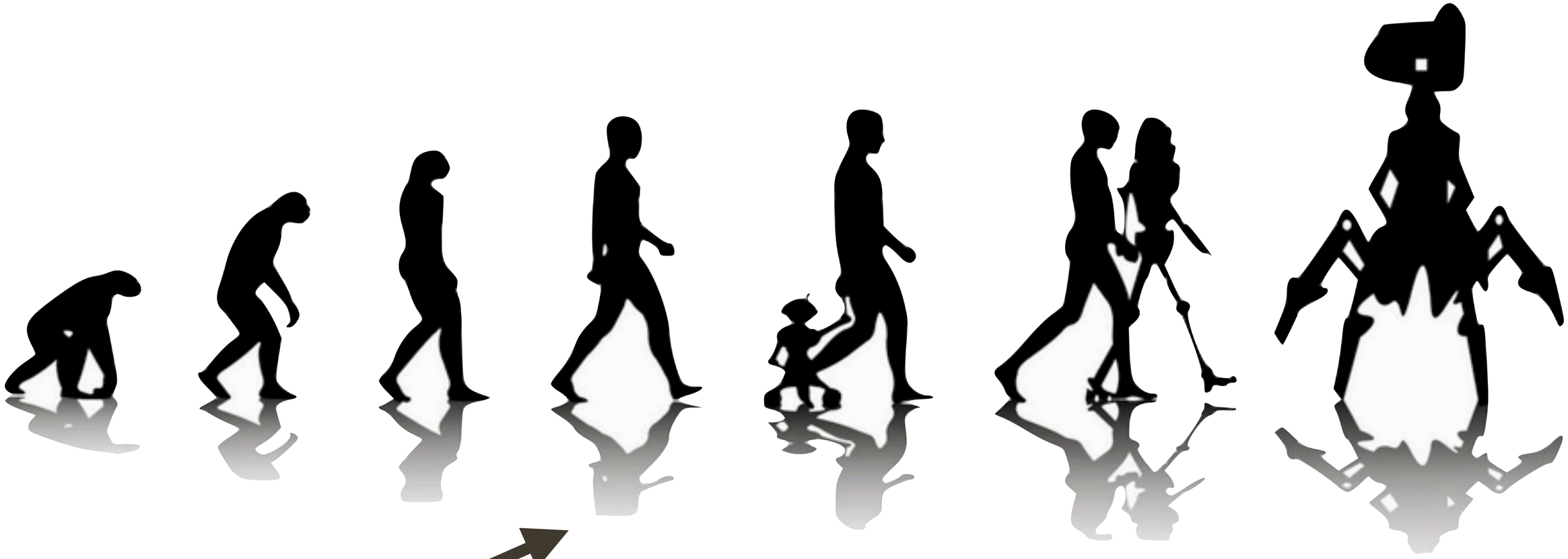

Deep
Thoughts



use version control



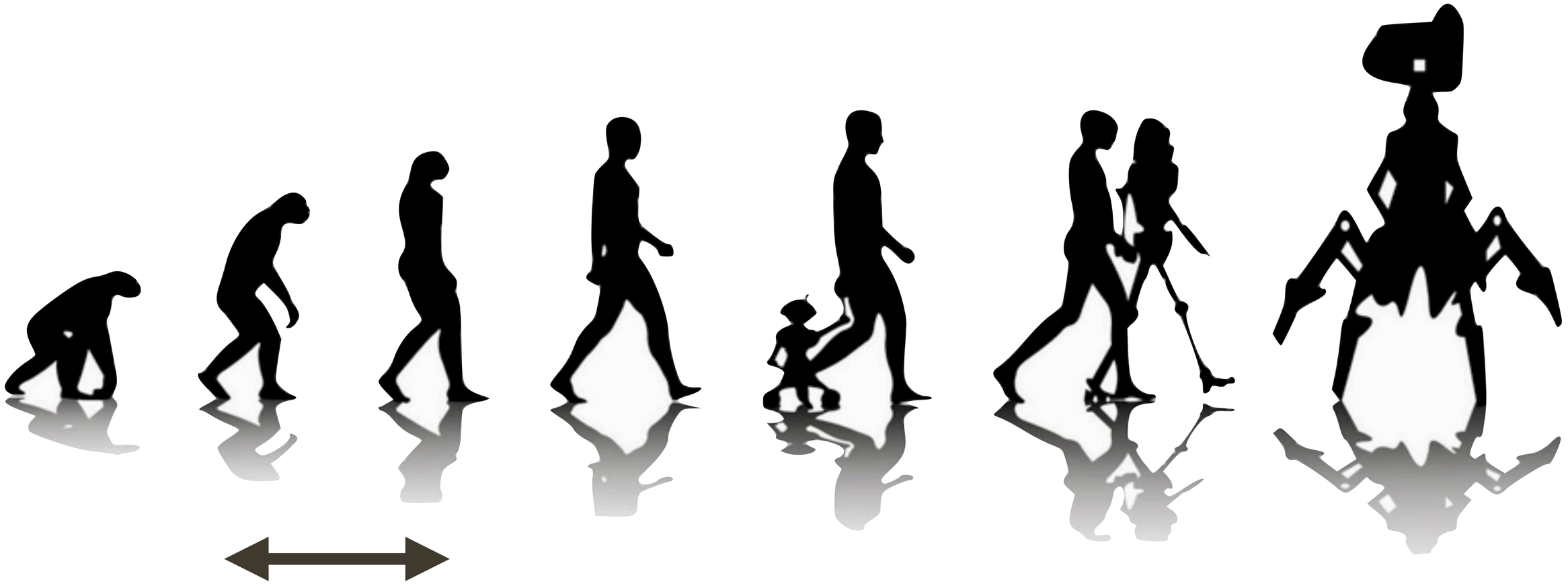
we teach Git + GitHub



"commit"

a file or project state that is **meaningful to you**

for inspection, comparison, restoration

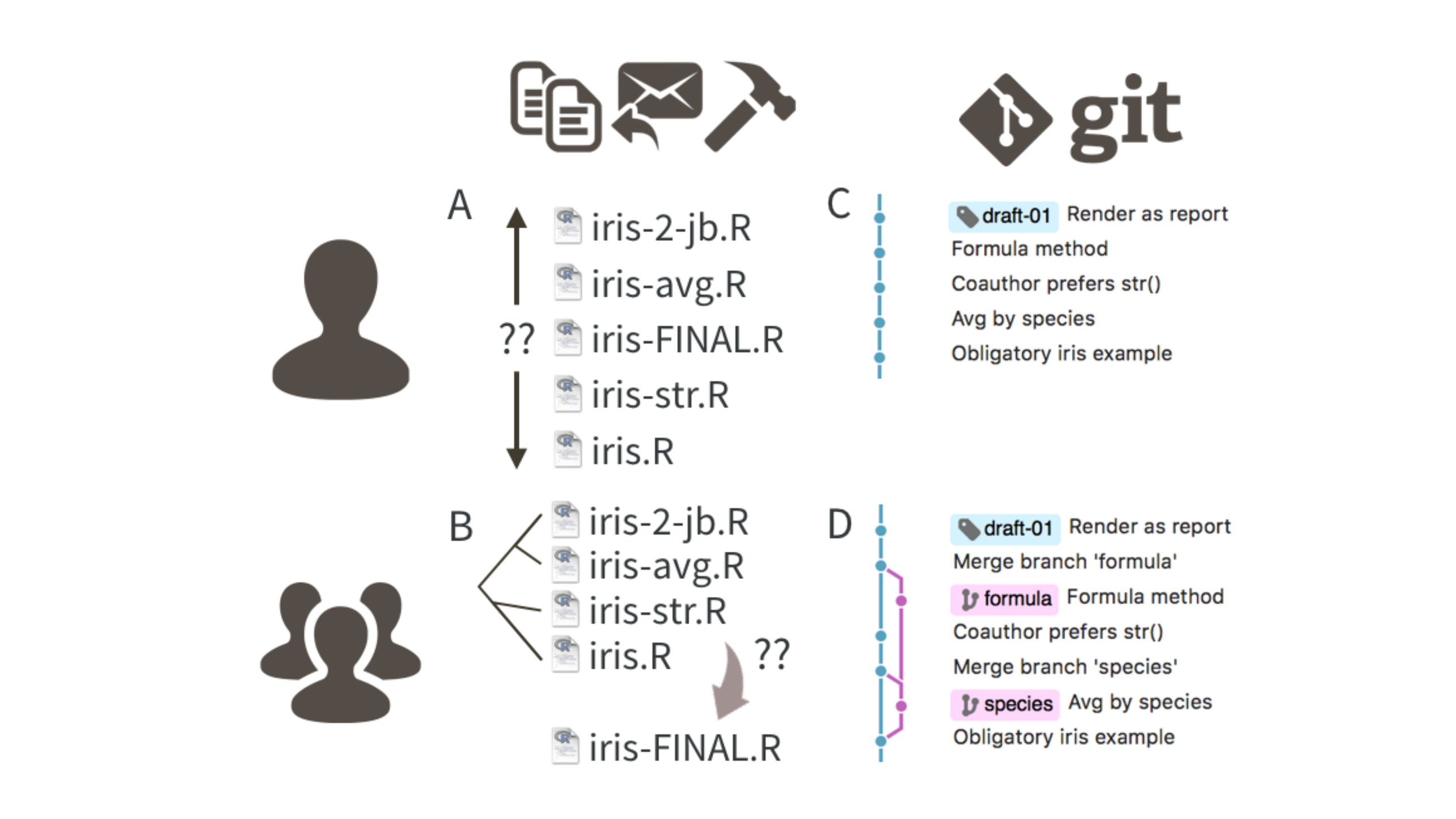


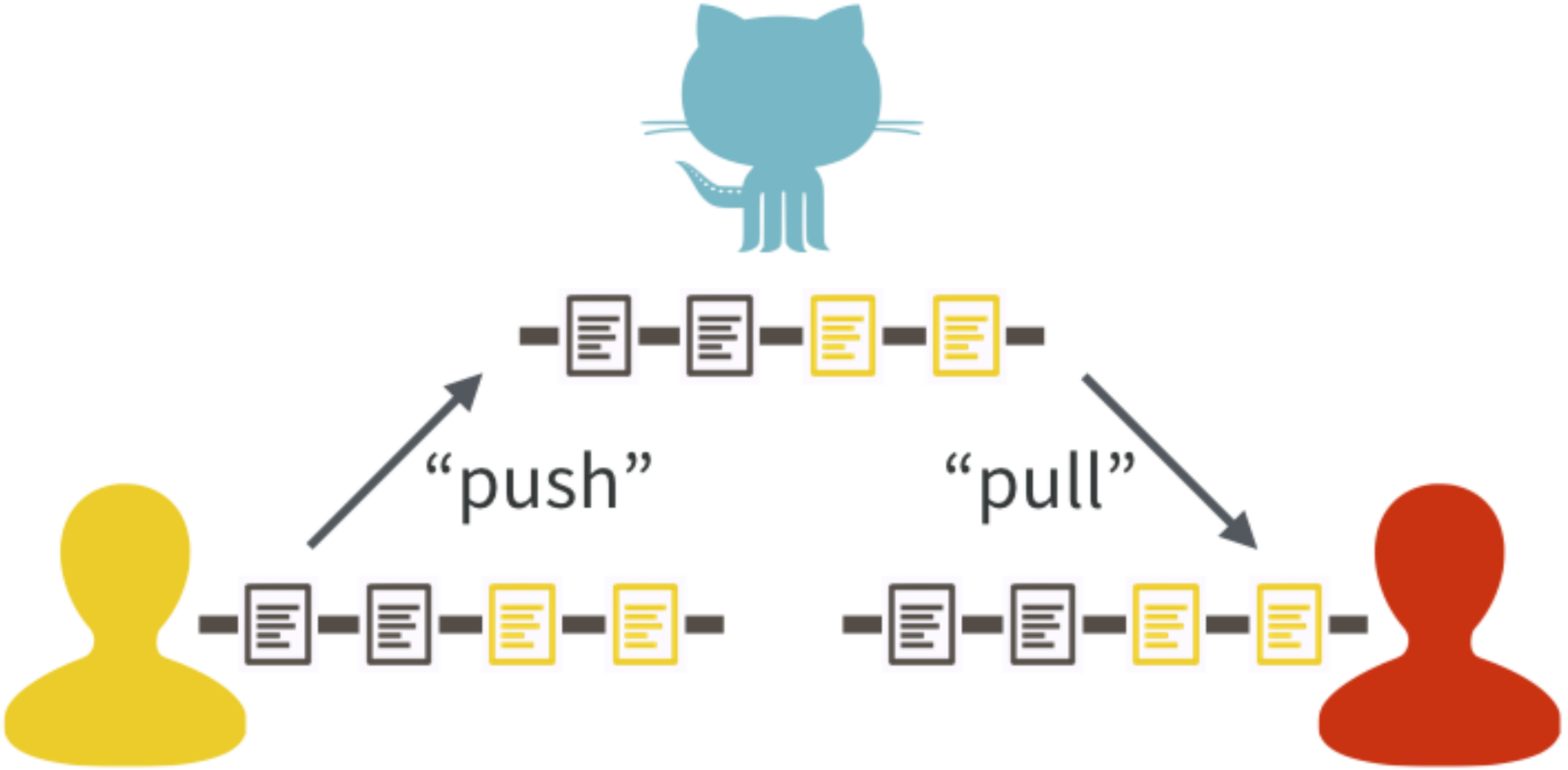
Δ

"diff"

What changed here?

Why?

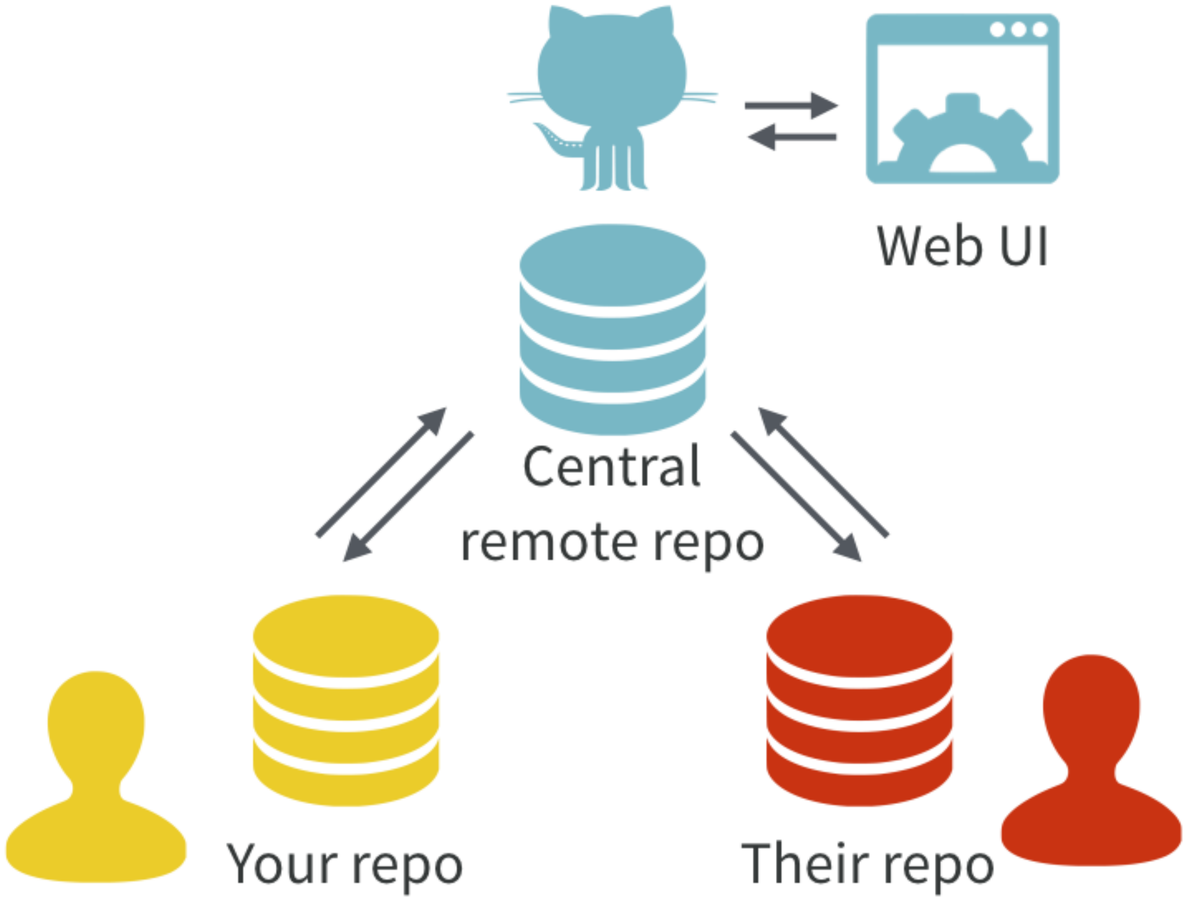

A
iris-2-jb.R
iris-avg.R
??
iris-FINAL.R
iris-str.R
iris.R
B
iris-2-jb.R
iris-avg.R
iris-str.R
iris.R
??
iris-FINAL.R
C
draft-01 Render as report
Formula method
Coauthor prefers str()
Avg by species
Obligatory iris example
D
draft-01 Render as report
Merge branch 'formula'
formula Formula method
Coauthor prefers str()
Merge branch 'species'
species Avg by species
Obligatory iris example
git



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

“If you wish to make an apple pie from scratch, you must first invent the universe.”

–Carl Sagan

“If you wish to make an apple pie from scratch, you must first invent the universe.”

–Carl Sagan

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 + GitKraken

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


Them
You
not your
problem

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

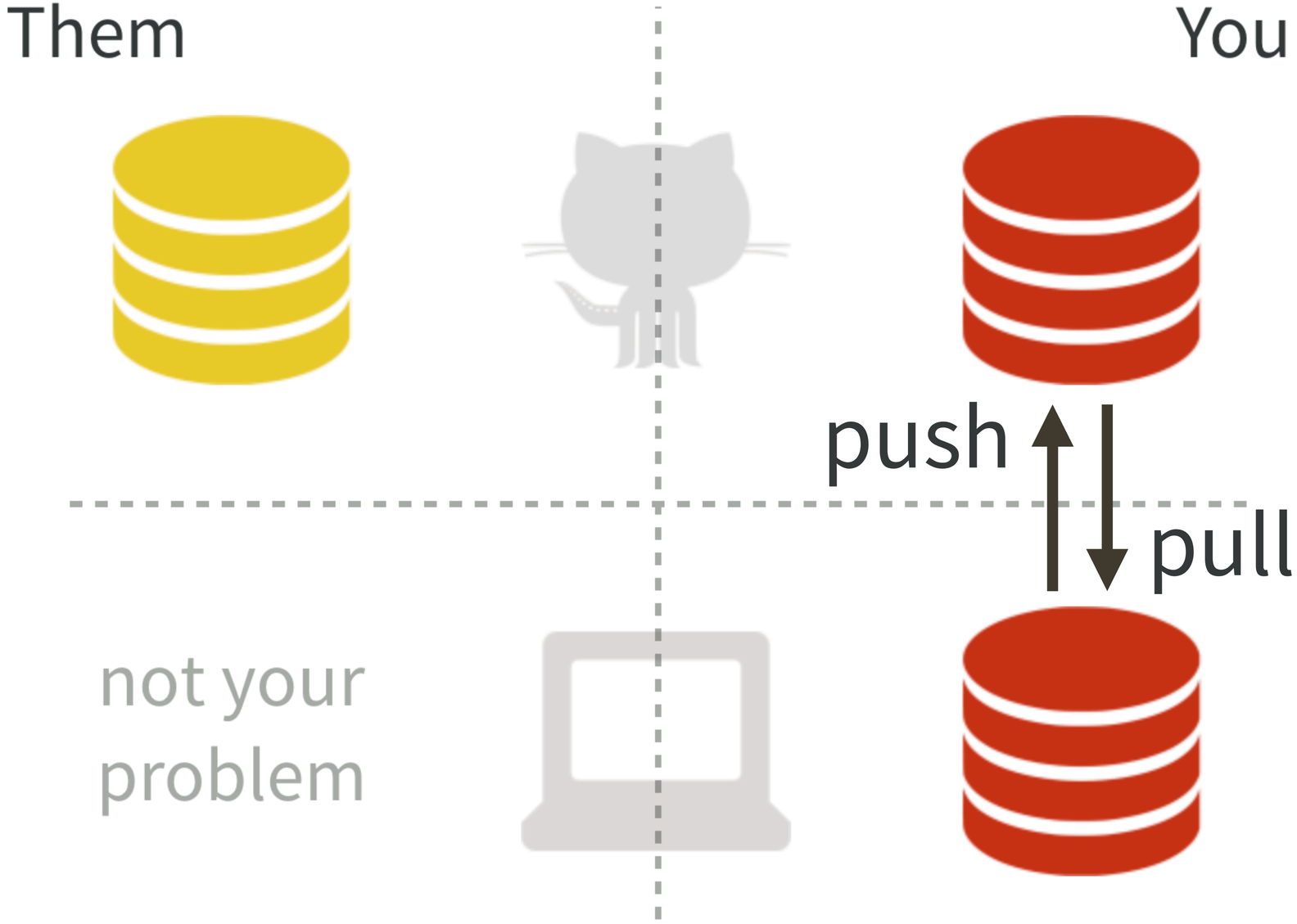
Project initiation strategies, the remote case

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

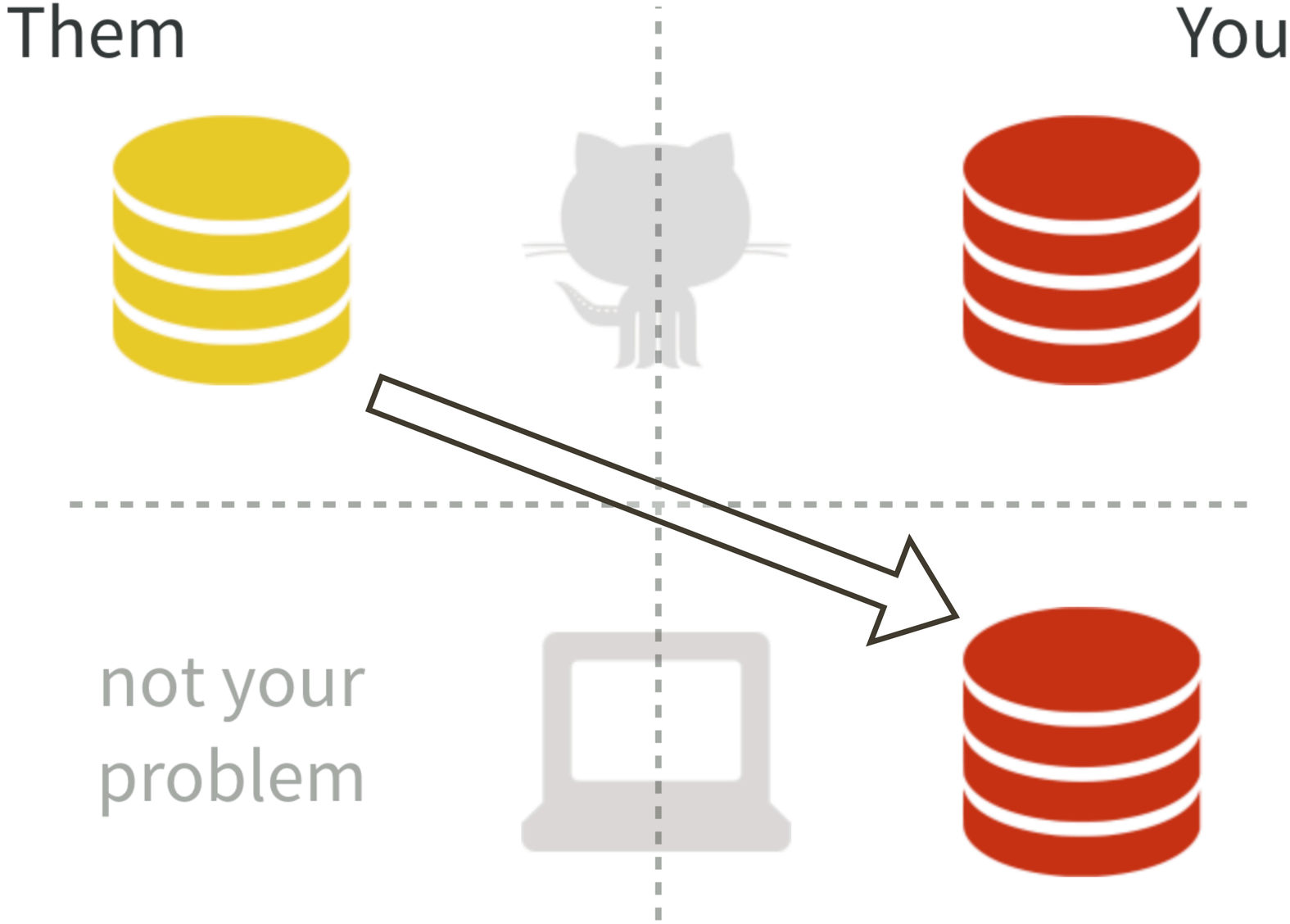
Make a local Project from a remote repo, Yours ~~or Theirs~~ = "clone"

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

"clone"

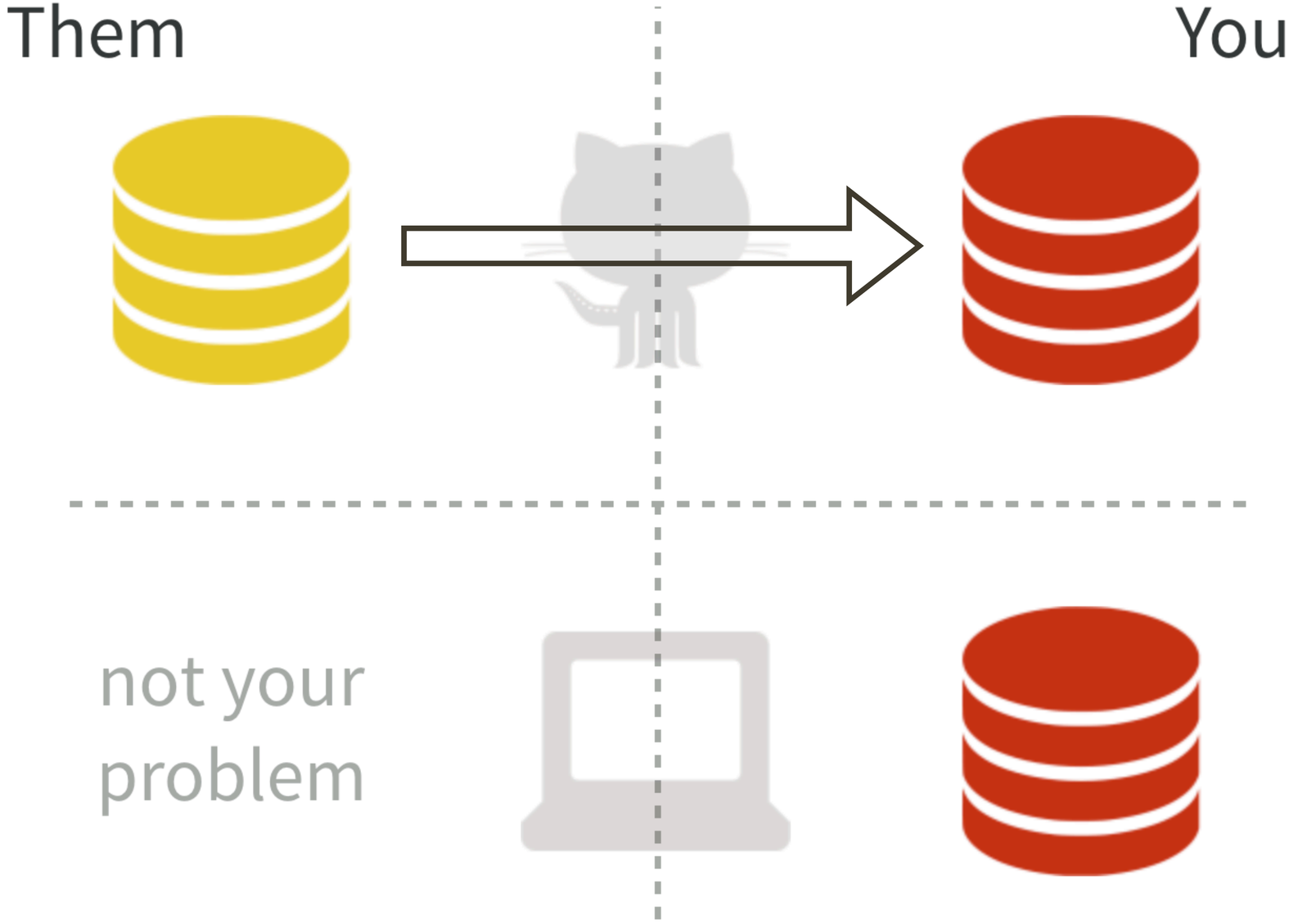


daily work, your stuff

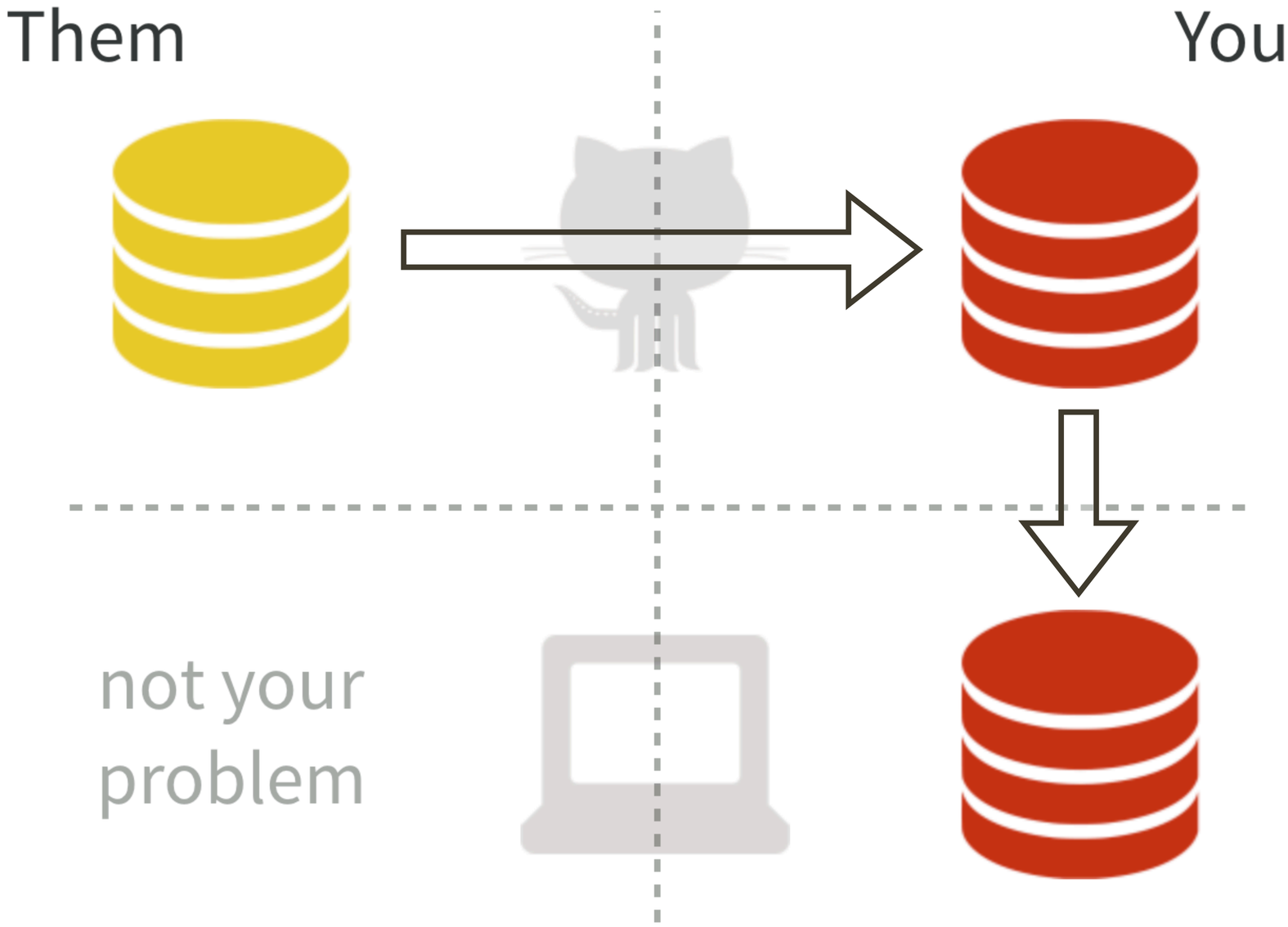


"clone"

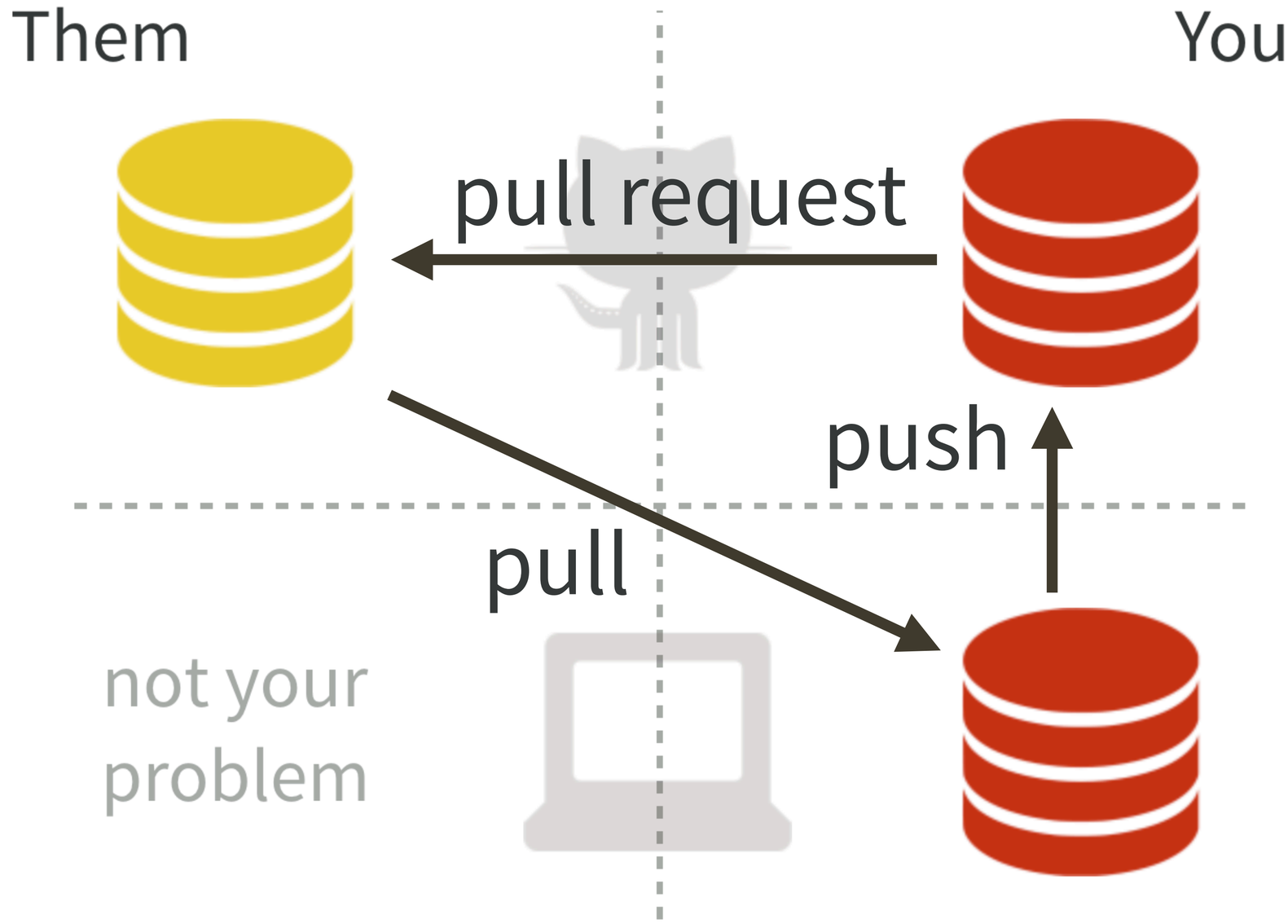
*not as useful as you might think



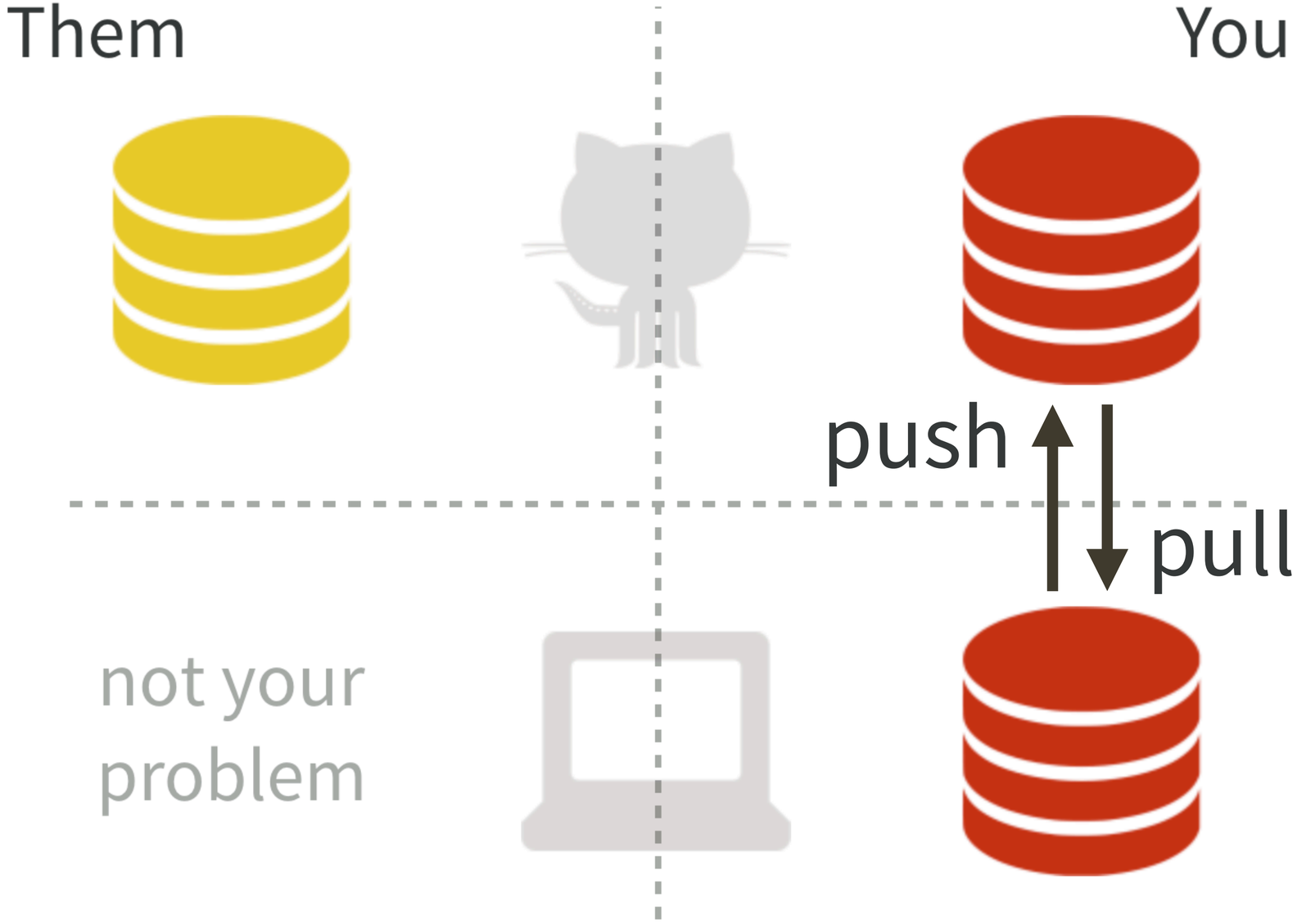
"fork"



"fork and clone"



contribute to other people's stuff



daily work, your stuff

"New project, GitHub first"
Why do I emphasize this?

My diagrams omit two big technical points:

- remotes
- branches

I want you to bake some pies before messing around with remotes and branches.

"New project, GitHub first" workflow

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

I suggest:

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

coordinated work through this:
https://happygitwithr.com/new-github-first.html

Create a new .R file.

Use a little bit of code developed earlier today.

Doesn't matter much what it is.

Just make sure it does something.

What changed in Git pane?

Inspect the diff.

Stage.

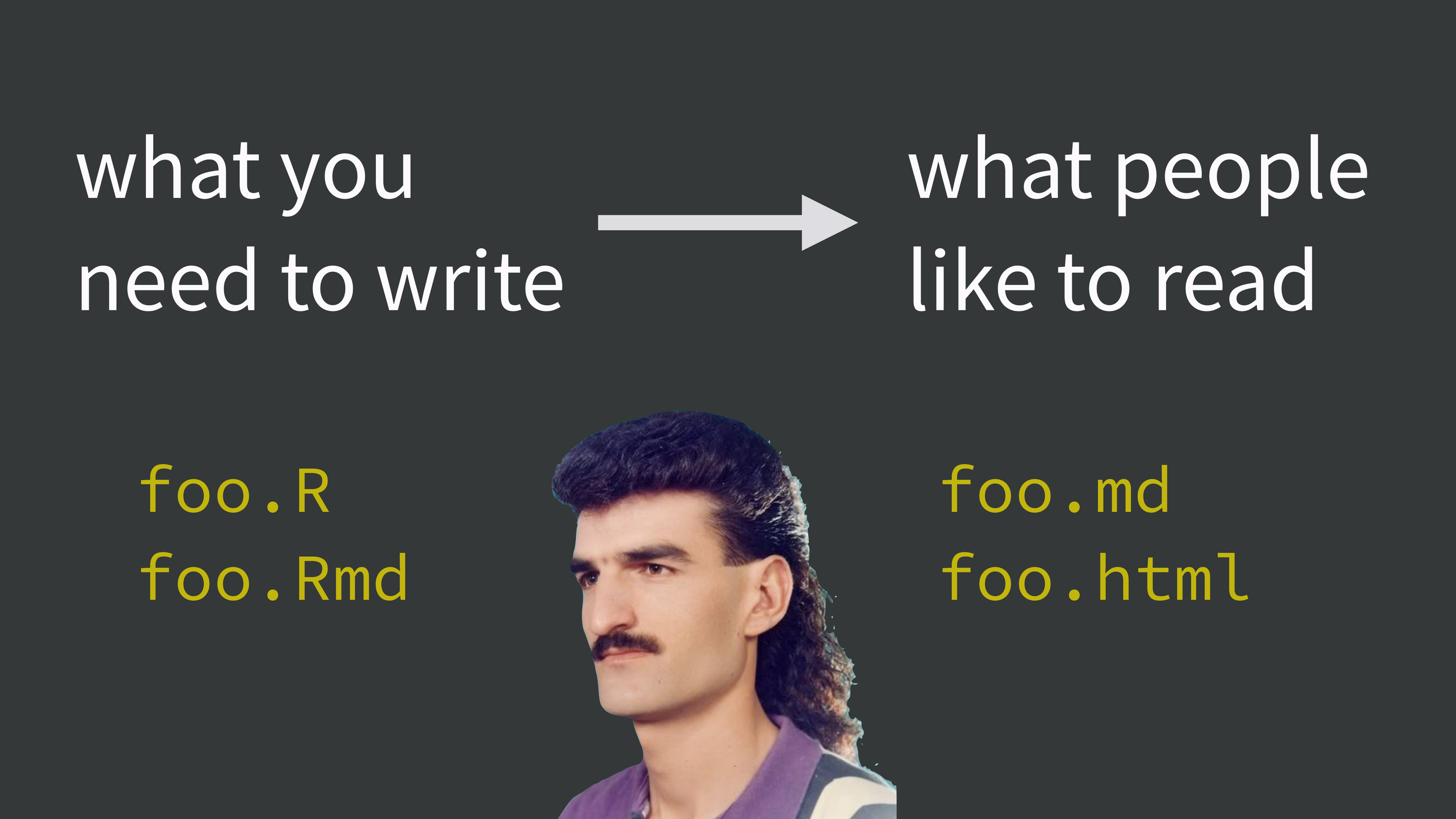
Commit.

Push.

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

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

what you need to write

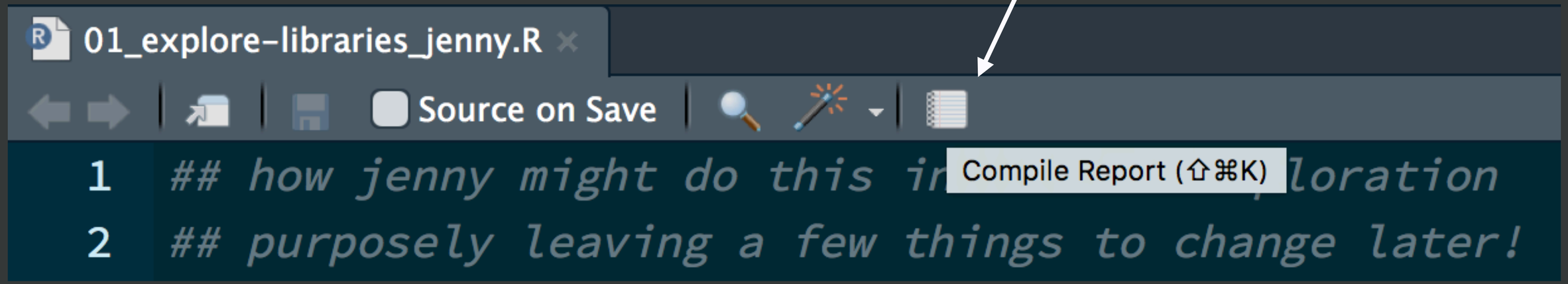


what people like to read

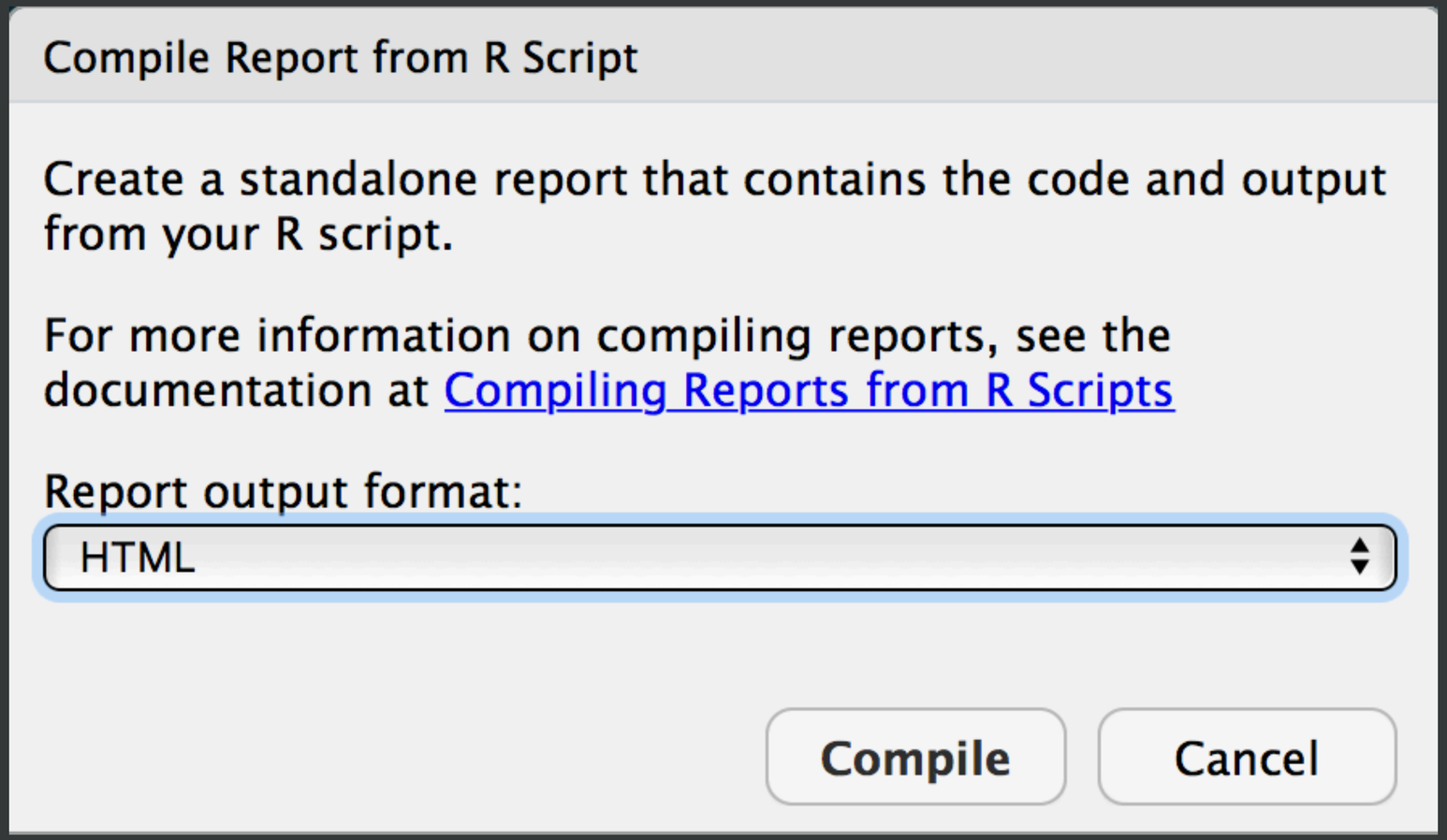
`foo.R`
`foo.Rmd`

`foo.md`
`foo.html`

Compile Report



≈ `rmarkdown::render("whatever.R")`



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?

248 lines (201 sloc) | 723 KB

Raw | Blame | History

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="generator" content="pandoc" />


<meta name="author" content="jenny" />


<title>01_explore-libraries_jenny.R</title>


<meta name="viewport" content="width=device-width, initial-scale=1" />

<script src="data:application/x-javascript;base64,LyohCiAqIEJvb3RzdHJhcCB2My4zLjUgKGh0dHA6Ly9nZXRib290c3RyYXAuY29tKQogKiBDb3B
<script src="data:application/x-javascript;base64,LyoqCiogQHByZXNlcnZlIEhUTUw1IFNoaXYgMy43LjIgfCBAYWZhcmthcyBAamRhbHRvbiBAam9u
<script src="data:application/x-javascript;base64,LyohIFJlc3BvbmQuanMgdjEuNC4yOiBtaW4vbWF4LXdpZHRoIG1lZGlhIHF1ZXJ5IHBvbHlmaWx
<script src="data:application/x-javascript;base64,CgovKioKICogalF1ZXJ5IFBsdWdpbjogU3RpY2t5IFRhYnMKICoKICogQGF1dGhvciBBaWRhbiBM
<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 (the "---" matter!)

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
https://happygitwithr.com/workflows-browsability.html

Start porting your library exploration work over.

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 wtf-packages-report project over (or the example solution), 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

Why did I make you create a GitHub PAT?

git operations via ssh	
example	`git clone git@github.com:OWNER/REPO.git`
creds	local private ssh key + public key on GitHub

git operations via https	
example	`git clone https://github.com/OWNER/REPO.git`
creds	username + password (password can be GITHUB_PAT)

GitHub API requests via REST	
example	`curl -H "Authorization: token $GITHUB_PAT" https://api.github.com/user/repos`
creds	GITHUB_PAT

```
usethis::create_from_github()
usethis::pr_push()
usethis::pr_pull()
```

Also: rate limiting

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 coverage.

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

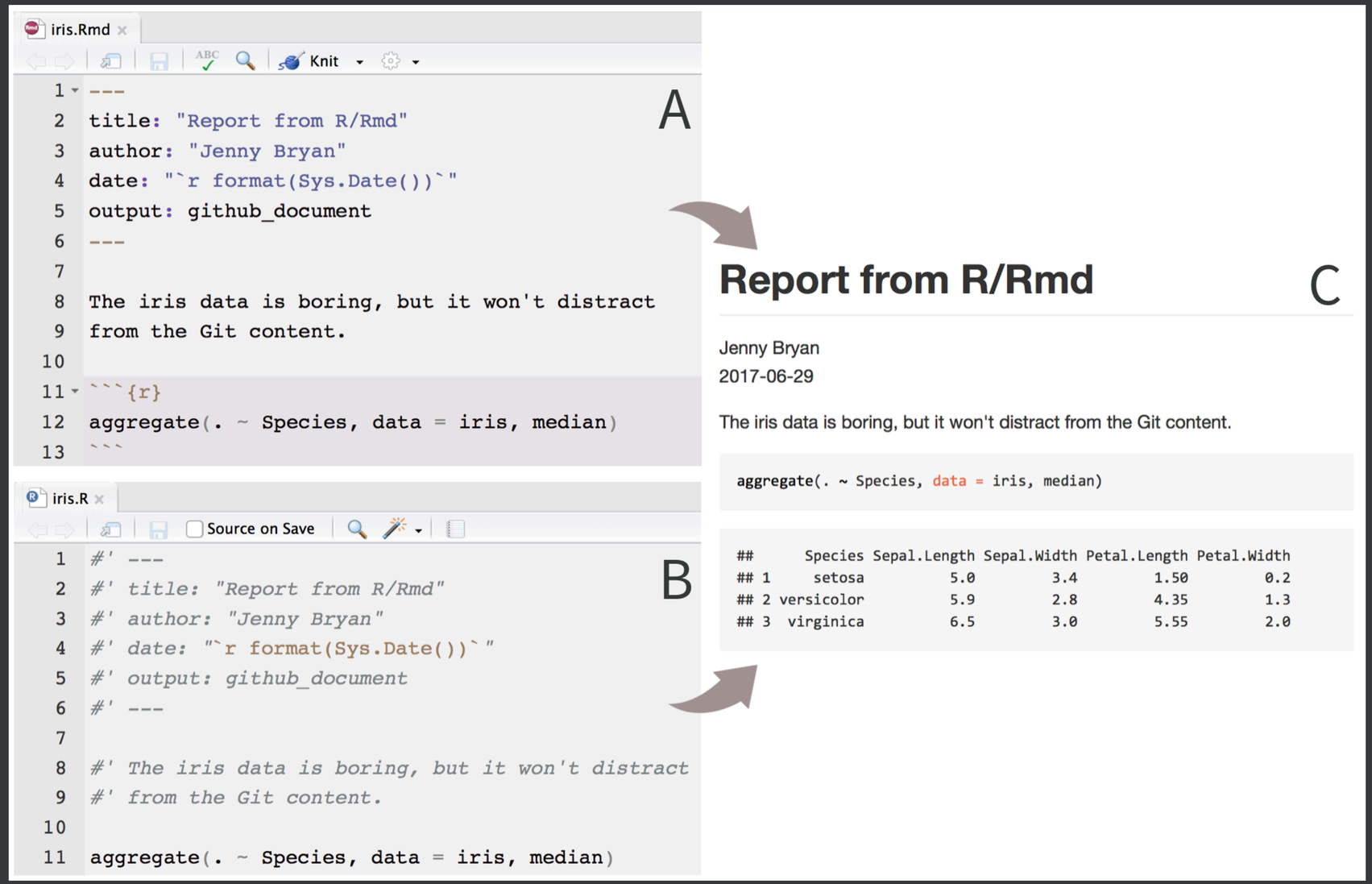
What now? Game time decision! Possibilities:

Equivalence between R and Rmd.

Use the secret README in the packages-report project.

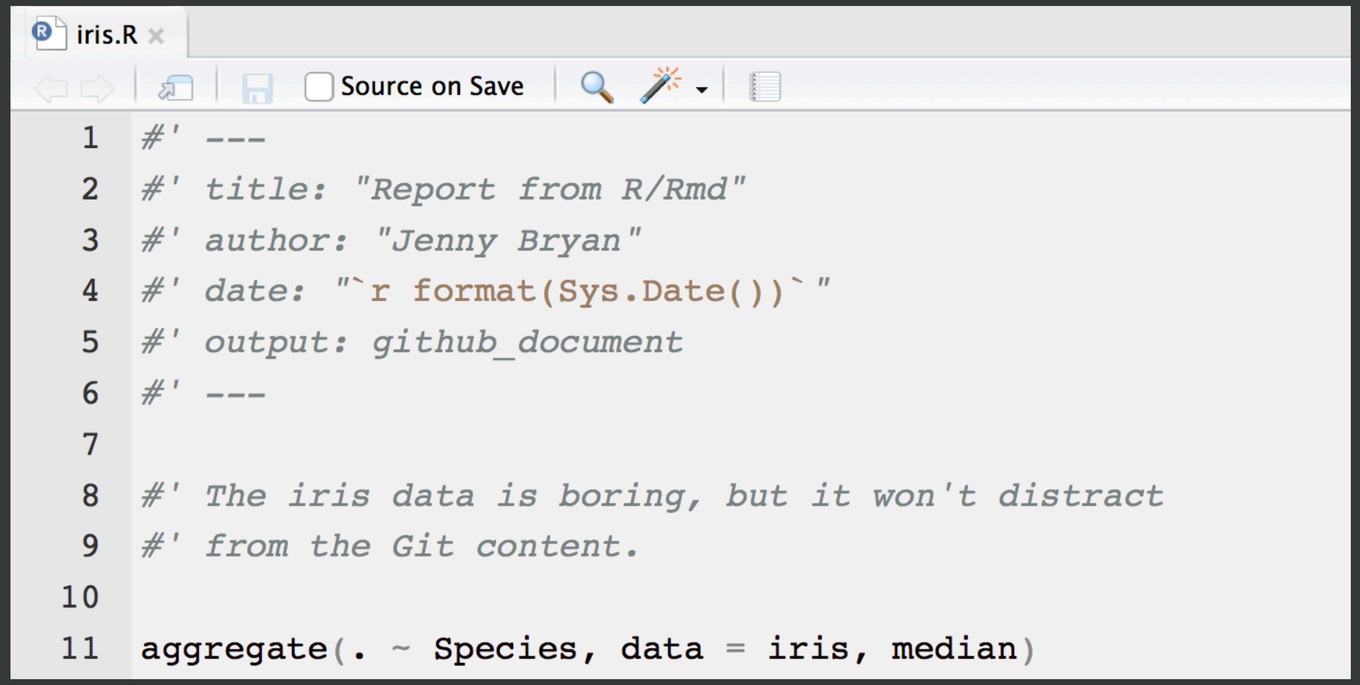
GitHub Pages, the Simple Version.

Equivalence between .Rmd and .R



From "Excuse Me, ..." article

.R

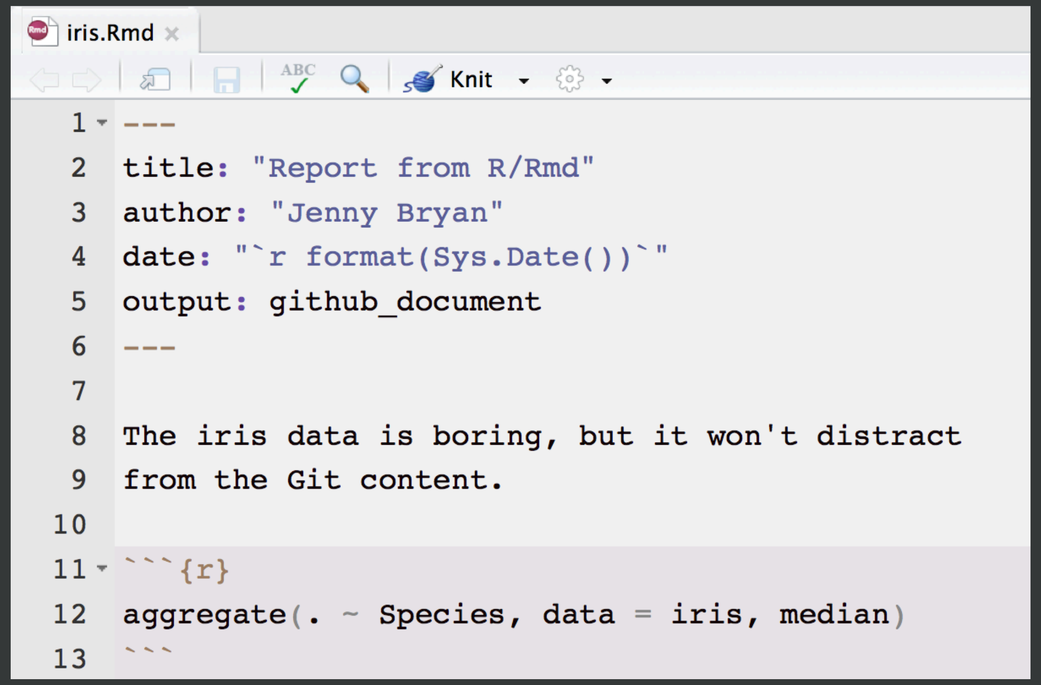


R code is top-level

Use #' comment for prose

#+ for chunk header

.Rmd



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
---
```

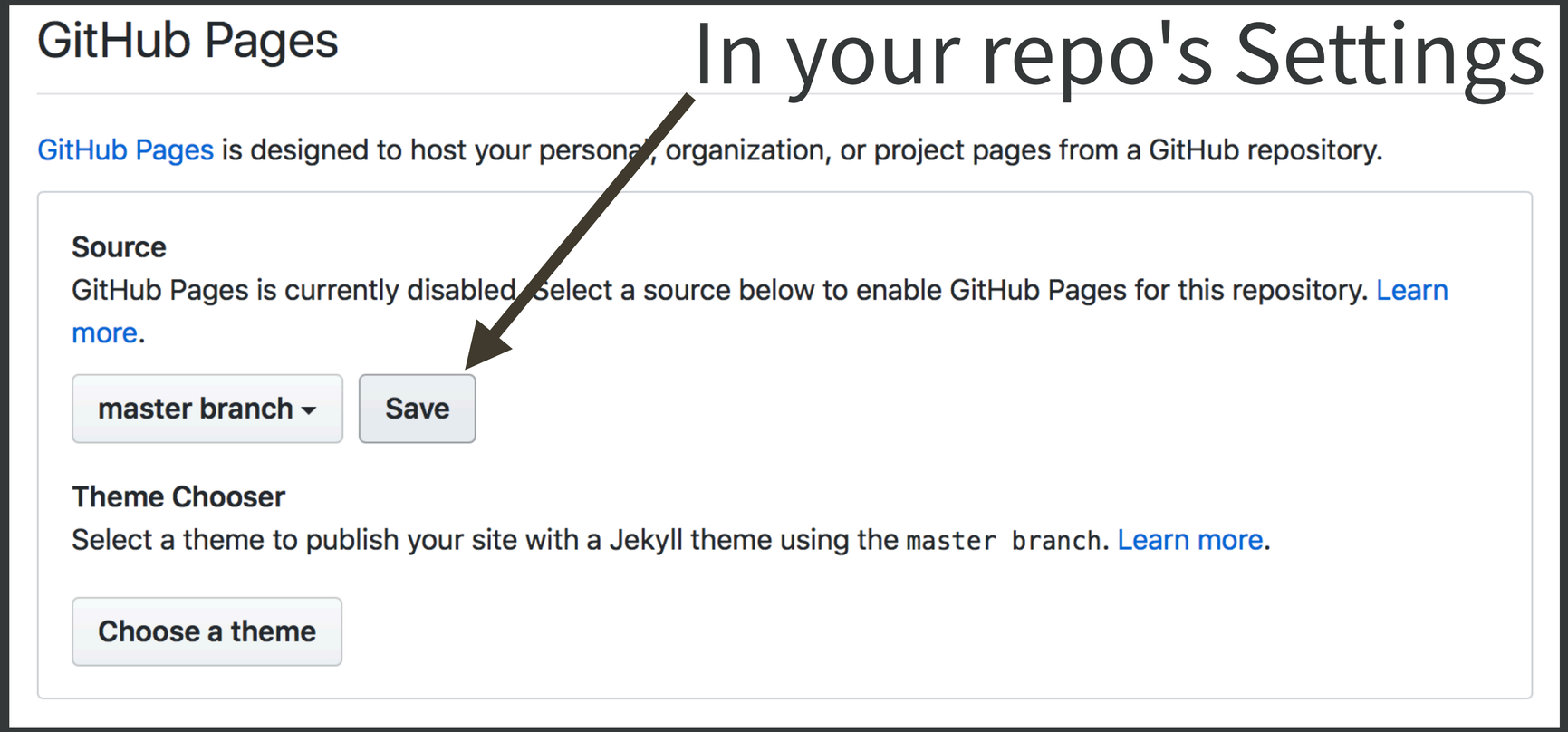
foo.Rmd → foo.md

```
---
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

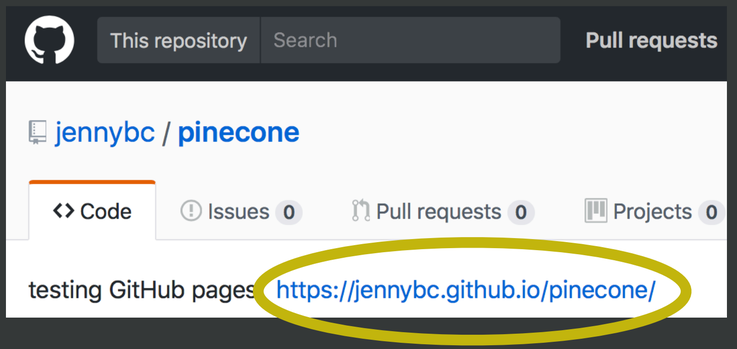
Hot tips for simple GitHub Pages

`README.md` becomes `index.html`, by default

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

👍 `[foo](foo)`, `[foo](foo.md)`, `[foo](foo.html)`

Record your site URL as your repo's website



wrap up here

when we return to Git/GitHub, we'll wrap up loose ends:

- branches and remotes
- useful daily workflows