

# Using Git with GitHub (Part 2)

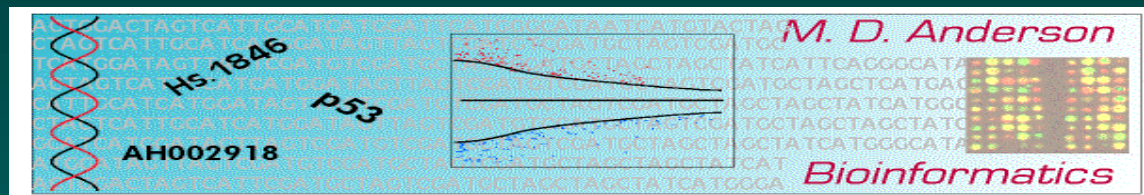
Keith A. Baggerly

Bioinformatics and Computational Biology

UT M. D. Anderson Cancer Center

[kabagg@mdanderson.org](mailto:kabagg@mdanderson.org)

SISBID, July 21, 2015



---

## Version Control and Sharing

In Part 1, we looked at how you could apply git in your own projects, including tracking, branching, and project flow.

But, as we've noted, the real power of version control is how it can help with collaborative efforts; GitHub is an excellent example of this.

Collaboration involves working with “remote” repositories, and requires a few more steps, but the basics are similar.

We'll introduce this in the context of modifying our `toyPackage`.

We'll also look more at how Rstudio's graphical interface highlights some stuff we know how to do.

---

---

## Going to GitHub

Now let's share our package with others.

The easiest way to do this is to post it on GitHub as a public repository.

If there're only a few people you want to see the repository, then you can either pay a small fee to maintain some private repositories, or ask at your institution about whether local repositories are available.

A common variant of this is [GitLab](#).

For now, let's work with GitHub.

---

---

## Registering at GitHub

Registration and a few public repositories can be had for free!

You do need to supply your name, email, and a password.

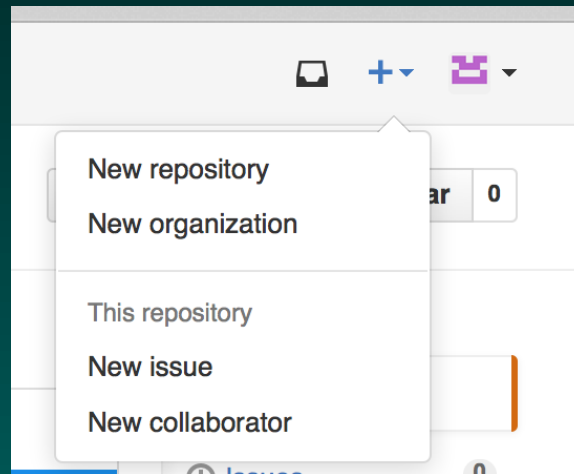
To keep things simple, please use the same values for these as you used when configuring git on your own machine.

---

# Creating a GitHub Repository

Since sharing repos is the main reason for GitHub's existence, they try their darnedest to make this easy.

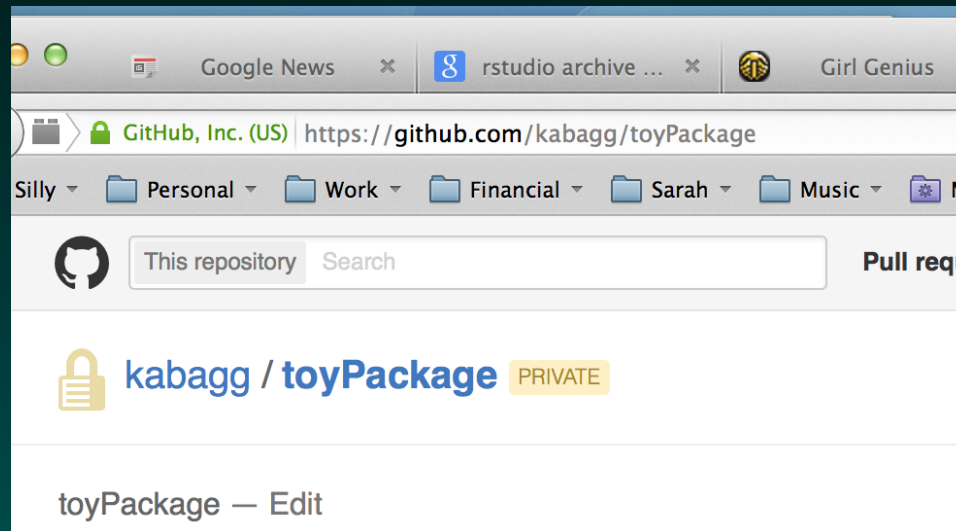
In the upper right of every GitHub page there's a "+" pulldown menu which will let you create a "New Repository".



When you do, it'll ask for a name and a short description; let's use the package name for both (i.e., "toyPackage")

# Congratulations, it's a Repo!

This repo not only exists, it has a web page! The url is `https://github.com/yourUsername/yourPackageName`



Now, GitHub really wants *stuff* in its repos. It'll encourage you to put some files (such as README.md) in your repo right away. We won't, because we're going to fill the repo with material from our local machine using "git push".

---

## “Push” Setup

In order to “push” our package to GitHub, we need to leave Rstudio and go to a shell for a few direct invocations.

Using the “Tools” option from the Rstudio panel will invoke a shell for us, and (if we invoke this with our `toyPackage` open) will shift folder locations to where we want.

Within the shell, we now need to do two things:

- (1) tell Git where we want to put our remote repository, and
- (2) actually push it there.

## Telling Git Where to Go

First, we supply git with location of the remote repo

```
git remote add origin \  
    https://github.com/kabagg/toyPackage.git
```

This will tweak your git configuration (the “config” file in your .git/ folder) by adding info:

```
[remote "origin"]  
    url = https://github.com/kabagg/toyPackage.git  
    fetch = +refs/heads/*:refs/remotes/origin/*
```

This is where git keeps track of such settings.



---

## What if you Get it Wrong?

I ask because I did the first time I tried this ;).

Don't edit the config file directly; that's a bad habit to get into.

Rather, try

```
git remote rm origin
```

This will properly remove (rm) the mistaken entry for origin from the config files, and let you supply the correct one.

---

## Pushing to GitHub

Now we push to GitHub (and get display arrows in Rstudio!)

```
git push -u origin master
```

```
Username for 'https://github.com': kabagg
```

```
Password for 'https://kabagg@github.com':
```

```
Counting objects: 25, done.
```

```
Delta compression using up to 8 threads.
```

```
Compressing objects: 100% (18/18), done.
```

```
Writing objects: 100% (25/25), 3.87 KiB | 0 bytes,
```

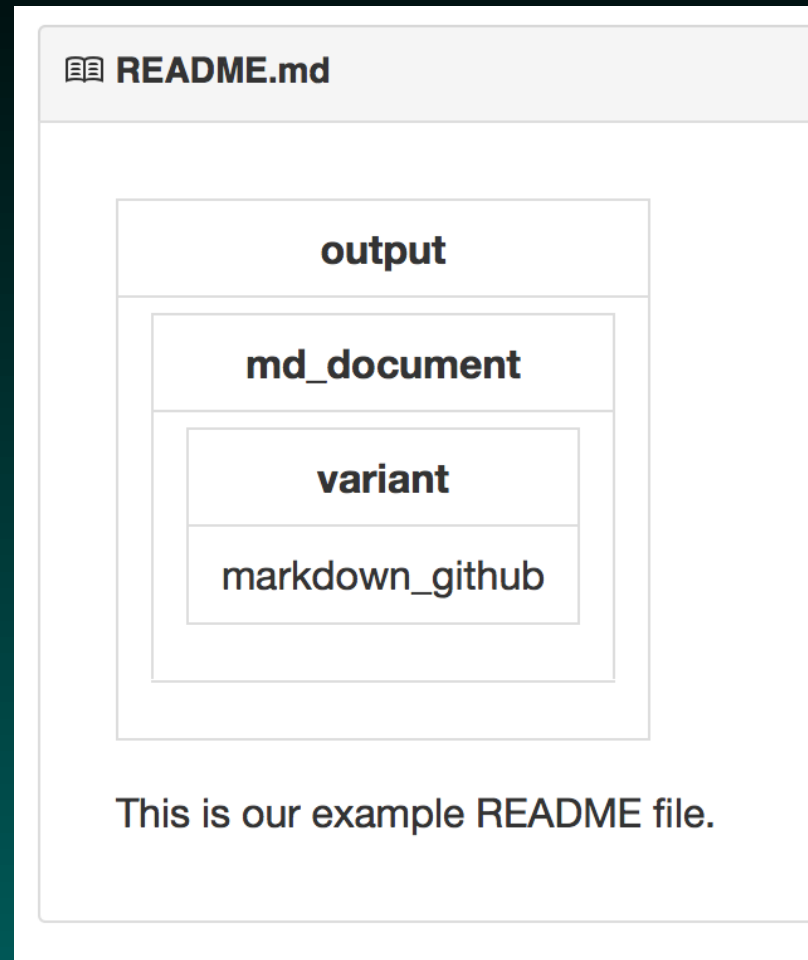
```
Total 25 (delta 2), reused 0 (delta 0)
```

```
To https://github.com/kabagg/toyPackage.git
```

```
 * [new branch]      master -> master
```

```
Branch master set up to track remote branch master
```

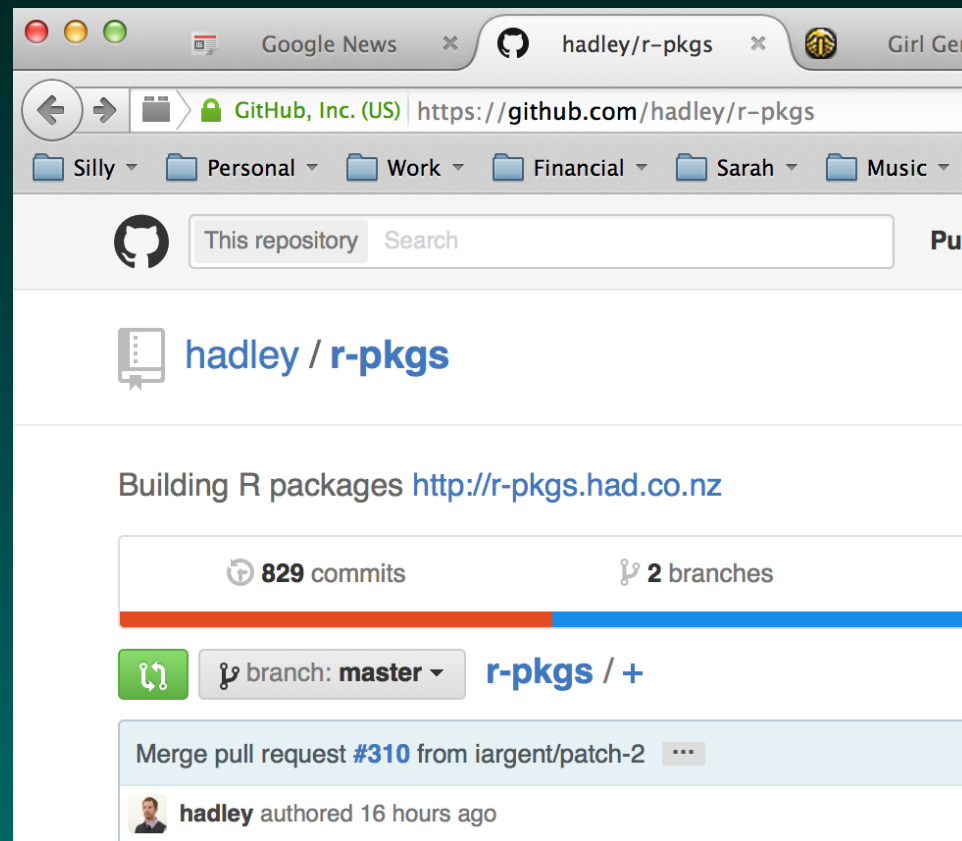
# Looking at our README.md on GitHub



We have a webpage for free!

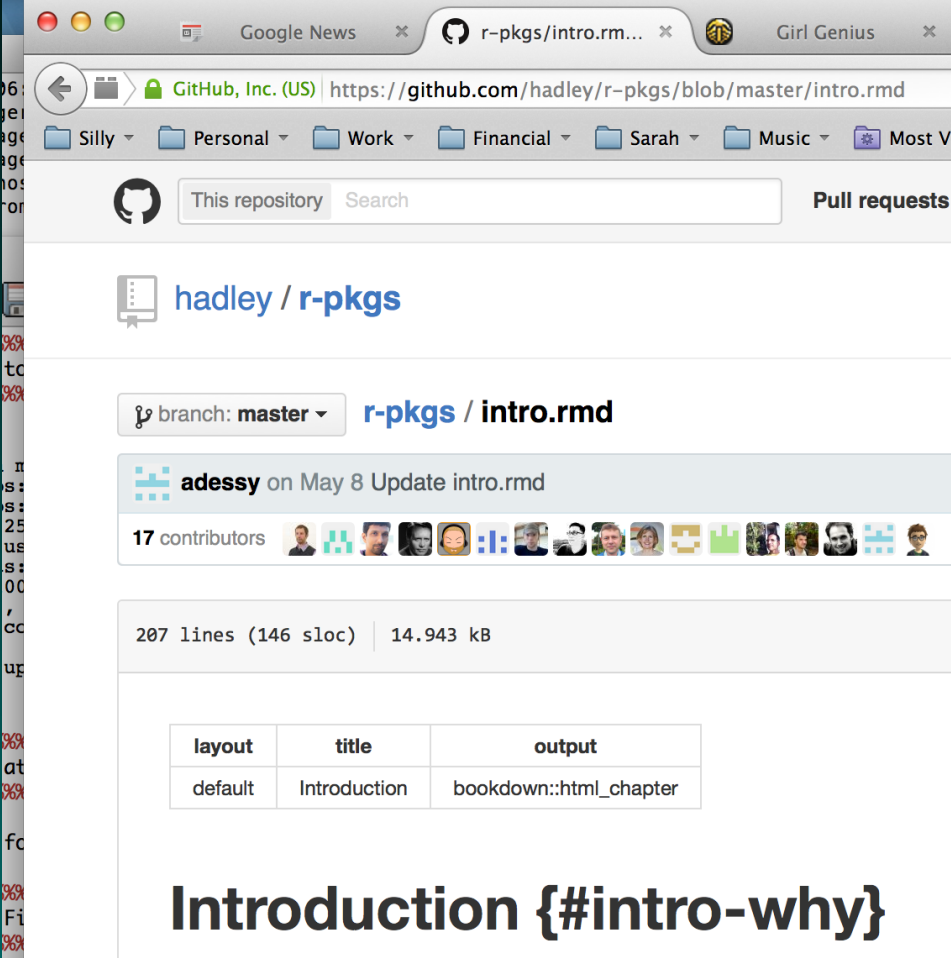
# Others' Files on GitHub

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



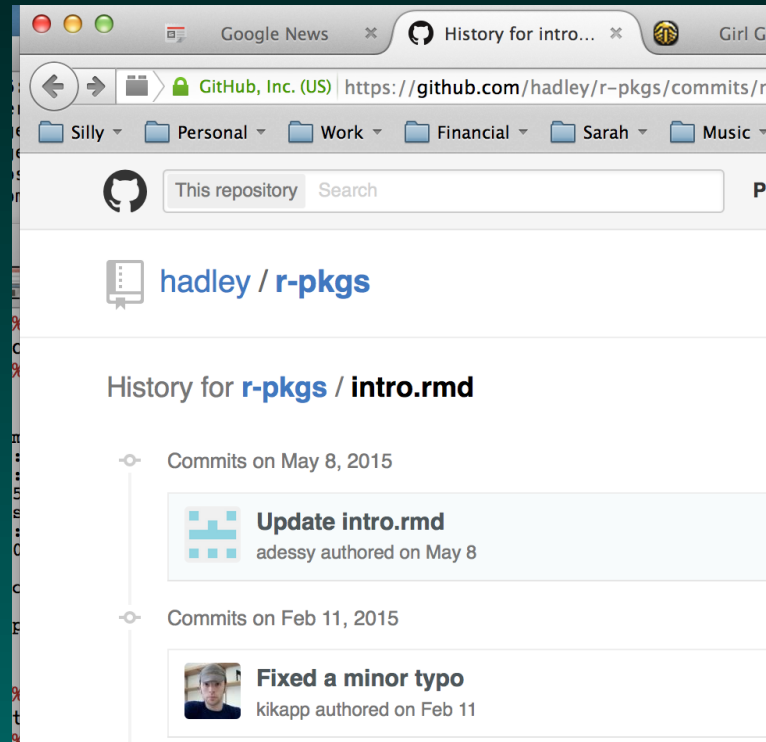
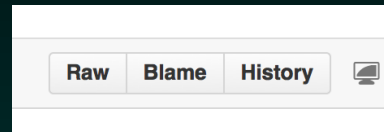
The screenshot shows a web browser window with the GitHub repository page for 'hadley / r-pks'. The URL is <https://github.com/hadley/r-pkgs/blob/master/intro.rmd>. The page displays the file 'intro.rmd' on the 'master' branch. A commit by 'adessy' on May 8 is shown, updating 'intro.rmd'. Below the commit, there are 17 contributors. The file statistics show 207 lines (146 sloc) and 14.943 kB. A table at the bottom of the file preview shows the layout, title, and output for the file.

layout	title	output
default	Introduction	bookdown::html_chapter

Introduction {#intro-why}

# Checking the History


Checking the history shows when edits were made





# Checking an Edit



Checking an edit shows the specific changes



## Fixed a minor typo

 master (#251)

 **kikapp** authored on Feb 11

 Showing **1 changed file** with **1 addition** and **1 deletion**.

2   intro.rmd

		@@ -91,7 +91,7 @@ You'll start by learning about
91	91	The final three chapters describe general best p
92	92	
93	93	* [Git and github](#git): mastering a version co
94		- colloborate with others, and is useful even fo
	94	+ collaborate with others, and is useful even fo
95	95	you to easily undo mistakes. In this chapter y
96	96	popular git and GitHub combo with RStudio.
97	97	
		

## Editing our Toy Package

Now let's go back to our machine and change something.

In particular, let's add another function, "plotSquare.R" (I'll let you guess what this does).

Now we need to  
edit the roxygen function comments  
edit .Rbuildignore  
and then, using devtools,

```
document()  
build()  
install()  
check()
```



## Ready to Commit?

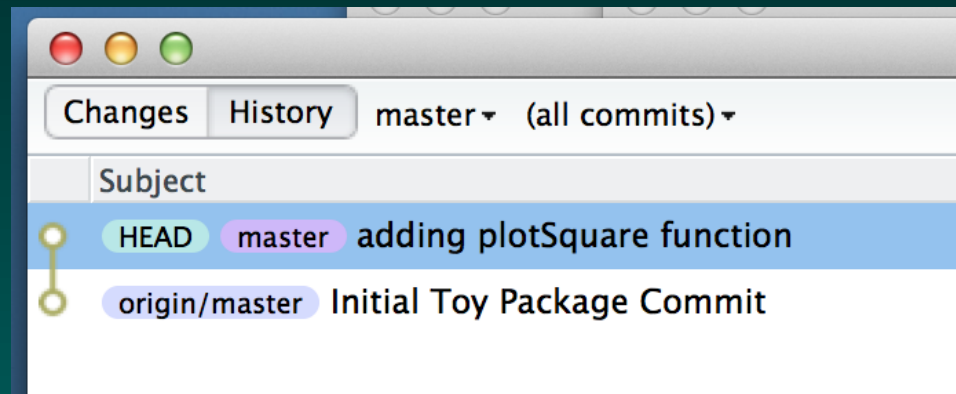
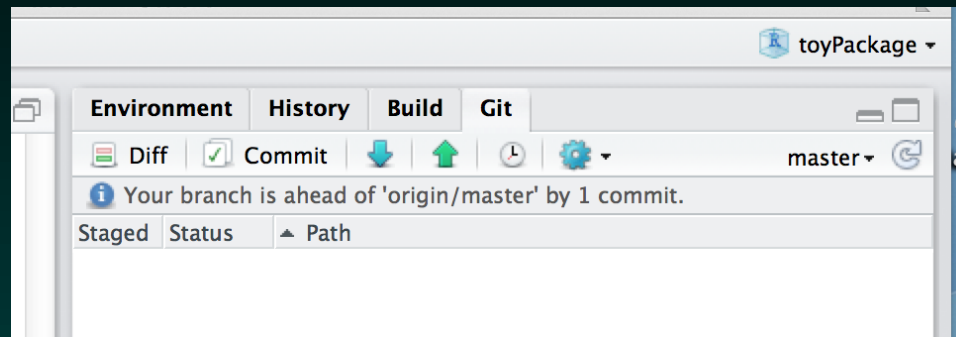
The above edits should change

R/plotSquare.R  
man/plotSquare.Rd  
NAMESPACE  
.Rbuildignore

Once our revised package passes “check”, stage the changed files and commit the changes.

# Where Are We Now?

We're now *ahead* of GitHub



The commit history shows a chain


# Pushing the Changes to GitHub

The git pane “up arrow” will push changes to GitHub


```
Git Push  
To https://github.com/kabagg/toyPackage.git  
86d9855..88831a6  master -> master
```



toyPackage — Edit

2 commits 1 branch

 branch: master ▼ **toyPackage** / +

adding plotSquare function

 **kabagg** authored 7 minutes ago

 R	adding plotSquare function
 data	Initial Toy Package Commit

---

## Working with Someone Else

There's also a *down* arrow (for “pulls”).

If we're doing our edits locally, where are these changes coming from?

*Other people!*

If the changes are from one of your collaborators (someone else with permission to directly push changes to the repository), then simply pulling down the changes will synchronize your repo with GitHub's (there's no problem with pulling the version you already have).

But others can suggest changes as well.

---

## 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.

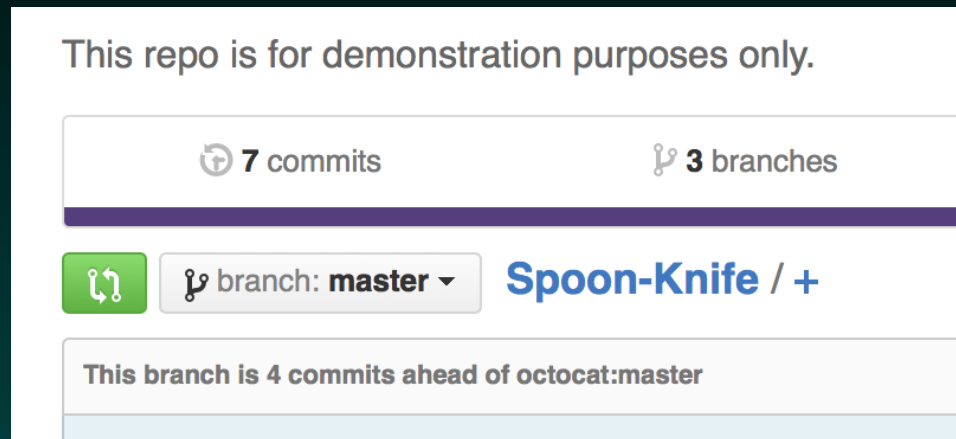
Push your edited version of the repo back to GitHub.

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.



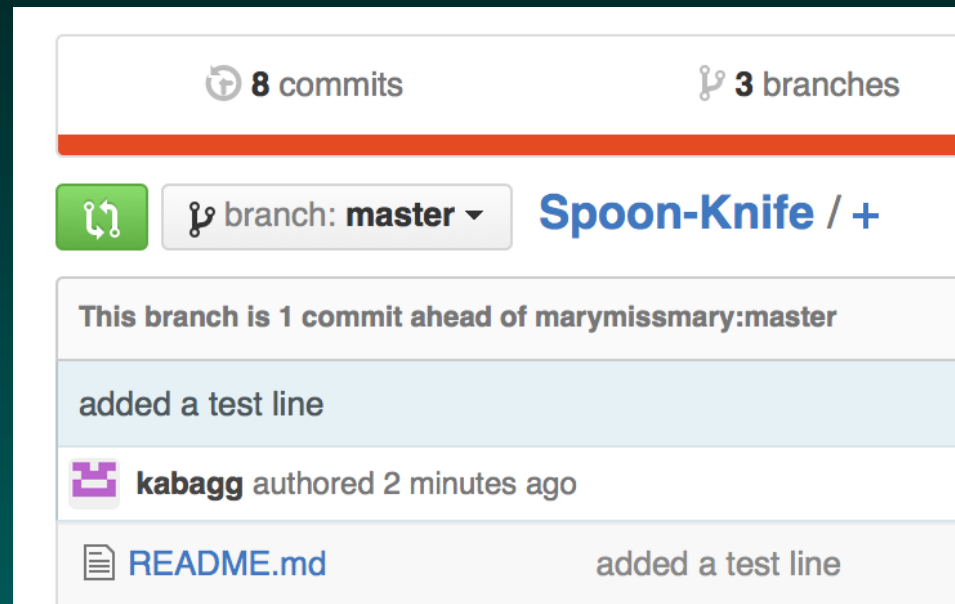
Now, within Rstudio, go to  
File/New Project/Version Control/Project Setup  
and give it the git url.

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

## Edit and Push

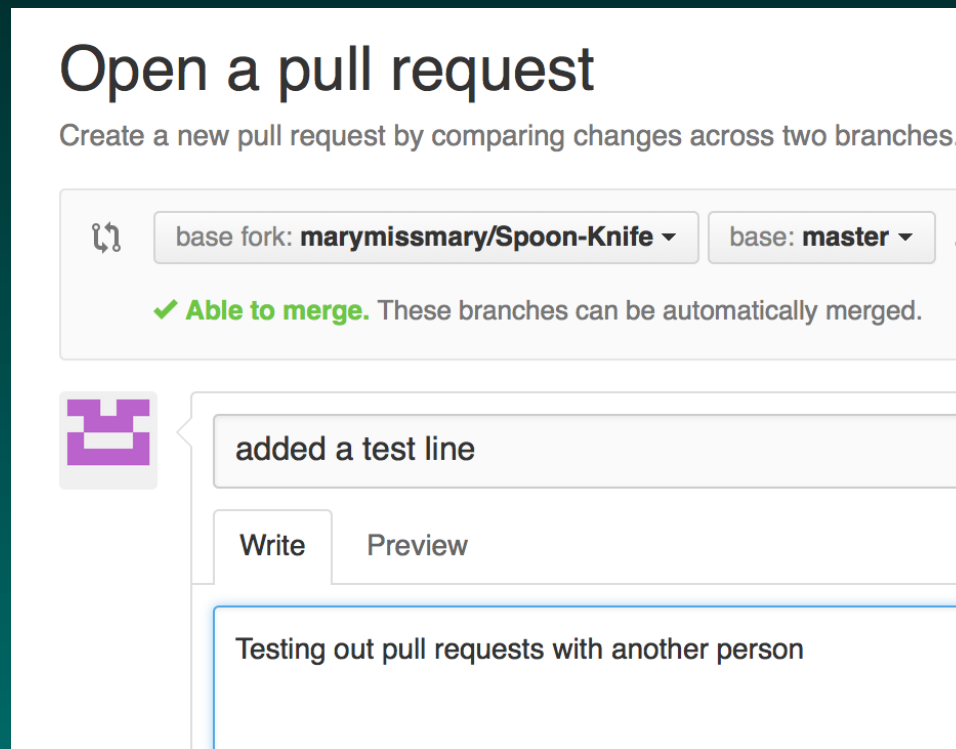
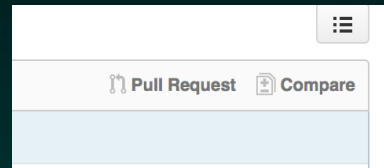
After making your edits, use the Rstudio commit popup to ignore the Rproj file and commit your desired changes.

Then push the changes over to GitHub



# Submit the Pull Request

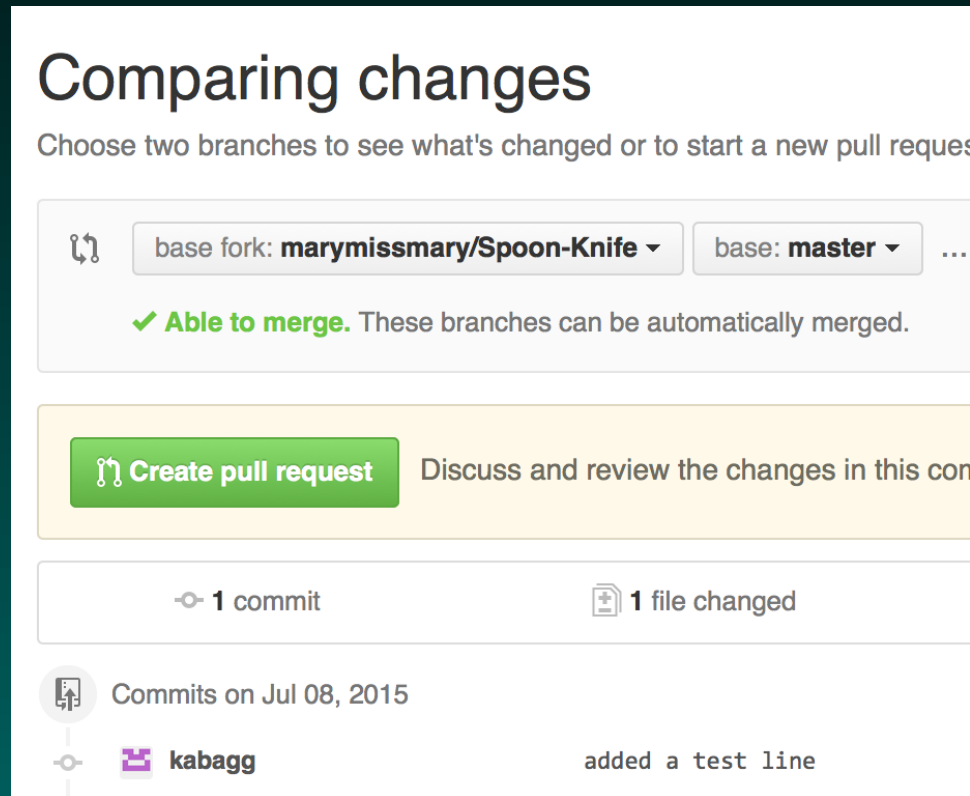
Assemble the pull request (PR), along with a brief comment about what you're trying to do.





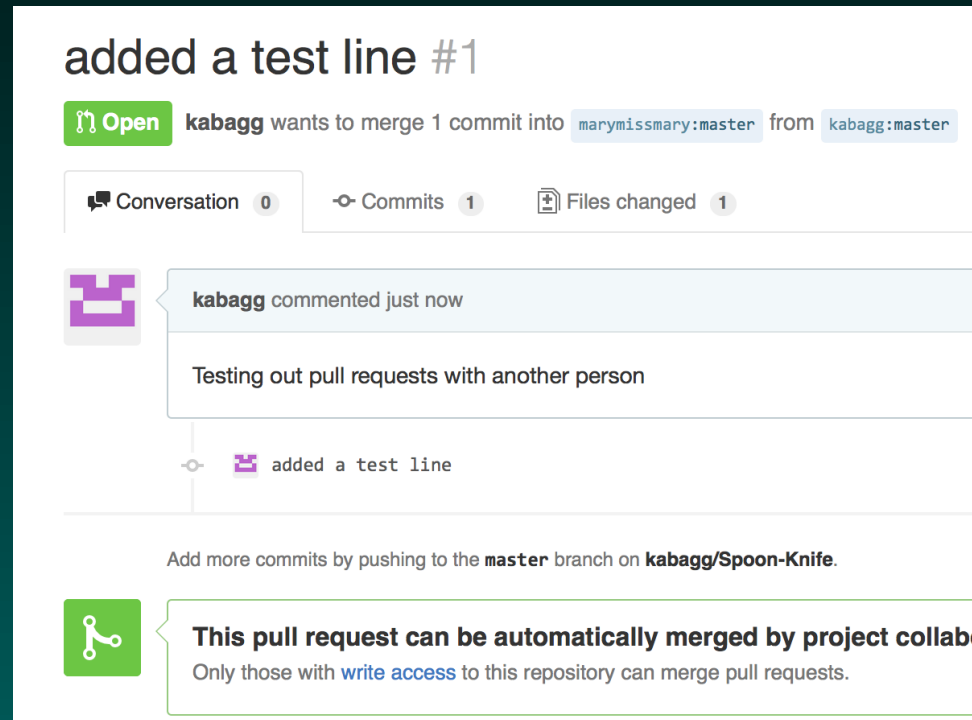
# You Get Some GitHub Feedback

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)




The screenshot displays a GitHub pull request interface. At the top, it says "added a test line #1". Below this, a green "Open" button is followed by the text "kabagg wants to merge 1 commit into marymismary:master from kabagg:master". A navigation bar shows "Conversation 0", "Commits 1", and "Files changed 1". The main content area shows a conversation between "kabagg" and the repository owner. "kabagg" commented "Testing out pull requests with another person". Below this, a commit is shown with the message "added a test line". At the bottom, a green box with a merge icon contains the text: "This pull request can be automatically merged by project collaborator. Only those with write access to this repository can merge pull requests."



added a test line #1

 Open kabagg wants to merge 1 commit into marymismary:master from kabagg:master


 Conversation 0  Commits 1  Files changed 1

 kabagg commented just now

Testing out pull requests with another person

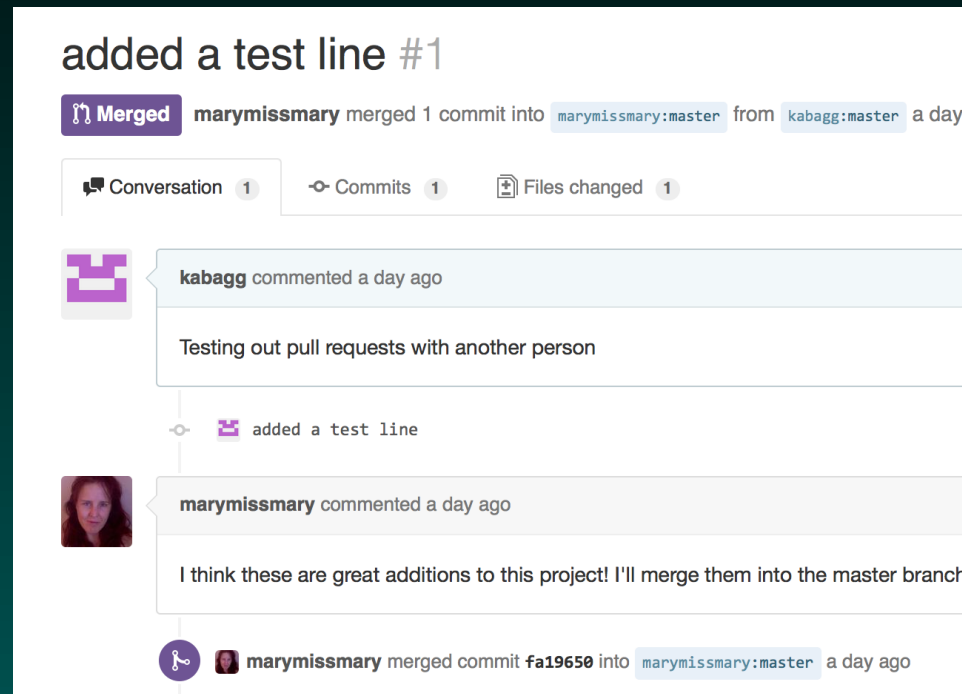
  added a test line

Add more commits by pushing to the **master** branch on **kabagg/Spoon-Knife**.

 This pull request can be automatically merged by project collaborator. Only those with [write access](#) to this repository can merge pull requests.

# 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 ;).

The screenshot displays a GitHub pull request titled "Embellished description of plotCircle #1". At the top, a green "Open" button is followed by the text "marymismary wants to merge 2 commits into kabagg:master from marymismary:master". Below this, a navigation bar shows "Conversation 0", "Commits 2", and "Files changed 1". A comment from "marymismary" is shown, stating: "Expanded description in plotCircle to briefly describe what the function does." Below the comment, the commit history is listed: "Rohrdanz, Mary A and others added some commits a day ago". Two commits are shown: "Embellished description of plotCircle ..." with hash 5bd4c15, and "making a merge conflict ..." with hash fe03da. A message below the commits says: "Add more commits by pushing to the master branch on marymismary/toyPackage." At the bottom, a green box states: "This pull request can be automatically merged. You can also merge branches on the command line." A "Merge pull request" button is visible. The bottom of the interface shows a "Write" tab, a "Preview" tab, and a note that "Markdown supported" and "Edit in fullscreen" are available.

## **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.

## Back to Rstudio

Some of these changes (pushing and pulling) are used often enough that Rstudio has added specific buttons to address them.

How does it represent some of the other stuff that's going on, e.g. branches?

When do pictures improve on the command line?

Let's revisit some stuff from before...

# The Naming History

Author	Date	SHA
Baggerly,Keith A <kabagg@mdanderson.org>	2015-07-08	88831a60
Baggerly,Keith A <kabagg@mdanderson.org>	2015-07-07	86d98557

```
BCBMC02L30BVFFT:toyPackage kabaggerly$ git log
commit 88831a608ec7119231101ad28828449e7013479f
Author: Baggerly,Keith A <kabagg@mdanderson.org>
Date:   Wed Jul 8 10:16:52 2015 -0500

    adding plotSquare function

commit 86d98557c3cd3671b25d217eaafa2e93d1589b68
Author: Baggerly,Keith A <kabagg@mdanderson.org>
Date:   Mon Jul 6 21:57:01 2015 -0500

    Initial Toy Package Commit

    This is a minimal functioning R package.
```

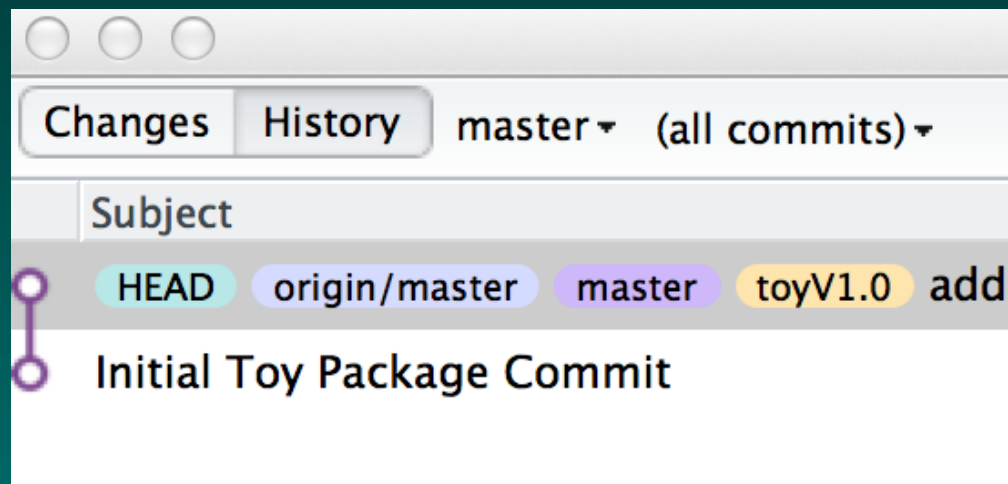
Every commit is “named” with a secure hash (SHA). Any unique part of the SHA can id the commit.

## Tagging Commits

If we want to refer to a commit by something other than its SHA, we can assign a “tag” to that commit.

```
git tag toyV1.0 88831
git tag
toyV1.0
```

This tag also appears in Rstudio's commit page

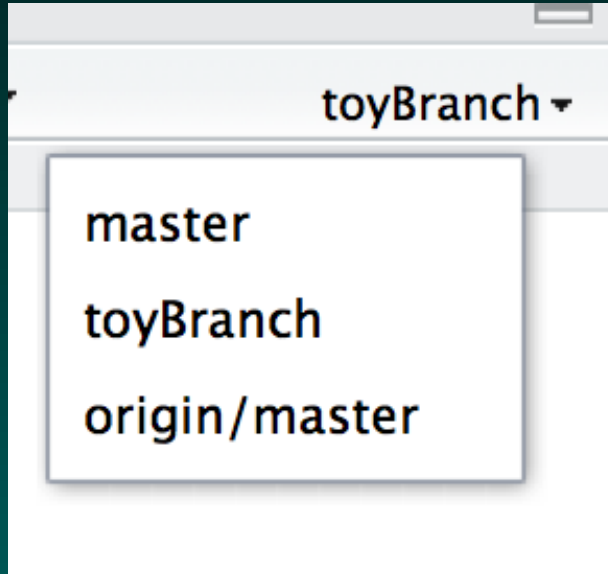




## Adding a Branch

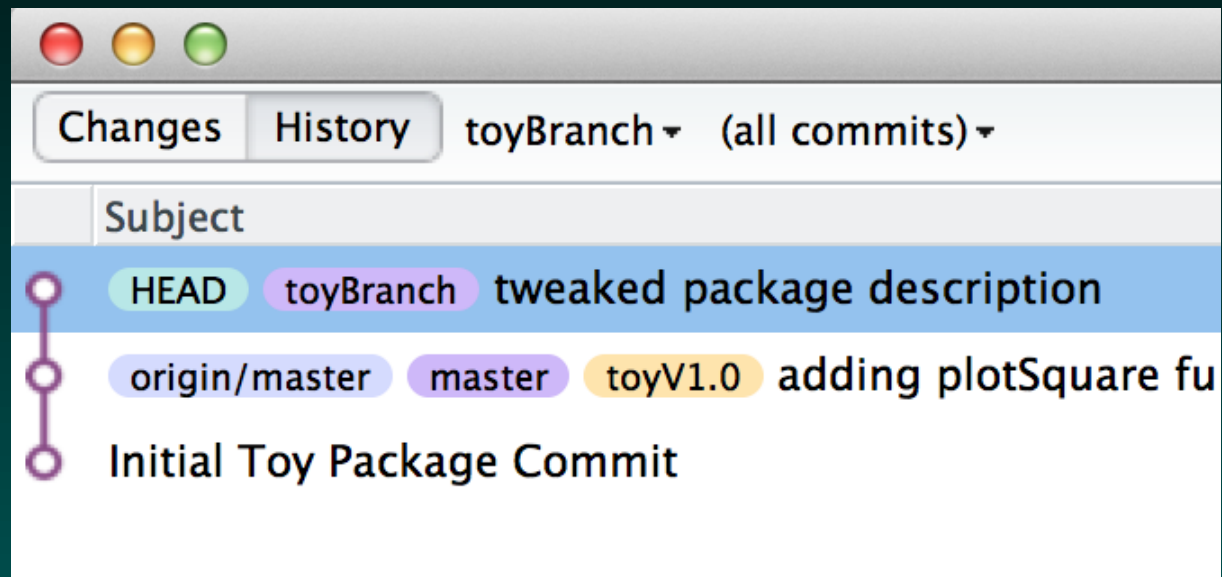
We can shift to a new named branch as follows:

```
> git checkout -b toyBranch master  
Switched to a new branch 'toyBranch'
```



# Tweak Something

I added a sentence to the package documentation, saved, staged, and committed.



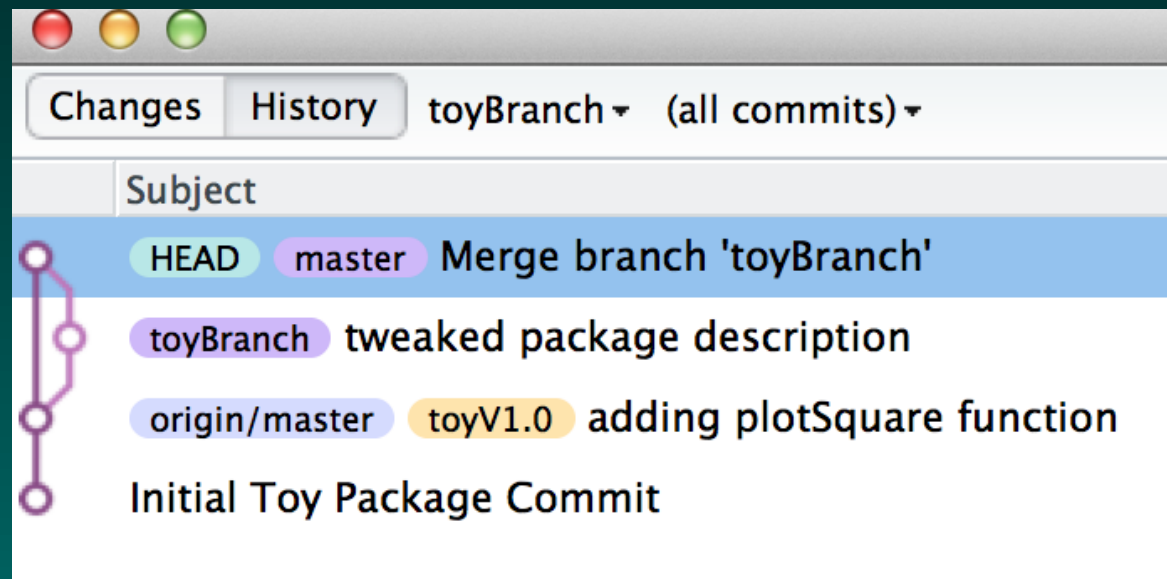
## Merge in the Change

“Checkout” shifts us to the specified branch, here “master”

```
git checkout master
```

```
git merge --no-ff toyBranch -m "minor change"
```

The “--no-ff” option keeps the history around pictorially

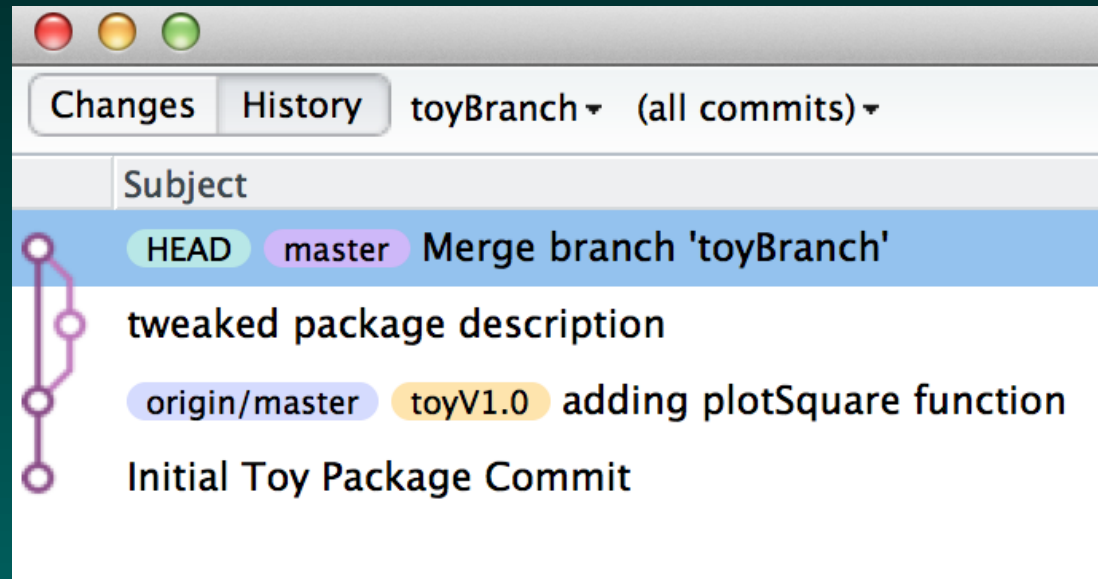


## Prune the Branch

Since I'm not taking this further, I also delete the branch, and will work with a new branch from master if needed.

```
git branch -d toyBranch
```

Deleted branch toyBranch (was 0d8fcd9).



## How We Can Use Git

ok, we've introduced collaboration via GitHub, and we've tried to illustrate how Rstudio (or other graphical interfaces) can clarify or help with some steps.

So, what can you apply git to?

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

homework submissions (precise timestamps!)

editing/commenting on reports

and lots of other stuff!