AGe

Introduction to Linux and PBS server

Emile R. Chimusa
(emile.chimusa@uct.ac.za)
Division of Human Genetics
Department of Pathology
University of Cape Town

AGe

Outlines:

1. Ubuntu.
2. Getting started.
3. Transferring files.
4. Streamlining data manipulation.
5. Shell Scripting.

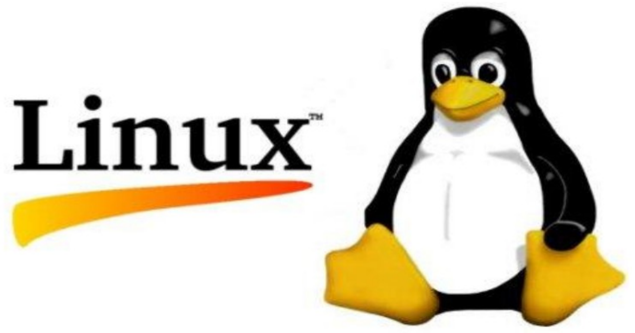
AGe

Assessment:

AGe

Overview of Linux

- Linux refers to a family of operating systems modeled off of Unix
- Can perform many of the same functions as Windows or OS X
- Built in a collaborative, open-source environment
  - Anyone may use, modify, or distribute the Linux kernel
  - Anyone can develop software to run on the Linux kernel
  - Many programmers collaborate to develop or improve Linux programs
  - Many Linux operating systems and add-on programs are free

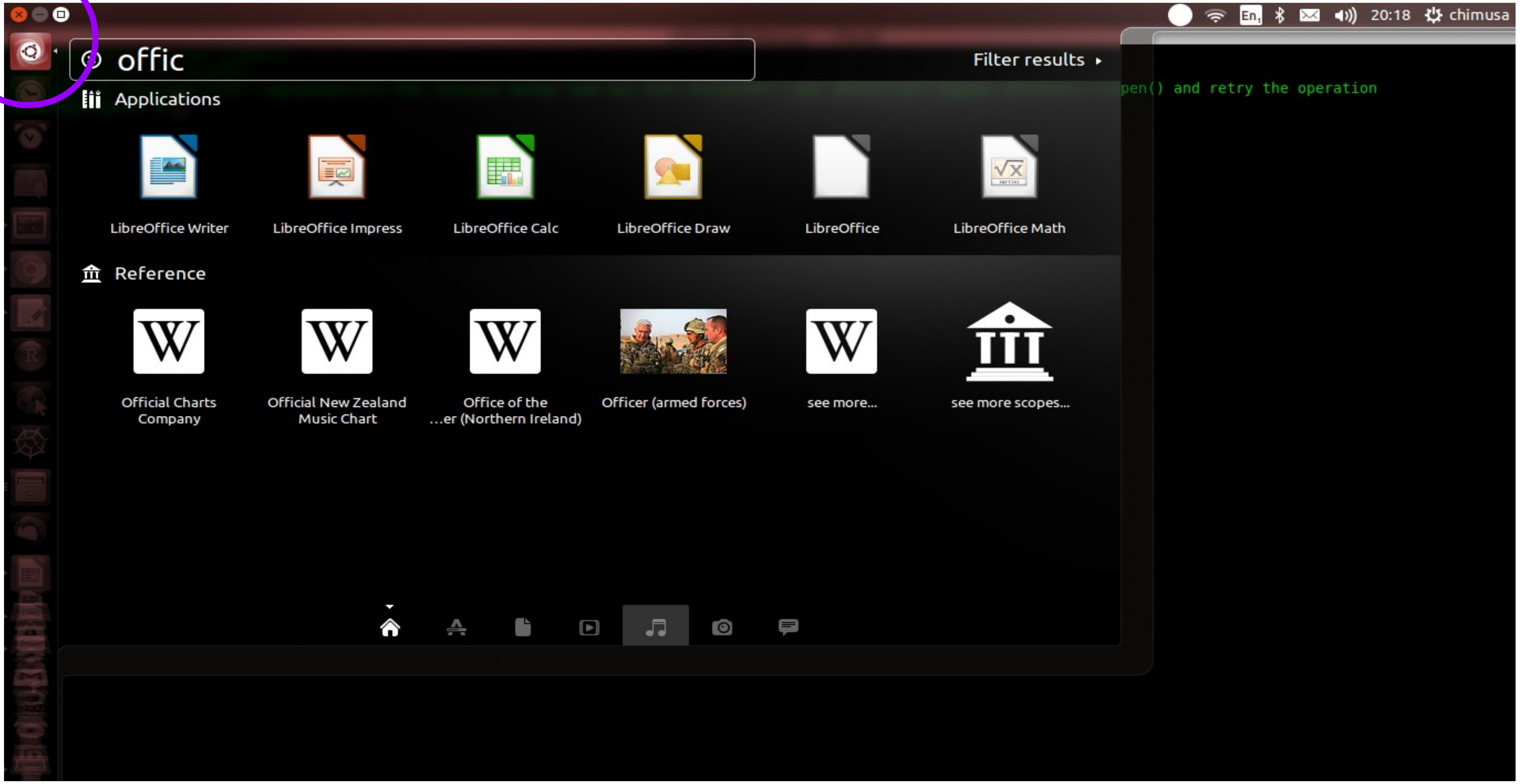


Source: http://scienceblogs.com/gregladen/category/technology/linux/

AGe

I.Ubuntu: Ubuntu Desktop

T.he default Desktop in Ubuntu is **Unity**, but what we are seeing is known as the **Gnome** Desktop. Ubuntu has an built-in Office Suite of programs which you can access from the **Dashboard**



```
pen() and retry the operation
```

AGe

I.Ubuntu: Ubuntu Desktop

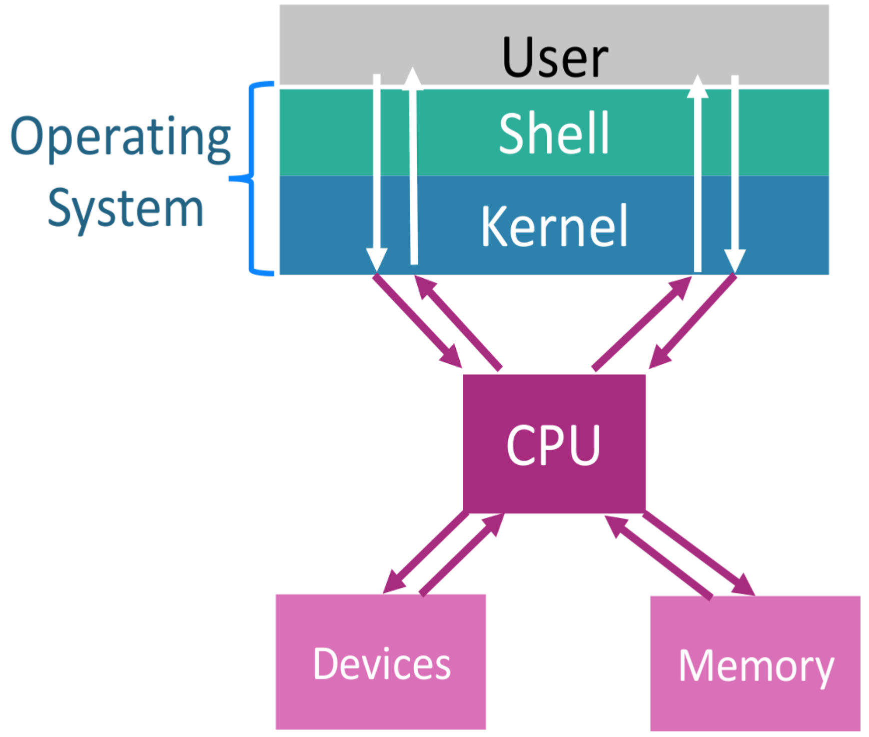
Command-line tools are the mainstay of analysis of large data sets. Good candidate examples for command-line analysis are:

Files that are too large to open on your computer using a text editor or Excel
Manipulations of data which are repetitive or laborious to perform manually
Any analysis that is different from what is available in programs with graphical interfaces.

AGe

I. Ubuntu: Similarities to Windows

- A kernel is the core component of an OS
  - Windows operating systems have kernels, but since they are not open-source or packaged separately for programmers to build off, they are less-often discussed
- Manages system resources (Memory, Processes, Input and Output Devices)
- When a user does something in the shell (the OS's user interface and applications), the kernel translates the command and prioritizes it against other requests for resources, so that it can be understood and executed by the CPU



AGe

I. Ubuntu: More Advantages

- Cheaper
  - Because Linux is open-source, there is often no cost for flavors or their applications
  - Still many more Windows programs because programmers have a financial incentive to develop them
- Graphical User Interface (GUI) can be changed on Linux
  - Windows and Linux both have GUIs to make it easier for users to navigate them
  - Linux menus, icons, and other components can be more customized than Windows ones
- Some tasks can only be run in the command line
  - Command line is a GUI alternative that allows users to execute commands by text
- Less malware on Linux
  - Since most organizations still use Windows, those systems are more lucrative targets for attackers
  - The open-source Linux community sometimes develops and distributes patches more quickly than Microsoft and Apple
- Certain hardware is not built to work with Linux
  - Hardware developers are not as likely to make their products compatible with Linux, as many more people use Windows

AGe

I. Ubuntu: The root account

- Account types: User and Root
- Root – the Linux Administrator account
  - Like the built-in Administrator in Windows, Linux comes with a built-in root account
  - A system can have multiple root accounts
  - Users can switch whether their actions are carried out as a user or root
  - When someone enacts root permissions, he or she can access all of the files and run all commands on a system, as well as set policies for other users
- Root actions require a password in both GUI and command line
- Authentication vs. Authorization
  - Root users are authorized to do many different tasks, but they must first authenticate their identity by entering their password

Source: http://eswalls.com/wp-content/uploads/2014/01/i-am-root.png

Source: http://www.cyberciti.biz/faq/authentication-vs-authorization/

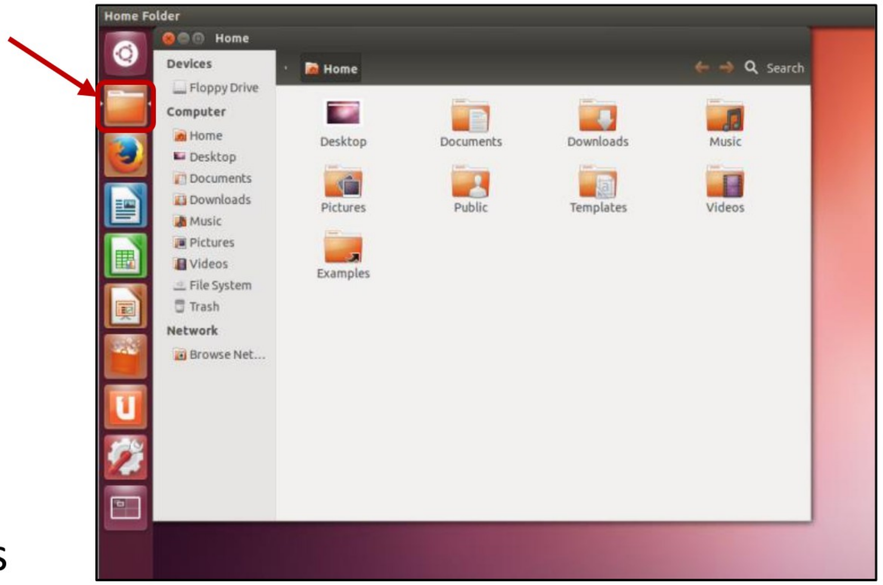
To be user user (a root/sudoer) tape on terminal :

>$ sudo **OR** **>$ sudo su**

AGe

I.Ubuntu: File System

- Different than the Windows file system
  - Does not specify on which drive a folder is stored and uses forward slashes (/) to identify root directories
- Example:
  - Windows: C:\Documents\hello.txt
  - Linux: /home/CyberPatriot/hello.txt
- Important folders:
  - /home: stores each user's documents, media files, etc. Users can only access their own folders, unless they have enacted root permissions
  - /boot: contains startup files and kernel files. Should not be modified unless you are an expert user.
- The file system can be accessed by clicking the orange folder on your Ubuntu menu bar



AGe

I.Ubuntu: File System

/bin	→	contains several useful programs that can be used by the root user and standard users. For example, the ls program is located in /bin.
/boot	→	contains files used during startup.
/dev	→	contains files that represent hardware components of the system. When data is written to these files it is redirected to the corresponding hardware device.
/etc	→	contains system configuration files.
/home	→	the user home directories.
/initrd	→	information used for booting.
/lib	→	software components used by many different programs.
/lost+found	→	files saved during system failures are stored here.
/misc	→	for miscellaneous purposes.
/mnt	→	a directory that can be used to access external file systems, such as CD-ROMs and digital cameras.

AGe

I.Ubuntu: File System

/opt	usually contains third-party software.
/proc	a virtual file system containing information about system resources.
/root	the root user's home directory.
/sbin	essential programs used by the system and by the root user.
/tmp	temporary space that can be used by the system and by users.
/usr	programs, libraries, and documentation for all user-related programs.
/var	contains files to which the system writes data during the course of its operation.

AGe

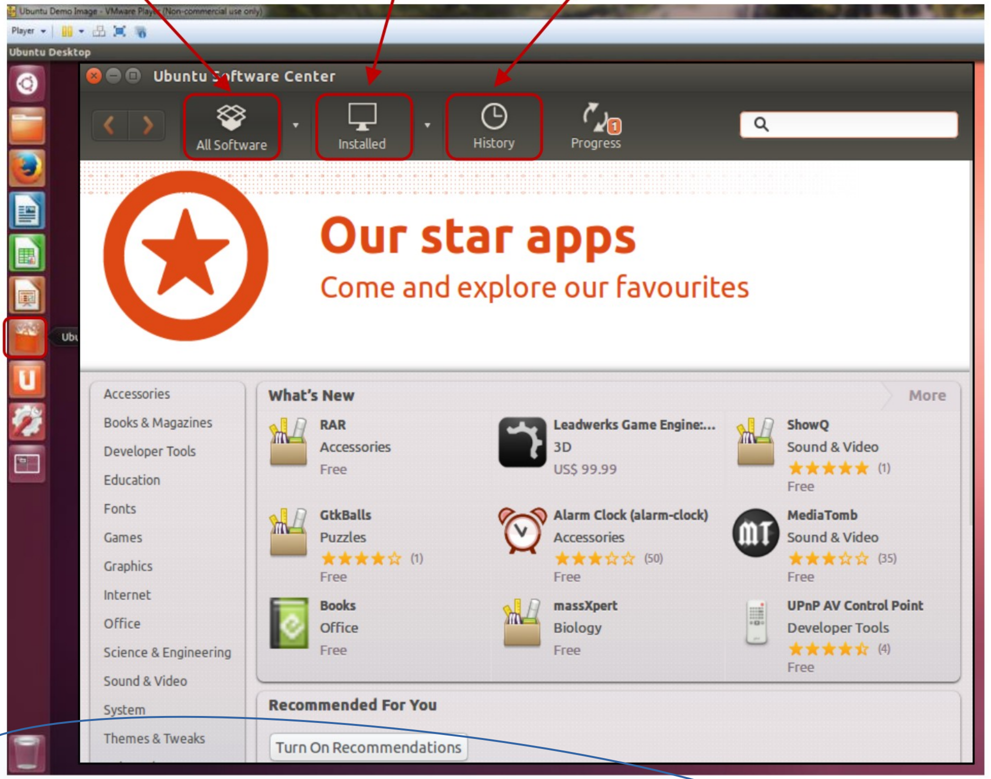
Ubuntu: Add/Remove Software

- Linux software is bundled into packages
- Packages are managed by package managers
  - In Ubuntu, the package manager is called "Ubuntu Software Center."
  - It looks and functions a lot like Mac's App Store
  - Many programs are free
- To access Ubuntu Software Center, click the shopping bag on your Ubuntu menu bar
- Users must enact root permissions to install, uninstall, or modify software.

Use to find and download free and pay-for-use software

Use to manage or uninstall software you have already installed

Use to view a log of all the recent software installs, removals, and updates on your system



```
> sudo apt-get update
```

```
> sudo apt-get install NAMESOFTWARE
```

AGe

I. Ubuntu: Add/Remove Software

1. Installing software

```
>$ sudo apt-get  install name_software
```

2. Updating software and system

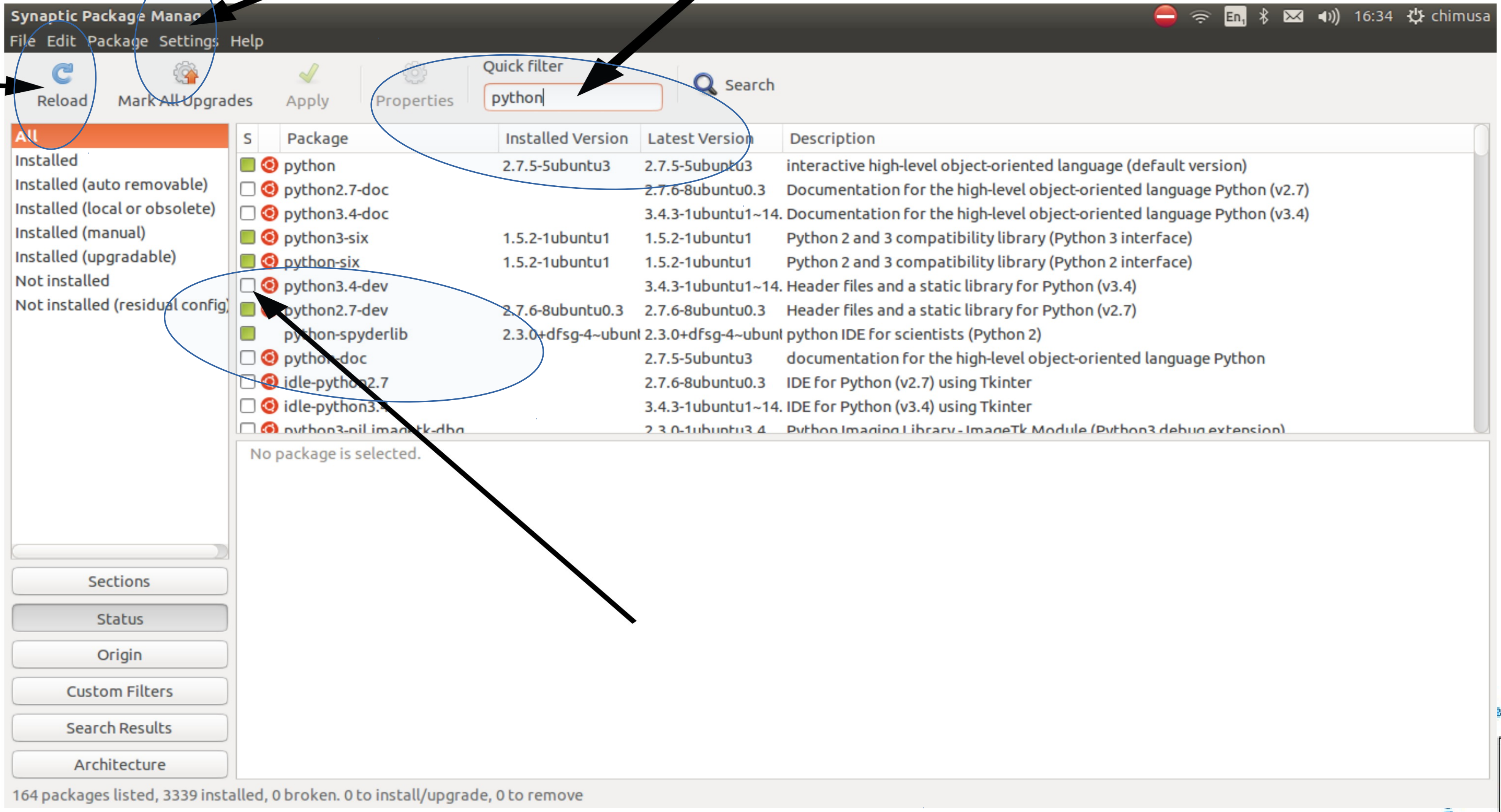
```
>$ sudo apt-get update
```

3. Removal of software

It is great and more safe to install from Ubuntu synaptic manager software that come with Ubuntu or from the setup repository.

AGe

I.Ubuntu: Add/Remove from Synaptic mnager



Quick filter: python

S	Package	Installed Version	Latest Version	Description
🟩	python	2.7.5-5ubuntu3	2.7.5-5ubuntu3	interactive high-level object-oriented language (default version)
☐	python2.7-doc		2.7.6-8ubuntu0.3	Documentation for the high-level object-oriented language Python (v2.7)
☐	python3.4-doc		3.4.3-1ubuntu1~14.	Documentation for the high-level object-oriented language Python (v3.4)
🟩	python3-six	1.5.2-1ubuntu1	1.5.2-1ubuntu1	Python 2 and 3 compatibility library (Python 3 interface)
🟩	python-six	1.5.2-1ubuntu1	1.5.2-1ubuntu1	Python 2 and 3 compatibility library (Python 2 interface)
☐	python3.4-dev		3.4.3-1ubuntu1~14.	Header files and a static library for Python (v3.4)
🟩	python2.7-dev	2.7.6-8ubuntu0.3	2.7.6-8ubuntu0.3	Header files and a static library for Python (v2.7)
🟩	python-spyderlib	2.3.0+dfsg-4~ubunt	2.3.0+dfsg-4~ubunt	python IDE for scientists (Python 2)
☐	python-doc		2.7.5-5ubuntu3	documentation for the high-level object-oriented language Python
☐	idle-python2.7		2.7.6-8ubuntu0.3	IDE for Python (v2.7) using Tkinter
☐	idle-python3.4		3.4.3-1ubuntu1~14.	IDE for Python (v3.4) using Tkinter
☐	python3-pil.imagetk-dbg		2.3.0-1ubuntu3.4	Python Imaging Library - ImageTk Module (Python3 debug extension)

No package is selected.

164 packages listed, 3339 installed, 0 broken. 0 to install/upgrade, 0 to remove

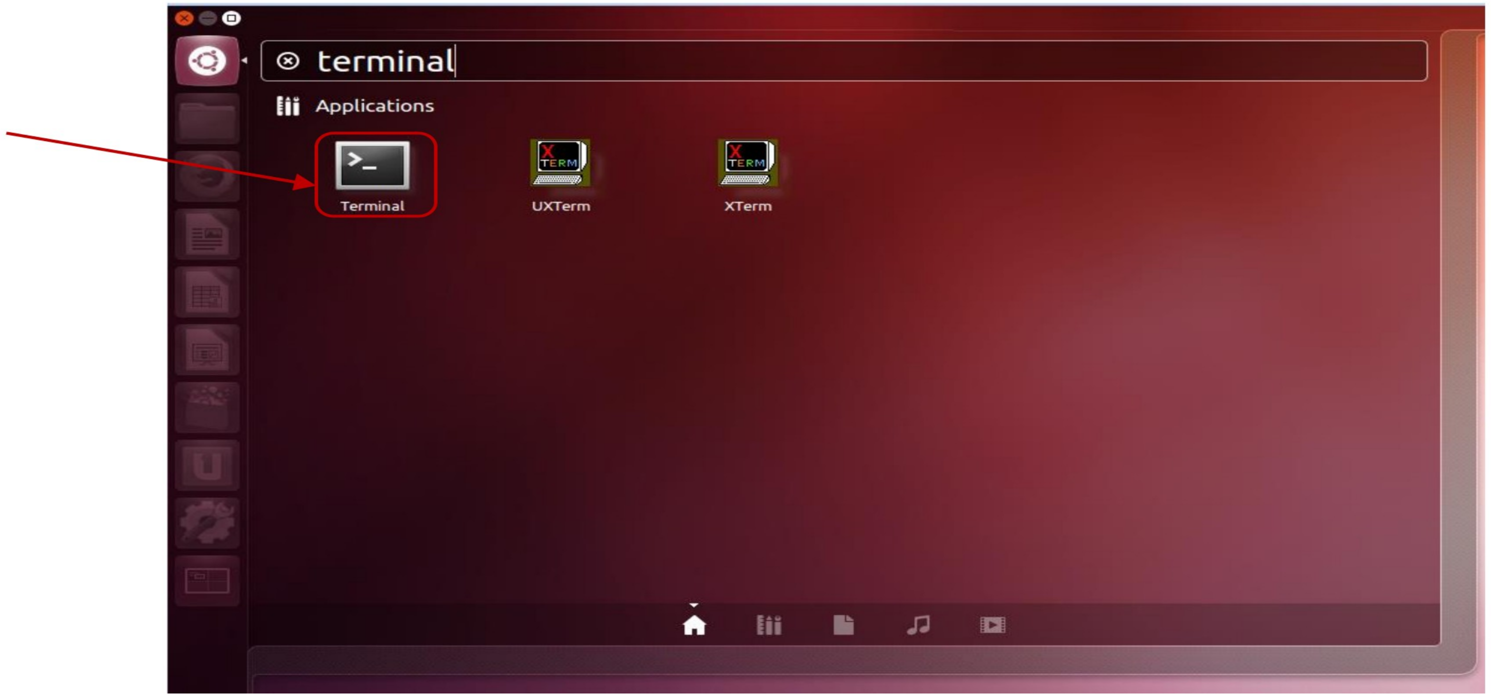
> **sudo apt-get --yes install synaptic**

> **sudo synaptic**

AGe

II. Getting Started: Terminal

- Command line is accessed through a program called Terminal
- Click the Ubuntu Button on the Ubuntu menu bar→ Search "Terminal" → Open Terminal



By default the terminal prompts at your home folder. Connecting remotely to Linux PBS, you will be prompted to your home directory (folder).

AGe

II. Getting Started: Terminal

Opening a terminal to connect to linux system or PBS server:

1. **Mac OS X** includes a Terminal application (located in the Applications >> Utilities folder), which can be used to connect to other systems.

1. From **Ubuntu** launch Terminal (Ctrl + Alt + T) and at the command prompt. Use dash board to search for a particular software

1. On **Windows** systems you can use a variety of programs to connect to a Linux system.
PuTTY is free and the most used.

By default the terminal prompts at your home folder.
Connecting remotely to Linux PBS, you will be prompted to your home directory (folder).

AGe

II. Getting Started: Terminal

Opening a terminal to connect to linux system or PBS server:

Terminal from both Mac OS X and Ubuntu uses same command lines to access PBS:

➔1. From **Ubuntu** launch Terminal (Ctrl + Alt + T) and at the command prompt.

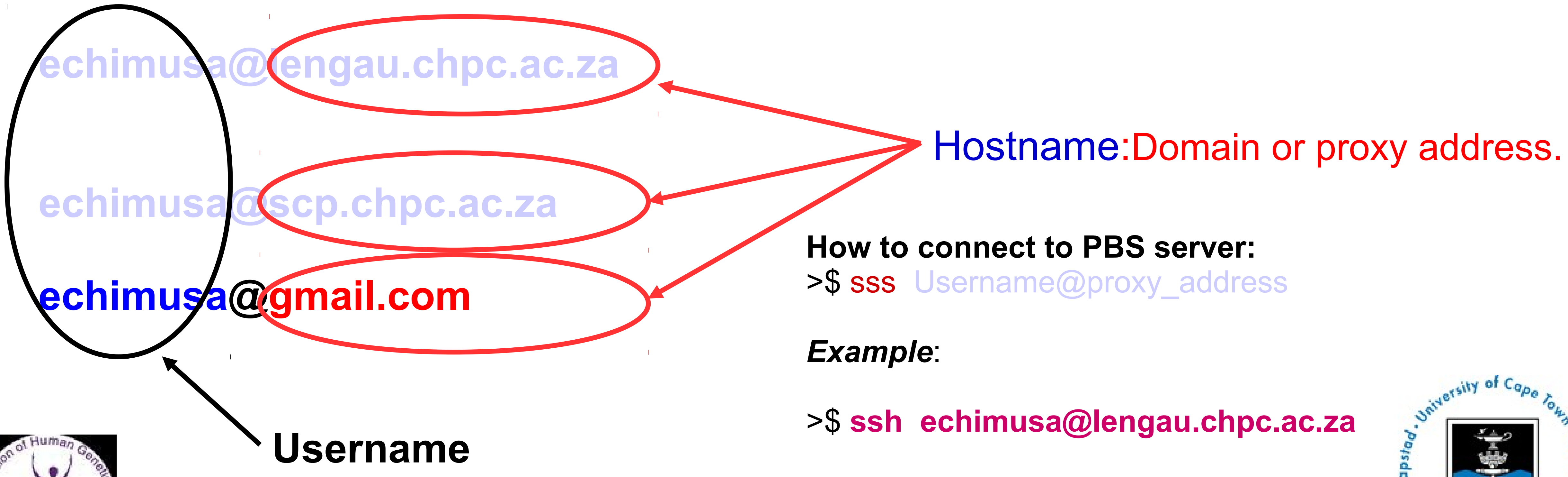
➔2. On **Windows** systems you can use a variety of programs to connect to a Linux system. PuTTY is free and the most used.

AGe

II. Getting Started: Terminal

Proxy server: is a dedicated computer acting as an intermediary between an endpoint device, such as a computer, and another server from which a user or client is requesting a service.

Example:



How to connect to PBS server:

```
>$ sss  Username@proxy_address
```

Example:

```
>$ ssh  echimusa@lengau.chpc.ac.za

>$ ssh  -X echimusa@lengau.chpc.ac.za
```

AGe

II. Getting Started: Terminal

Once connected you are at your home folder
For example:

/home/echimusa

Data Segregation; Where should files be stored?
Classify your Data

- How important is it?
- Can it be recreated?
- Is this dataset currently in use?
- Will this data be used by others?
- Does this data need to be backed up?

Put your data in the appropriate space

- Home Directory : /home/**your_user_name**
- Group Space :
- Network Mounted Scratch : **/mnt/lustre/users/your_user_name**

AGe

II. Getting Started: Basic command

When you sign in you will be located in your home directory.
To see where this directory is located in the file system, use the **pwd** command:

For example:

```
echimusa@login2 ~]$ pwd
/home/echimusa
```

Now you should be in the home directory. To see what is inside of this directory, use the ls command (**ls** stands for list):

```
[echimusa@login2 ~]$ ls
get-pip.py  hapfuse  MarViN1  soft  supportmix  vcftools
```

To change to a different directory, use the **cd** command (**cd** means change directory):

```
[echimusa@login2 ~]$ cd /mnt/lustre/users/echimusa/
```

You can supply certain alias terms to the cd command. One of these is the ˜ character, which represents your home directory (/home/echimusa/). Another is **..**, which represents the directory above the current directory.

AGe

II. Getting Started: Basic command

What does it do?
List files in the current directory

Usage
`$ ls        # list all files in current directory`
`$ ls *.txt # list all text files`

Options	What does it do?
-l	List file details
-h	Human-readable file sizes (e.g. 3.3MB vs. 3354123)
	Lots of build in sorting options (size, type, last update, etc.)

AGe

II. Getting Started: Basic command

⚠ You can supply certain alias terms to the cd command. One of these is the ˜ character, which represents your home directory (/home/echimusa/). Another is .., which represents the directory above the current directory. Let try this

```
$ cd˜
$ cd ..
```

To see which user you are signed in as, use the **whoami** command:

```
[echimusa@login2 ~]$ whoami
```

To see who else is signed in to the same system, use the who command:

```
[echimusa@login2 ~]$ who
embele   pts/0       2017-05-04 11:10 (10.128.23.174)
abarde   pts/1       2017-05-06 20:30 (146.141.41.149)
zfakhar  pts/3       2017-05-06 18:41 (2.146.1.71
```

To see current data:

```
[echimusa@login2 ~]$ date
Sat May  6 22:24:28 SAST 2017
```

II. Getting Started: Basic command

⚠ To create your own directories use the **mkdir** (make directory) command:

```
[echimusa@login2 ~]$ mkdir proteins
[echimusa@login2 ~]$ cd proteins/
[echimusa@login2 proteins]$ ls
[echimusa@login2 proteins]$
```

```
[echimusa@login2 proteins]$ pwd
/home/echimusa/proteins
```

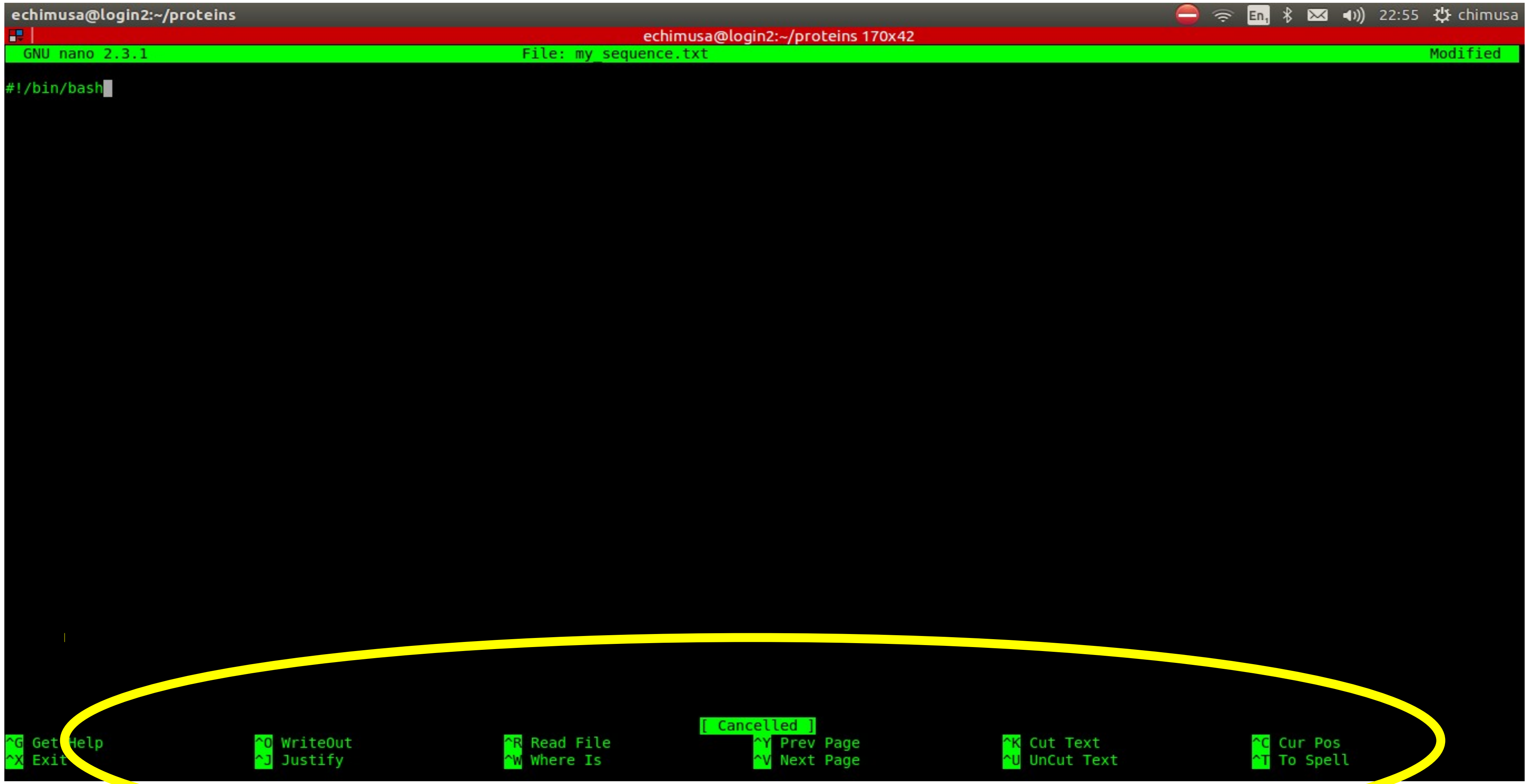
To create new file, let use touch and nano see who else is signed in to the same system, use the who command:

```
[echimusa@login2 proteins]$ touch my_sequence.sh
[echimusa@login2 proteins]$ ls -l my_sequence.sh
-rw-rw-r-- 1 echimusa echimusa 0 May  6 22:49 my_sequence.sh
```

AGe

II. Getting Started: Basic commandc

```
[echimusa@login2 proteins]$ nano my_sequence.sh
```



```
echimusa@login2:~/proteins                                                         22:55  chimusa
                                echimusa@login2:~/proteins 170x42
  GNU nano 2.3.1                        File: my_sequence.txt                          Modified

#!/bin/bash

                                          [ Cancelled ]
^G Get Help       ^O WriteOut       ^R Read File      ^Y Prev Page      ^K Cut Text       ^C Cur Pos
^X Exit           ^J Justify        ^W Where Is       ^V Next Page      ^U UnCut Text     ^T To Spell
```

AGe

II. Getting Started: Utility less

What does it do?

View a file or data stream with navigation options

Usage

```
$ less my_file
$ cat *.txt | less # 'less' is often used at the end of a chain
```

Options	What does it do?
-S	Don't wrap long lines
←↑→↓	Arrow keys to navigate in file (PageUp, PageDown, Home, End work too)
/	[in program] Search for text, use n and N to see next/previous matches

AGe

II. Getting Started: Utility cat/zcat

What does it do?

Dumps files on disk to STDOUT (for feeding into another program)

Usage

```
$ cat my_file                # lines of file scroll over screen
```

Any other command

```
$ zcat my_file.gz | program
```

```
$ cat my_file | program # lines of file enter program as STDIN
```

```
$ cat *.txt | program   # concatenate all text files as STDIN
```

Cmd *char	Operate or apply on all files/folders with extension "**char**".
Cmd chr|*	Operate or apply on all files/folders with prefix "**char**".
Cmd ***chr***	Operate or apply on all files/folder containing the string "**char**".

Any piece of string mainly from file or folder name

Should be any Linux command (cmd)

II. Getting Started: Utility top/kill

Opening an application or run a job locally on background: & at the end of the command line

Example:

```
$ firefox &
```

To see all running processes in your machine

```
$ top
```

A screen showing all running applications will be displayed. Each with an id (PID), and the user names. Ctrt + c to terminate the screen. PID can be used to force the stop

To kill or stop a running process

```
$ kill PID
```

Running process id

AGe

II. Getting Started: Modules on PBS

CHPC uses the GNU modules utility, which manipulates your environment, to provide access to the supported software in **/apps/**.

For a list of available modules:

```
[echimusa@login2 ~]$ module avail
```

To see currently loaded modules:

```
[echimusa@login2 proteins]$ module list
```

To remove all modules:

```
[echimusa@login2 proteins]$ module list
```

To load a modules:

```
[echimusa@login2 proteins]$ module load name_module
```

AGe

II. Getting Started: Job script parameters

my_sequence.sh

```
#!/bin/bash
#PBS -N Xchr
#PBS -q smp
#PBS -P CBBI0818
#PBS -l select=1:ncpus=24
#PBS -l walltime=48:00:00
#PBS -M echimusa@gmail.com

module load chpc/BIOMODULES
module load chpc/python/2.7.11
```

- q**stat**: View queued jobs. (eg. **qstat -u** user_name), or to see what are on each queue (**qstat** -Q).
- **qsub** :Submit a job to the scheduler.
- **qdel** :Delete one of your jobs from queue (**qdel** ID_of_your_job).

AGe

III. Transferring files

From both Mac and Ubuntu, we use the terminal to transfer the data from local to remote computer or from a remote to local machine. We commonly use **scp, rsync, wget (curl)**

Synthax:

a)
```
rsync options source destination
```

-au: update files that are newer in the original directory

b)
```
scp options source destination
```

-r: if copying folder

c)
```
wget options source -o path destination
```

From the ftp or internet source such as http://www.whatever.com/filename.txt

-nc => --no-clobber
-N => Turn on time-stamping
-r => Turn on recursive retrieving

Is optional if copying in to current folder

AGe

III. Transferring files: Examples

Example 1:

1. Launch the Terminal program.

2. Switch to your home directory on the Mac or Ubuntu using the command **cd ˜** .

3. Create a text file containing your home directory listing using **ls -l > myprotein.txt**
(you will learn more about the meaning of > later).

4. Now use the scp command on your Mac/Ubuntu to transfer the file you created to your CHPC account. This command requires two values: the file you want to transfer and the destination. Be sure to replace “user” with your user name, and replace hostname with the real hostname or IP address of the CHCP system you want to connect to:

```
$ scp myfiles.txt user@hostname:~
```

You should be prompted for your user account password. Remember, in the above example you are running the scp (can try also rsync) command on your Mac/Ubuntu desktop, not from your CHPC account.

AGe

III. Transferring files: Examples

Example 1:

Now, delete the myprotein.txt file on your Mac, and see if you can use scp to retrieve the file from your Linux account:

1. In the terminal on your Mac, switch to your home directory using cd ˜ .

2. Delete the myprotein.txt file using rm myprotein.txt.

3. Use the scp command to copy myprotein.txt from your Linux account back to your Mac. Remember to replace “hostname” and “user” with the appropriate values when you enter the command:

```
$ scp user@hostname:~/myprotein.txt  ./
```

Current directory (.)

AGe

III. Transferring files: Assignment I

To test your ability to transfer files to your CHPC account, download the following file to your Mac/Ubuntu/Window local system form
http://web.cbio.uct.ac.za/~emile/AGe/sample_sequences.zip
Once the file has been downloaded to your local machine, use the file transfer methods outlined above to transfer the file to your at the folder protein
(in scratch, /mnt/lustre/users/username).

Once the data is at CHPC, unzip it, it will be a folder and in side contain zip files, thus unzip them too.

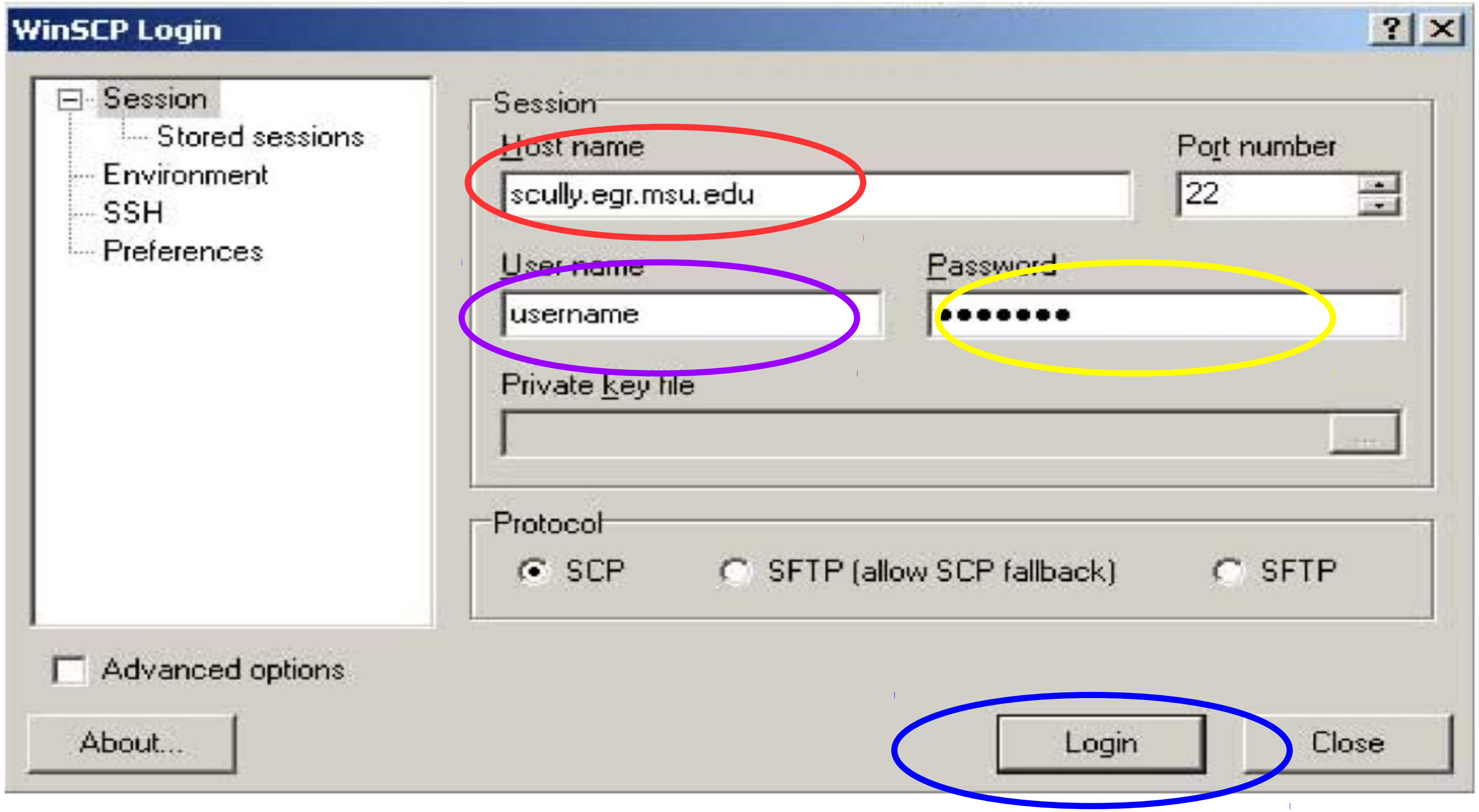
Be sure to perform this exercise, as the sample_sequences.zip file will be used later on.

AGe

III. Transferring files

Transferring data from Windows: we can use winscp software:

To use WinSCP, launch the program and enter the appropriate information into the Host name, User name, and Password text areas. Click Login to connect to the remote system. Once you are connected you should be able to transfer files and directories between systems using the simple graphical interface by dragging file to .


WinSCP Login

Session
Host name: scully.egr.msu.edu
Port number: 22
User name: username
Password: ●●●●●●●
Private key file:

Protocol: [✓ SCP] SFTP (allow SCP fallback) SFTP

AGe

III. Transferring files

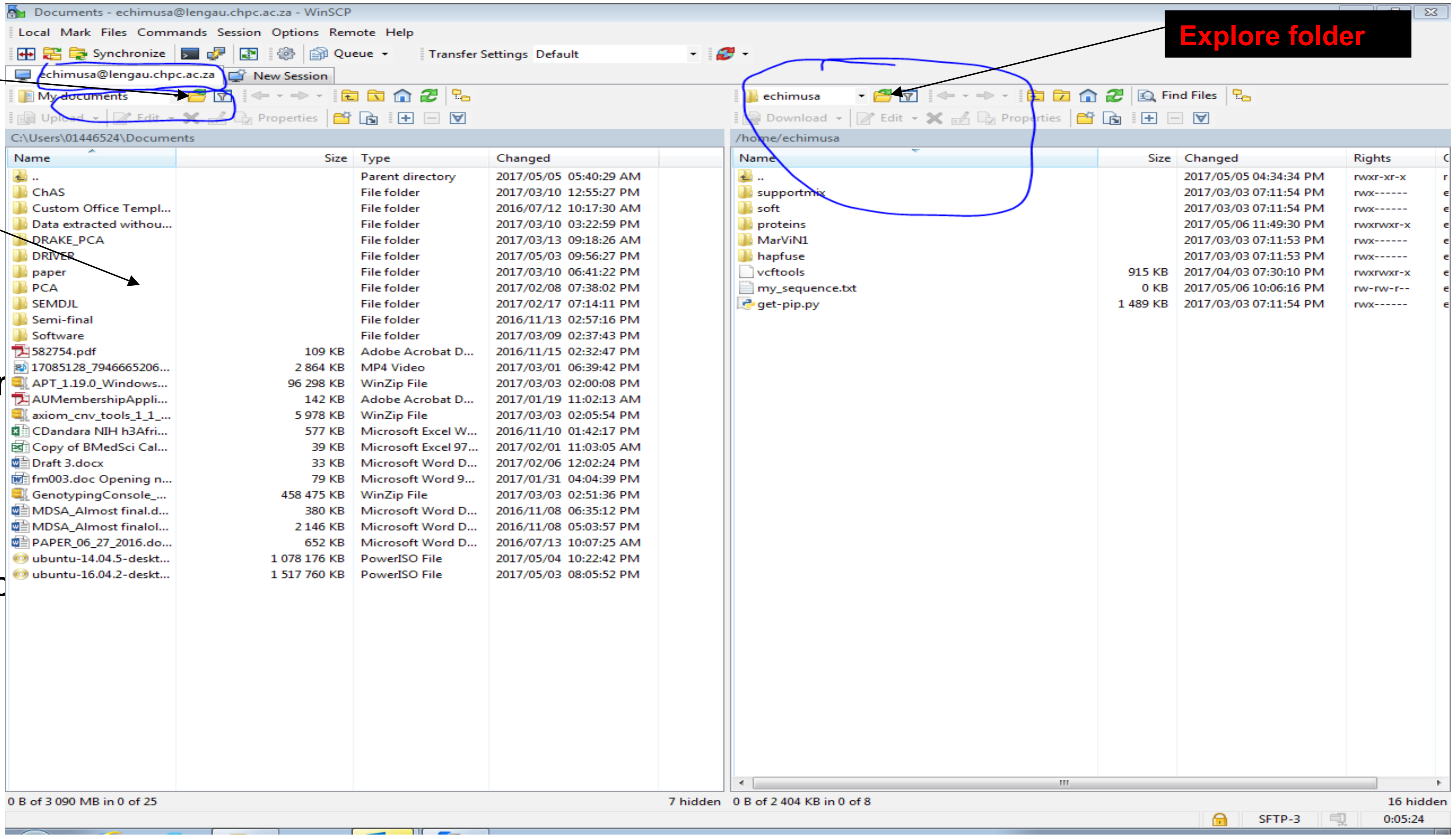
Transferring data from Windows: we can use winscp software:

Explore folder

Local machine

Explore folder

Once you are connected you should be able to transfer files and directories between systems by dragging files o folder in between.


Documents - echimusa@lengau.chpc.ac.za - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
echimusa@lengau.chpc.ac.za
New Session
My documents
Upload
Edit
Properties
echimusa
Find Files
Download
Edit
Properties

C:\Users\01446524\Documents

Name	Size	Type	Changed
..		Parent directory	2017/05/05 05:40:29 AM
ChAS		File folder	2017/03/10 12:55:27 PM
Custom Office Templ...		File folder	2016/07/12 10:17:30 AM
Data extracted withou...		File folder	2017/03/10 03:22:59 PM
DRAKE_PCA		File folder	2017/03/13 09:18:26 AM
DRIVER		File folder	2017/05/03 09:56:27 PM
paper		File folder	2017/03/10 06:41:22 PM
PCA		File folder	2017/02/08 07:38:02 PM
SEMDJL		File folder	2017/02/17 07:14:11 PM
Semi-final		File folder	2016/11/13 02:57:16 PM
Software		File folder	2017/03/09 02:37:43 PM
582754.pdf	109 KB	Adobe Acrobat D...	2016/11/15 02:32:47 PM
17085128_7946665206...	2 864 KB	MP4 Video	2017/03/01 06:39:42 PM
APT_1.19.0_Windows...	96 298 KB	WinZip File	2017/03/03 02:00:08 PM
AUMembershipAppli...	142 KB	Adobe Acrobat D...	2017/01/19 11:02:13 AM
axiom_cnv_tools_1_1_...	5 978 KB	WinZip File	2017/03/03 02:05:54 PM
CDandara NIH h3Afri...	577 KB	Microsoft Excel W...	2016/11/10 01:42:17 PM
Copy of BMedSci Cal...	39 KB	Microsoft Excel 97...	2017/02/01 11:03:05 AM
Draft 3.docx	33 KB	Microsoft Word D...	2017/02/06 12:02:24 PM
fm003.doc Opening n...	79 KB	Microsoft Word 9...	2017/01/31 04:04:39 PM
GenotypingConsole_...	458 475 KB	WinZip File	2017/03/03 02:51:36 PM
MDSA_Almost final.d...	380 KB	Microsoft Word D...	2016/11/08 06:35:12 PM
MDSA_Almost finalol...	2 146 KB	Microsoft Word D...	2016/11/08 05:03:57 PM
PAPER_06_27_2016.do...	652 KB	Microsoft Word D...	2016/07/13 10:07:25 AM
ubuntu-14.04.5-deskt...	1 078 176 KB	PowerISO File	2017/05/04 10:22:42 PM
ubuntu-16.04.2-deskt...	1 517 760 KB	PowerISO File	2017/05/03 08:05:52 PM

0 B of 3 090 MB in 0 of 25 7 hidden

/home/echimusa

Name	Size	Changed	Rights	C
..		2017/05/05 04:34:34 PM	rwxr-xr-x	r
supportmix		2017/03/03 07:11:54 PM	rwx------	e
soft		2017/03/03 07:11:54 PM	rwx------	e
proteins		2017/05/06 11:49:30 PM	rwxrwxr-x	e
MarViN1		2017/03/03 07:11:53 PM	rwx------	e
hapfuse		2017/03/03 07:11:53 PM	rwx------	e
vcftools	915 KB	2017/04/03 07:30:10 PM	rwxrwxr-x	e
my_sequence.txt	0 KB	2017/05/06 10:06:16 PM	rw-rw-r--	e
get-pip.py	1 489 KB	2017/03/03 07:11:54 PM	rwx------	e

0 B of 2 404 KB in 0 of 8 16 hidden

SFTP-3 0:05:24

AGe

III. Transferring files: Large data

Transferring large genomic data may takes weeks:

Screen is a full-screen software program that can be used to multiplexes a physical console between several processes (typically interactive shells). It offers a user to open several separate terminal instances inside a one single terminal window manager.

```
[echimusa@login2 ~]$ screen option
There is no screen to be resumed.
```

-r : to resume existing screen.
-S :name_screen: create new screen

Once in an open screen, press **ctrl+shift+A+D** you can detach and back to the terminal without stop the process or even logout without stop the process.

screen -r allow you to screen list of screens, screen -r id_screen allows you to go to the screen.

Screen is useful when copying large data from web or other to PBS server such CHPC.

AGe

III. Transferring files: Example

Transferring large genomic data may takes weeks:

From your CHPC account tape the follows

```
[echimusa@login2 ~]$ screen -S jobs1
```

Once in the screen is open,

```
[echimusa@login2 ~]$ top
```

Now, press at the same time **ctrl+shift+A+D**

```
[echimusa@login2 ~]$ screen -r
```

Should be able to see your screen, the number before your screen name is the id. You can use that **id** to go back to you screen by,

```
[echimusa@login2 ~]$ screen -r xxxx
```

AGe

III. Transferring files: UCT research data

UCT research data is a long term storage server. You cannot perform any data analysis from there. You will need to move the to the productive server such CHPC or UCT/HPCT.

How from your Ubuntu local machine to mount UCT research data, so that you can transfer them?

Here is how you can mount a driver in Linux,

1. Create empty folder at your local machine;

```
chimusa@chimusa:~$ mkdir
LAPTOP/
```

2. Mount the researchdata.uct.ac.za to LAPTOP/:

```
chimusa@chimusa:~$ sudo mount -t cifs -o user=01446524 //researchdata.uct.ac.za/HumanGenetics   LAPTOP/
```

Need to be super user of your local machine

Your UCT ID

UCT Hostname rersearch data

Human genetics folder at research data

Your local folder

AGe

III. Transferring files: Working with tar and zip file

Let get zip file from NCBI database,

```
>$ wget ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/GFF/alt_CHM1_1.1_top_level.gff3.gz
```

To extract this

```
>$ gunzip  alt_CHM1_1.1_top_level.gff3.gz
```

Now let zip this with tar (to have zip format with .tar or .tar.zip

1. `>$ tar -cvf  alt_CHM1_1.1_top_level.gff3`
2. `>$ tar -cvzf alt_CHM1_1.1_top_level.gff3.tar.zip alt_CHM1_1.1_top_level.gff3`

To extract a tar file use the tar command, this time with the -x (extract) option:

```
>$ tar -xvf alt_CHM1_1.1_top_level.gff3.tar
>$ tar -xvzf alt_CHM1_1.1_top_level.gff3.tar.zip
```

Note to compress a folder using tar/zip/gzip you need a recursive -r parameter :

```
>$ zip -r my_folder.zip my_folder
```

AGe

III. Transferring files: file permissions, using chmod

```
>$ chmod go-r somefile
>$ ls -l
```

The go-r portion of the above chmod command means “from the group (g) and other (o) permission sets take away the read permission (r).” To allow everyone to read the file somefile you could modify the permissions using the following:

```
>$ chmod a+r somefile
```

The a in the above command means “all users”. To refer to different types of users separately, use u (user who owns the file), g (group that owns the file), and o (other users).

r (read permission) indicates that the file can be read.
w (write permission) indicates that the file can be modified.
x (execute permission) indicates that the file can be executed as a program.

IV. Streamlining data manipulationc

Utilities: `man`

What does it do?
Displays a “manual” page for another tool, detailing expected input data and optional flags

Usage
`$ man less`

Options	What does it do?
Q	[within program] Exits the current man page

IV. Streamlining data manipulation

Utilities: `rm`

What does it do?
Deletes (removes) a file **PERMANENTLY**

Usage
`$ rm my_file`

Options	What does it do?
`-r`	Force-remove a directory (will fail by default)
`-i`	Confirm each deletion event

IV. Streamlining data manipulation

Utilities: cp

What does it do?
Copy a file

Usage
`$ cp my_file my_copy`

Options	What does it do?
`-i`	Warn before overwriting a file
`-r`	Copy recursively (needed for copying directories)

IV. Streamlining data manipulation

Utilities: `mv`

What does it do?
Move a file to a new location (or rename a file)

Usage
`$ mv my_file MY_FILE   # rename my_file as MY_FILE`
`$ my /my_dir/my_file . # move file “here” (.)`

Option s	What does it do?
`-i`	Warn before overwriting a file

Path syntax

$ is used to indicate the start of a command (your prompt may look different)

```
$ command
```

/ is the root of the file system (`C:\` on Windows, typically; note the backslash)

```
$ ls /
```

A subdirectory of the root directory

```
$ ls /my_dir/
```

. refers to the current directory

```
$ wc -l ./my_file.txt
$ cp /my_dir/my_file.txt . # copy file “here”
```

.. refers to the parent directory

```
$ mv ../my_file.txt .
```

IV. Streamlining data manipulation

Utilities: grep

What does it do?
Isolate lines of a data stream (file or STDIN) that **match a pattern**

Usage : egrep (or grep) [command line options] <pattern> [path]

```
$ grep pattern my_file
$ cat *.txt | grep pattern
```

Options	What does it do?
`-P`	Richer pattern options (regular expressions); more on these in a later lecture
`-v`	Isolate lines that DO NOT match the pattern (invert the match)
`-i`	Case-insensitive match
`-f`	Specify a file of patterns to match (slow if there are lots of options)

IV. Streamlining data manipulation

Utilities: grep

. (dot) : a single character.
? : the preceding character matches 0 or 1 times only.
* : the preceding character matches 0 or more times.
+ : the preceding character matches 1 or more times.
{n} : the preceding character matches exactly n times.
{n,m} : the preceding character matches at least n times and not more than m times.
[agd] : the character is one of those included within the square brackets.
[^agd] : the character is not one of those included within the square brackets.
[c-f] : the dash within the square brackets operates as a range. In this case it means either the letters c, d, e or f.
() : allows us to group several characters to behave as one.
| (pipe symbol) : the logical OR operation.
^ : matches the beginning of the line.
$: matches the end of the line.

IV. Streamlining data manipulation

Utilities: `grep (or egrep) some Examples`

Using the mysamples.txt @ : http://web.cbio.uct.ac.za/~emile/AGe/mysampledata.txt

1. Identify any line with two or more vowels in a row.
```
>$ grep '[aeiou]{2,}' mysampledata.txt
```
2. Print out any line with a 2 on it which is not the end of the line.
```
>$ egrep '2.+' mysampledata.txt
```

1. Print out lines match "Lerato" and line number as well
```
>$ grep -n "Lerato"  mysampledata.txt
```
2. Print out lines of whic number 2 as the last character on the line.
```
>$ egrep '2017$' mysampledata.txt
```

1. Print out line which contains either 'Raj' or 'Clinical' or 'Genetics'.
```
>$ egrep 'Raj|Clinical|Genetics' mysampledata.txt
```

1. Print out Sunrname begins with A - K.
```
>$ egrep '^[A-K]' mysampledata.txt
```

IV. Streamlining data manipulation

Utilities: cut

What does it do?
Isolate tab-delimited columns of a data stream. First column is #1 (not #0 as in Python).

Usage
`$ cut -f2 my_file   # isolate the 2nd column of the file`
`$ cut -f2,3 my_file # isolate columns 2 and 3`
`$ cut -f2-5 my_file # isolate columns 2 THROUGH 5`
`$ cut -f3- my_file  # isolate columns 3 to END (python [2:] slice)`

Options	What does it do?
`-f`	Select columns (fields)
`-t`‘*char*’	Break columns on the specified character instead of tab, e.g. `-t‘,’` for .csv file

IV. Streamlining data manipulation

Utilities: cut

Let use the data GIANT_BMI.EUR.gwas.assoc.txt.zip, GIANT_BMI.EUR.gwas.assoc.txt.gzip and a normal tab separation file hmp2012_metadata.tsv. All these files are located at **http://web.cbio.uct.ac.za/~emile/AGe/**

1. Download these file at local CHPC account, using **wget**

```
$ wget http://web.cbio.uct.ac.za/~emile/AGe/xxxxxxx
```

2. To view the first 4 lines of zip file, use zcat (for .gzip) and unzip for (.zip), following pipe (|),

```
$ zcat GIANT_BMI.EUR.gwas.assoc.txt.gzip | head -n 4
$ unzip GIANT_BMI.EUR.gwas.assoc.txt.zip | head -n 4
$ cat hmp2012_metadata.tsv | head -n 4
```

3. You can use grep to search pattern, cut to extract columns of interest and use > to redirect the output to a new file.

Example:

```
$ zcat GIANT_BMI.EUR.gwas.assoc.txt.gzip | cut -f1,3,4 | gzip  -c > emile.txt.gz
```

IV. Streamlining data manipulation

Utilities: `sort`

What does it do?
Sort the lines of a data stream (alphabetically)

Usage
`$ sort my_file`

Options	What does it do?
`-r`	Reverse the sort
`-k` *N*	Sort on the value of **WHITESPACE**-delimited column *N*
`-t`‘*char*’	Specify the delimiter character (e.g. `-t‘\t’` for tab)
`-n`	Perform a numeric sort (otherwise `10` comes before 2)

IV. Streamlining data manipulation

Utilities: `uniq`

What does it do?
Isolate the unique **ADJACENT** lines of a data stream

Usage
`$ sort my_file | uniq # w/o sorting, non-adjacent repeats missed`

Options	What does it do?
-c	Count the occurrence of each unique line

IV. Streamlining data manipulation

Excise:
From data in hmp2012_metadata.tsv located at **http://web.cbio.uct.ac.za/~emile/AGe/**
1) gzip the file.
2) delete hmp2012_metadata.tsv
3) How many patients with BODY_SITE = Stool, rewrite these patients sorted by VISIT retain only column SAMPLE_ID, VISIT and BODY_SITE into a new gzipped file.

IV. Streamlining data manipulation

Utilities: `wc`

What does it do?
Counts the lines, words, and characters of a data stream

Usage
`$ wc my_file`

Options	What does it do?
`-l`	Only report line count (faster, and often all you want)
`-w`	Only report word count
`-c`	Only report character count

IV. Streamlining data manipulation

Utilities: head/tail

What does it do?
Stream the first/last (`tail/head`) lines of a data stream (default 10)

Usage
`$ head my_file`
`$ head -n 100 my_file | tail # Stream lines 91-100`

Options	What does it do?
`-n` *N*	Stream *N* lines instead of default 10

IV. Streamlining data manipulation

Utilities: `column`

What does it do?

"Normalize" the widths of column entries (Excel*ify* your data)

Usage

```
$ column -t my_file
```

Options	What does it do?
`-s`'*char*'	Specify the delimiter character (e.g. `-s'\t'` for tab); default = any whitespace

IV. Streamlining data manipulation

Utilities: sed

What does it do?
Edit a data stream, most often used for find/replace operations

Usage
$ sed “s/*find*/*replace*/g” my_file
$ sed “s/apple/banana/g” my_file # replace all instances of “apple” with “banana”

Options	What does it do?
-i	Edit file “in place” (use with caution)
	“*find*” can be a regular expression, and “*replace*” can use captured elements of the pattern (this will make more sense after our regular expressions lecture).

IV. Streamlining data manipulation

Utilities: awk

What does it do?

Is specifically designed for quickly manipulating space delimited data

Usage

```
$ awk ‘/<pattern>/’ file #search the file and output line with match pattern
$ awk ‘{<code>}’ file # put a little awk program in the curly braces
```

Example:

Let download data from NCBI

```
$ wget ftp://ftp.ncbi.nlm.nih.gov/genomes/H_sapiens/GFF/alt_CHM1_1.1_top_level.gff3.gz
$  zcat alt_CHM1_1.1_top_level.gff3.gz | awk '{if ( ($3=="gene") && ($4 > 10000) && ($4 < 20000) ) print $0}'
```

IV. Streamlining data manipulation
Utilities: awk

```
$  zcat alt_CHM1_1.1_top_level.gff3.gz | awk '{if ( ($3=="gene") && ($4 > 10000) && ($4 < 20000) ) print $0}'
```

In the previous code, $3==”gene” asks for the entry in the third field to be “gene.” The next two fragments request for the values in the fourth field (i.e. $4) to be between 10000 & 20000. Thus we have found all the features annotate as genes in a 10000bp region of this genome. Notice that each these three commands were enclosed in a pair of brackets within an outer pair of brackets. This gave a command of the form:
((Condition1) && (Condition2) && (Condition3))
After this came the fragment print $0 which asked awk to print the entire line if the 3 conditions are true.

Another example (what does this do?):

```
$ gzip alt_CHM1_1.1_top_level.gff3.gz
$ awk '{if (($5 - $4 > 1000) && ($3 == "gene")) print $1, $2, $4, $5, \
$9}' alt_CHM1_1.1_top_level.gff3 > myoutput.gff3}
```

Above we specify which columns to output.

IV. Streamlining data manipulation
Utilities: awk

More a example with GIANT_BMI.EUR.gwas.assoc.txt dataset

1. Choose rows where column MAF > 0.05 :

```
>$ awk '$4>0.05' GIANT_BMI.EUR.gwas.assoc.txt > output.cSNP.txt
```

2. Extract column SNP,MAF,PVALUE:

```
>$ awk '{print $1,$4,$7}' GIANT_BMI.EUR.gwas.assoc.txt > output.PV.txt
```

or

```
>$ awk 'BEGIN{OFS="\t"}{print $2,$4,$5}' GIANT_BMI.EUR.gwas.assoc.txt
```

3. Show rows between 20th and 80th:

```
>$ awk 'NR>=20&&NR<=80' GIANT_BMI.EUR.gwas.assoc.txt> output.sub.txt
```

4.calculate the average of PVALUE:

```
>$ awk '{x+=$7}END{print x/NR}' input.avP.txt
```

IV. Streamlining data manipulation

Utilities: awk

More a example with GIANT_BMI.EUR.gwas.assoc.txt dataset

5.Calculate the sum of column 5 and 6 and put it at the end of a row or replace the first column:

```
>$ awk '{print $0,$5+$6}' GIANT_BMI.EUR.gwas.assoc.txt
>$ awk '{$1=$5+$6;print}' GIANT_BMI.EUR.gwas.assoc.txt
```

How to join two files based on a common column, support is the column 1:

```
>$ awk 'BEGIN{while((getline<"file1.txt")>0)l[$1]=$0}$1 in l{print $0"\t"l[$1]}' file2.txt >
output.txt
```

More tutorials are onlines

V. Shell Scripting

V.1: Variables:

The shell uses variable and the simple assignment of value to a variable is by =
eg.

exe_plink=”/mnt/lustre/users/echimusa/myprotein/plink”

To use a variable

$exe_plink or ${exee_plink}

V.2 If statement:

```
If statement
Type 1:
if condition is true
then
        do this
if
Type 2:
if condition is true
then
        do this
else
        do that
if
```

V. Shell Scripting

V.2 If statement:

```
exe_plink=”/mnt/lustre/users/echimusa/myprotein/plink”
path=”/mnt/lustre/users/echimusa/myprotein/”

if [ ! -d "${exe_plink}" ];then
        mkdir  ${path}MAF
        chmod +rw ${path}MAF
else
       ${exe_plin} --help
fi
```

V.3 Iteration in Shell

#simple for loop

```
for i in 1 2 3
do
    echo "==>$i"
done
 or
For chr in {1..22}
do
     echo “this chromose ${chr}
done
```

while loop – syntax

```
while [ condition ] do

      code block;

done
```

V. Shell Scripting

V.3 Iteration in Shell

Let download GWAS data GWAStutorial.bed, GWAStutorial.fam and GWAStutorial.bim at http://web.cbio.uct.ac.za/~emile/Age/.

The download data is for the whole genome. We split is into chromosemes, convert into VCF and after submit the job at CHPC.

1. Create a text file call emile_vcf.sh
2. We have to a) put the shell header and as we wish to qsub to node, we then load PBS headers at the top of the created file

```
#!/bin/bash
#PBS -N test
#PBS -q smp
#PBS -P CBBI0818
#PBS -l select=1:ncpus=24
#PBS -l walltime=48:00:00
```

3. Now let load potential module needed

```
module add chpc/BIOMODULES
module add chpc/gnu/parallel-20160422
```

V. Shell Scripting

V.3 Iteration in Shell

Let download GWAS data GWAStutorial.bed, GWAStutorial.fam and GWAStutorial.bim at http://web.cbio.uct.ac.za/~emile/Age/.

The download data is for the whole genome. Let split it into chromosemes, convert into VCF by running the task on CHPC cluster.

4. Let declare or setup some variable to specify working directory, where to find software and where to read Inputs and write the output

```
exe_plink=”/mnt/lustre/users/echimusa/myprotein/plink”
path=”/mnt/lustre/users/echimusa/myprotein/”
out=”/mnt/lustre/users/echimusa/myprotein/”
inputbed=”/mnt/lustre/users/echimusa/myprotein/GWAStutorial”
CHR=22

if [ -f "${exe_plink}" ];then
      echo “Plink software exists”
fi
if [ -f “{inputbed}.bed”];then
        mkdir ${out}
        chmod +rw ${out}
fi
```

V. Shell Scripting

V.3 Iteration in Shell

Let execute plink

```
${exe_plink} –bfile ${inputbed} –chr ${CHR} –recode vcf –out ${out}GWAS.${CHR}
${exe_plink} –bfile ${inputbed} –chr ${CHR} –freq –out ${out}GWAS.${CHR}
```

Let discuss the rest of the code in both shell scripts, qsub_emile_vcf.sh and emile_vcf.sh

?