Git Bash (Shell): To communicate with servers

$ pwd: print work directory (where you are)

$ ls (list of files in directory)

$ ls –F (show which is folder and which is file)

$ cd Pictures (change directory)

Use Tab to make shell finish your directory and sentences

$ mkdir ususwc (to create folder in your directory)

$ nano mynewfile.txt to create and edit text files

$ cp data.txt new\_data.txt (copy)

$ mkdir temp

$ mv new\_data.txt temp/

$ cd ../ go up one level

Mv (to rename files)

Cat data\_greencanyon\_2013.txt (to see what’s inside)

$ Sort

$ sort --help (to access help)

$ sort data\_greencanyon\_2013.txt –k 3 –n (-k to sort based on 3rd col, -n to sort numerically)

Use > to put the output in a different file, use >> to append to a file

$Tail -1 sorted\_counts.txt (give me the last row only) -3, -5, …

$history

$ !42 (line)

$ ctrl+r to re-write a command

| take an output and pass it to the right

Sort data\_greencanyon\_2013.txt –k 3 -n| tail -1

To find the data row with the second most individual, using pipes, sort, head and tail

$sort data\_greencanyon\_2013.txt –k 3 –n |tail -2|head -1

To read a program already written on python:

$python species\_counts.py data\_greencanyon\_2013.txt |sort –k 2 –n > output.txt

$ ls \*.txt or $ls data\* to see files

$ cat data\* (show all in data\* files)

$ cat data\* |python species\_counts.py | sort –k 2 –n

**For Loop:**

$for datafile in data\*

> do

> echo $datafile (print the name out)

>python species\_counts.ps $datafile (do this for each data file starts with data)

>done

On a nano text editor, enter the code and save as species\_count\_all\_files.sh

$ bash species\_count\_all\_files.sh

**Git – Version Control**

$ git init spring\_plants (create a git file)

$ cd spring\_plants

$ git status (how the commands alter the repos)

$ nano plant\_list.txt

Creating a file – untracked

$ git add plant\_list.txt (getting ready to add to the repository)

$ git congif --global user.name “Ayman Afifi”

$ git congif --global user.email “Ayman Afifi”

$ git commit plant\_list.txt –m “initial commit. Add plant list to repo”

**Working to staging: Add. Staging to Repo: Commit**

$ git log (to see changes by whom?)

$ git mv (to rename a file or move it)

$ git comit –m “rename site.txt to …” (only one file to commit, no name to name it)

$ git add . (add all files)

$ git rm (remove from repo)

$git diff (spot differences between versions)

$git log (to see all versions)

$ git checkout aadsada41123 (enough numbers)

$git checkout master (to go back to the original file)

**Recovering previous file states:**

After modifying a file, git checkout plan\_list.txt is what you use to revert a modification, but there is no way to go back to the original file because it is not staged.

Therefore, always stage a file first (add)

Once added, use reset

DAY 2:

Python

Download files and save to python file in ususwc

In the Bash navigate to the folder where the files are saved and type

$ ipython notebook

**SQL**

Ctrl+; to run

Select year from surveys;

Select DISTINCT year From surveys (only distinct years no repetition

SELECT year,month,day,wgt/1000.0 FROM surveys; (1000.0 to divide over float and keep)

SELECT \* FROM surveys WHERE species="DM";

SELECT \* FROM surveys WHERE year>=2000;

SELECT \* FROM surveys WHERE (year>=2000) AND (species="DM");

SELECT \* FROM surveys WHERE (species="DM") OR (species="DO") or (species="DS");

SELECT day, month,year, species, wgt/1000.0 as kg FROM surveys WHERE (plot=1) AND (kg>0.075);

SELECT \* FROM surveys WHERE (year>=2000) AND (species IN ("DM", "DO", "DS")); #instead of OR

Sorting:

SELECT \* FROM species ORDER BY species; or ASC for ascending

SELECT \* FROM species ORDER BY species DESC; (descending order)

SELECT \*

FROM species

WHERE species IS NOT NULL

ORDER BY species; [where there is data only]

SELECT COUNT(\*) FROM surveys; count raws including NULL

SELECT Count(wgt) FROM surveys; excluding NULL

SELECT sum(wgt) FROM surveys;

SELECT round(sum(wgt)/1000.0,2) as kg FROM surveys;

-- my SQL: (comment line)

SELECT species, AVG(wgt)

FROM surveys

GROUP BY species;

**-- Join tables:**

SELECT \*

FROM surveys JOIN species

ON surveys.species = species.species\_id;

SELECT surveys.year, surveys.month, surveys.day, species.genus, species.species

FROM surveys JOIN species

ON surveys.species = species.species\_id;