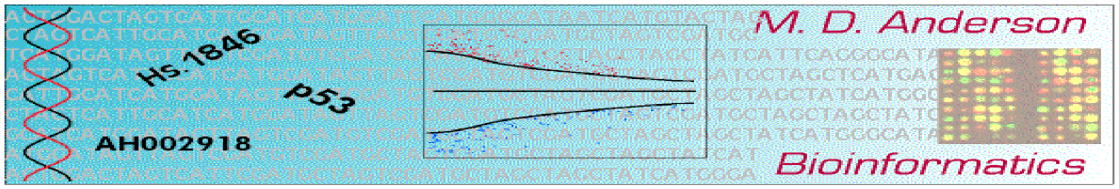
Git/GitHub 3: Collaboration

Keith A. Baggerly
Bioinformatics and Computational Biology
UT M. D. Anderson Cancer Center
`kabagg@mdanderson.org`

SISBID, July 17, 2018



Learning from Others

GitHub is wonderful for browsing.

You can browse repos assembled by people who've been doing this for a while and see what they do.
(e.g., Hadley Wickham, Yihui Xie, Jeff Leek, Jenny Bryan)

Many people now post the development versions of their R packages to GitHub before they appear on CRAN, and

devtools::install_github()

is designed to install such packages directly.

Interacting with Others

You can work with others on GitHub in a few different ways.

The most direct way is if you create a repo and invite someone else to join you (under Settings/Collaborators).

I invited Karl to collaborate on the course github page, `https://github.com/kabagg/sisbid_2018_rr`

As a collaborator, Karl can push files directly to the repo (he has write access).

Because I'm working with someone else on this project, every time I get ready to add something I pull the latest repo version, so I'll have any changes he's made in place.

Interacting Without Write Access

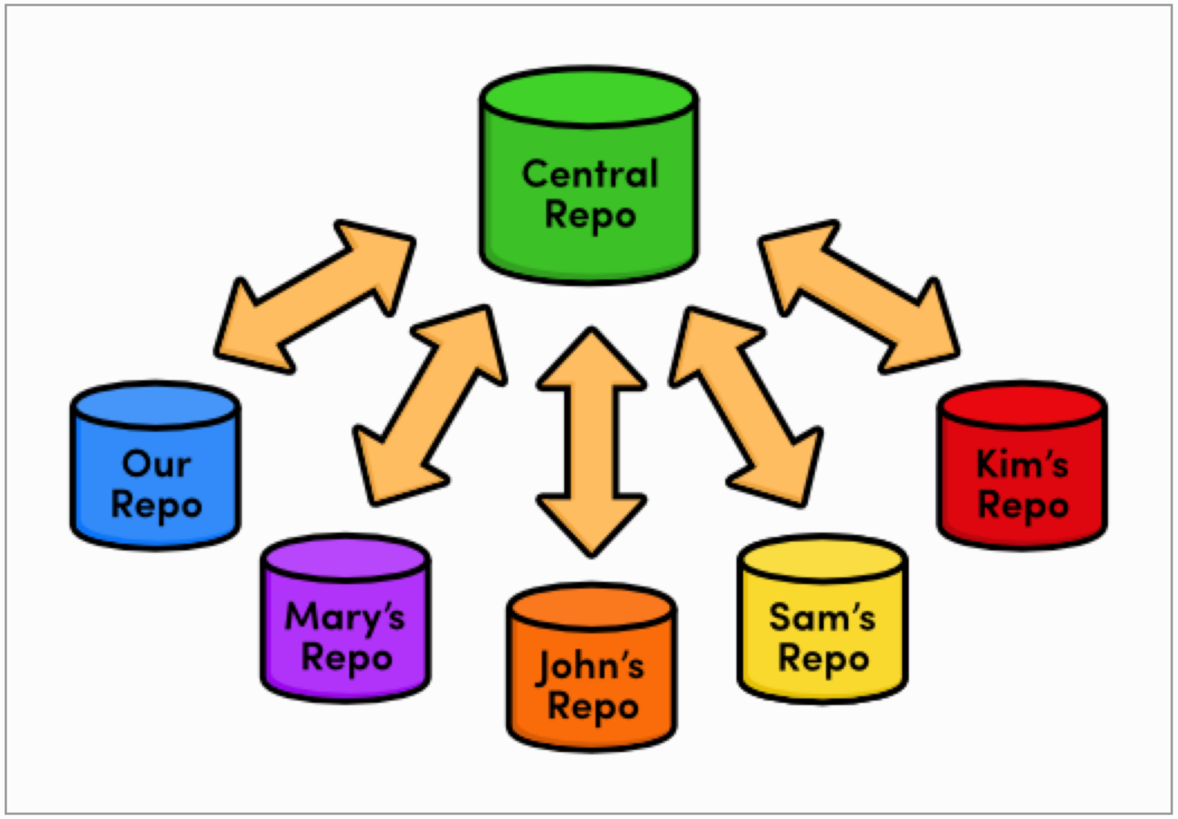
Occasionally, however, you may have an idea about how to improve a repo belonging to someone you’re not collaborating with.

You can suggest your fix using a process very similar to what we saw when we pushed a non-master branch up to GitHub.

You can send them a pull request.

Here’s how it works.

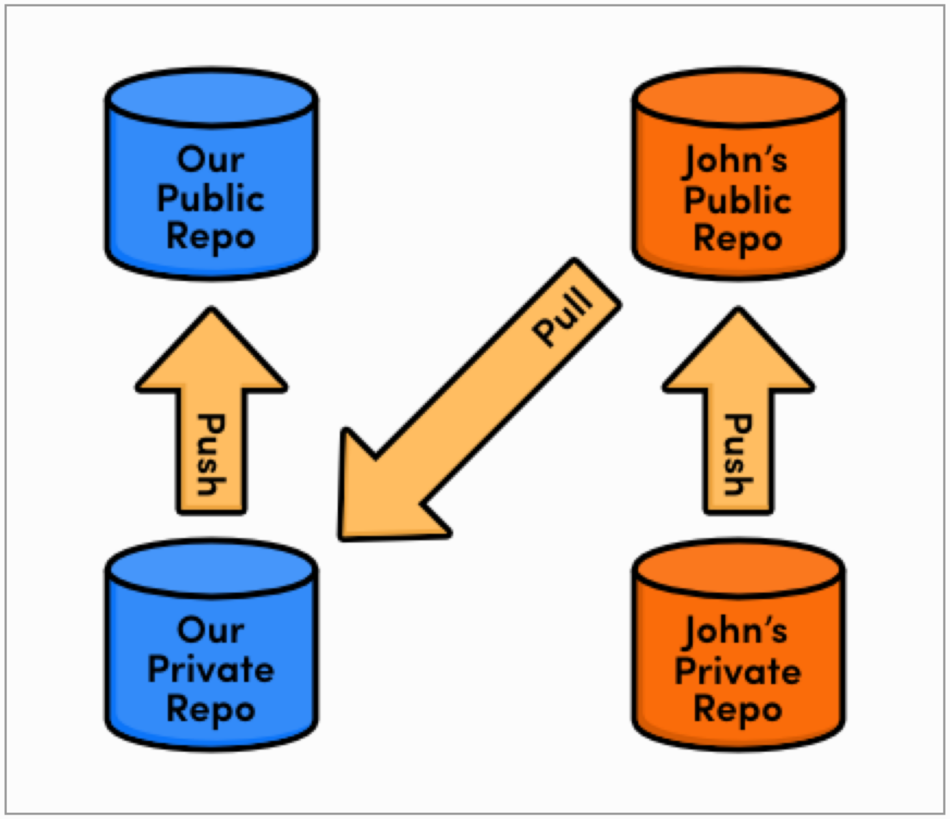
Central (bare) repos



The centralized workflow with many developers

from Ry's Git Tutorial, Centralized Workflows
Bare repos: good FINDABLE, SHAREABLE archives.

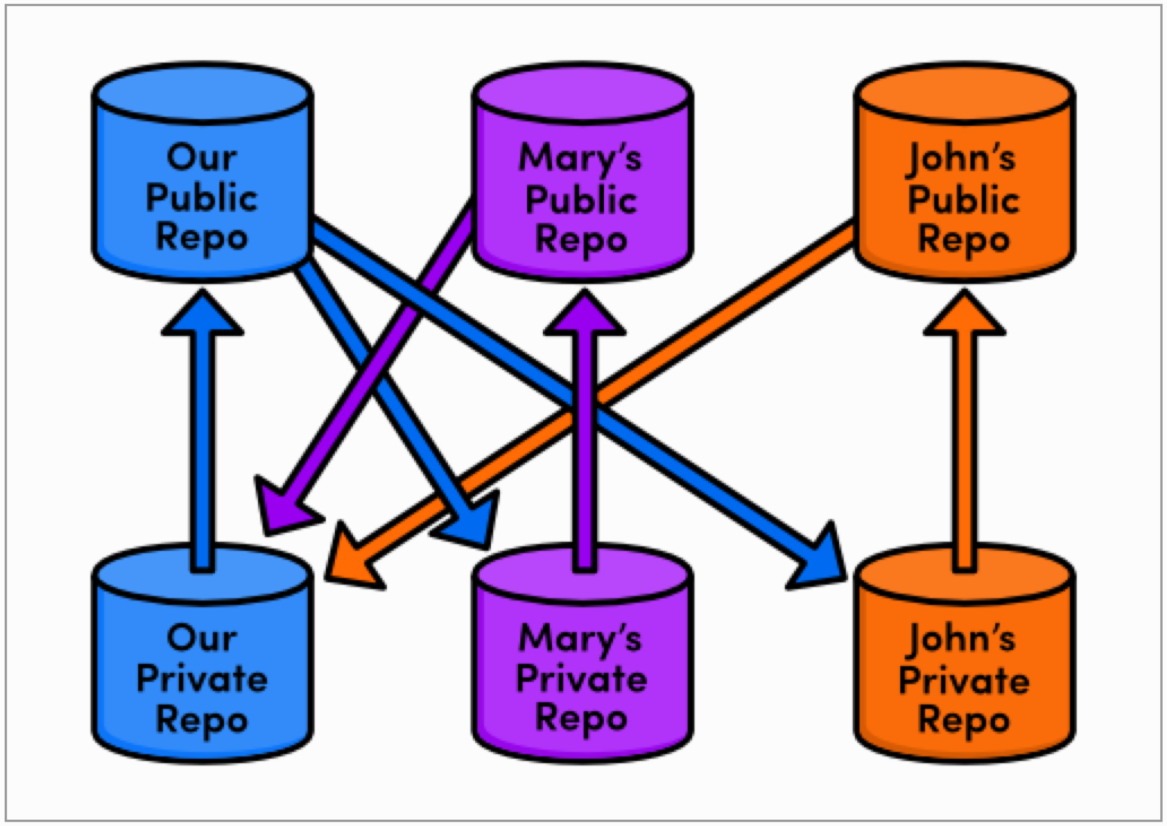
Sharing with Others



The integrator workflow

from Ry's Git Tutorial, Distributed:
Fork, Clone, Edit, Push, Wave, Fetch or Pull

Suggestions from Others



The integrator workflow with many developers

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

Fork Their Repo

You don’t have write access to their repo on their account, but you can read it and copy it.

In the upper right hand corner of the page for their repo is a button labeled “Fork”.

Forking the repo creates a copy of it in your GitHub account, and you have write permission to this copy.

Once you’ve forked their repo, clone a copy to your machine using the approach we’ve followed so far.

Set up a new R Project,
link the project to your GitHub page,
and pull.

Branch, Edit, Push

Per Hadley’s R Packages book section on git

“When working on a forked repo, I recommend that you don’t work on the master branch. Because you’re not really working on the main line of development for that repo, using your master branch makes things confusing.”

So, we create a new branch like “dev-fixit”, make our changes, and push our dev branch to GitHub.

If you initiate a pull request now, it will get sent to the owner of the initial repo.

Only those with write access to the initial repository can merge pull requests.

The Discussion May be Longer

If you’re just pointing out basic things like typos, they may simply accept the changes with thanks.

If you’re suggesting something more substantive, the original author may ask you to follow some style guidelines and help them check that your suggestion works.

You can make edits and push new changes to your GitHub development branch; as long as the pull request is open, changes are pulled in automatically.

If Changes Come Thick and Fast

If the repo you’re viewing is being updated frequently, you may need to check that things haven’t changed between the time you forked the repo and the time you’re submitting changes.

We won’t go into this here, we’ll simply note that it can be done.

Hadley’s book has more details.

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

Other Things To Discuss if Time (1)

squashing commits

Backtracking, working with earlier commits, and undoing changes

git reset

git revert

git checkout SHA

git branch

git tag

Other Things To Discuss if Time (2)

contributing and guidelines

GitHub pages

Licenses

Is it useful to have a file like “scratchpad.txt” in .gitignore?

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.