Functional Annotation of Metagenomes

In the previous lab we learned how shotgun metagenome data can be used to ascertain the taxonomic structure of microbial communities. However, the real utility of metagenome data is that it encodes information about the functional capacity of these microbial communities. In this lab we will explore the functional capacity of select metagenomes from the human microbiome project.

Objectives

1. Gain an understanding of how functional metagenomics analysis can be conducted
2. Analyze the functional capacity of the three distinct body sites
3. Gain exposure to the databases used to annotate the reads from shotgun metagenome studies
4. Examine functional differences across metagenomes

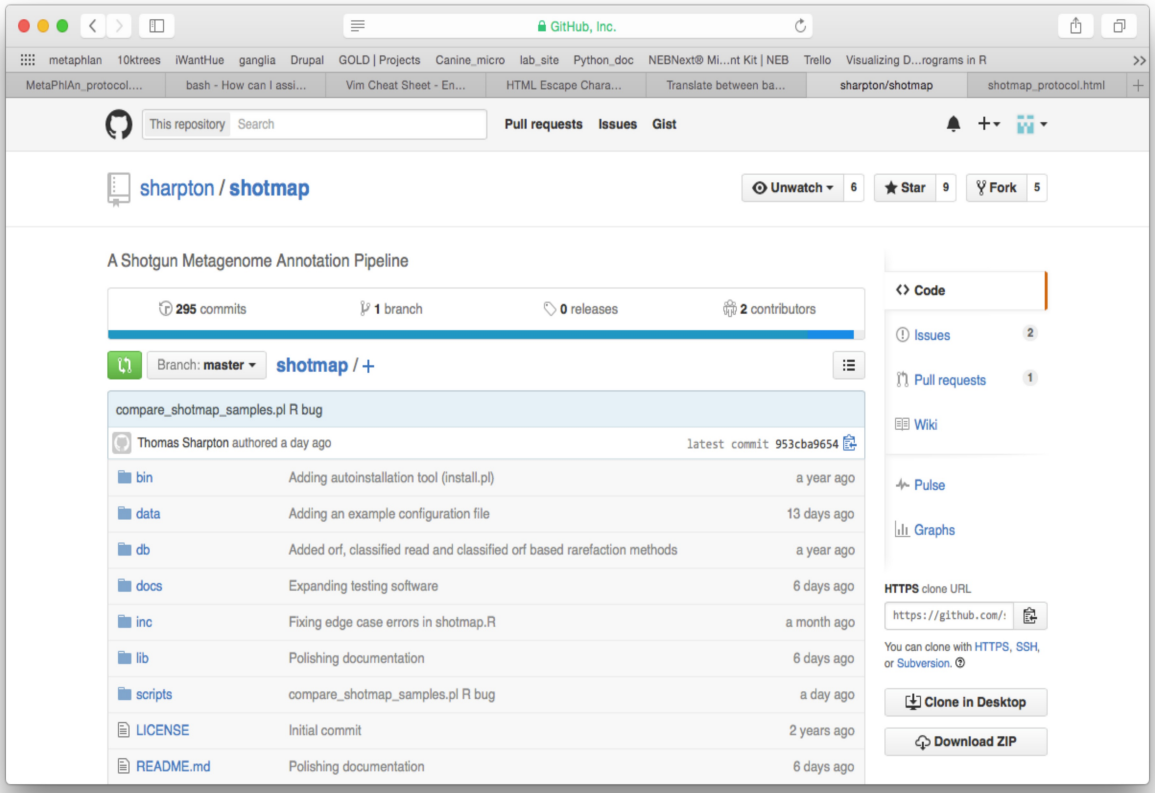
Protocol

There are many tools for annotating metagenomes in this lab we will be focusing on one such tool, ShotMAP. ShotMAP annotates metagenomic reads based on their protein classifications. ShotMAP, like many other bioinformatic algorithms is free and publicly available on GitHub. GitHub is a free software hosting service that allows scientists (and normal folks) to build, modify, and collaborate on software. This, and other services like it (BitBucket, GitLab, etc.), are an important resource for bioinformaticians, data scientists, and pretty much everyone else. Given the importance of this resource lets start by taking a look.

1. Start by getting your environment right

```
$ bash
```

2. Open your web browser (on your PC) and navigate to http://www.github.com/sharpton/shotmap and you will see something like this.



As an bioinformatician much of the software you use will likely be free and hosted on sites like GitHub so it is a good idea to become familiar with the site.

3. Next let's try our hands at pulling down our first git repo. Since ShotMAP is already installed on the server, you will be pulling down a different repository. Navigate to https://github.com/chrisgaulke/mcb525 and take a look at the README.md file (by clicking on it)to make sure you are in the right place.

4. Now we need to set up the a location for the git hub repo, let's call this directory dev.

```
$ cd
$ mkdir dev
$ cd dev
```

5. Next you need to find the location of the git hub repo. You can find this in the right side panel of the git hub repo in a box titled `HTTPS clone URL`. Copy this url and return to the terminal.

6. To pull down the repo all you need to do is enter the command below.

```
$ git clone <repo_address>
```

There should be several files in this repo: a README.md text file, a python script called hi_pi.py, a category file, a sample ID to body site mapping file, and a .shotmap file that we will use to set some environmental variables for use later.

7. For now let's see what hi_pi.py does.

```
$ cd mcb525
$ python hi_pi.py
```

Well most of the repos in Github will be a little more useful than that. Some repos, like ShotMAP, will also require some installation after the repo is pulled down, but that is a conversation for another day.

8. There was one useful item in that git repo, the .shotmap file. As you may have guessed we need to source this file for it to work, but first let's take a look at it.

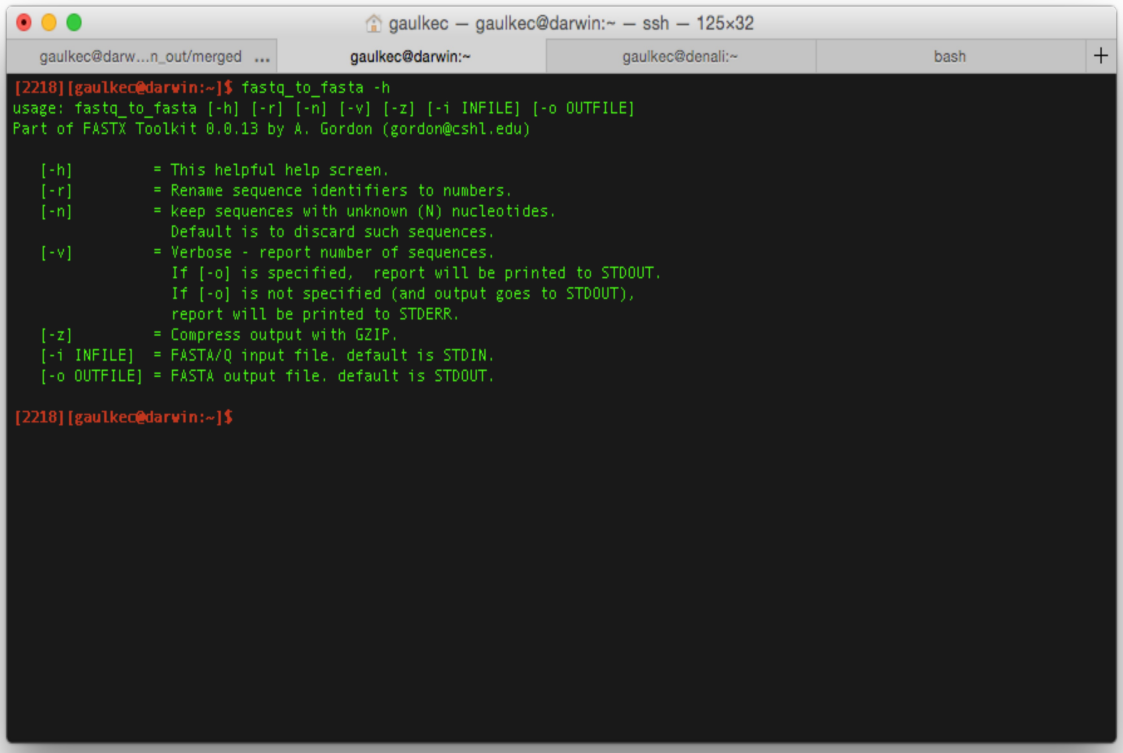
```
$ less .shotmap
$ source .shotmap
```

Now all we have to do is push our files into ShotMAP and we will have the answer to all of our questions... But we have a problem. Recall our files are in fastq format, but ShotMAP expects, no, demands fasta formated files. What do we do? This might seem a bit dramatic but it actually is a frequent annoyance encountered when working with large datasets. We could write our own script to handle this issue or we could look to see if there are any tools out there that have already been written to do the thing we want. Lucky for us, someone has already done the heavy lifting, and even better it is already installed systemwide on the CGRB infrastructure . The converter is cleverly named fastq_to_fasta.

9. Let's take a look at what it wants for input.

```
$ fastq_to_fasta -h
```

You should see something like this.



10. Now that we know what it wants we just need to test it. Lets give it a fastq file and see if it works.

```
$ cd
$ mkdir shotmap_input_files
$ cd shotmap_input_files
$ fastq_to_fasta -i ~/raw_metagenome/SRS011405.fastq
-o test.fa
```

What an error?!?!?! Hmmm... it turns out that this is a very common thing to happen the first time you try to run software. You should not be discouraged (or surprised) when this happens. As it turns out the scoring system used during base calling was different from what this script was expecting. It wanted the quality score (fourth line of the entry in fastq files) to be in PHRED 64 format and we gave it PHRED 33 (the distinction is not really important for our purposes except that it is breaking things). But wait there is no way to change this, or at least it isn't written into the documentation. Alright time to write our own script right? Well actually sometimes there are hidden options (I like to think of them as easter eggs, you know like the Chrono Trigger secret developer room or the Disco party in the Stanley Parable). I will be honest, I found the answer on a forum designed for sequencing nerds (seqanswers.com). A quick google search of the error can often lead you to an answer that will save you hours of scripting or searching for another tool to do the job.

What was the error?

11. Let's try again this time with the option that should solve our problem

```
#Note the -Q 33 option specifies that our quality
score data is in PHRED 33 format
$ fastq_to_fasta -i ~/raw_metagenome/SRS011405.fastq
-o test.fa -Q 33
```

12. Now let's check to make sure we didn't lose any reads along the way. For this we can use the `wc -l` command, which will give us the number of lines in the file. All of the original files should have 800,000 lines which translates to 200,000 sequences (recall there are four lines per fastq entry). So there should be 400,000 lines in our fasta file.

```
$ wc -l test.fa
#The -l option tells wc that we only want it to
return the number of lines
```

But, wait we are missing some reads here. What happened? Turns out the default behavior is to filter reads that have Ns (ambiguous bases) in them. In order to override this default behavior we needed to pass the `-n` option which will force `fastq_to_fasta` to keep all those reads that contain Ns. Let's keep those reads so we will have the same number of reads in each file.

How many sequences are present in this file (**Hint** sequences not lines)?

Note that you should always check the input requirements of the software you are using. It would be a drag if we kept N containing reads if this option would cause problems during downstream analysis.

What does shotMAP want for input (**Hint** you will probably need to take a look at the repo again)?

```
$ fastq_to_fasta -i ~/raw_metagenome/SRS011405.fastq
-o test.fa -Q 33 -n
```

13. Check one last time to be sure we haven't lost data.

```
$ wc -l test.fa
```

How many sequences? What about lines?

Looks good now. Let's apply it across all the samples, but also lets pass the `-z` option to compress the files and save space.

```
$ cd ~/raw_metagenome
$ my_files=$(ls ~/raw_metagenome)
$ for f in ${my_files}
> do
> g=${f/.fastq/.fa}
> fastq_to_fasta -i ~/raw_metagenome/${f} -o $g.gz -Q
33 -n -z
> done &
```

This probably deserves some explanation.

- `my_files=$(ls ~/raw_metagenome)`: Here we are using a bash command, `ls`, to assign the variable my_files the output of `ls`. This is similar to when you assign x = 2 in algebra, except here we are assigning file names instead of numbers. Note when we refer to this variable we us the '$' in front of it to tell bash it is a variable and not just text.
- `for f in ${my_files}`: Here we initiate a `for loop`. Some have mentioned that for loops are a pretty intermediate level programming skill and might be too advanced for this class. Perhaps, but I believe the benefits of being exposed to for loops outweigh the confusion they might cause. Mainly, for loops allow us to process large numbers of things (files in this case) identically (i.e., the output will be uniform and reproducible). If you prefer process each file by hand but be careful to make sure the output of this script is formatted exactly the same. For those of you who are adventurous (or lazy, which is ok as laziness is actually a virtue in some programming circles...I am not kidding.) this part of the command says for each file in the list of files in ${my_files} run the commands that follow. Note each file is assigned to the new variable ${f} one at a time until all the files have been processed. The file is not appended to ${f}, but overwrites it.
- `do`: This tells the computer to do the follow commands.
- `g=${f/.fastq/.fa}`: This command is a bit advanced, but what it is doing is taking the file ${f}, replacing the .fastq ending with .fa, and assigning the new file name toa new variable ${g}
- `> fastq_to_fasta ...`: This is the actual command that is being run for each file in the `for loop` everytime ${f} appears the current file assigned to ${f} is inserted. Similarly, when $g appears the modified file name is inserted.
- `done &` This tells the computer that we are done giving it commands and the `&` put the job in the background

What is in ${my_files} (**Hint** try using echo)?

14. The last thing we need to do is to move these files to the a new directory

```
#Note the mv command might be new to you. This
command is really useful it allows you to move files
and directories around. It also allows your to rename
files and directories.
$ cd
$ mkdir metagenome_fasta
$ mv ~/raw_metagenome/*.fa.gz ~/metagenome_fasta
```

Now that the files are ready let's have a look at the basic operation of ShotMAP.

15. The first step in running ShotMAP is to create a database to align your reads to. Although fully automated, this can be a computationally intensive process. Luckily, we have a copy of the database already and all you have to do to have access to it is source the .shotmap file we pull down in our GitHub repo.

    We will access the repo by pointing ShotMAP to the variable named $sm_database. The variable contains the path to the database.

16. Now that we have the fasta files and database ready we can run ShotMAP. Your input directory will be ~/raw_metagenome (make sure you enter the full path)

    IMPORTANT!!!! Make sure you are putting your job on the correct machine. The instructor and TAs will give your group instructions on how to do this. If you don't follow directions here you will cause a pileup on the server and make your instructor...Displeased.

At this point elect one group member who will run the commands from here forward. We don't have the resources for everyone to run the scripts. However, every group member is responsible for understanding and being capable of executing these scripts. You are all also for answering the questions herein and including the answers in your lab notebook. All group members are also responsible for obtaining and adding the figures generated from this script. Note not all figures need to be included but a at the minimum the figures that correspond to the figures pictured in this workflow should be included.

```
$ cd ~/shotmap_out/
$ perl $SHOTMAP_LOCAL/scripts/shotmap.pl -
i=/nfs1/Teaching/home/<user_name>/raw_metagenome/ -
d=$sm_database -
o=/nfs1/Teaching/home/<user_name>/shotmap_out --
nprocs=8 --prerare-samps=50000 --ags-method=none &>
/nfs1/Teaching/home/<user_name>/shotmap_out/shotmap_e
rr.log &
```

17. Does the above command look like a bunch of gibberish? Well maybe then some explanation of the options is in order.

- `perl`: This is the language that ShotMap is written in and here we are telling the computer to use this language to process the commands in this software.
- `$SHOTMAP_LOCAL`: Recall $SHOTMAP_LOCAL is just a variable that contains the path to the directory containing shotmap.
- `-i`: is the input directory that contains the files we want ShotMap to process. Note shotMAP can also process individual files.
- `-o`: is the output directory where we want ShotMap to put the files.
- `--nprocs=8`: This indicates the number of processors we want to use.
- `--prerare-samps=50000`: This option sets the depth of rarefaction before the reads enter ShotMap. Rarefaction allows the us to compensate for differences in the number of reads between samples by randomly subsampling the data (i.e., it grabs a set number of sequences per file). Typically rarefaction depth is set to a number that is semi-meaningful, here it is necessary to subsample due to time constraints.
- `&>`: sends standard error (i.e., error messages that would normally come up on screen like 'ls: ./rando: No such file or directory') to a file called mcb_shotmap_err.log.txt this is useful for troubleshooting and documenting failed runs.
- `--ags-method=none`: Turns off average genome size normalization (This is not ideal, but it is necessary for this script to run on watson).

18. Broken?!?!?! What happened? Luckily we have asked ShotMAP to redirect STDERR (standard error) to a logfile so we can figure this out.

```
$ less shotmap_err.log.txt
```

19. Right, we forgot that ShotMAP wants fasta files not fastq. We know this is the issue because ShotMAP tells us under the heading "LOADING PROJECT" in the logfile. This time we will point ShotMAP to the directory with the fasta files.

```
#The input directory should be the full path to the
fasta file folder created above
#note we also will need to specify the clobber option
to overwrite the directories shotmap made in the last
run and you will need to specify the paths for input
directories and output directories
$ perl $SHOTMAP_LOCAL/scripts/shotmap.pl -i=
<full_path_to_input_dir> -d=$sm_database -o=
<full_path_to_shotmap_out> --nprocs=8 --prerare-
samps=50000 --clobber --ags-method=none &>
shotmap_err.log.txt
```

What was the name of the directory we created for the fasta files? Why did we create the fasta files?

This will probably take a while (~1hr)

Post Processing

1. Now that our run has finished let's have a look at the results

```
$ ls shotmap_out
```

There should be several subdirectories in this directory including:
- logs
- output
- parameters
- scripts
- A subdirectory for every sample

Take a minute to explore these directories and familiarize yourself with the data structure that ShotMAP creates. For information on what each directory contains see https://github.com/sharpton/shotmap/blob/master/docs/workflow_synopsis.md.

2. Now that we have the annotated data it is time to run some test. For the next step we will need a mapping file. Luckily, ShotMAP has created one for you (you will have plenty of time to worry about making one of you own in the QIIME module), all you have to do is add a column that specifies the body site. The file called body sites from GitHub repo you pulled down maps sample IDs to body sites. It might be worth briefly going over what a tab delimited file is quickly.

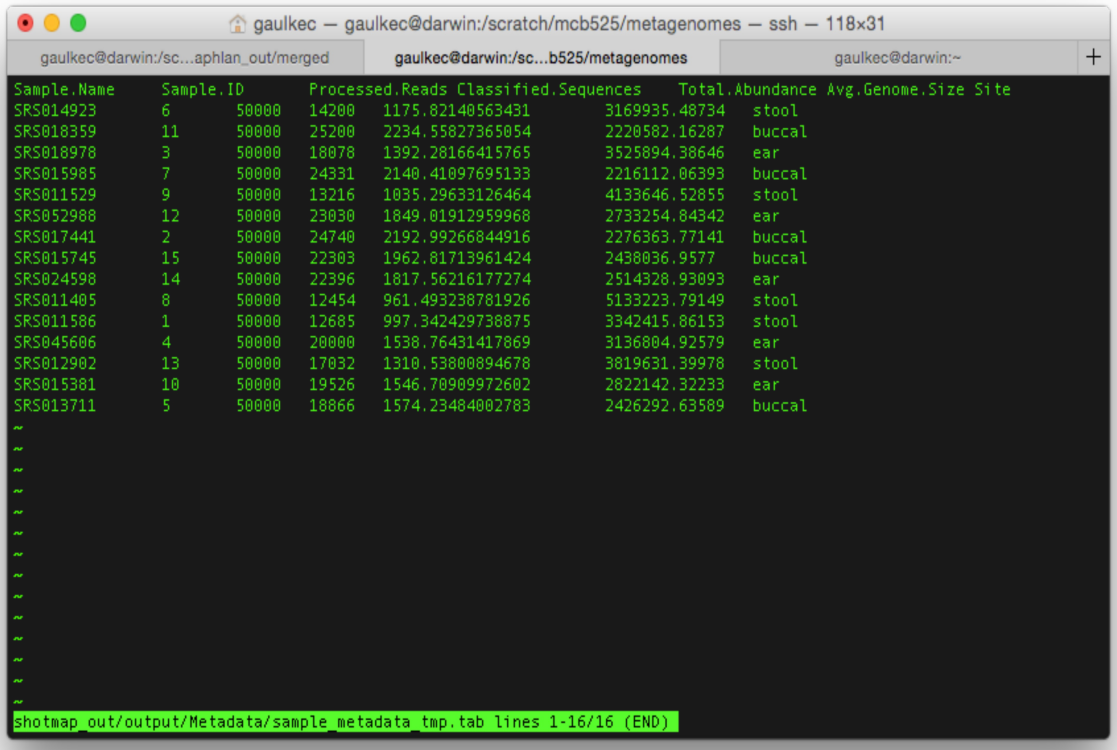
```
#Tab delimited files are files that are composed of
columns separated by a tab as below
SampleID<tab>Barcode<tab>Height<tab>Weight<tab>Diet<t
ab>etc
#The delimiter in this case is a tab, but you can use
other delimiters if you so choose (e.g., spaces,
comma, semicolon, colon, etc.)
```

The file to be modified is `~/shotmap_out/output/Metadata/sample_metadata_tmp.tab`. You can find the sample to site mapping file here ~/dev/mcb525. You will need to enter vim and start

editing mode. Sometimes people find it helpful to open two terminal windows for this type of exercise, one for the sample_metadata_tmp.tab file and one for the sample to site mapping file which will be found in the github repo you cloned earlier.

```
$ vim
~/shotmap_out/output/Metadata/sample_metadata_tmp.tab
```

Then type `a` to enter edit mode. Now you can edit the file.After you are done it should look something like this (the sample names will likely be in a different order).



Once you are complete you can exit vim a number of ways. First hit `esc` then enter one of the following: `q!` then `enter`(this will exit **without** saving), `wq` then `enter` (will save the changes to the file and then will exit) or `w` then `enter` then `q` then `enter`. I prefer `wq` then `enter`.

Remember this should be a tab delimited text file so be sure that when you add the sites column you add a tab first. If you aren't sure if you put tabs in or not you can use vim to check.

Enter vim by typing `vim` the type `:set list` then hit return. Now you can see that each `tab` is displayed as an `^I` and each newline is an `$`.

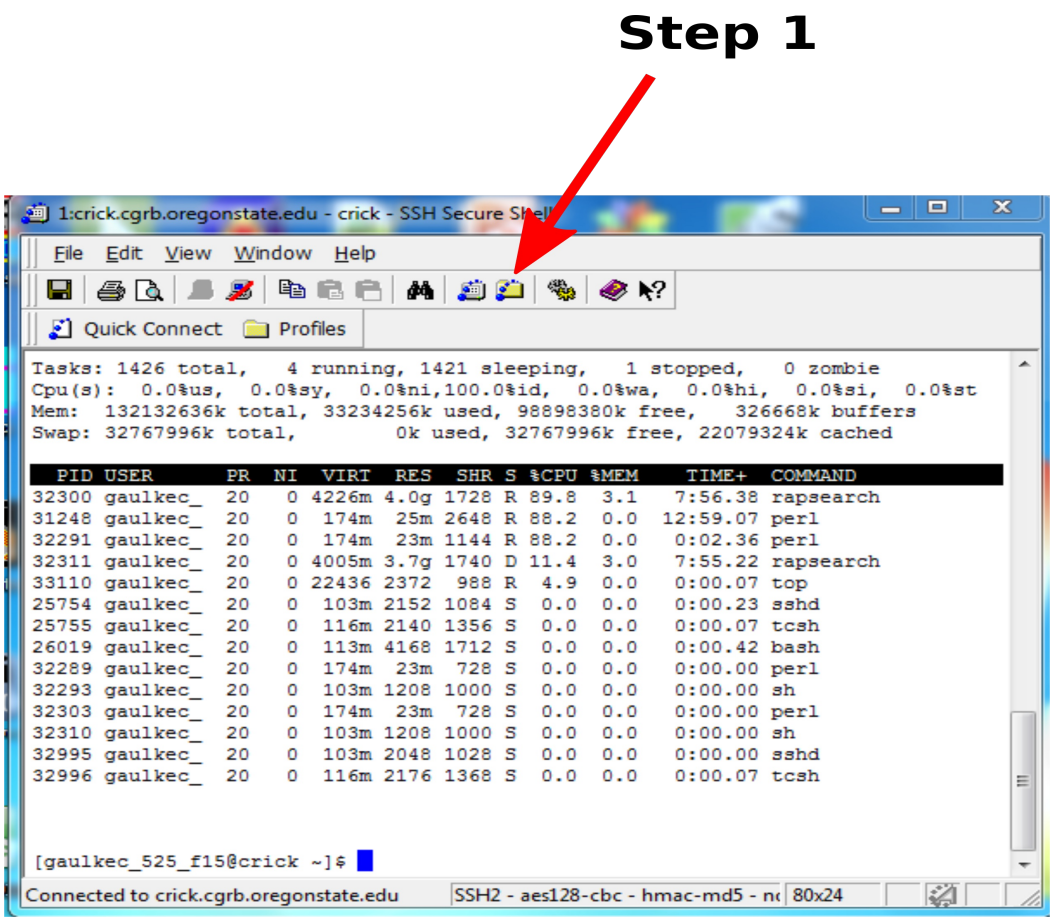
3. Now that we have everything in place we can run the compare script.

   IMPORTANT!!!! Remember you must specify full paths to the input database here and the field category file. Where is the category file located (think github)?

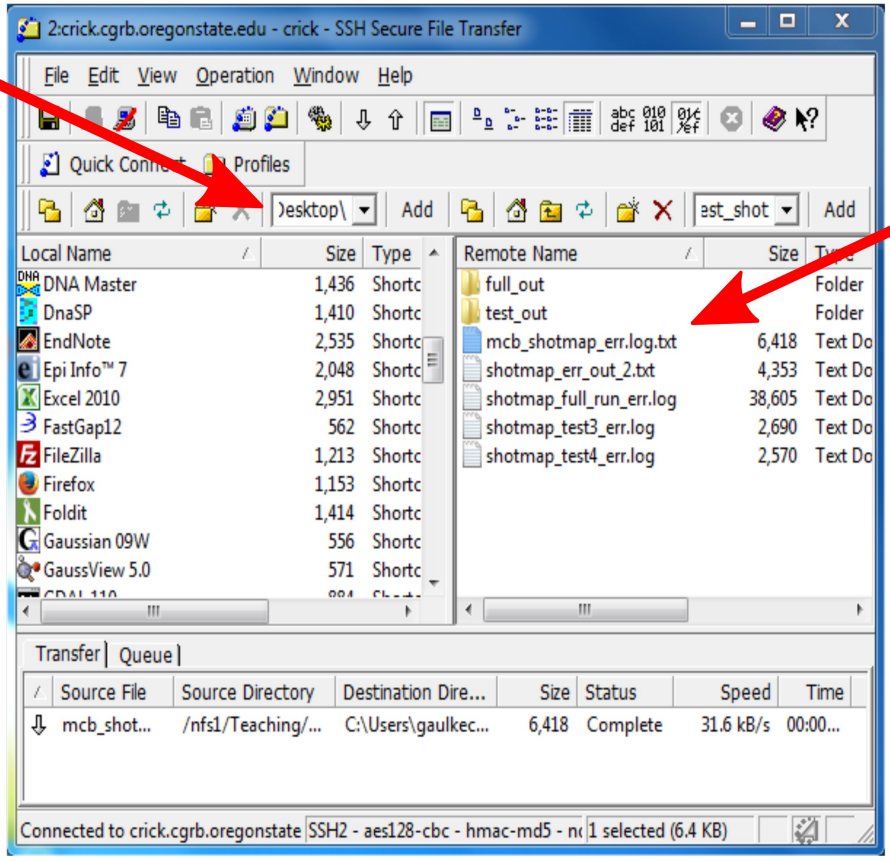
```
$ cd ~/shotmap_out
$ perl
$SHOTMAP_LOCAL/scripts/compare_shotmap_samples.pl -
i=/nfs1/Teaching/home/gaulkec_525_f15/shotmap_out/ -
m=sample_metadata_tmp.tab -o=compared_samples -
d=counts --cat-fields-file <path_to_cat_file>
```

4. This is going to produce a lot of output and in order to see any of it we are going to need to pull this data down.

    - Step 1: Click on the open new file transfer window icon.
    - Step 2: Set the location you want to download the files.
    - Step 3: Select the directory you want to download, right click and select download file.

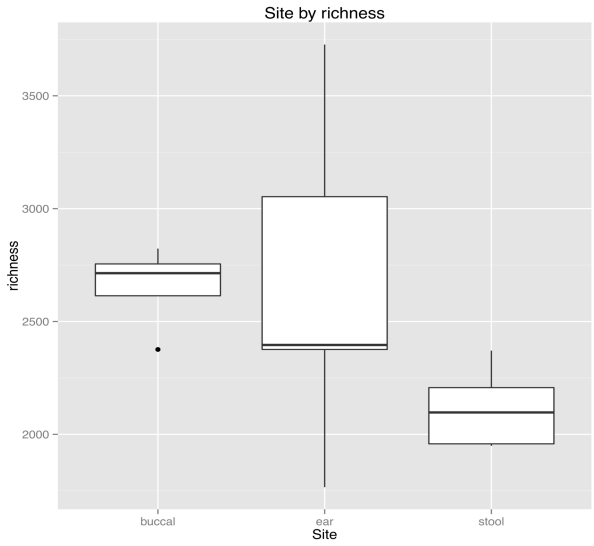


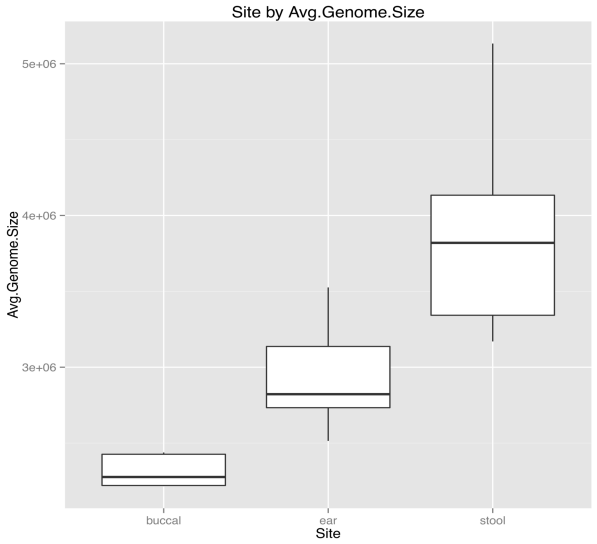
Step 2

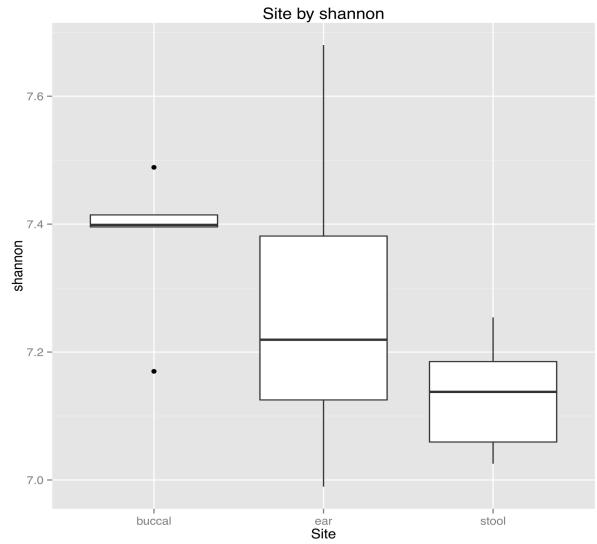


Step 3

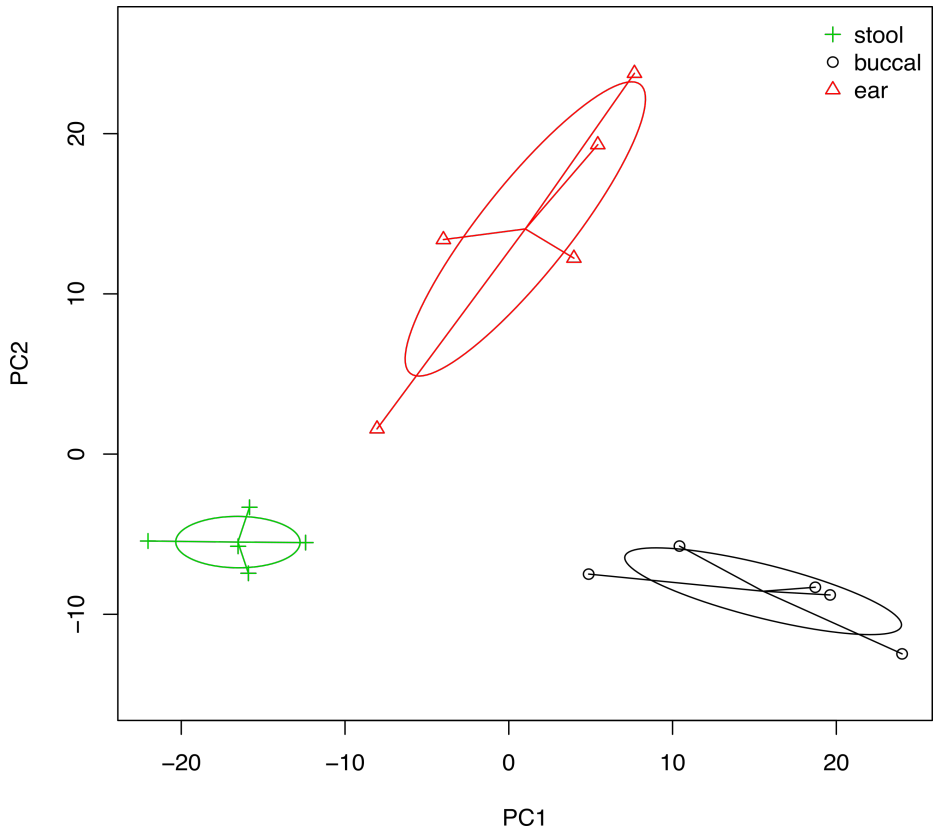
5. Let's take a look at some of the data. Now that the data is local (i.e., on you machine) you can simply click on the indicated files and they will open. Start in the alpha diversity directory and take a look at the alpha diversity stats first this should be located in a file called categorical_kwtests.tab. It seems like the communities are stratified in terms of alpha-diversity across nearly every parameter. Let's take a look at some of the plots.

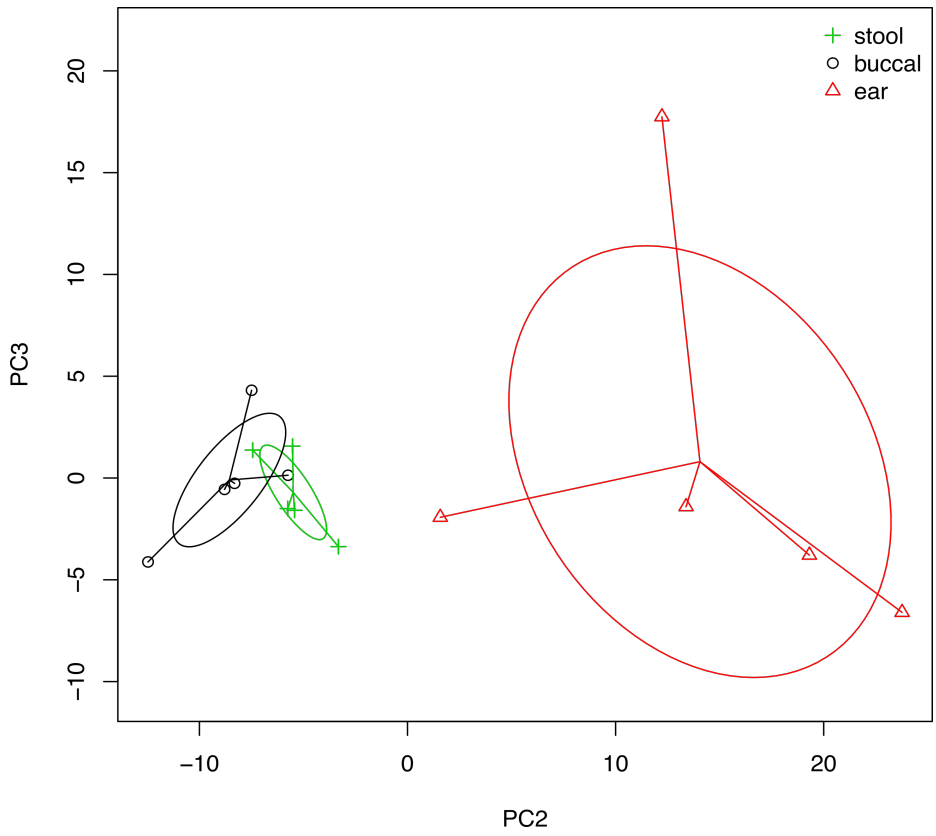






22. Next let's take a look at the beta-diversity. This will be in the directory titled beta diversity.





It looks like there these body sites are also stratified by beta-diversity. There are a lot of other plots in this directory. Take a minute to look over these plots. Do all the graphs look the same? Which plots show the most seperation in the data by group? Which show the least?

23. Finally, since there are clear differences in beta-diversity between the three body sites let's have a look at the specific protein families that stratify these sites. The file we want to look at is in the Families directory and should be named kw-Site.tab. Find a protein family that looks interesting to you. This family should have a good raw p-value (rawp; $p < 0.05$) and a good corrected p-value (BH or qvalue; $q < 0.1$). If you are

feeling indecisive just pick K00027

What protein family did you choose? Why?

Until now we really have no idea what these protein family IDs actually correspond to, but there is an easy way to find out. Here we are going to us the KEGG API to find out the identity and functions of the protein family you selected. The KEGG API is an internet interface for the KEGG database that allows to rapidly query the database from a web browser and the command line (if we so choose)

24. Navigate to http://rest.kegg.jp/get/<your_ko_id>.

    If you wanted to download this file from the internet what command would you use? Think back to yesterday?

    For K00027 it appears that this protein family is involved in three pathways. We can also take a look at the other members of these pathways by using the same get command, but replace the KO ID with the pathway ID.

    What might the protein family's pathway membership tell us about the functional capacity of different microbial communities?

25. Navigate to http://rest.kegg.jp/get/koXXXXX. This page will give you information about the pathway, the proteins in it, and lots of other information. If you prefer an image view Navigate to http://rest.kegg.jp/get/koXXXXX/image

    Include a pathway image in your lab notebook and a justification why you chose that pathway. Also include 2-3 other protein families that belong to this pathway? Do you think protein families or pathways are more informative physiologically? Why?

    Covering all aspects of the KEGG API or KEGG database are beyond the scope of this lab, but if you are interested in learning more check out http://www.kegg.jp/kegg/rest/keggapi.html, and http://www.genome.jp/kegg/.