# Using Git, Part 3: Collaboration
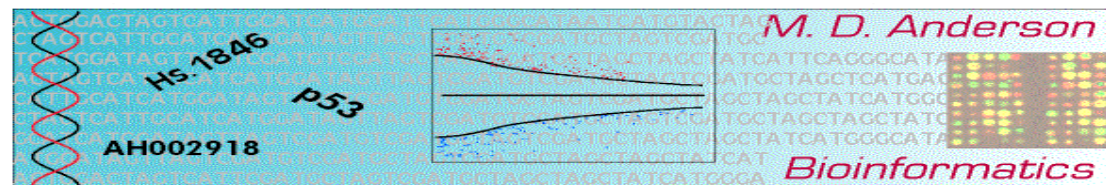
Keith A. Baggerly

Bioinformatics and Computational Biology

UT M. D. Anderson Cancer Center

`kabagg@mdanderson.org`

SISBID, July 18, 2017

# Sharing vs Collaborating

A static webpage anyone can download from is sharing.

If multiple people interact on a final project, that's collaborating.
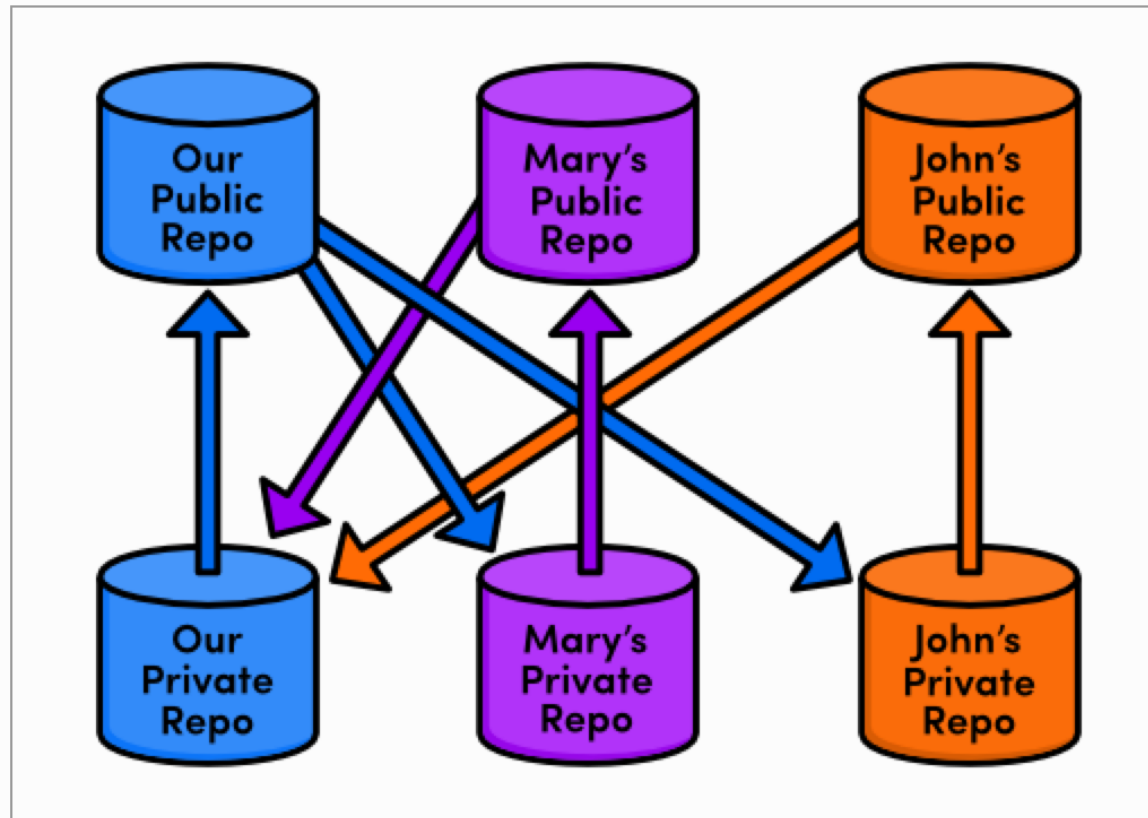
Your number of collaborators need not be large.

I often work with just one or two analysts on assembling reports, where after a few iterations, we're done.

Here, we can specify a few known collaborators who can all push to and pull from the repo (e.g., Karl and I are collaborators on the Module 3 repo).

It's possible to give or get feedback from others

# Suggestions from Others



*The integrator workflow with many developers*

more from Ry's Git Tutorial, Distributed: Everyone can share!
Pull requests

# How Other Changes Work

Let's say you're looking at someone else's repo because it relates to your work, and you notice a bug. If you see a fix, you can suggest this as follows.

fork the repo you're looking at. This creates a copy of the repo in your GitHub account which you own and can edit.

clone your repo copy from GitHub to your local machine.

Make your edits.

Push your edited version of the repo back to GitHub.

Send the original owner a pull request, asking them to pull the edits you've made into their repo.

# Fork and Clone

Clicking the + sign next to the repo name will fork it for you.
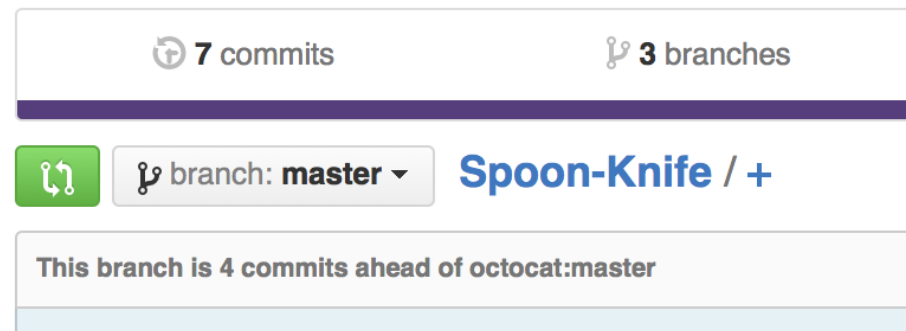


Now, within Rstudio, go to
File/New Project/Version Control/Project Setup

and give it the git url.

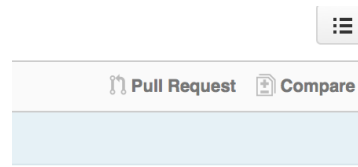This will set up a local project repo for you (with a .Rproj file)

# Edit and Push

After making your edits, use the Rstudio commit popup to ignore the Rproj file and commit your desired changes.
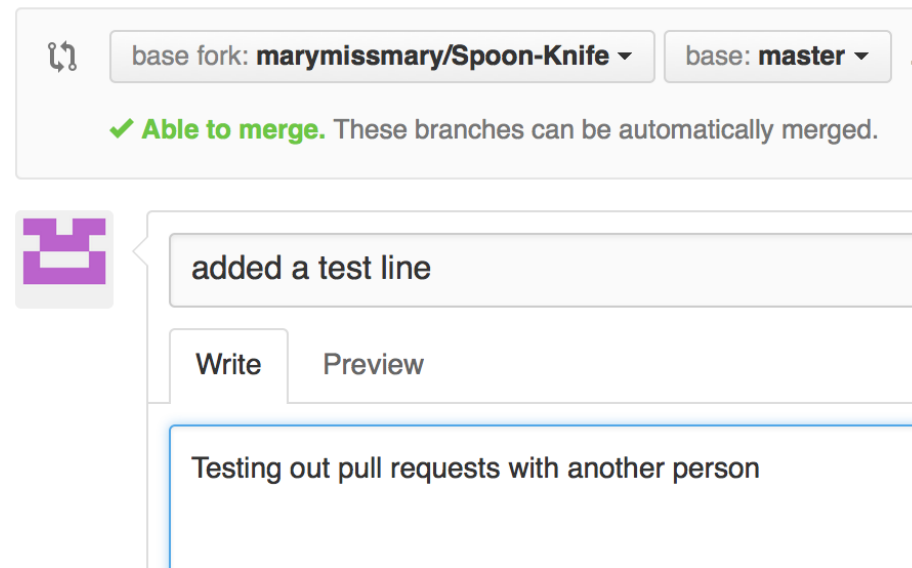
Then push the changes over to GitHub
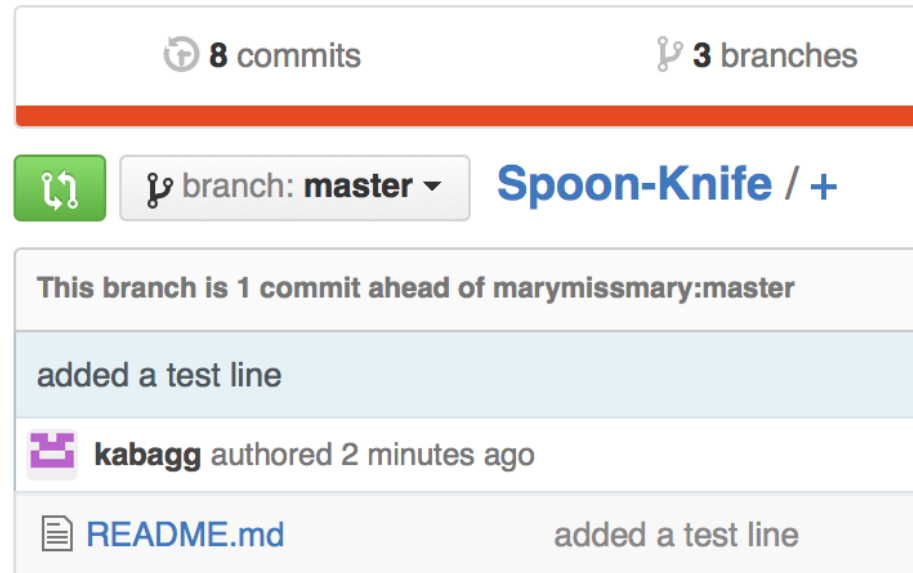
# Submit the Pull Request

Assemble the pull request (PR), along with a brief comment about what you're trying to do.

# You Get Some GitHub Feedback

Even before you submit the PR, GitHub will indicate whether including the suggested changes is easy.
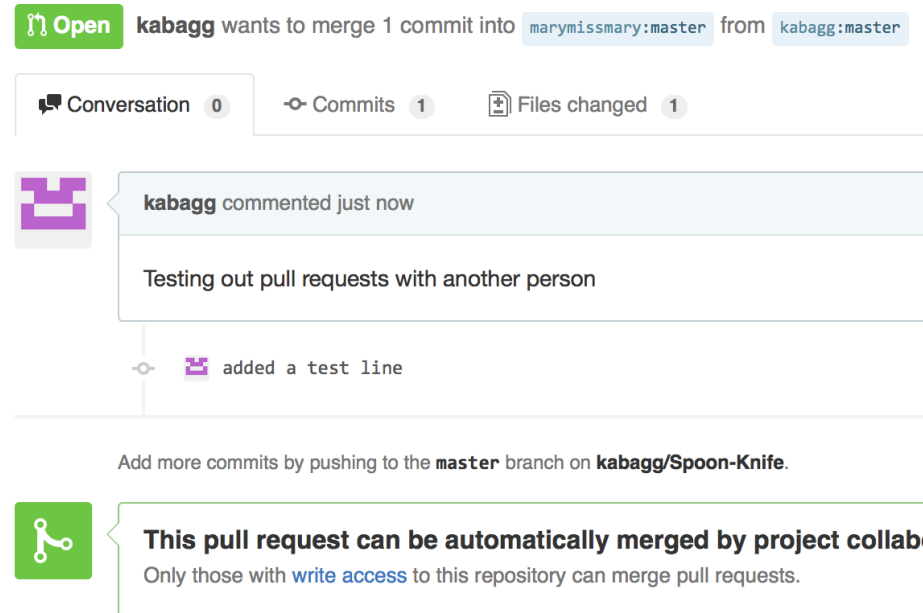
# Welcome to the Conversation

Submitting the PR starts a "conversation" with the owner (initially by email)

# Welcome to the Conversation (2)

The owner can choose to include your suggestion (or not)

# Including Suggestions from Others

Of course, they can comment on your code as well ;).

# pull = fetch + merge

When you pull down data from a remote repo, you're actually doing two things:
(1) fetching the data (as a new branch), and
(2) trying to merge that branch with what you've got in place.

Sometimes you may want to split these steps apart, so you can "fetch" the data and edit it before trying to merge it. This may let you avoid merge conflicts, or simply reduce their number.

# Sending Yourself a Pull Request

Not all pull requests need to be to others; you can send them to yourself as well.

Why?

If you're actively interacting with others, this may be a way of getting input from them before making a change

If, in the request, you mention them by name with TheirGitHubID, they'll be specifically pinged (don't do this to folks you don't know)

# Sometimes I have Issues

Sometimes I notice a problem with (or a potential improvement to) a package I or someone else has developed, but I don't know how to fix it (yet).

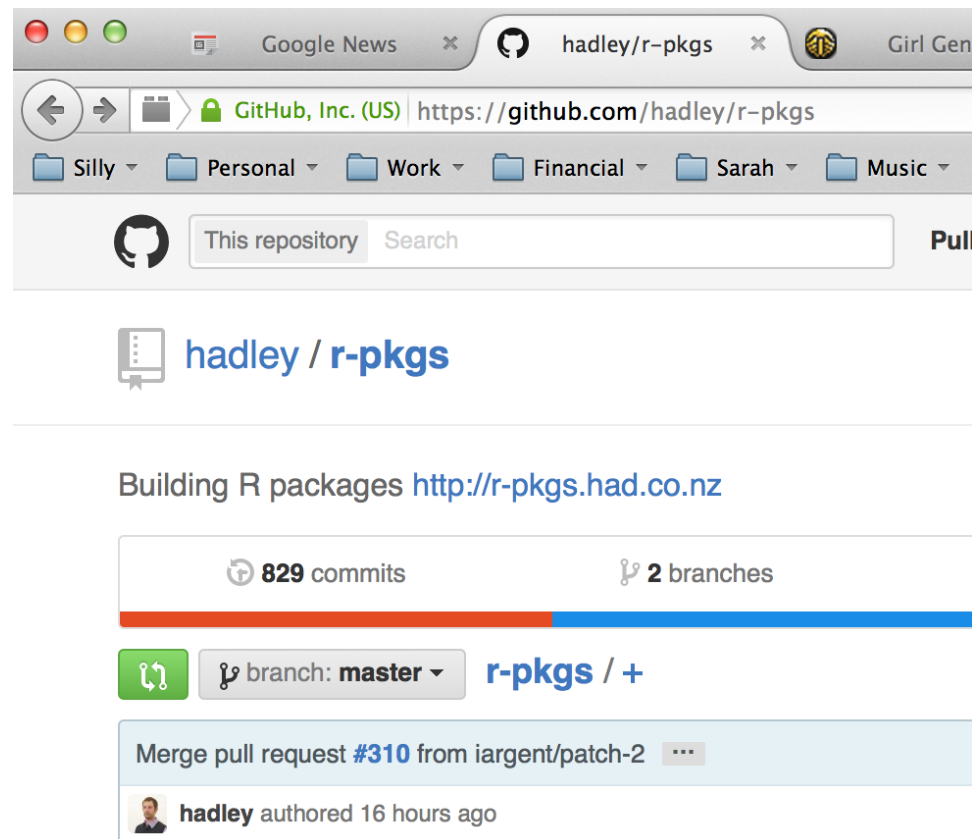In this case, I can raise it as an Issue with the repository

Not all issues are for problems! You can use issues as part of a "to do" list for the next release of your package

Issues get assigned numbers.

If a new commit to the main branch mentions "Fixes" IssueNumber, the issue is "closed" with a link to that commit.

# Others' Files on GitHub

While we're here, let's look around. Exploring others' repos is a great way to learn. Here're some snapshots from Hadley's *R Packages book* repo

# An Introduction

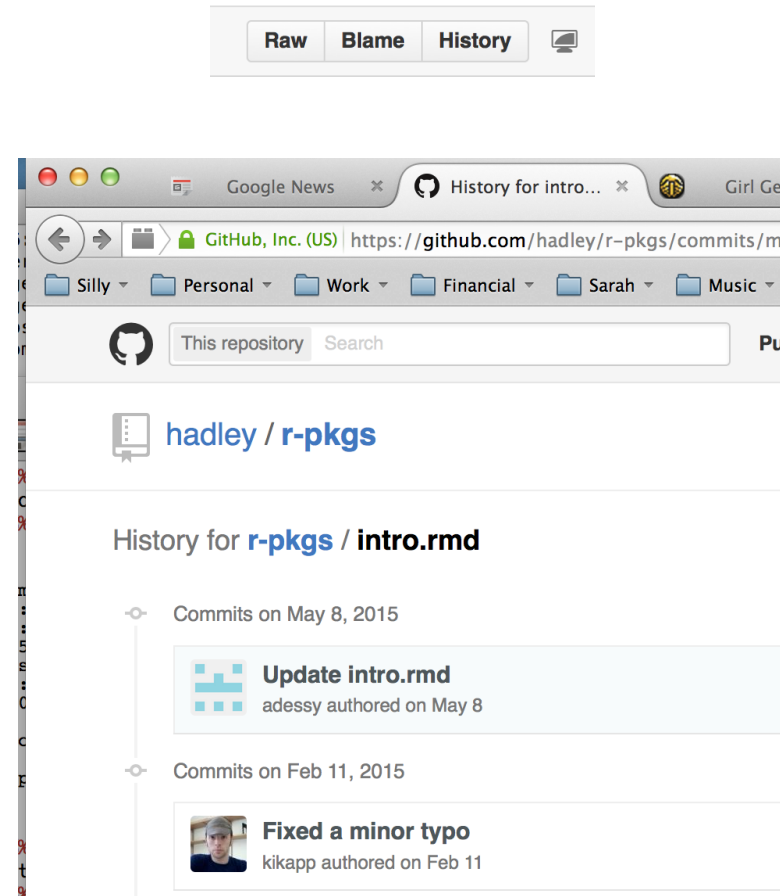Zooming in on specific files shows the contents

# Checking the History

Checking the history shows when edits were made

# Checking an Edit

Checking an edit shows the specific changes

# Improving Signal to Noise

Find a few folks who you think do good work, and explore their repositories to see what they do (and possibly why they do it).

Hadley Wickham

Yihui Xie

Jenny Bryan

RStudio folks

Jeff Leek

# Learning More

Git will require practice, and more examples than we've provided here. Fortunately, there are many good examples on the web.

Scott Chacon's Pro Git, also in book form

Ry's Git Tutorial, also in e-book form

Jon Loeliger, Version Control with Git, 2e (Amazon)

GitHub Help

# How We Can Collaborate with GitHub

So, what can you apply this to?

R packages

analyses for papers

grant applications (e.g., references, multiple drafts)

homework submissions (precise timestamps!)

editing/commenting on reports

and lots of other stuff!