Advanced UNIX and Data Processing

UNIX wildcard characters

```
cd GEN220_data/data
ls
Ecoli_K-12.fasta.gz                  Ncrassa_OR74A_InterproDomains.tab.gz rice_chr6_3kSNPs_f
Nc20H.expr.tab.gz                    codon_table.txt                      rice_random_exons.
Nc3H.expr.tab.gz                     numbers.txt                          yeast_orfs-to-chr
```

If I wanted to see all the files which end in `.txt`

```
ls *.txt
codon_table.txt numbers.txt

ls -l *.txt
-rw-r--r--@ 1 jstajich  staff  938 Oct  7 14:43 codon_table.txt
-rw-r--r--@ 1 jstajich  staff  291 Oct  7 14:43 numbers.txt

ls r*
rice_chr6_3kSNPs_filt.bed.gz rice_random_exons.bed

ls r*.gz
rice_chr6_3kSNPs_filt.bed.gz

# count lines in muliple files
wc -l *.txt
        64 codon_table.txt
       100 numbers.txt
       164 total

# count all lines in muliple compressed files
zcat *.gz | wc -l
348854
```

GZIP

Compression of files with gzip

```
$ gzip file.fa
# will produce
file.fa.gz
```

To uncompress

```
$ gunzip file.fa.gz
# will produce
```

```
file.fa
```

Searching for text with grep

Powerful pattern seaching with `grep`

Simple search for a text string:

```
$ grep Chr11 /bigdata/gen220/shared/data-examples/examples/random_exons.csv
Chr11,14656670,14656778
Chr11,3528895,3530426
Chr11,16238576,16239304
```

To get the count of number of lines that match a pattern use the `-c` option.

```
$ grep -c Chr11 /bigdata/gen220/shared/data-examples/examples/random_exons.csv
3
```

What if we wanted to count the number of times Chr1 showed up?

```
$ grep Chr1 /bigdata/gen220/shared/data-examples/examples/random_exons.csv
Chr11,14656670,14656778
Chr1,1147485,1147562
Chr12,22130532,22130707
Chr10,19029658,19029760
Chr11,3528895,3530426
Chr12,23125462,23125634
Chr1,4249358,4249468
Chr11,16238576,16239304
Chr12,9264478,9264617
Chr1,18658403,18658693
Chr12,9488597,9489239
Chr1,12152,12435
Chr1,43214981,43215253
```

How can we make this a more specific query? Well we know the ',' comes after
so we can include that in the search.

```
$ grep Chr1, /bigdata/gen220/shared/data-examples/examples/random_exons.csv
Chr1,1147485,1147562
Chr1,4249358,4249468
Chr1,18658403,18658693
Chr1,12152,12435
Chr1,43214981,43215253
```

If you want to invert the search and find lines that DO NOT match the pattern
use the `-v` option.

```
$ grep -c Chr1, /bigdata/gen220/shared/data-examples/examples/random_exons.csv
5
```

```
$ grep -v -c Chr1, /bigdata/gen220/shared/data-examples/examples/random_exons.csv
25
```

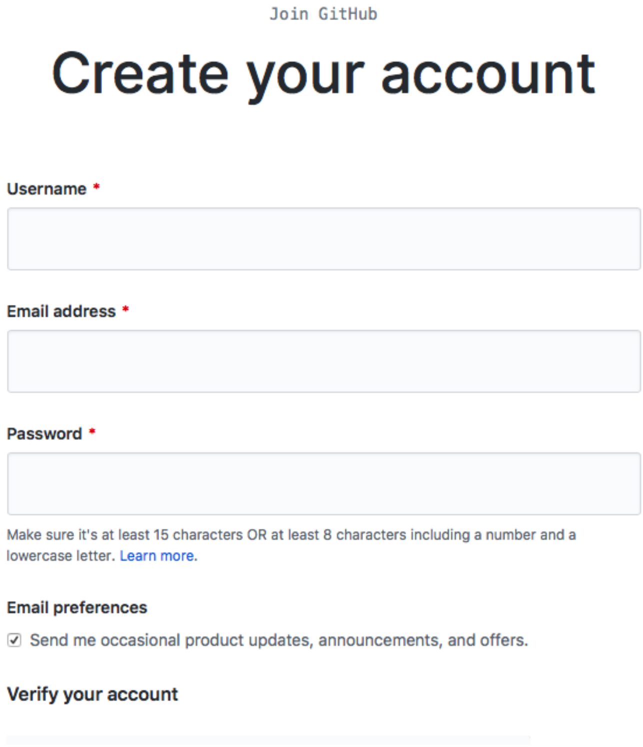
Git and Github

Version control is useful for sharing code, keeping track of versions of software and code (or any text). Distributed version control allows multiple people to work on the same project or code.

Github is a free* resource for code sharing and supports a great deal of the software development among open source projects.

Creating Github Account

https://github.com/join?source=header



Join GitHub

Create your account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

Email preferences

☑ Send me occasional product updates, announcements, and offers.

Verify your account

Figure 1: github

After you create your account - you need to setup SSH keys on your account to simplify check-in and checkout.

You need to add SSH keys to your account and these keys should be

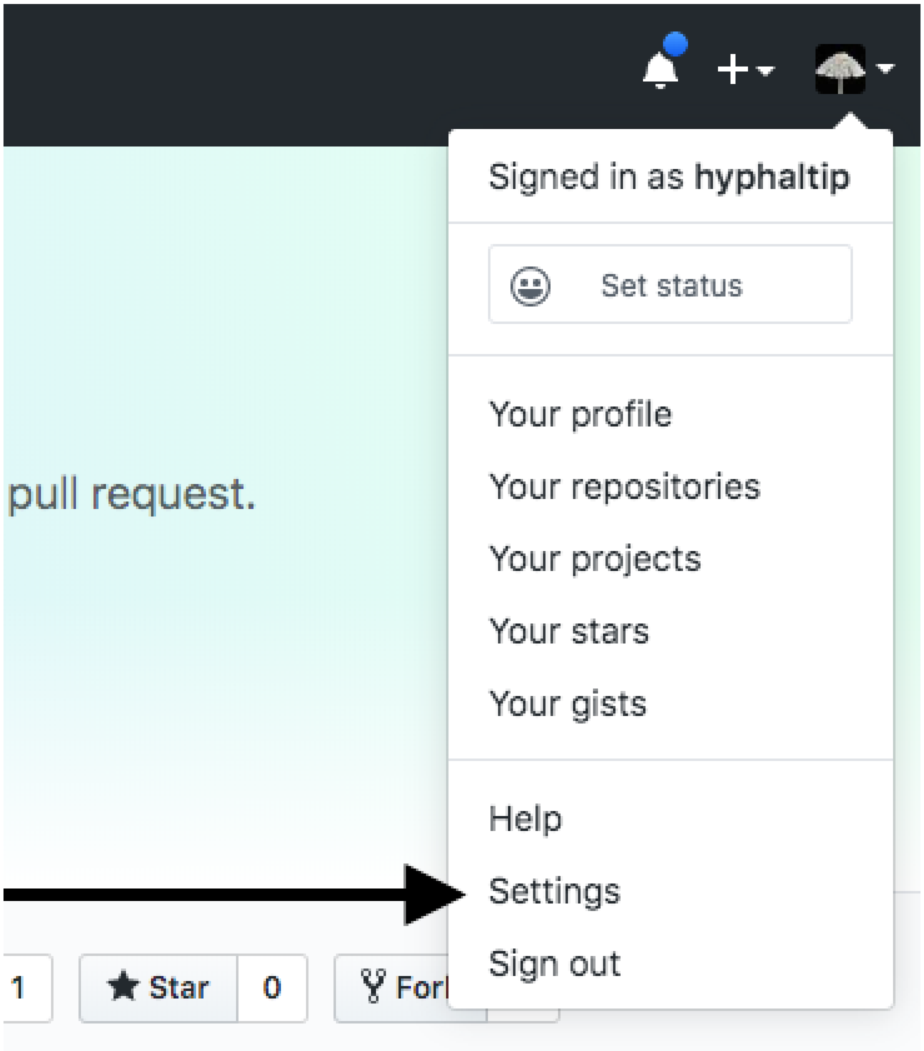
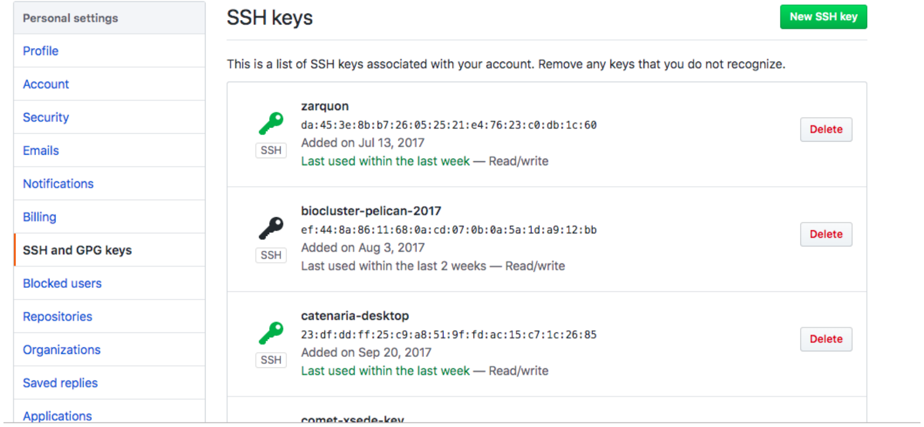


Figure 2: github

stored on the computer you are doing the check outs from (eg the cluster). Follow the directions here https://help.github.com/en/articles/connecting-to-github-with-ssh on how to create key pairs. This provides simple guide * https://help.github.com/en/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent * add the key to your account: https://help.github.com/en/articles/adding-a-new-ssh-key-to-your-github-account

Note that creating these same pairs on your local laptop and copying the public key to your HPCC account. Some basic info is here as well. https://biodataprog.github.io/GEN220/Resources/SSH_keys



SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

zarquon
da:45:3e:8b:b7:26:05:25:21:e4:76:23:c0:db:1c:60
Added on Jul 13, 2017
Last used within the last week — Read/write

biocluster-pelican-2017
ef:44:8a:86:11:68:0a:cd:07:0b:0a:5a:1d:a9:12:bb
Added on Aug 3, 2017
Last used within the last 2 weeks — Read/write

catenaria-desktop
23:df:dd:ff:25:c9:a8:51:9f:fd:ac:15:c7:1c:26:85
Added on Sep 20, 2017
Last used within the last week — Read/write

comet-xsede-key

Figure 3: github

Preparing Homework

Click on piazza links for homework submission:

https://piazza.com/ucr/fall2021/gen220001/resources

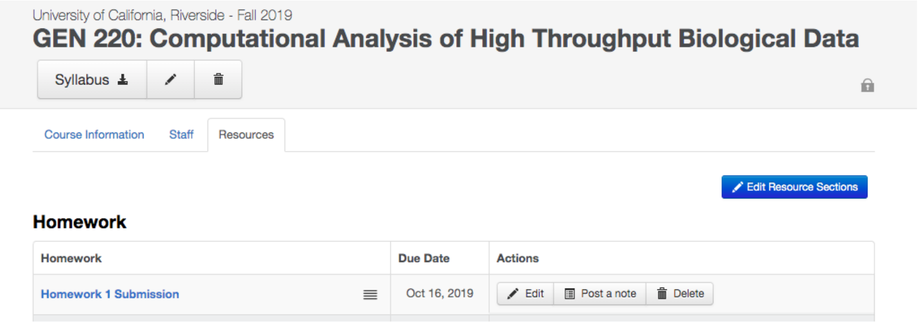
You should link your UCR netID to your github account so I can figure out who has which homework.

Setting up a repository

Click through the links and accept setting up the repository.

Checking out code

Now you have created a repository for your homework. It has been prepopulated with code framework I started for you.



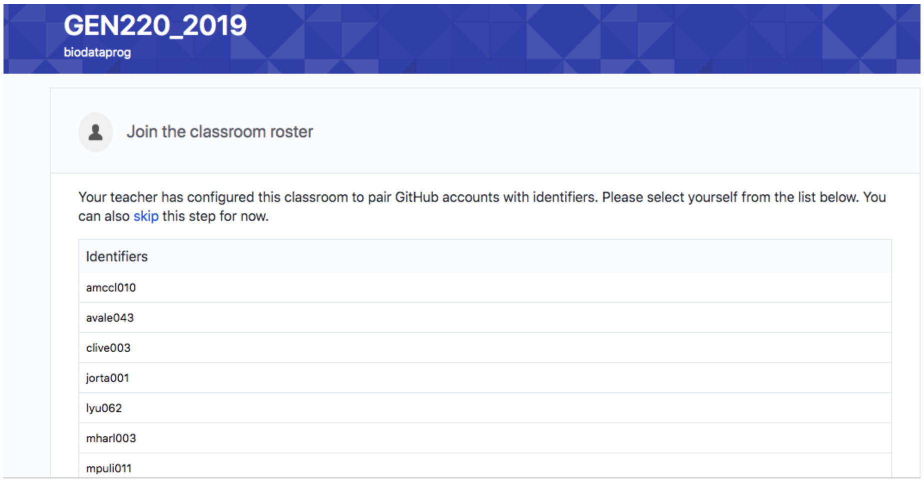
University of California, Riverside - Fall 2019

GEN 220: Computational Analysis of High Throughput Biological Data

Homework

Homework	Due Date	Actions
Homework 1 Submission	Oct 16, 2019	Edit Post a note Delete

Figure 4: piazzahw

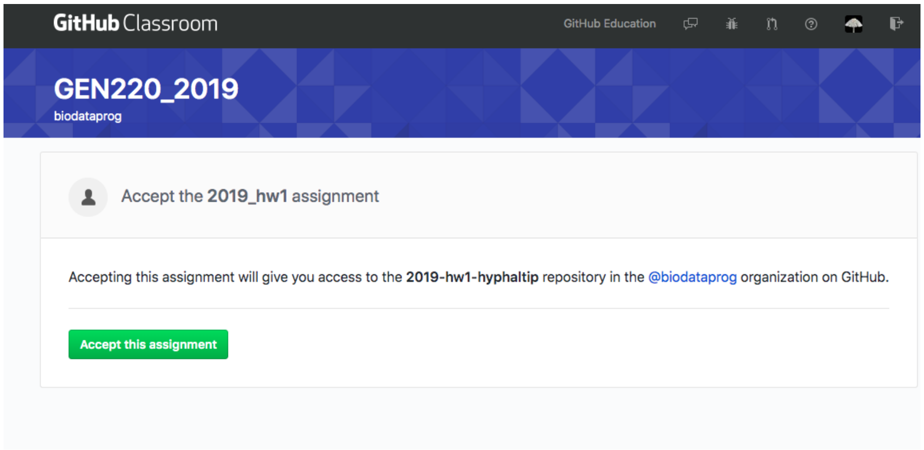


Join the classroom roster

Your teacher has configured this classroom to pair GitHub accounts with identifiers. Please select yourself from the list below. You can also skip this step for now.

Identifiers
amccl010
avale043
clive003
jorta001
lyu062
mharl003
mpuli011

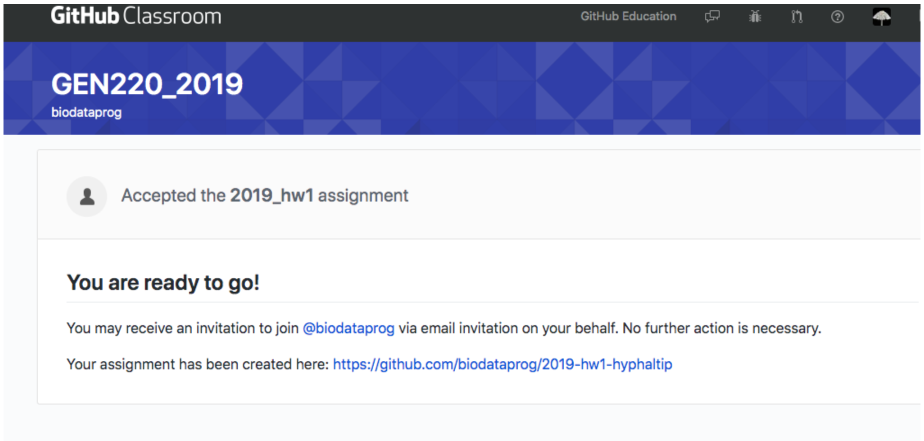
Figure 5: piazzahwlink



Accept the **2019_hw1** assignment

Accepting this assignment will give you access to the **2019-hw1-hyphaltip** repository in the @biodataprog organization on GitHub.

Figure 6: githubrepo



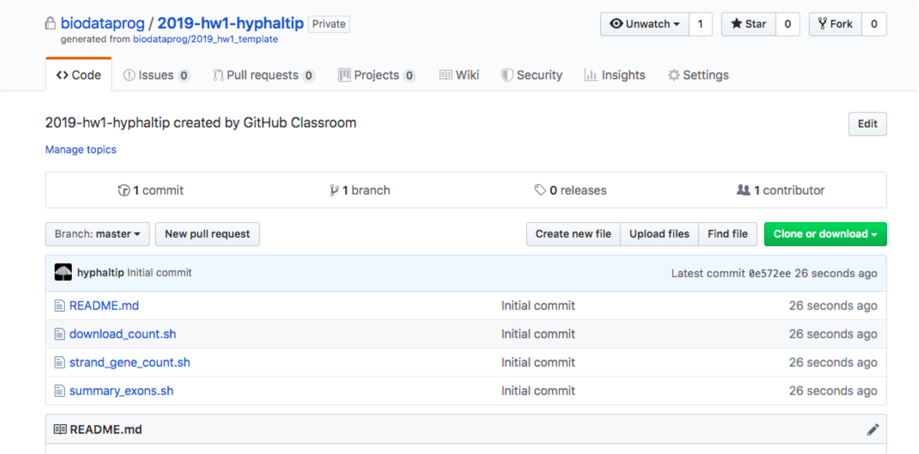
Accepted the **2019_hw1** assignment

You are ready to go!

You may receive an invitation to join @biodataprog via email invitation on your behalf. No further action is necessary.

Your assignment has been created here: https://github.com/biodataprog/2019-hw1-hyphaltip

Figure 7: githubrepo



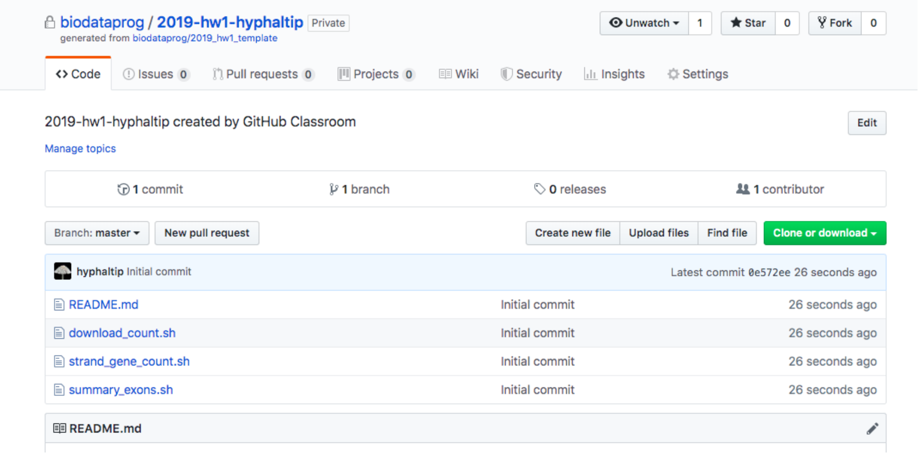
2019-hw1-hyphaltip created by GitHub Classroom

hyphaltip Initial commit — Latest commit 0e572ee 26 seconds ago

README.md	Initial commit	26 seconds ago
download_count.sh	Initial commit	26 seconds ago
strand_gene_count.sh	Initial commit	26 seconds ago
summary_exons.sh	Initial commit	26 seconds ago

Figure 8: githubrepo

You want to check out this repository on the cluster (will also work to check out to your laptop).



biodataprog / 2019-hw1-hyphaltip Private
generated from biodataprog/2019_hw1_template

2019-hw1-hyphaltip created by GitHub Classroom

hyphaltip Initial commit — Latest commit 0e572ee 26 seconds ago

README.md	Initial commit	26 seconds ago
download_count.sh	Initial commit	26 seconds ago
strand_gene_count.sh	Initial commit	26 seconds ago
summary_exons.sh	Initial commit	26 seconds ago

See the link in this window:

Go to your command line (on the cluster and check out your repository - you will be changing YOURUSERNAME to the login you use on github.

```
git clone git@github.com:biodataprog/2021-homework1-YOURUSERNAME.git
```

If you cannot get this to work you can revert to using https but you will need to enter your **Github** username and password each time you want to commit which is annoying. Note the instructions on how to use the Git access token for https connection.

The equivalent would look like this (except for the YOURNAME part) `git clone https://github.com/biodataprog/2021-homework1-YOURNAME.git`

Making changes

Edit changes locally using nano or your favorite editor. When you are done you can commit these changes to the repository with git commit.

```
git commit -m "message" file_changed.sh
```

Git add

If you create additional files to track you can add them to the system. You need to tell Git which files you want to track. This is done with `add`

```
$ git add file1.sh file2.sh data/dat.tab
```

Git commit

To save the changes in the repository you need to commit them. This commit is accompanied by a message with `-m` option

```
$ git commit -m "A helpful message"
```

If you forget to include a message it will prompt you

```
$ git commit
# will spawn an editor for you to write a message
```

Last step - git push

To Sync your code on HPCC (or your laptop) wherever you have a git repository checked out - you still need to save and push these changes to the github "cloud". You can do this by typing

```
git push
```

- you will be asked to enter a username (your git username) and your password - which is the authentication token you generated before. This is effectively a new password that can be more easily thrown away, as compared to your github account password so this increases security of your account. so you create this token (Personal Access Token) and it is a long set of letters and numbers. your username is still your username. So when you go to `git clone`, or `git pull`, or `git push` you will need to put in your username and this token as the password.

So you'll have some notepad you can copy from ready to grab this each time, but it is kind of annoying. To overcome that you can also use the following to cache (eg save) your password in a process that is running on the cluster for a certain period of time. Default is 15 minutes but you can even set that cache time to hours or days or more (but only is valid for while you are logged into that computer I believe).

This explains how you can save your username/password so you don't have to enter the username and password each time. https://stackoverflow.com/questions/5343068/is-there-a-way-to-cache-https-credentials-for-pushing-commits

To get new changes

If you are collaborating on a project and someone else makes changes to the repository, you need to sync their changes with yours. You do this by typing

```
git pull
```

Git resources

More links and helpful tutorial here

https://guides.github.com/activities/hello-world/ from github.