

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](#)

"FINAL".doc



FINAL.doc!



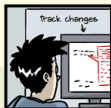
FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc



FINAL_rev.22.comments49.
corrections.10. #@\$%WHYDID
ICOMETOGRADSCHOOL?????.doc

© 2012
Jesse Cohn

WWW.PHDCOMICS.COM

<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

PUBLIC

kbroman / Talk_MAGIC

Unwatch 1

Star 0

Fork 0

Talk for MAGIC workshop in Cambridge, UK, 12 June 2013 — Edit

97 commits

1 branch

0 releases

1 contributor

branch: master

Talk_MAGIC

Greatly simplify the public domain stuff in the ReadMe

kbroman authored 15 days ago

latest commit f1777ef192

Figs

Add crazy table from preCC paper

4 months ago

Perf

Add lines_of_code_by_version.csv to repository

4 months ago

R

Another fix regarding map expansion in 8-way RIL by selfing at k=0

4 months ago

.gitignore

Add lines_of_code_by_version.csv to repository

4 months ago

Makefile

Revise README to link to version for web

4 months ago

ReadMe.md

Greatly simplify the public domain stuff in the ReadMe

15 days ago

magic.tex

Fix two slight bugs in slides:

4 months ago


ReadMe.md

Talk for MAGIC Workshop in Cambridge, UK

These are slides for a talk I will give at the [Workshop on MAGIC-type populations](#) in Cambridge, UK, on 12 June 2013.

The PDF is [here](#).

To the extent possible under law, [Karl Broman](#) has waived all copyright and related or neighboring rights to "MAGIC design and other topics". This work is published from: United States.



<> Code

Issues 0

Pull Requests 0

Wiki

Pulse

Graphs

Network

Settings

HTTPS clone URL

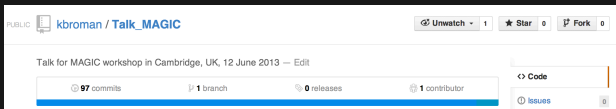
<https://github.com>

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

Example repository



Greatly simplify the public domain stuff in the ReadMe



kbroman authored 15 days ago


latest commit [f177ef192](#)




Figs	Add crazy table from preCC paper	4 months ago
Perl	Add lines_of_code_by_version.csv to repository	4 months ago
R	Another fix regarding map expansion in 8-way RIL by selfing at k=0	4 months ago
.gitignore	Add lines_of_code_by_version.csv to repository	4 months ago
Makefile	Revise Readme to link to version for web	4 months ago
ReadMe.md	Greatly simplify the public domain stuff in the ReadMe	15 days ago
magic.tex	Fix two slight bugs in slides:	4 months ago

rights to "MAGIC design and other topics". This work is published from: United States.




Example history


PUBLIC  **kbroman / Talk_MAGIC**

 Unwatch 1  Star 0  Fork 0


branch: master **Talk_MAGIC** / Commits


Sep 27, 2013

 **Greatly simplify the public domain stuff in the ReadMe** f1777ef192
kbroman authored 15 days ago [Browse code](#)


 **Fix url in ReadMe.md file** o6515823f9
kbroman authored 15 days ago [Browse code](#)


Jun 17, 2013


 **Another fix regarding map expansion in 8-way RIL by selfing at k=0** 280a0482f2c
kbroman authored 4 months ago [Browse code](#)


 **Fix two slight bugs in slides:** 51d4a09c8b
- 8-way RIL by selfing: map expansion = 1 at k=0
- Slight repair to definition of 3-pt coincidence
kbroman authored 4 months ago [Browse code](#)


Jun 10, 2013


 **Change one page number** e8e0688615
kbroman authored 4 months ago [Browse code](#)


 **Add missing paren** f4975dee6e
kbroman authored 4 months ago [Browse code](#)


 **who's -> who is** 886f20f098
kbroman authored 4 months ago [Browse code](#)

 **rubbish -> bad** e0fbf2f647
kbroman authored 4 months ago [Browse code](#)

 **Add link to R/qtl page** 4edf3e8d76
kbroman authored 4 months ago [Browse code](#)

 **Revise slide re analysis issues** 14ebb1eeb5
kbroman authored 4 months ago [Browse code](#)

 **italicize 'de novo'** 45dda4b4c7
kbroman authored 4 months ago [Browse code](#)

 **replace plain right arrow with fat arrow** 8bbe305d6c
kbroman authored 4 months ago [Browse code](#)

Example commit

PUBLIC kbroman / Talk_MAGIC

Unwatch 1 Star 0 Fork 0

Fix two slight bugs in slides:

- 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 e0e8608 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)
25 25 #####
26 26 # Eight-way
27 27 #####
28 -mesibf8 <- function(k)
29 - 4 - (((1)/(2)))^(k-2)
+mesibf8 <- function(k) {
+  if(k==0) return(1)
+  4 - (((1)/(2)))^(k-2)
+}
30 32
31 33 mesibX8 <- function(k)
32 34 ((14)/(3)) - (((30 + 14*sqrt(5))/(15))) * (((1+sqrt(5))/(4)))^k - (((30 - 14*sqrt(5))/(15))) * (((1-sqrt(5))/(4)))^k)
```

2 magic.tex View file @ 51d4aa9

```
... @@ -636,7 +636,7 @@
636 636 \hspace{20mm} {\color{myblue} = $\mathsf{Pr}(\text{rec'n in 23}) \setminus
637 637 \setminus \text{rec'n in 12}) /
638 638
639 - Pr(\text{rec'n in 12})}$$$
+ Pr(\text{rec'n in 23})}$$$
640 640
641 641 \item
642 642 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 "How to display data badly";

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



[Explore](#) [Gist](#) [Blog](#) [Help](#)


 kbroman   


PUBLIC 

 kbroman /

Great repository names are short and memorable. Need inspiration? How about [scaling-octo-wookie](#).

Description (optional)

☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

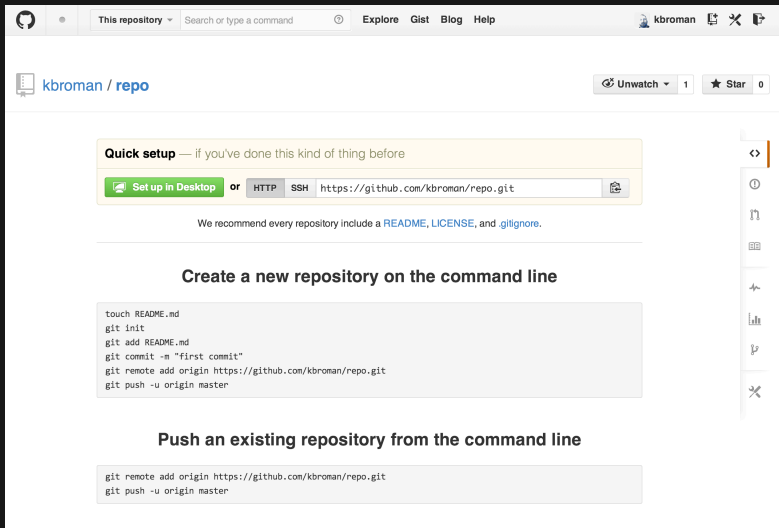
☐ **Initialize this repository with a README**
This will allow you to `git clone` the repository immediately.

Add .gitignore: **None**

Add a license: **None** ⓘ



Create repository

Set up GitHub repository



The screenshot shows the GitHub web interface for a repository named 'repo' by user 'kbroman'. The top navigation bar includes the GitHub logo, a dropdown for 'This repository', a search bar, and links for 'Explore', 'Gist', 'Blog', and 'Help'. The repository header shows 'kbroman / repo' with 'Unwatch' (1) and 'Star' (0) buttons. A 'Quick setup' section offers a 'Set up in Desktop' button or an 'SSH' link to 'https://github.com/kbroman/repo.git'. Below this, a recommendation states: 'We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).' The main content area has two sections: 'Create a new repository on the command line' with a code block for initializing a new repo, and 'Push an existing repository from the command line' with a code block for pushing an existing repo.

Quick setup — if you've done this kind of thing before

 **Set up in Desktop** or **HTTP** **SSH** `https://github.com/kbroman/repo.git` 

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/kbroman/repo.git
git push -u origin master
```

Push an existing repository from the command line

```
git remote add origin https://github.com/kbroman/repo.git
git push -u origin master
```

Configuration file

Part of a .git/config file:

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

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

[remote "brian"]
url = git://github.com/byandell/ctl.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`

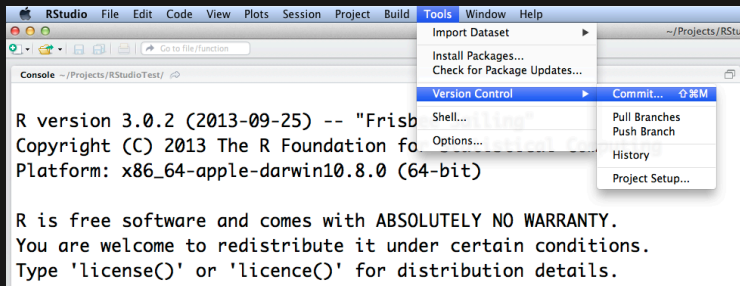
```
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

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

The screenshot shows the GitHub repository settings page for a public repository named 'RStudioTest' by user 'kbroman'. The page is divided into several sections: 'Options' (Collaborators, Service Hooks, Deploy Keys), 'Settings' (Repository Name, Default Branch), 'Features' (Wikis, Restrict edits to Collaborators only, Issues), 'GitHub Pages' (Automatic Page Generator), and a 'Danger Zone' at the bottom. The 'Danger Zone' contains three critical actions: 'Make this repository private', 'Transfer Ownership', and 'Delete this repository'. The 'Delete this repository' button is highlighted in red.

Options

- Collaborators
- Service Hooks
- Deploy Keys

Settings

Repository Name: RStudioTest [Rename](#)

Default Branch: master [⌵](#)

Features

☒ **Wikis**
GitHub Wikis are the simplest way to let others contribute content. Any GitHub user can create and edit pages to use for documentation, examples, support or anything you wish.

☐ **Restrict edits to Collaborators only**
Public Wikis will still be readable by everyone.

☒ **Issues**
GitHub Issues adds lightweight issue tracking tightly integrated with your repository. Add issues to milestones, label issues, and close & reference issues from commit messages.

GitHub Pages

Create a beautiful site for your project with our [GitHub Pages](#) generator.
Author your content in our markdown editor, select a theme, then publish.
To publish a page manually, push an HTML or [jekyll](#) site to a branch named gh-pages. [More info](#).

[Automatic Page Generator](#)

Danger Zone™

Make this repository private
Please [upgrade your plan](#) to make this repository private.

Transfer Ownership
Transfer this repo to another user or to an organization where you have admin rights. [Transfer](#)

Delete this repository
Once you delete a repository, there is no going back. [Delete this repository](#)

Git at Statistics, UW-Madison

- ▶ Easy to use, free infinite private repositories.
- ▶ Not as nice of interface to review code: Rely on GUI or private web page.
- ▶ When your ssh account expires, your access to them expires.

Git at Statistics, UW-Madison

Setup (on server):

- ▶ Connect to server

```
ssh bigmem01.stat.wisc.edu
```

Consider using kinit + aklog if logging on frequently

- ▶ Make Folder

```
cd Repositories
```

```
mkdir NewRepository
```

- ▶ Initialize Server Repository

```
cd NewRepository
```

```
git init
```

Git at Statistics, UW-Madison

Usage (on client, e.g. laptop):

- ▶ Clone/Pull onto other systems

```
git clone ssh:\\bigmem01.stat.wisc.edu\\-[user]\\Repositories\\NewRepository
```

- ▶ Make changes, and commit

```
git add -i
```

```
git commit -m 'An informative message here.'
```

- ▶ Push changes back

```
git push origin
```


Open source means everyone can see my stupid mistakes.

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