Using Github for research and life

Matthew Malishev[1]*
[1] *Department of Biology, Emory University, 1510 Clifton Road NE, Atlanta, GA, USA, 30322*

Contents

Date: 2018-11-21
R version: 3.5.0
*Corresponding author: matthew.malishev@gmail.com
This document can be found at https://github.com/darwinanddavis

Install git

Install git on your computer following the instructions here.

Once git is on your computer, you can now access its features using either just your local computer for version control or your Github account.

Create a Github account

Create your new Github account here. Some tips on creating an account:

- Choose a username that you plan to keep. Something that represents your professional acumen, e.g. not "matt_loves_hiphop86"
- Github is universal and really useful. You can connect to programming, troubleshooting, userX sites, and coding libraries, e.g. CodePen, using your Github account, so plan for longevity.

Feel free to navigate my personal Github page. Everything is publicly available.

www.github.com/darwinanddavis

Some essential elements of your Github page:

- Your repositories. This is where you store your online information.
- Your gits. These are the digital footprints of your changes. We use this for version control.
- Your README.md file. This tells users what your repo contains, instructions for running code, troubleshooting, version control, links to external web sources, and other git specific elements, such as program/package versions.

Here are some screenshots of what you'll see on your own github page.

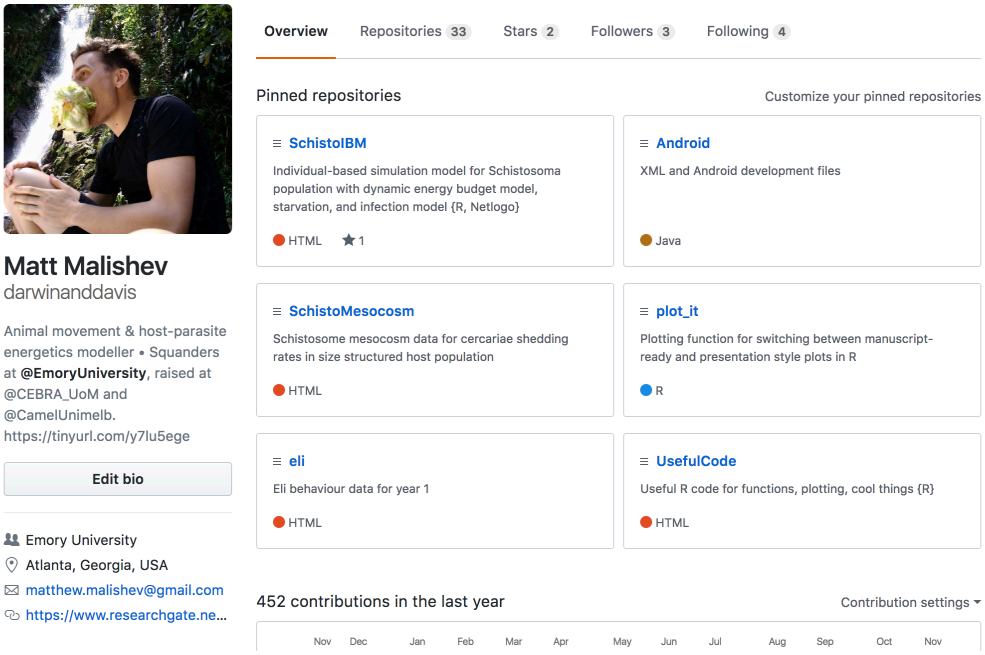

Figure 1: Github loading page

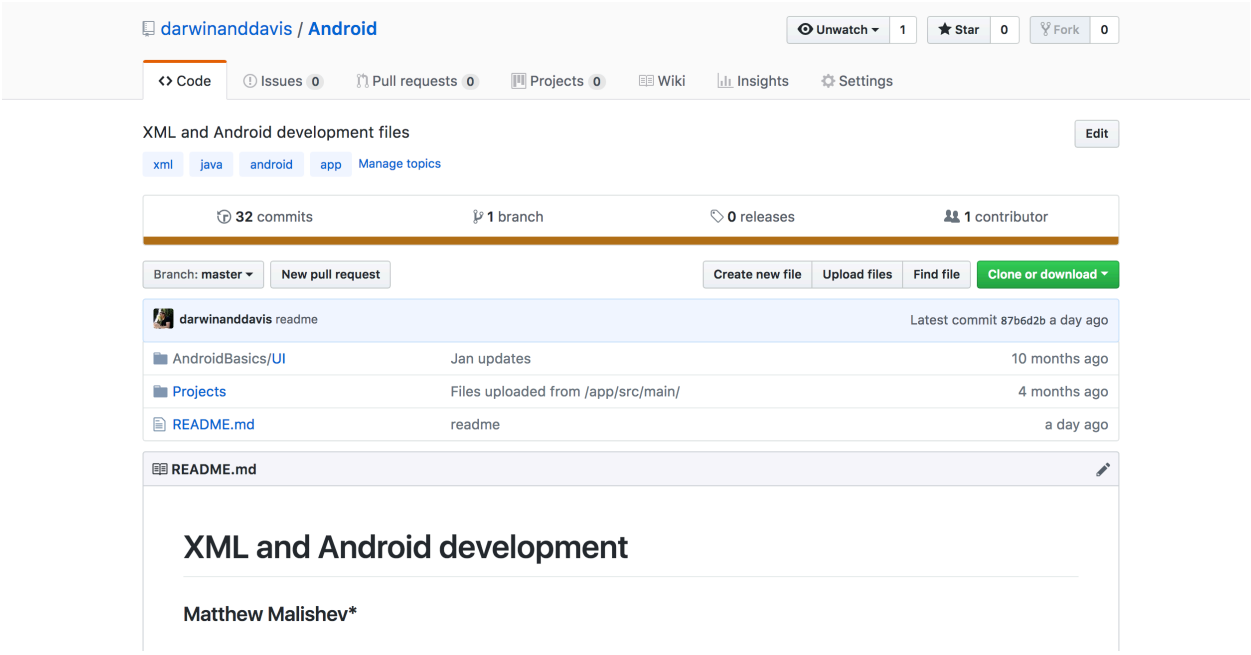

Figure 2: Repository loading page

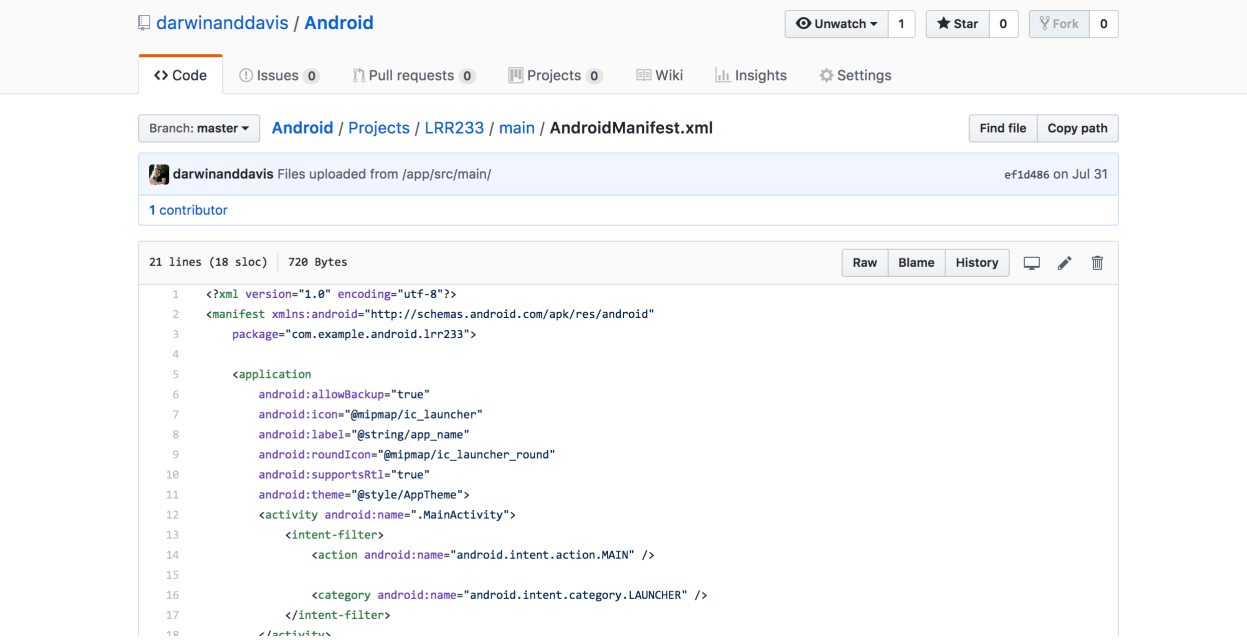

Figure 3: Inside of a file in a repository

Running the simulation model

Download the instructions for your operating system

Windows
Mac OSX

Parameters and variables

Individual

Notation follows the standard DEB notation, except in the following aspects:

- Rates are indicated with '_rate' rather than a dot
- '*ee*' is scaled reserve density of hosts in `Netlogo`

Table 1. Definitions of model parameters for individual hosts and **parasites**. Dimensions and units: -, dimensionless; cm, centimetres; J, Joules; L, length.

Parameter	Definition	Dimension (unit)	Environment
L	structural length	cm	`NetLogo`
ee	scaled reserve density	J (cm^3)	`NetLogo`
D	host development	??	`NetLogo`

Figure 4: Example of a README file

Using git and Github

We'll be using the command line to talk with git.

- In Mac, this is found in *Applications* > *Terminal*.
- In Windows, it's under *Start*, then in the Search line type *"cmd"*.

Commmon `git` syntax

Note: commands require spaces between terms.

Common `git` phrases

```
init # initialise your git
push # push your changes to a remote repository
pull # pull changes made remotely to match local git changes
fetch # re-align git changes from origin (remote) to master (local) branch
```

initialise a new git (local)

```
git init
```

add all files in directory to git (local)

```
git add .
```

add individual file (local)

```
git add abstract.txt
```

check git activity (local)

```
git status
```

add remote origin source to push git (remote)

```
# two options
get remote set-url origin https://github.com/darwinanddavis/newtest.git
git remote add origin https://github.com/darwinanddavis/newtest.git
```

push git changes to origin (your remote location) from your master (local) branch

```
git push origin master
```

check latest git activity (local)

```
git log
```

check what remote locations you have available to push your gits

```
git remote -v # v = verbose
```

add another remote destination (on github) called 'github' (remote) and push your staged git (file changes) to that remote location from your master (local) branch

```
git remote add github https://github.com/darwinanddavis/newtest.git
git push github master
```

See these references for a brief intro to using the command line in Mac and Windows.

Here is a brief intro. At least familiarise yourself with these before the workshop.

Useful command line syntax

Note: commands require spaces between terms.

`cd ~/Documents` change working dir to 'Documents'. `cd ..` move one level up

`pwd` print current working dir

`ls` list files in working dir

`mkdir newfolder` make new working dir

`touch text.txt` create new file

More useful syntax

Note: commands require spaces between terms.

copy files from *source* to *destination*. e.g. cp /Users/mydir/README.txt ~/Documents
`cp source destination`

copy all folders, subfolders, and files from *source* to *destination*
`cp -R source destination`

move files or folders from *source* to *destination* (no need for `-R`)
`mv source destination`

move multiple files with the * wildcard, which copies all .rtf files. The tilde (~) symbol is a shortcut for your Home folder, which contains '/Desktop'.
`cp ~/Desktop/*.rtf ~/Documents`

rename files
```
mv ~/Desktop/MyFile.rtf ~/Desktop/MyFile-old.rtf
cp ~/Desktop/MyFile.rtf ~/Documents/MyFile-old.rtf
```

Example of command line workflow

Install 'gitbash' to use Linux/Mac capabilities if not already

Open *Terminal/cmd*

```
cd ~/Documents/ # change working dir
ls # list dir contents
```

Open *Finder/Windows*. Make a new project on your local comp.

```
# create new project
### <b>
cd ~/Documents
### </b>
# create new file
### <b>
touch test.txt
open test.txt
### </b>
```

```
# make a new folder
### <b>
mkdir newgit
### </b>
# navigate to that folder
### <b>
cd newgit
ls -a
### </b>
```

Create a new file in the command line

```
# navigate to your new git repo
### <b>
pwd
cd ~/Documents/newgit
### </b>

# move the new file into the git repo
### <b>
mv ~/Documents/test.txt ~/Documents/newgit
ls
### </b>
```

References

Installing git

Sign up to Github

Version control with git

Terminal in Mac

Command line in Windows

Maintainer

Matt Malishev
Github | Website
matthew.malishev [at] emory.edu