## BE 434/535 Biosystems Analytics

The field of "bioinformatics" is biology plus data science. This course assumes you have some understanding of the Unix command line and some programming experience. At the conclusion of this course, you should be able to:

- Write, test, and document programs in bash and Python
- Use the source code management system Git to version, share, and distribute code
- Use parallelization techniques and hardware (HPC) to run programs faster
- Package and distribute software to create reproducible workflows

## Ocelote (UA HPC)

Given that our class will have students on a variety of operating systems (Windows, OSX, Linux), we will use the HPC (high performance computing) cluster at the University of Arizona for our work.

To gain access to Ocelote, students must:

1) Have a terminal application ("Terminal" or "iTerm2" on OSX, Gitbash on Windows)
2) Sign up for UA's NetID+ (https://netid-plus.arizona.edu/)
3) Be sponsored (https://portal.hpc.arizona.edu/portal/)

Once you have these, open a terminal and type:

```
ssh <NetID>@hpc.arizona.edu
```

You will be prompted for additional authentication for NetID+. If all goes well, you should see something like this:

```
Last login: Sat Jan  5 07:53:30 2019 from ip72-200-123-88.tc.ph.cox.net
This is a bastion host used to access the rest of the environment.

Shortcut commands to access each resource
-----------------------------------------
Ocelote:                El Gato:
$ ocelote               $ elgato
```

Or you may see this (e.g., if you have enabled the menu with the `menuon` command):

```
===============
HPC.ARIZONA.EDU
===============
Please select a target system to connect to:
```

```
(1) Ocelote
(2) El Gato
(Q) Quit
(D) Disable menu
```

Either way, proceed to log in to Ocelote for your work. If you are uncertain which machine you are on, use `hostname`. If you are on the bastion host, it will be "gatekeeper.hpc.arizona.edu." Once you are on Ocelote, the hostname should be something like "login1" or "login2."

## SSH Keys

If you would like to avoid the 2-factor authentication, then copy your SSH private key to the target system like so:

1) On your local machine and `cd ~/.ssh`. If you do not have one, then run `ssh-keygen`.
2) Copy the contents of the `id_rsa.pub` file (the "public" part of your key). If you do not have one, then run `ssh-keygen`.
3) Login to the target system and `cd ~/.ssh`. If you do not have one, then run `ssh-keygen`.
4) Edit the `authorized_keys` file (e.g., with `nano`) and paste in the public key. Save and exit your editor.
5) Set the permissions with `chmod 600 authorized_keys`
6) Test your login from your local machine.

## Git

You will use the "Git" source code management system (https://git-scm.com/) get class materials, maintain versions of your code, and turn in your assignments. Git was originally created by Linus Torvalds, the creator of Linux, for managing the Linux source code.

### Github

Github is a commercially company that hosts Git repositories. We will use their free service to host public repositories, but they make money by hosting private repos. Recently Github was purchased by Microsoft, so they probably will be around at least until the end of the semester. The advantage to use Github (or Gitlab) is that, when you `push` your code to Github, you will have an "off-site" backup. That is, if your computer were to crash, a copy of the repo would still exist on Github's server. It is not necessary to use Github to use Git. You can `git init` and directory you'd like and maintain a repo there. Github has a useful web interface and the ability to add your instructors as “collaborators” allowing us to `push` and `pull` from your repos.

So, step one is to create a Github account and then share your username with your instructors. I suggest you add your public SSH key (see “SSH Keys” above) into your Github “Settings/SSH and GPG Keys” so that you can more easily push and pull into your repositories. You’ll need to add a key from each machine you intend to work from, i.e., both your laptop and Ocelote.

## Forking the course repo

Once you have an account, visit the course repo in a web browser. Create a copy of our repository into your own account by clicking the “Fork” button in the upper-right corner.

```
https://github.com/hurwitzlab/biosys-analytics
```

You can see your Github repo from github.com, but you must use the command line `git` program to check it out, add files, etc. If you are working on Ocelote or any other Unix platform, it’s quite likely that Git is already installed. Check the version like so:

```
[hpc:login3@~]$ git --version
git version 2.2.2
```

If you are on Windows, you will probably find it easiest to install “Gitbash” which will give you a Unix-like command line with `git` and many other Unix utilities (but apparently not `wget`).

### Cloning your repo locally

Now you can `clone` your repository to your machine(s) like so:

```
$ git clone git@github.com:yourusername/biosys-analytics.git
Cloning into 'biosys-analytics'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 16 (delta 1), reused 12 (delta 0), pack-reused 0
Receiving objects: 100% (16/16), 104.71 KiB | 0 bytes/s, done.
Resolving deltas: 100% (1/1), done.
Checking connectivity... done.
```

## Adding course repo upstream

Your version of the repository is a copy of the “hurwitzlab” repo at the time you forked it. To get new content from the “hurwitzlab” repo, you will add the “hurwitzlab” Github repo as an “upstream” repo:

```
git remote add upstream https://github.com/hurwitzlab/biosys-analytics.git
```

When you need to get new content from the “hurwitzlab” repo, `pull` from the upstream repo’s “master” branch:

```
git pull upstream master
```

## Committing your work

There are three git commands you must use to put your files into Github so that they can be seen by others:

- add
- commit
- push

A basic workflow is:

```
$ echo hello > hello.txt
$ git add hello.txt
$ git commit -m 'added hello' !$
$ git push
```

The `-m` argument to `commit` is the commit message. If you don’t specify a message, you will be dropped into your `$EDITOR` to create one.

**If you cannot see your work the Github web interface, then I cannot check it out and grade it.**

# About The Author

> “Computer programming has always been a self-taught, maverick occupation.” - Ellen Ullman

My undergraduate degree was in English and music. I learned to program on my own and on the job starting in 1995, so I started relatively late and wasn’t formally trained in programming until much later in my MS program. I say this so you know that everyone starts somewhere, some later than others, but it’s like the joke “When is the best time to plant a tree? Thirty years ago. When is the second best time? Now.” Now is a great time to become a programmer. I look forward to teaching you what I’ve learned in course of 20+ years of programming in industry and bioinformatics.