Version control

with git and GitHub

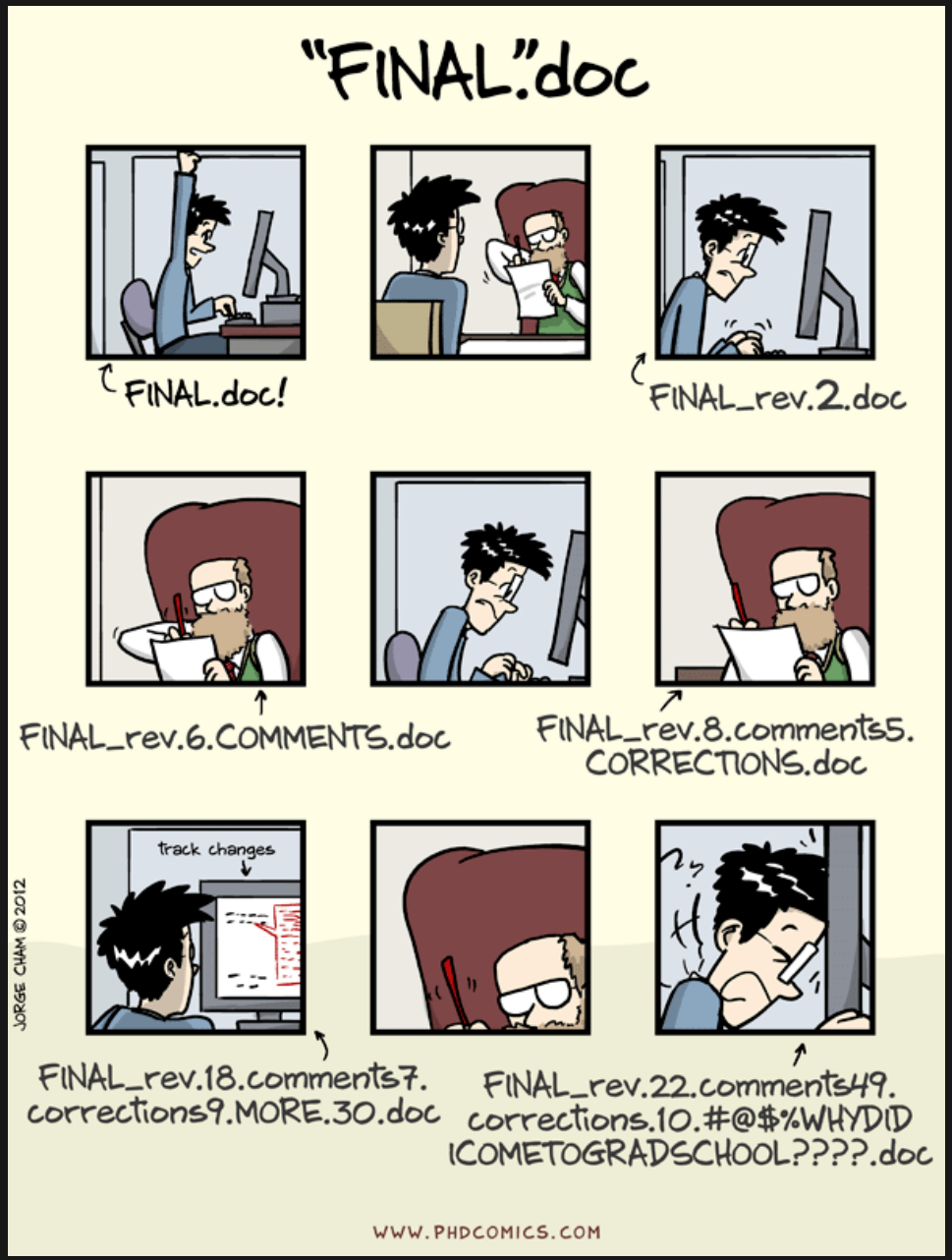
Karl Broman

Biostatistics & Medical Informatics, UW–Madison

biostat.wisc.edu/~kbroman
github.com/kbroman
@kwbroman

Course web: bit.ly/tools4rr

Slides prepared with Sam Younkin



http://www.phdcomics.com/comics/archive.php?comicid=1531

Methods for tracking versions

- Don't keep track
- Save numbered zip files
- Formal version control

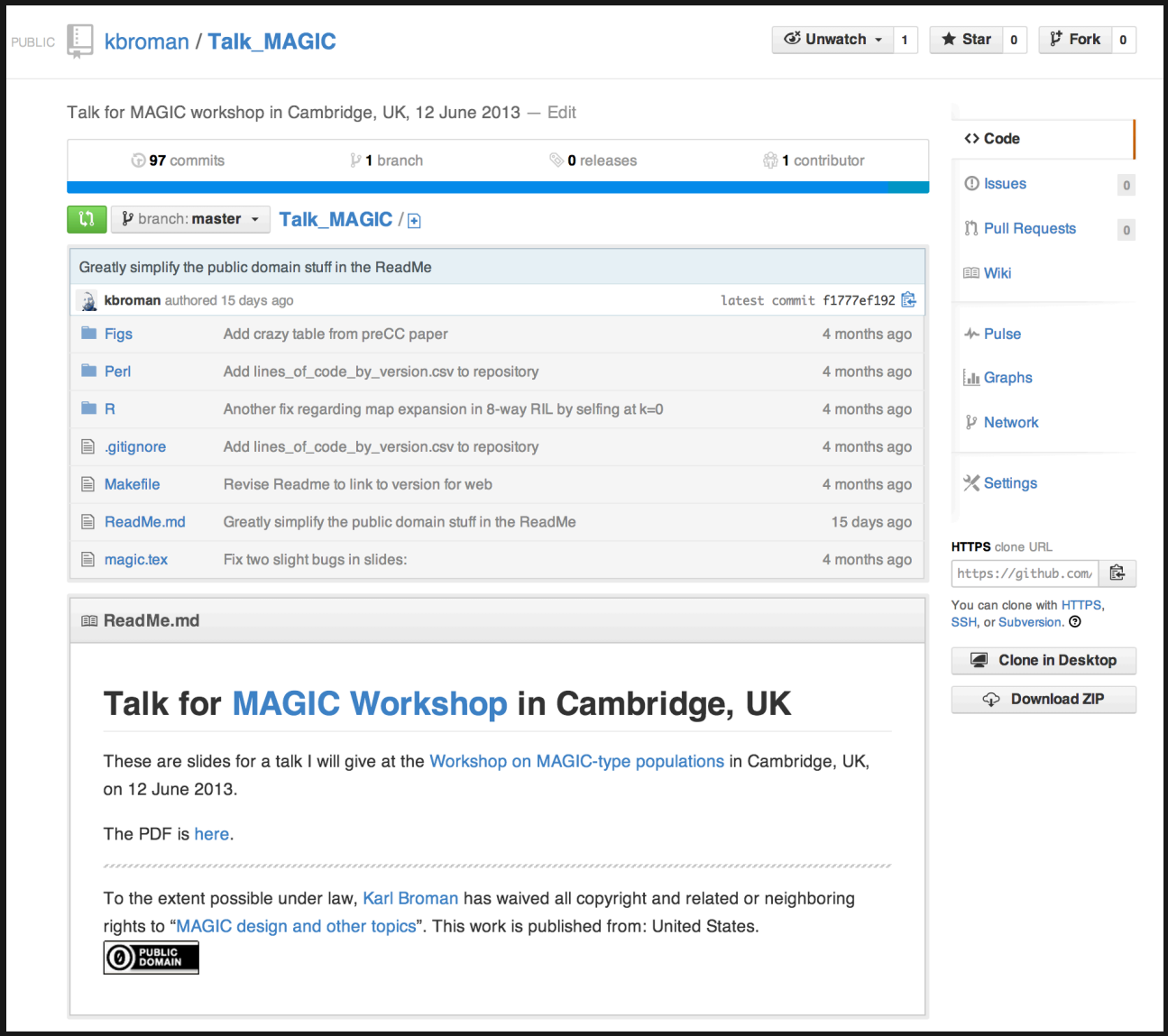
Suppose it stops working…

- Don't keep track
  - good luck!
- Save numbered zip files
  - Unzip versions and `diff`
- Formal version control
  - Easy to study changes back in time
  - Easy to jump back and test

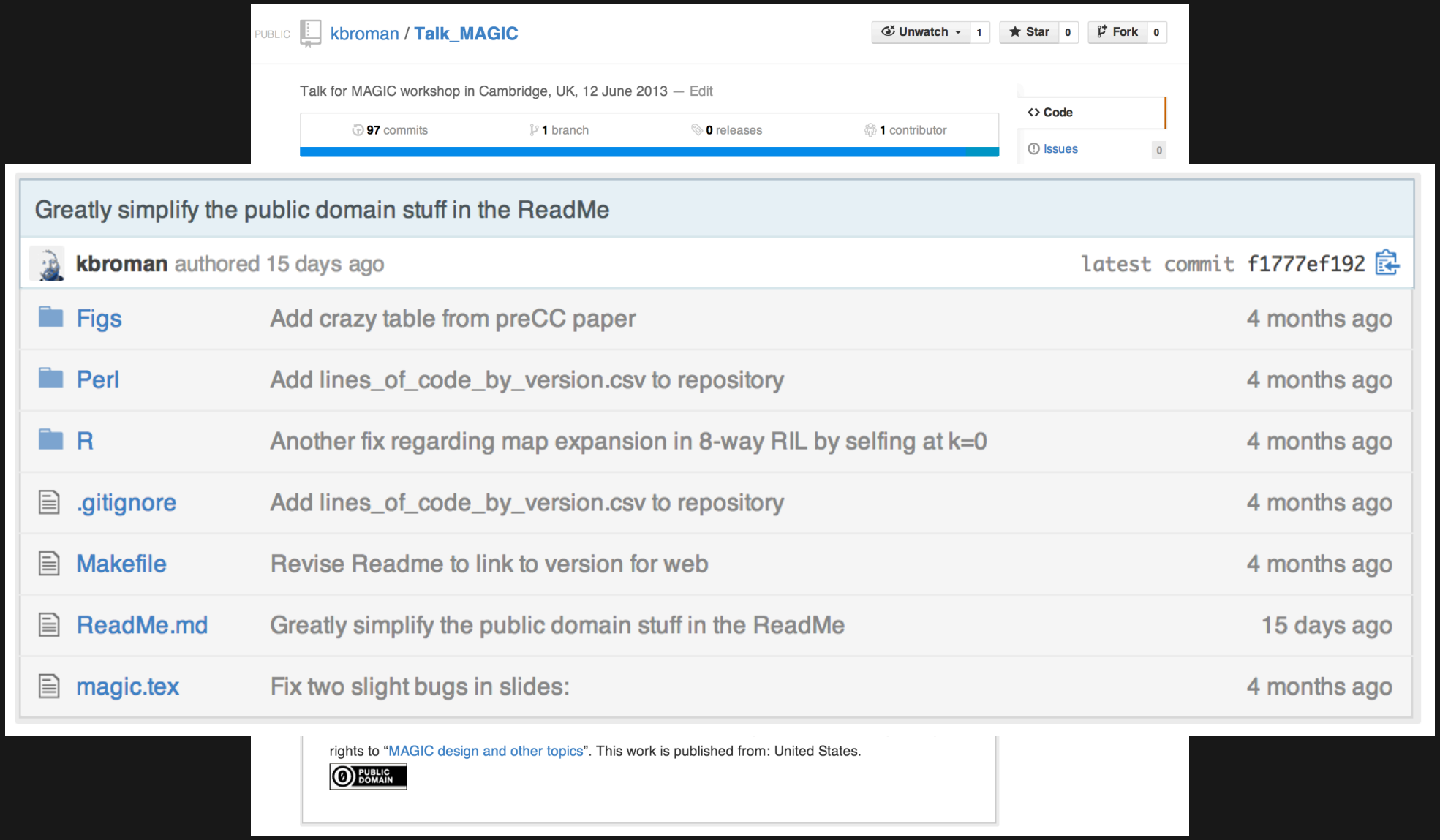
Why use formal version control?

- History of changes
- Able to go back
- No worries about breaking things that work
- Merging changes from multiple people

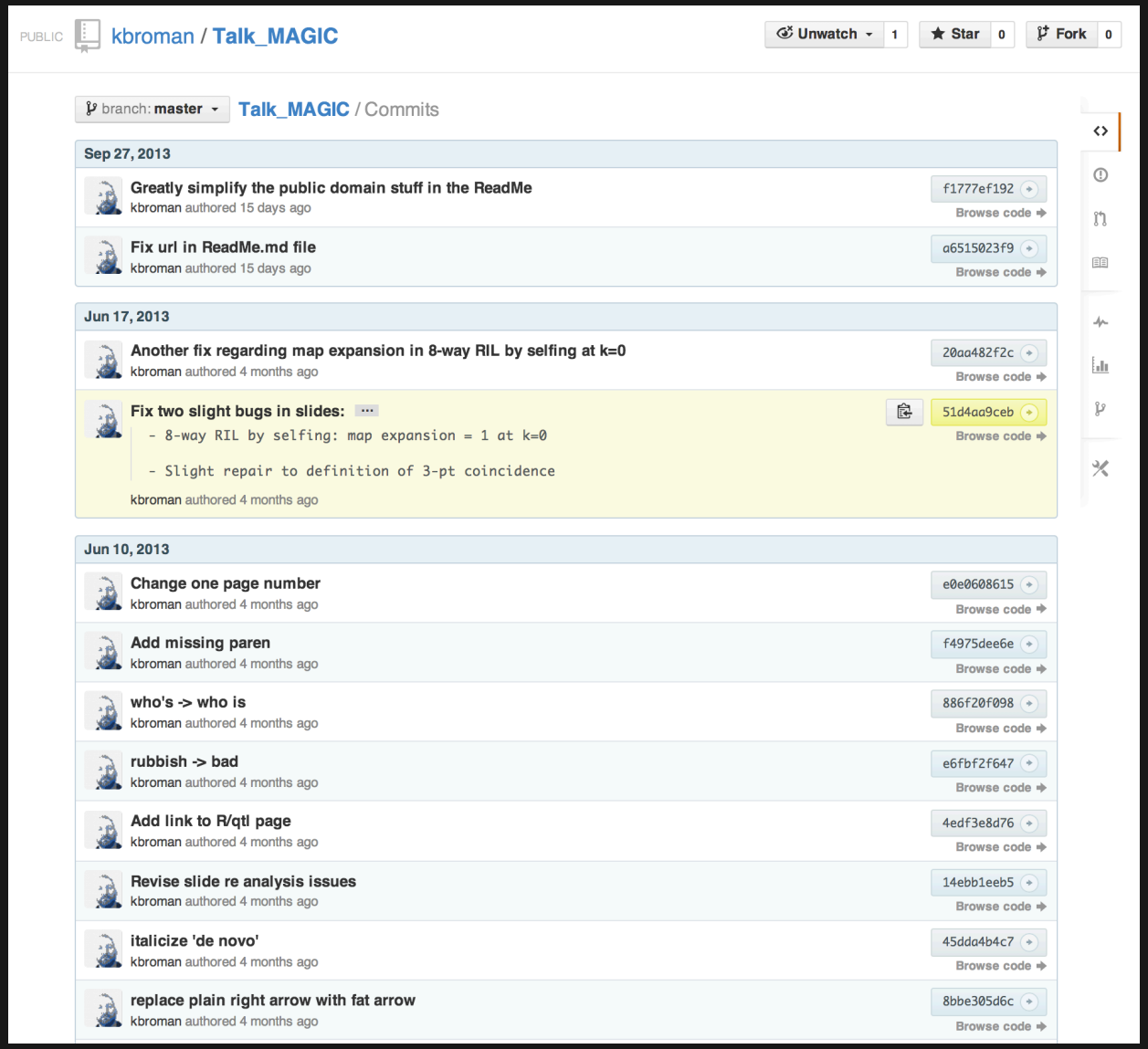
Example repository



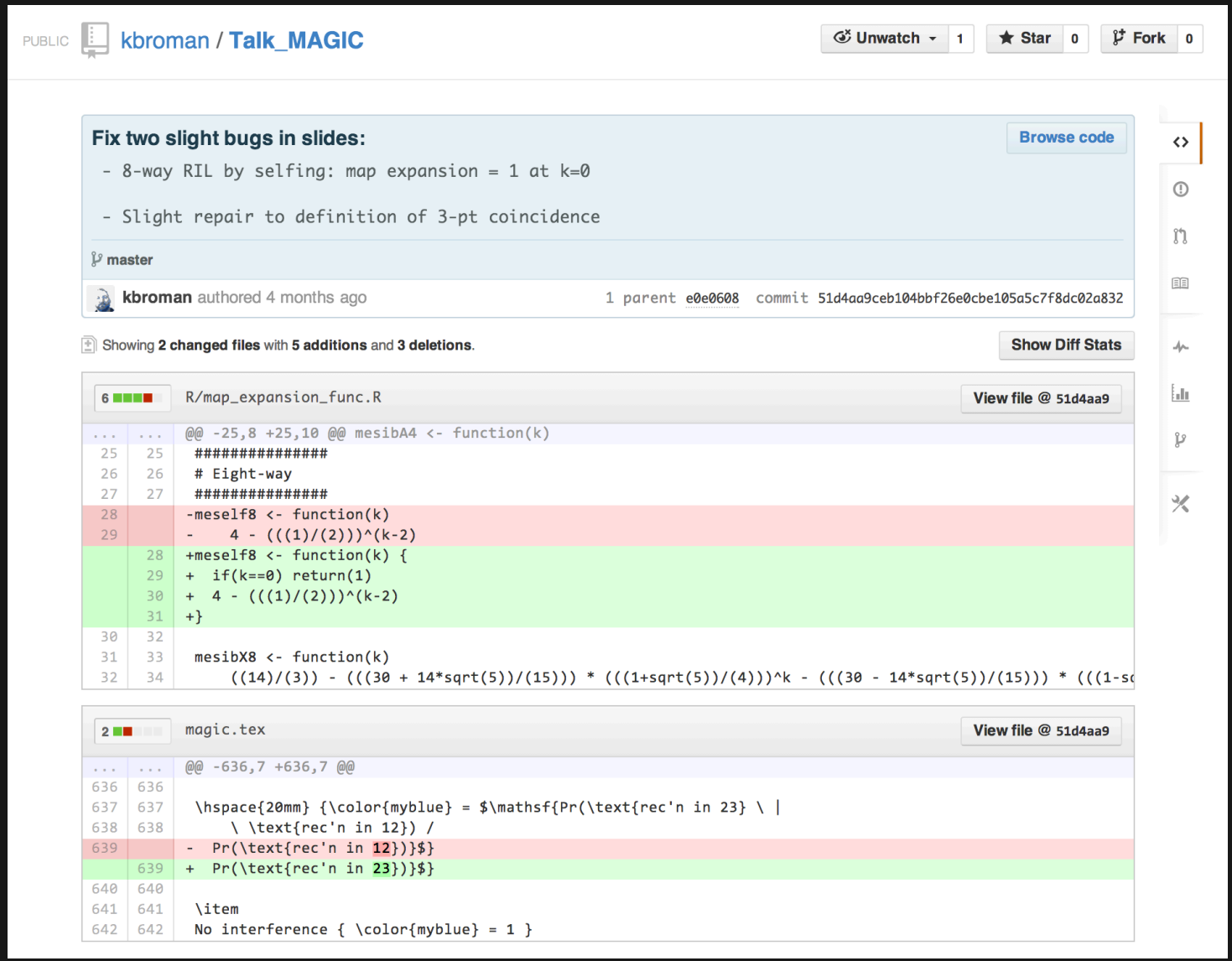
Example repository



Example history



Example commit



PUBLIC kbroman / Talk_MAGIC

Unwatch 1 | Star 0 | Fork 0

Fix two slight bugs in slides:

Browse code

```
- 8-way RIL by selfing: map expansion = 1 at k=0

- Slight repair to definition of 3-pt coincidence
```

master

kbroman authored 4 months ago — 1 parent e0e0608 commit 51d4aa9ceb104bbf26e0cbe105a5c7f8dc02a832

Showing **2 changed files** with **5 additions** and **3 deletions**. Show Diff Stats

6 R/map_expansion_func.R — View file @ 51d4aa9

```
@@ -25,8 +25,10 @@ mesibA4 <- function(k)
 ################
 # Eight-way
 ################
-meself8 <- function(k)
--    4 - (((1)/(2)))^(k-2)
+meself8 <- function(k) {
+  if(k==0) return(1)
+  4 - (((1)/(2)))^(k-2)
+}

 mesibX8 <- function(k)
     ((14)/(3)) - (((30 + 14*sqrt(5))/(15))) * (((1+sqrt(5))/(4)))^k - (((30 - 14*sqrt(5))/(15))) * (((1-s
```

2 magic.tex — View file @ 51d4aa9

```
@@ -636,7 +636,7 @@

 \hspace{20mm} {\color{myblue} = $\mathsf{Pr(\text{rec'n in 23} \ |
     \ \text{rec'n in 12}) /
-  Pr(\text{rec'n in 12})}$}
+  Pr(\text{rec'n in 23})}$}

 \item
 No interference { \color{myblue} = 1 }
```

What is git?

- Formal version control system
- Developed by Linus Torvalds (developer of Linux)
  - used to manage the source code for Linux
- Tracks any content (but mostly plain text files)
  - source code
  - data analysis projects
  - manuscripts
  - websites
  - presentations

Why use git?

- It's fast
- You don't need access to a server
- Amazingly good at merging simultaneous changes
- Everyone's using it

What is GitHub?

- A home for git repositories
- Interface for exploring git repositories
- Real open source
  - immediate, easy access to the code
- Like facebook for programmers
- Free 2-year "micro" account for students
  - github.com/edu
- (Bitbucket.org is an alternative)
  - free private repositories

Why use GitHub?

- It takes care of the server aspects of git
- Graphical user interface for git
  - Exploring code and its history
  - Tracking issues
- Facilitates:
  - Learning from others
  - Seeing what people are up to
  - Contributing to others' code
- Lowers the barrier to collaboration
  - "There's a typo in your documentation." vs. "Here's a correction for your documentation."

Basic use

- Change some files
- See what you've changed

```
git status
git diff
git log
```

- Indicate what changes to save

```
git add
```

- Commit to those changes

```
git commit
```

Basic use

- Change some files
- See what you've changed

```
git status
git diff
git log
```

- Indicate what changes to save

```
git add
```

- Commit to those changes

```
git commit
```

- Push the changes to GitHub

```
git push
```

Basic use

- Change some files
- See what you've changed

```
git status
git diff
git log
```

- Indicate what changes to save

```
git add
```

- Commit to those changes

```
git commit
```

- Push the changes to GitHub

```
git push
```

- Pull changes from your collaborator

```
git pull
```

Basic use

- Change some files
- See what you've changed

```
git status
git diff
git log
```

- Indicate what changes to save

```
git add
```

- Commit to those changes

```
git commit
```

- Push the changes to GitHub

```
git push
```

- Pull changes from your collaborator

```
git fetch
git merge
```

Initialize repository

- Create (and `cd` to) a working directory
  - For example, `~/Docs/Talks/Graphs`
- Initialize it to be a git repository
  - `git init`
  - Creates subdirectory `~/Docs/Talks/Graphs/.git`

```
$ mkdir ~/Docs/Talks/Graphs
$ cd ~/Docs/Talks/Graphs
$ git init
Initialized empty Git repository in ~/Docs/Talks/Graphs/.git/
```

Produce content

- Create a `README.md` file

```
## Talk on &ldquo;How to display data badly&rdquo;

These are slides for a talk that I give as often as possible,
because it's fun.

This was inspired by Howard Wainer's article, whose title I
stole: H Wainer (1984) How to display data badly.
American Statistician 38:137-147

A recent PDF is
[here](
http://www.biostat.wisc.edu/~kbroman/talks/graphs2013.pdf).
```

Incorporate into repository

- Stage the changes using `git add`

```
$ git add README.md
```

Incorporate into repository

- Now commit using `git commit`

```
$ git commit -m "Initial commit of README.md file"
[master (root-commit) 32c9d01] Initial commit of README.md file
 1 file changed, 14 insertions(+)
 create mode 100644 README.md
```

- The `-m` argument allows one to enter a message
- Without `-m`, `git` will spawn a text editor
- Use a meaningful message
- Message can have multiple lines, but make 1st line an overview

Using git on an existing project

- `git init`
- Set up `.gitignore` file
- `git status` (did you miss any?)
- `git add .` (or name files individually)
- `git status` (did you miss any?)
- `git commit`

Removing/moving files

For files that are being tracked by git:

Use `git rm` instead of just `rm`
Use `git mv` instead of just `mv`

```
$ git rm myfile
$ git mv myfile newname
$ git mv myfile SubDir/
$ git commit
```

A few points on commits

- Use frequent, small commits
- Don't get out of sync with your collaborators
- Commit the sources, not the derived files

  (R code not images)
- Use a `.gitignore` file to indicate files to be ignored

```
*~
manuscript.pdf
Figs/*.pdf
.RData
.RHistory
*.Rout
*.aux
*.log
*.out
```

First use of git

```
$ git config --global user.name "Jane Doe"
$ git config --global user.email "janedoe@wisc.edu"

$ git config --global color.ui true

$ git config --global core.editor emacs

$ git config --global core.excludesfile ~/.gitignore_global
```

Getting started with GitHub

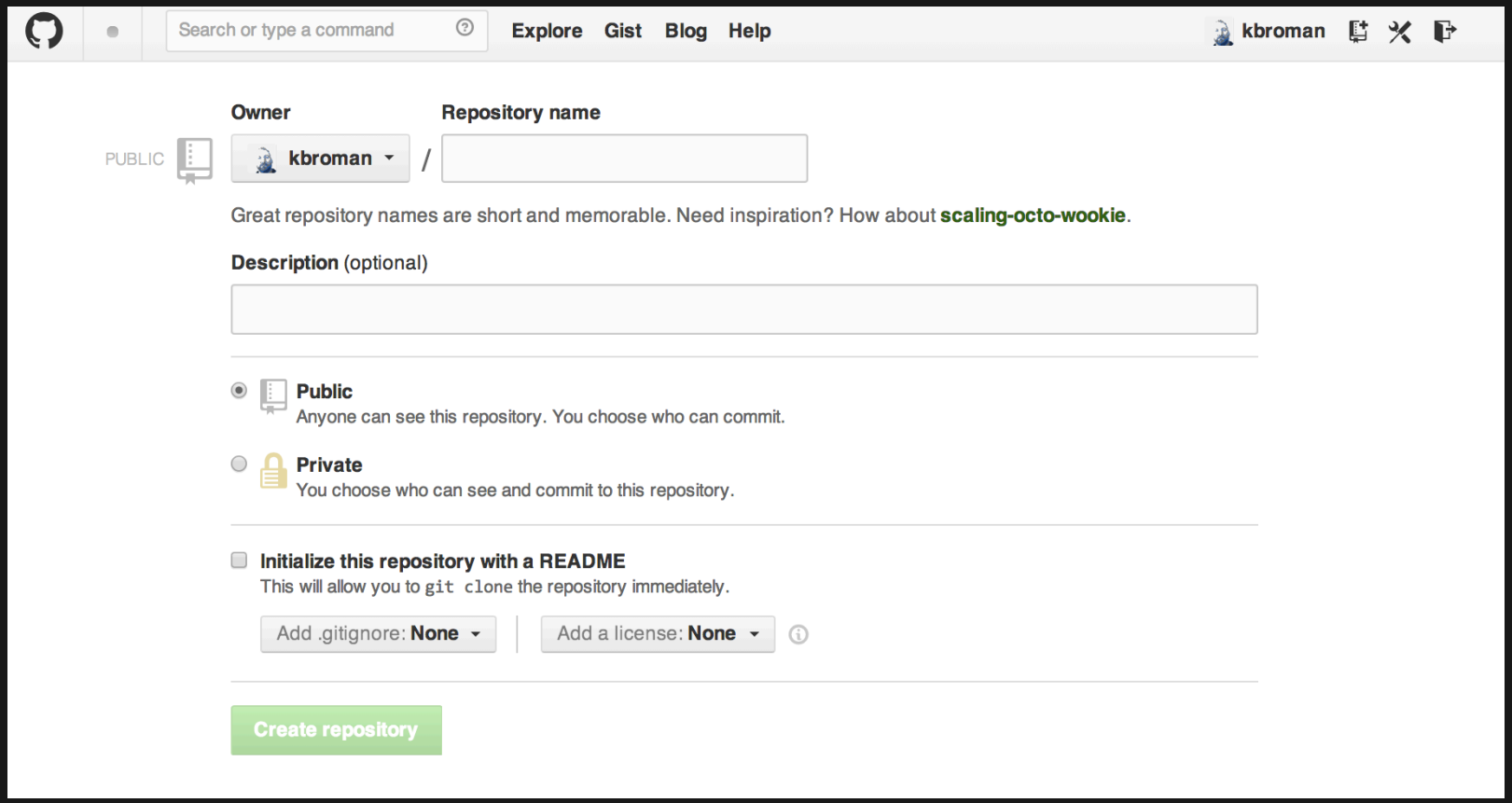
- Get an account
- Set up ssh keys
  - Look for files `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub`
  - `ssh-keygen -t rsa -C "janedoe@wisc.edu"`
  - Copy contents of `~/.ssh/id_rsa.pub`
- Add SSH key at GitHub
  - Account settings
  - SSH Keys
  - Add SSH key
  - Paste contents of `~/.ssh/id_rsa.pub`
- Similar thing at BitBucket

Set up GitHub repository

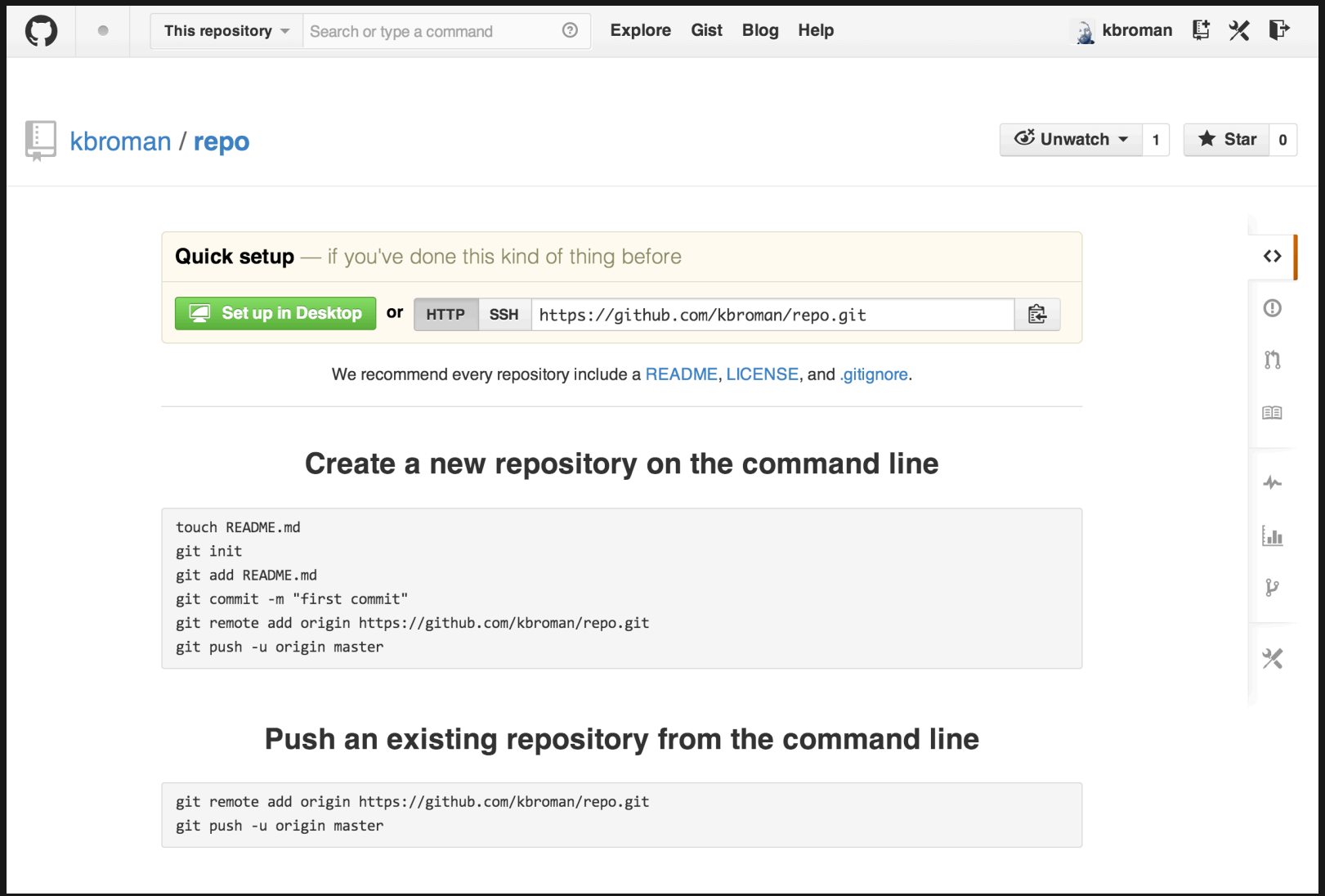
- Click the "Create a new repo" button
- Give it a name and description
- Click the "Create repository" button
- Back at the command line:

```
git remote add origin git@github.com:username/repo
git push -u origin master
```

Set up GitHub repository



Set up GitHub repository



Configuration file

Part of a `.git/config` file:

```
[remote "origin"]
url = git@github.com:kbroman/qtl.git
fetch = +refs/heads/*:refs/remotes/origin/*

[branch "master"]
remote = origin
merge = refs/heads/master

[remote "brian"]
url = git://github.com/byandell/qtl.git
fetch = +refs/heads/*:refs/remotes/brian/*
```

Branching and merging

- Use branches to test out new features without breaking the working code.

```
git branch devel
git branch
git checkout devel
```

- When you're happy with the work, merge it back into your master branch.

```
git checkout master
git merge devel
```

Issues and pull requests

- Problem with or suggestion for someone's code?
  - Point it out as an Issue
- Even better: Provide a fix
  - Fork
  - Clone
  - Modify
  - Commit
  - Push
  - Submit a Pull Request

Suggest a change to a repo

- Go to the repository:

  `http://github.com/someone/repo`

- Fork the repository

  Click the "Fork" button

- Clone your version of it

  `git clone git@github.com:username/repo`

- Change things locally, `git add`, `git commit`
- Push your changes to *your* GitHub repository

  `git push`

- Go to *your* GitHub repository
- Click "Pull Requests" and "New pull request"

Pulling a friend's changes

- Add a connection

```
git remote add friend git://github.com/friend/repo
```

- If you trust them, just pull the changes

```
git pull friend master
```

- Alternatively, fetch the changes, test them, and *then* merge them.

```
git fetch friend master
git branch -a
git checkout remotes/friend/master
git checkout -b friend
git checkout master
git merge friend
```

- Push them back to your GitHub repo

```
git push
```

Merge conflicts

Sometimes after `git pull friend master`

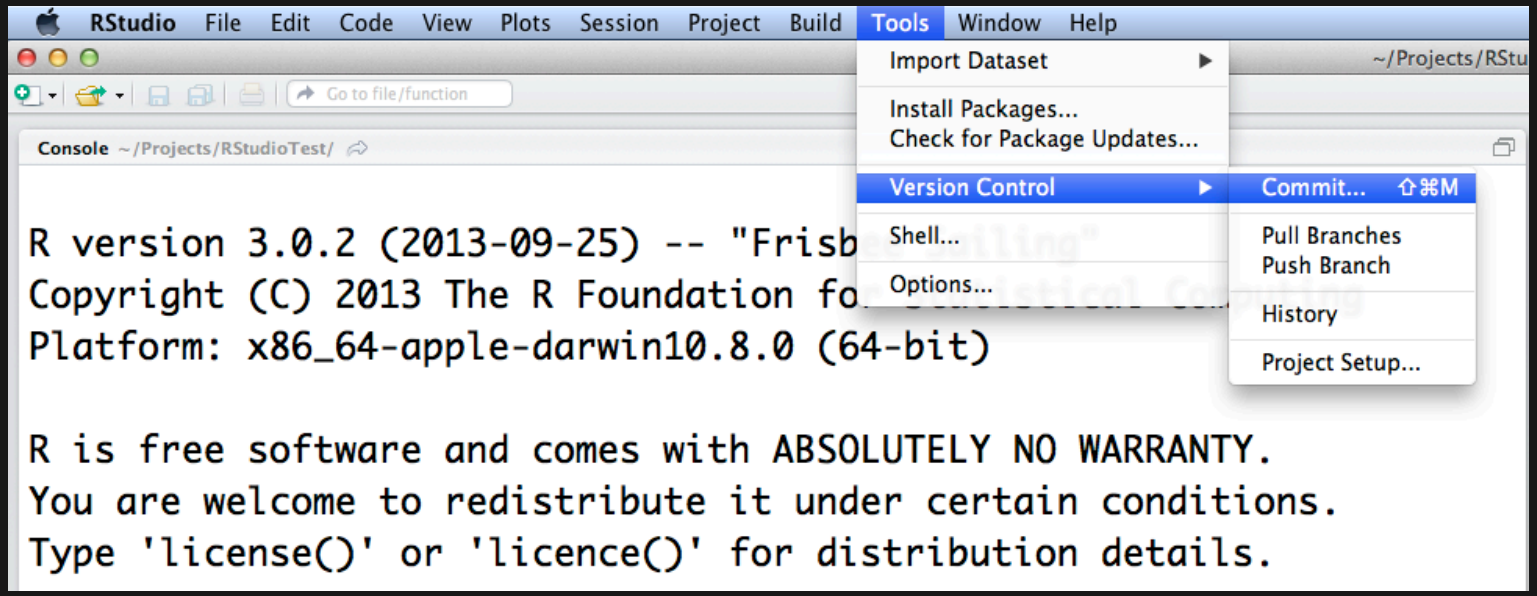
```
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Inside the file you'll see:

```
<<<<<<< HEAD
A line in my file.
=======
A line in my friend's file
>>>>>>> 031389f2cd2acde08e32f0beb084b2f7c3257fff
```

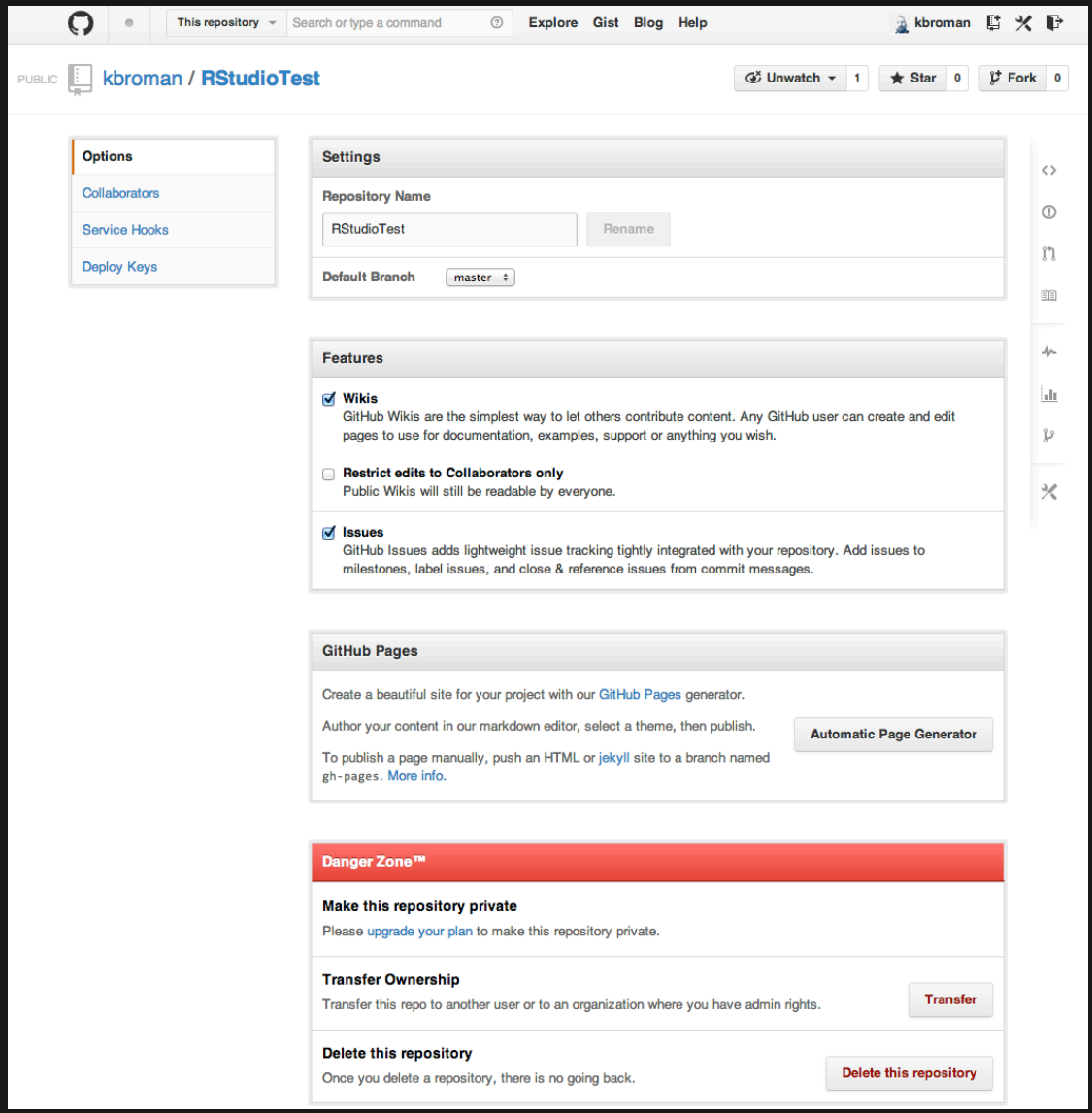
Edit, add, commit, push, submit pull request.

git/GitHub with RStudio



See GitPrimer.pdf or RStudio page

Delete GitHub repo



Open source means everyone can see my stupid mistakes.

Version control means everyone can see every stupid mistake I've ever made.

bit.ly/stupidcode