Getting Started with GitHub in R

Dr Caitie Kuempel Post-doc, University of Queensland
c.kuempel@uq.edu.au Special thanks to Dr. Julie Stewart
Lowndes and Dr. Alexa Fredston for sharing materials

Quick Intro

- Who am I?
- Disclaimer
  - I have been using github for about 1.5 years
  - I am a conservation scientist, not a computer scientist
  - I am self/community taught thanks to the amazing R community
- Please comment in the chat window with any questions or other helpful tips/tricks/resources

What is Git?

- Git is a distributed version control system that keeps track of changes to files and projects over time

version control system that tracks changes to your work

online platform to organize/store your work + collaborate

Git — The version control tool that GitHub is built on top of.

GitHub — Our company and the name of our software. We build software and websites to help you interact with Git repositories in a nice way.

GitHub.com — The website you log into to view repositories online.

GitHub Desktop — An application that you can install on your computer to

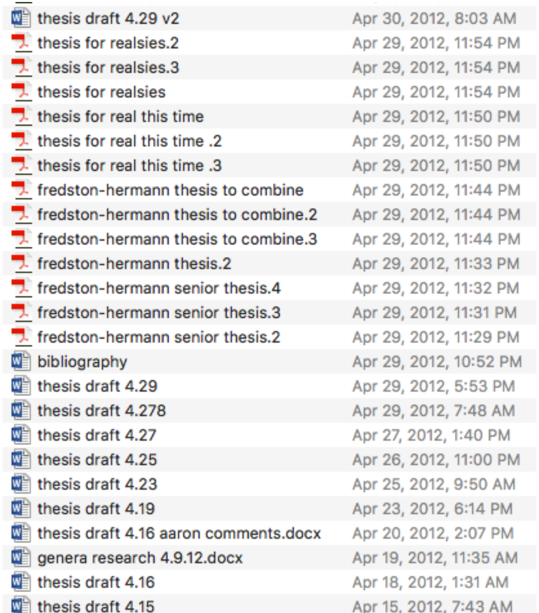
What is GitHub?

- **Github** is a website that hosts Git repositories online – making it easier to share code and collaborate

Similar to Dropbox, you have certain folders on your local computer that will be 'watched', with any changes able to be synced online. However, with GitHub, you have more control about what is synced, and how often. You can store, share, track changes and collaboratively edit many filetypes (including this presentation!) using any application. There are many other great features, including a to-do list you can share with collaborators (called Issues).

Why learn and use GitHub?

- Does this look familiar?



thesis draft 4.29 v2	Apr 30, 2012, 8:03 AM
thesis for realsies.2	Apr 29, 2012, 11:54 PM
thesis for realsies.3	Apr 29, 2012, 11:54 PM
thesis for realsies	Apr 29, 2012, 11:54 PM
thesis for real this time	Apr 29, 2012, 11:50 PM
thesis for real this time .2	Apr 29, 2012, 11:50 PM
thesis for real this time .3	Apr 29, 2012, 11:50 PM
fredston-hermann thesis to combine	Apr 29, 2012, 11:44 PM
fredston-hermann thesis to combine.2	Apr 29, 2012, 11:44 PM
fredston-hermann thesis to combine.3	Apr 29, 2012, 11:44 PM
fredston-hermann thesis.2	Apr 29, 2012, 11:33 PM
fredston-hermann senior thesis.4	Apr 29, 2012, 11:32 PM
fredston-hermann senior thesis.3	Apr 29, 2012, 11:31 PM
fredston-hermann senior thesis.2	Apr 29, 2012, 11:29 PM
bibliography	Apr 29, 2012, 10:52 PM
thesis draft 4.29	Apr 29, 2012, 5:53 PM
thesis draft 4.278	Apr 29, 2012, 7:48 AM
thesis draft 4.27	Apr 27, 2012, 1:40 PM
thesis draft 4.25	Apr 26, 2012, 11:00 PM
thesis draft 4.23	Apr 25, 2012, 9:50 AM
thesis draft 4.19	Apr 23, 2012, 6:14 PM
thesis draft 4.16 aaron comments.docx	Apr 20, 2012, 2:07 PM
genera research 4.9.12.docx	Apr 19, 2012, 11:35 AM
thesis draft 4.16	Apr 18, 2012, 1:31 AM
thesis draft 4.15	Apr 15, 2012, 7:43 AM

Why learn and use Github?

- **Version control** - Track all of your changes clearly and unambiguously, without proliferating files or losing old versions
- Used for **many programming languages**, although we'll only cover its interface with RStudio.
- **Facilitates collaboration** because multiple people can work on code together
- Great **organizational tool** for individual projects too.
- **Back up code**
- **Easily switch between computers**
- **Undo mistakes**
- **'comment' on changes**
- **Project management**
- And much **more!**

What you'll need for this tutorial

You don't need to be proficient in R to understand this workshop!

- You will need:
  - An updated version of RStudio on your laptop
  - An account on github.com
- Please ask questions in the chat box as we go
- You can read more about why and how to use GitHub from:
  - Hadley Wickham
  - Karl Broman
  - Ben Best

Workshop Outline

1. Github Vocabulary
2. GitHub Structure
3. GitHub Workflow
4. RStudio Practice
5. Non-R Options
6. Best Practices
7. Resources

Github Vocabulary

- **Repository (‘repo’)** - the most basic element of GitHub. Imagine as a project’s folder which contains all of the project files (including documentation), and stores each file’s revision history. Repositories can have multiple collaborators and can be either public or private.
- **Commit** - or is an individual change to a file (or set of files). Github keeps a record of the specific changes commited along with who made them and when.
- **Push** - to push means to send your committed changes to a remote repository on GitHub.com. For instance, if you change something locally, you can push those changes so that others may access them.

Github Vocabulary

- **Pull** - refers to when you are fetching in changes and merging them. For instance, if someone has edited the remote file you're both working on, you'll want to pull in those changes to your local copy so that it's up to date.
- **Issue** - are suggested improvements, tasks or questions related to the repository. They can be created by anyone (for public repositories), and are moderated by repository collaborators.
- **Clone** - is a copy of a repository that lives on your computer instead of on a website's server somewhere. When you make a clone, you can edit the files in your preferred editor and use Git to keep track of your changes without having to be online. The clone is still connected to the remote version so that you can push your local changes to keep them synced when you're online.

Github Vocabulary

- **Fork** - is a personal copy of another user's repository that lives on your account. Forks allow you to freely make changes to a project without affecting the original upstream repository. You can also open a pull request in the upstream repository and keep your fork synced with the latest changes since both repositories are still connected.
- **Branch** - is a parallel version of a repository. It is contained within the repository, but does not affect the primary or master branch allowing you to work freely without disrupting the "live" version. When you've made the changes you want to make, you can merge your branch back into the master branch to publish your changes.

You can find more github vocabulary here

GitHub Structure

Files are stored in **repositories**, owned by **users** / **organizations**.

Same structure across all orgs/repos: familiar, easy to navigate.

Here are some examples to explore later on.

- **repositories**: github-intro, dplyr, ggplot2
- **users**: jules32, hadley, jennybc
- **organizations**: twitter, netflix, rstudio, nceas, ohi-science

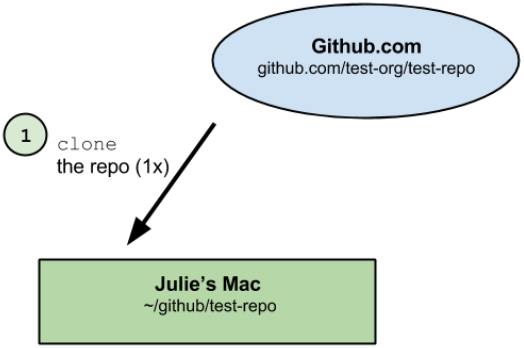
GitHub Workflow and Vocabulary

Let's discuss how this works first, and then we're going to practice with RStudio.

You come across code for an R package that will randomly download 10 dog pictures from the internet given the name of a dog breed. Obviously, you need this on your computer, so you're going to **clone** it.

GitHub Workflow and Vocabulary

- **clone**: download an identical copy - a ‘clone’ - of a repository to your local computer. Unlike most downloads, cloned repositories can still be synced with the online version(s).



GitHub Workflow and Vocabulary

Now that you have a local copy of this package, you edit the code to only download pictures of puppies. When you're at a good stopping point and want to save your progress, you 'commit' the code with an informative message describing the changes for your future self or anyone else who sees this code.

I think of commit as the GitHub version of 'save', but you also get to add a comment.

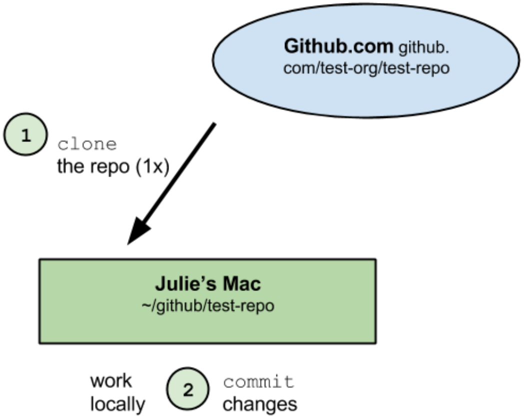
GitHub Workflow and Vocabulary

- **commit**: message associated with your changes (best practices)

1

GitHub Workflow and Vocabulary

- **commit**: message associated with your changes (best practices)



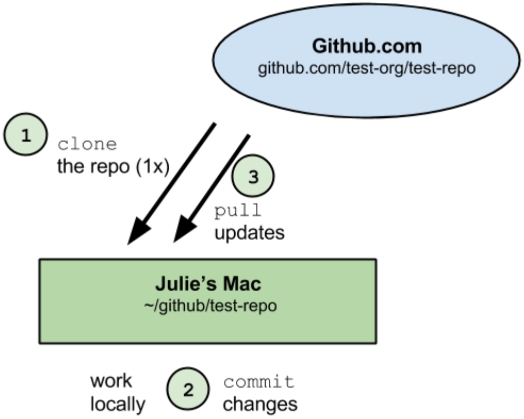
GitHub Workflow and Vocabulary

But wait! Before you share your newly updated R package with the world, you want to be sure the person who created it hasn't made any changes to the original (bug fixes, for example). To download new changes to the code from others, you 'pull' from GitHub. **This is also how you can switch computers and easily work on the same code.**

If there are differences that can be easily resolved (e.g., the R package developer fixed a bug by adding a line of code), your repository will be 'merged' accordingly. If the differences *conflict*, you have to resolve them manually by telling GitHub which changes to keep and which to ignore.

Github Workflow and Vocabulary

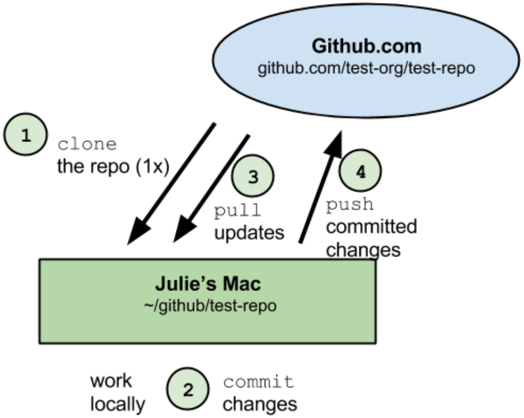
- **pull**: sync a repo on your computer with the online version. Do this frequently to avoid **merge conflicts** or working on outdated code.



GitHub Workflow and Vocabulary

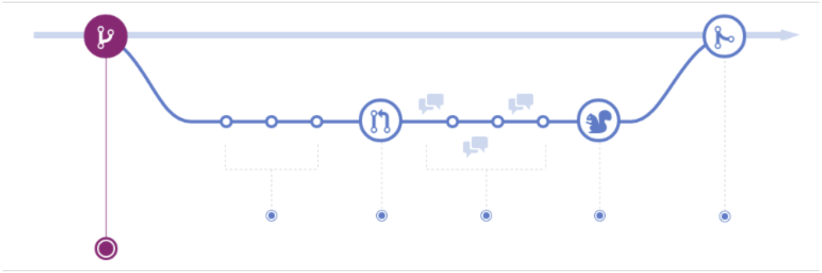
After committing your changes and pulling the repository again, you're ready to share your puppy image search code from your local computer to GitHub. To do this, you upload - or 'push' - the code.

- **push**: sync the online repo with your version, only possible after committing



GitHub Workflow: branches

GitHub's branching approach is very intuitive to some people, and not to others.



guides.github.com/introduction/flow

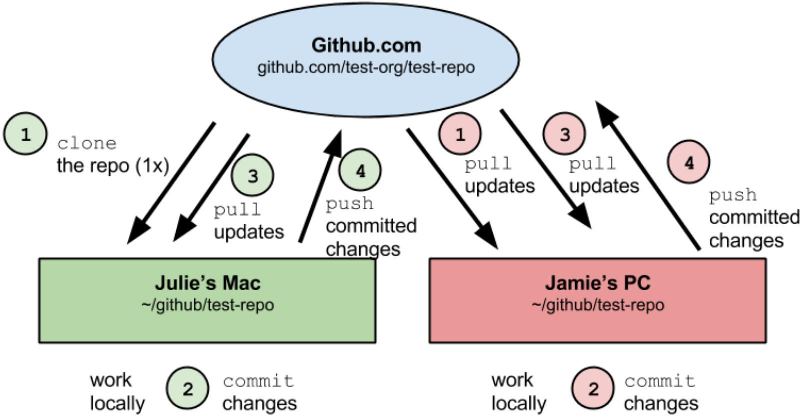
Github Workflow: branches

There are (at least) two ways to work on an existing repository:

- Clone it and then push and pull directly from the original (‘master’) - suitable for small group collaborations.
- Create a ‘fork’ in the branch so that you can make any changes to all of the repository contents without affecting the other branches, such as the master branch. This is suitable for when you just want to copy code, or for very large collaborations.

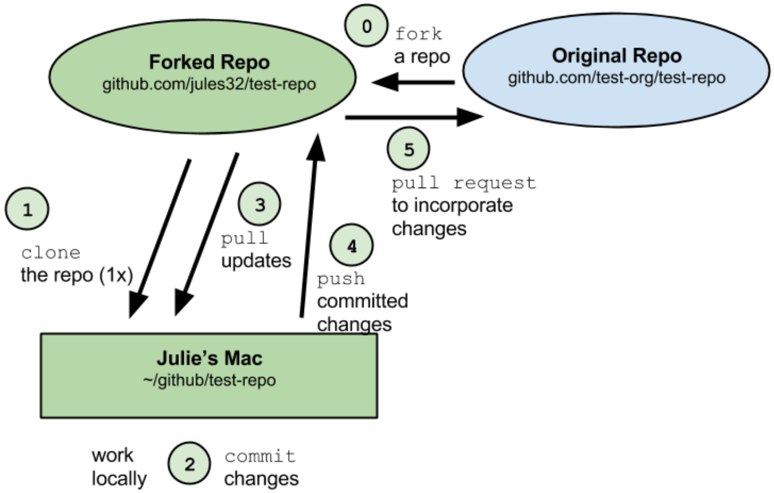
GitHub Workflow: sync with branch

sync ~ pull + commit + pull + push
All collaborators work independently but sync regularly



GitHub Workflow: fork and pull

fork + pull + commit + push + pull request



Github Workflow and Vocabulary: Summary

1. **fork a repo** to your user account
2. **clone the repo** to your computer
3. **edit a file**, inspect differences
4. **commit** changes
5. **pull**
6. **push**
7. repeat!

RStudio Practice

In order to follow the demo, git and RStudio must be installed on your laptop, and you must have a GitHub account. Don't worry, they're all free!

RStudio Practice

Open RStudio and navigate to tools -> Global Options. Go to Git/SVN and click ‘enable version control interface for RStudio projects’. If you have any issues, flag one of us down or check the RStudio help page.

One more thing: To use the GitHub interface in RStudio, you’ll need to work with R projects. R projects are associated with working directories and they help with version control and with switching between projects. more info

RStudio Practice

Try finding this repository, `getting_started_github`, on my github page [@cdkuempel](https://github.com/cdkuempel).

Remember, you can *clone* if you want a local copy on your computer from which you can contribute changes to my project, or *fork* if you want your own branch to mess with.

Click 'fork' on the top right. You will see a copy of this same repository, but rather than cdkuempel/github-intro-2, it's called **yourusername**/getting_started_github, and it shows where it was forked from.

Now, it's yours! To clone this repository so you can edit it in R, click the green 'code or download button', and then the clipboard icon to copy the URL.

RStudio Practice

1. Open RStudio and go to file -> new project
2. Click version control, then git
3. Paste the URL you just cloned into the first box.

The second box is for the name of the project, and the new directory you're creating on your computer. I almost always use the same name as the repository (in this case, `getting_started_github`).

RStudio Practice

4. Tell RStudio where to put this repo.

It is **strongly** recommended that you create a directory in your home directory called ‘github’, and put all your repositories and code there. To do this, put ‘~/github’ in the third box.

5. Finally, click ‘create project’. (If you get an error doing this, there could be many reasons, but the first thing I try is going back to the repository on GitHub, clicking ‘use SSH’ or ‘use HTTPS’ under the green ‘code’ button (whichever you didn’t use last time), and repeating everything on this slide. You might also have to manually create a ‘github’ folder in your home directory.)

RStudio Practice

Congratulations, you just cloned a repository from GitHub! It's now yours to edit in RStudio. Under 'files' in the bottom right window, you'll see everything in this repository, including this presentation. Open practicescript.R, make any edit you like, and save it.

RStudio Practice

You'll notice the top right pane has a tab called 'git'. Click on it, and you'll see all the files in this directory. Check the box next to practicescript.R and click 'commit' above. Enter a message describing your changes in the box on the top right and click 'commit'.

Don't forget to pull before you push! Because no one else should be editing your forked repository, there shouldn't be any changes when you pull. Finally, push your commit, and then go back to GitHub to see the changes in the repository.

RStudio Practice

If you wanted to create a new repository, you would go to the GitHub home page, click ‘new repository’ and follow the prompts, and then use the same process we just covered to clone it and start editing code in RStudio. You can drag and drop files from other locations into the repo directory on your local machine and then push them to GitHub, if they aren’t too big.

Non-R Options: Creating Repositories

On GitHub.com, you can create new repos

On your computer, you can create new repos in several ways:

- **GitHub Desktop**
- **RStudio**
- **shell/command line**

Non-R Options: Syncing

On GitHub.com, you can clone a repo to your computer.

On your computer, you can clone/sync repos in several ways:

- **GitHub Desktop**
- **RStudio**
- **shell/command line**

When you work on your computer, any edits you make to any files in your repo, using any program, will be tracked.

Best Practices

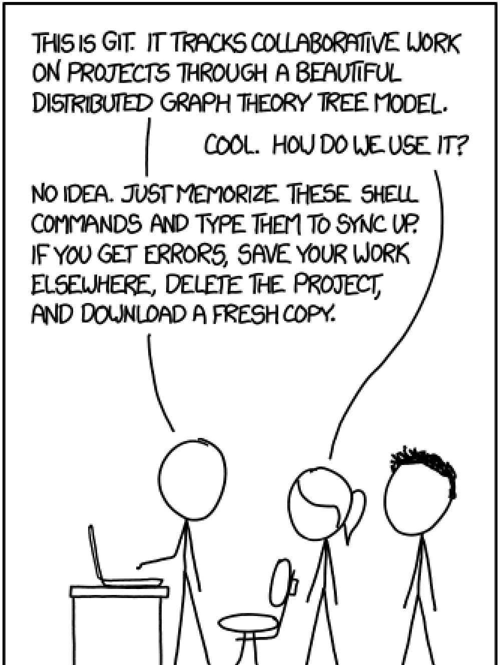
Pull often!

Commit frequently

Be mindful of filepaths

- We work from a in a folder in our home directory called ***'github'*** (all lowercase!), so that everyone can access the repo with the filepath beginning in `~/github`:
- **Windows**: `Users\[User]\Documents\github\`
- **Mac**: `Users/[User]/github/`

Thanks


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.

Resources

Learn more about GitHub:

- **GitHub Guides** by GitHub
- **Git and GitHub** by Hadley Wickham
- **Good Resources for Learning Git and GitHub** by GitHub
- **Learn Git Branching** by Peter Cottle
- **Git/GitHub Guide** by Karl Broman
- **Git & GitHub** by Ben Best

Just Google 'GitHub Tutorial...'

Thanks

- Email me: c.kuempel@uq.edu.au
- Follow me on Twitter: [@cdkuempel](https://twitter.com/cdkuempel?lang=en)
- Tweet #rstats, #github, #RLadies