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
Outlines:

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

Ubuntu Desktop

The 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.
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.
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
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
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

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

> $ sudo
OR

> $ sudo su
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
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.
Ubuntu: File System

bolt

\texttt{/opt} \quad \textbf{usually contains third-party software.}

\texttt{/proc} \quad \textbf{a virtual file system containing information about system resources.}

\texttt{/root} \quad \textbf{the root user’s home directory.}

\texttt{/sbin} \quad \textbf{essential programs used by the system and by the root user.}

\texttt{/tmp} \quad \textbf{temporary space that can be used by the system and by users.}

\texttt{/usr} \quad \textbf{programs, libraries, and documentation for all user-related programs.}

\texttt{/var} \quad \textbf{contains files to which the system writes data during the course of its operation.}
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.

```bash
> sudo apt-get update
> sudo apt-get install NAMESOFTWARE
```
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.
**Ubuntu:** Add/Remove from Synaptic Manager

- `sudo apt-get --yes install synaptic`
- `sudo synaptic`

---

> > sudo apt-get --yes install synaptic
> > sudo synaptic
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).
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).
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.
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:

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

Username: Domain or proxy address.

Example:

```
$ ssh echimusa@scp.chpc.ac.za
$ ssh -X echimusa@scp.chpc.ac.za
```

```
$ ssh echimusa@gmail.com
$ ssh -X echimusa@gmail.com
```
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
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.
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 built-in sorting options (size, type, last update, etc.)
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)
abarade 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
```
II. Getting Started: Basic commands

```bash
[echimusa@login2 proteins]$ nano my_sequence.sh
```
# II. Getting Started: Utility `less`

## What does it do?
View a file or data stream with navigation options

## Usage

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$ less my_file</code></td>
<td>View the file <code>my_file</code> with navigation options</td>
</tr>
<tr>
<td>`$ cat *.txt</td>
<td>less` # ‘less’ is often used at the end of a chain</td>
</tr>
</tbody>
</table>

## Option what does it do?

<table>
<thead>
<tr>
<th>Option</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-S</td>
<td>Don’t wrap long lines</td>
</tr>
<tr>
<td>←↑→↓</td>
<td>Arrow keys to navigate in file (PageUp, PageDown, Home, End work too)</td>
</tr>
<tr>
<td>/</td>
<td>[in program] Search for text, use n and N to see next/previous matches</td>
</tr>
</tbody>
</table>
II. Getting Started: Utility `cat/zcat`

### What does it do?
Dumps files on disk to STDOUT (for feeding into another program)

### Usage

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$ cat my_file</code></td>
<td># lines of file scroll over screen</td>
</tr>
<tr>
<td>`$ zcat my_file.gz</td>
<td>program`</td>
</tr>
<tr>
<td>`$ cat my_file</td>
<td>program`</td>
</tr>
<tr>
<td>`$ cat *.txt</td>
<td>program`</td>
</tr>
</tbody>
</table>

*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

To kill or stop a running process

$ kill PID

Running process id
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 module:

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

→ `qstat`: 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`).
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

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

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

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

Is optional if copying in to current folder
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.
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 ./
To test your ability to transfer files to your CHPC account, download the following file to your Mac/Ubuntu/Window local system from 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.
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 .
III. Transferring files

Transferring data from Windows: we can use winscp software:

Once you are connected you should be able to transfer files and directories between systems by dragging files or folder in between.
Transferring large genomic data may take 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 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
```

`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 as CHPC.
III. Transferring files: Example

Transferring large genomic data may take 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 your screen by,

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

Human genetics folder at research data

Your local folder

UCT Hostname research data
III. Transferring files: Working with tar and zip file

Let get zip file from NCBI database,

```bash
```

To extract this

```bash
$ 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. ```bash
   $ tar -cvf alt_CHM1_1.1_top_level.gff3
   ```
2. ```bash
   $ 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:

```bash
$ 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:

```bash
$ zip -r my_folder.zip my_folder
```
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.
# Utilities: man

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

## Usage
```
$ man less
```

## Option(s) What does it do?

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>[within program] Exits the current man page</td>
</tr>
</tbody>
</table>
Utilities: `rm`

<table>
<thead>
<tr>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deletes (removes) a file <strong>PERMANENTLY</strong></td>
</tr>
</tbody>
</table>

**Usage**

```
$ rm my_file
```

<table>
<thead>
<tr>
<th>Option</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-r</code></td>
<td>Force-remove a directory (will fail by default)</td>
</tr>
<tr>
<td><code>-i</code></td>
<td>Confirm each deletion event</td>
</tr>
</tbody>
</table>
### IV. Streamlining data manipulation

**Utilities: cp**

<table>
<thead>
<tr>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy a file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$ cp my_file my_copy</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Warn before overwriting a file</td>
</tr>
<tr>
<td>-r</td>
<td>Copy recursively (needed for copying directories)</td>
</tr>
</tbody>
</table>
## IV. Streamlining data manipulation

### Utilities: mv

<table>
<thead>
<tr>
<th>What does it do?</th>
<th>Move a file to a new location (or rename a file)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ mv my_file MY_FILE  # rename my_file as MY_FILE</td>
</tr>
<tr>
<td>$ my /my_dir/my_file .  # move file “here” (.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Warn before overwriting a file</td>
</tr>
</tbody>
</table>
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 .
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`

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

IV. Streamlining data manipulation
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/mysampleddata.txt

1. Identify any line with two or more vowels in a row.
   >$ grep '^[aeiou]{2,}$' mysampleddata.txt

2. Print out any line with a 2 on it which is not the end of the line.
   >$ egrep '2.+'$ mysampleddata.txt

1. Print out lines match "Lerato" and line number as well
   >$ grep -n "Lerato" mysampleddata.txt

2. Print out lines of which number 2 as the last character on the line.
   >$ egrep '2017$'$ mysampleddata.txt

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

1. Print out Surname begins with A - K.
   >$ egrep '^[A-K]' mysampleddata.txt
**Utilities: cut**

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

**Usage**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$ cut -f2 my_file</code></td>
<td># isolate the 2\textsuperscript{nd} column of the file</td>
</tr>
<tr>
<td><code>$ cut -f2,3 my_file</code></td>
<td># isolate columns 2 and 3</td>
</tr>
<tr>
<td><code>$ cut -f2-5 my_file</code></td>
<td># isolate columns 2 THROUGH 5</td>
</tr>
<tr>
<td><code>$ cut -f3- my_file</code></td>
<td># isolate columns 3 to END (python [2:] slice)</td>
</tr>
</tbody>
</table>

**Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-f</code></td>
<td>Select columns (fields)</td>
</tr>
<tr>
<td><code>-t</code> &quot;char&quot;</td>
<td>Break columns on the specified character instead of tab, e.g. <code>-t</code> ',' for .csv file</td>
</tr>
</tbody>
</table>
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
   
   ```bash
   $ 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 (|),
   
   ```bash
   $ 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:
   
   ```bash
   $ zcat GIANT_BMI.EUR.gwas.assoc.txt.gzip | cut -f1,3,4 | gzip -c > emile.txt.gz
   ```
### Utilities: sort

<table>
<thead>
<tr>
<th>What does it do?</th>
<th>Reverse the sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort lines of data stream (alphabetically)</td>
<td></td>
</tr>
</tbody>
</table>

#### Usage

```
$ sort my_file
```

<table>
<thead>
<tr>
<th>Options</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r</td>
<td>Reverse the sort</td>
</tr>
<tr>
<td>-k N</td>
<td>Sort on value of WHITESPACE-delimited column N</td>
</tr>
<tr>
<td>-t 'char'</td>
<td>Specify delimiter character (e.g. -t \t for tab)</td>
</tr>
<tr>
<td>-n</td>
<td>Perform numeric sort (otherwise 10 comes before 2)</td>
</tr>
</tbody>
</table>
## IV. Streamlining data manipulation

### Utilities: uniq

<table>
<thead>
<tr>
<th>What does it do?</th>
<th>Isolate the unique <strong>ADJACENT</strong> lines of a data stream</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage</strong></td>
<td>$ sort my_file</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td><strong>What does it do?</strong></td>
</tr>
<tr>
<td>- c</td>
<td>Count the occurrence of each unique line</td>
</tr>
</tbody>
</table>
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`

<table>
<thead>
<tr>
<th>What does it do?</th>
<th>Counts the lines, words, and characters of a data stream</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Usage</strong></th>
<th><code>$ wc my_file</code></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-l</code></td>
<td>Only report line count (faster, and often all you want)</td>
</tr>
<tr>
<td><code>-w</code></td>
<td>Only report word count</td>
</tr>
<tr>
<td><code>-c</code></td>
<td>Only report character count</td>
</tr>
</tbody>
</table>
## 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?

<table>
<thead>
<tr>
<th>Option</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n N</td>
<td>Stream N lines instead of default 10</td>
</tr>
</tbody>
</table>
## IV. Streamlining data manipulation

### Utilities: column

<table>
<thead>
<tr>
<th>What does it do?</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>“Normalize” the widths of column entries (Excelify your data)</td>
<td></td>
</tr>
</tbody>
</table>

### Usage

```
$ column -t my_file
```

### Options

<table>
<thead>
<tr>
<th>Option</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-s</code> <code>char</code></td>
<td>Specify the delimiter character (e.g. <code>-s \t</code> for tab); default = any whitespace</td>
</tr>
</tbody>
</table>
# IV. Streamlining data manipulation

## Utilities: sed

<table>
<thead>
<tr>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit a data stream, most often used for find/replace operations</td>
</tr>
</tbody>
</table>

## Option(s)

<table>
<thead>
<tr>
<th>Option</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Edit file “in place” (use with caution)</td>
</tr>
</tbody>
</table>

## Usage

- `$ sed "s/find/replace/g" my_file`
- `$ sed "s/apple/banana/g" my_file` # replace all instances of “apple” with “banana”

“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).
# 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

```bash
$ 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, \n$9}' alt_CHM1_1.1_top_level.gff3 > myoutput.gff3

Above we specify which columns to output.
IV. Streamlining data manipulation

Utilities: awk

More an 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=":"}{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){$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 ${exe_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 chromosome ${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 it into chromosomes, 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

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

3. Now let load potential module needed

```bash
module add chpc/BIOMODULES
module add chpc/gnu/parallel-20160422
```
V. Shell Scripting

V.3 Iteration in Shell

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

The download data is for the whole genome. Let split it into chromosomes, 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

```bash
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/GWAS.tutorial"
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

$\{\text{exe}_\text{plink}\} -\text{bfile} \ $\{\text{inputbed}\} -\text{chr} \ $\{\text{CHR}\} -\text{recode vcf} -\text{out} \ $\{\text{out}\}\text{GWAS.}\$\{\text{CHR}\}$

$\{\text{exe}_\text{plink}\} -\text{bfile} \ $\{\text{inputbed}\} -\text{chr} \ $\{\text{CHR}\} -\text{freq} -\text{out} \ $\{\text{out}\}\text{GWAS.}\$\{\text{CHR}\}$

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