

Introduction to **GWAS**

Basic Linux and the Shell

Christian Werner

(Quantitative geneticist and biostatistician) **EiB, CIMMYT**, Texcoco (Mexico)

Filippo Biscarini

(Biostatistician, bioinformatician and quantitative geneticist) **CNR-IBBA**, Milan (Italy)



HerrFaloppio

Oscar González-Recio

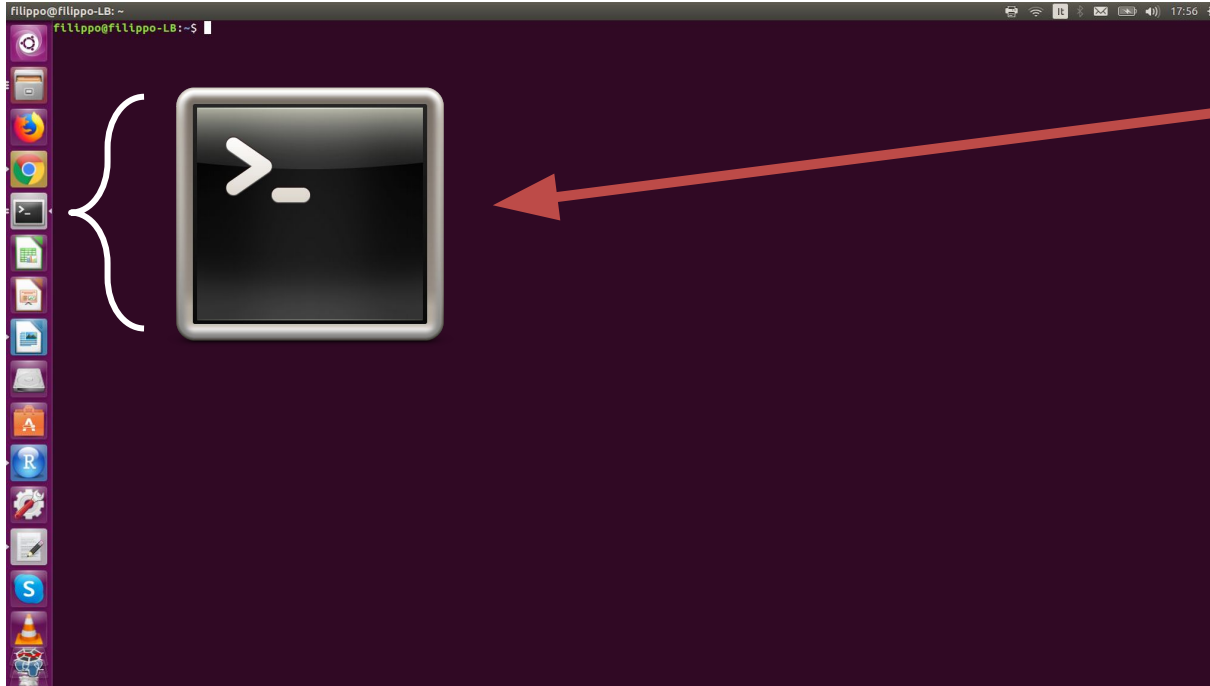
(Computer biologist and quantitative geneticist) **INIA-UPM**, Madrid (Spain)



OscarGenomics



a light touch on Linux and the command line - **let's start!**



click on the **terminal icon**
to launch “**the shell**”

- similar in Mac OS
- On Windows we use MobaXterm

a light touch on Linux and the command line - **let's start!**

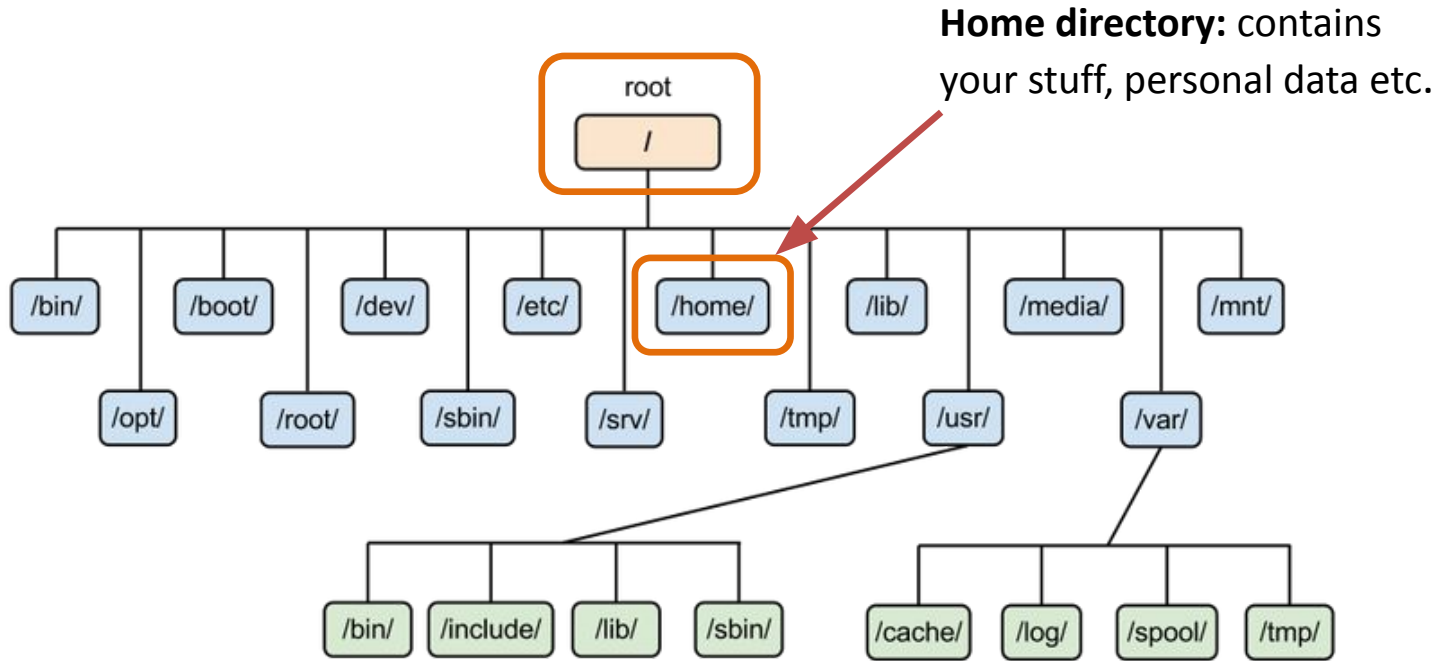
```
bash /Users/flavio
bash-3.2$ help
GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin16)
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

JOB_SPEC [θ]                (( expression ))
. filename [arguments]      :
[ arg... ]                  [[ expression ]]
alias [-p] [name=value] ... ] bg [job_spec ... ]
bind [-lpsVPS] [-m keymap] [-f fi break [n]
builtin [shell-builtin [arg ...]] caller [EXPR]
case WORD in [PATTERN] [PATTERN]. cd [-L|-P] [dir]
command [-pVv] command [arg ...] compgen [-abcdefgjkusv] [-o option
complete [-abcdefgjkusv] [-pr] [-o continue [n]
declare [-afFirtx] [-p] [name=val dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ...] echo [-neE] [arg ...]
enable [-pnds] [-a] [-f filename] eval [arg ...]
exec [-cl] [-a name] file [redirec exit [n]
export [-nf] [name=value] ... ] or false
fc [-e ename] [-nlr] [first] [last fg [job_spec]
for NAME [in WORDS ... ];] do COMMA for (( exp1; exp2; exp3 )); do COM
function NAME { COMMANDS ; } or NA getopt optstring name [arg]
hash [-lr] [-p pathname] [-dt] [na help [-s] [pattern ...]
history [-c] [-d offset] [n] or hi if COMMANDS; then COMMANDS; [ elif
```

the shell takes commands from the keyboard (the “**command line**”), interprets them and passes them on to the **operating system**.

Linux file system - **hierarchy**



Linux file system - **hierarchy**

A Path in Linux is a unique location to a file or a folder in a file system

Absolute Path

- Specified location of a file or directory from the root directory (/)
- “Complete path” from the start of the file system
- `cd /home/christian/music/Slayer`

Relative Path

- path related to the present working directory (pwd)
- Starts from the directory you are in at the moment
- `cd music/Slayer`

*print **w**orking **d**irectory*

`$~ pwd`

Overview

- **Change directory**
- **Check directory content**
- **Create and remove directories**
- **Copy and move directories and files**
- **Check files**
- **Create, modify and remove files**

Linux file system - **change directory**

change directory

\$~ cd

cd has different flavours ...

- `cd ..` (relative path - one directory up)
- `cd ../../` (relative path - two directories up)
- `cd music/Slayer` (relative path - only move down)
- `cd /user/music/` (absolute path - up and down)

Tab key

1x autocomplete path (if unique)

2x show all path options

Linux file system - **directory listing**

Check out what is in your folder:

list screen

\$~ ls *directory* (relative or absolute path)

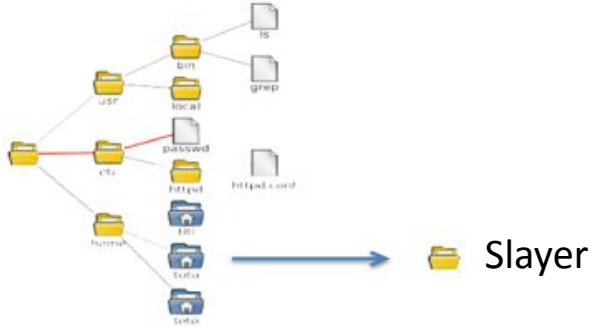
Enhanced options

- `ls -a` (including hidden files)
- `ls -l` (long format: more details)
- `ll` (alternative to `ls -l`)
- `ls -lr` (long + reverse order)
- `ls -lt` (long + sorted by time)

Check manual!

\$~ man ls

Linux file system - create and remove directories



make *directory*

```
$~ mkdir Slayer
```

remove *directory*

```
$~ rmdir Slayer
```

or

```
$~ rm -R Slayer
```

rm is more general

- removes each file specified in the command line
- by default, it does not remove directories (-R is necessary)

Linux file system - move (rename) and copy files

move (rename) a file or directory

```
$~ mv <path_to_file> <path_to_destination>
```

Danger of overwriting!!

Ask if file should be overwritten

```
$~ mv -i <path_to_file> <path_to_destination>
```

Do not overwrite file

```
$~ mv -n <path_to_file> <path_to_destination>
```

Do overwrite file

```
$~ mv -f <path_to_file> <path_to_destination>
```

copy a file

```
$~ cp <path_to_file> <path_to_destination>
```

Danger of overwriting!!

```
$~ cp -i <path_to_file> <path_to_destination>
```

```
$~ cp -n <path_to_file> <path_to_destination>
```

```
$~ cp -f <path_to_file> <path_to_destination>
```

Linux file system - check files without opening

`$~ more <path_to_file>` {scroll down with Enter/Space; quit with q}

`$~ head -n <path_to_file>` {n = number or rows to show; e.g. -10}

`$~ tail -n <path_to_file>` {same as head but from the end}

```
tail -n +2 my_file
```

← (view everything but first line)

Linux file system - word count

word count

`$~ wc <path_to_file>`

```
Harpia:CEU_daughter_high_coverage pabloorozco$ wc head10.sam
  10   31  231 head10.sam
```

lines

words

characters

`$~ wc -l <path_to_file>` how many lines?

`$~ wc -w <path_to_file>` how many words?

`$~ wc -m <path_to_file>` how many characters?

Text editors in the command line

Various text editors to choose from...

vi / vim
gedit
textpad
emacs

Create and open a file using vim

`$~ vim <my_file>`

Text editing in the shell is much more powerful than graphical text editors like notepad.

- Notepad++ is a nice graphical text editor



Exiting Vim

```
:w - Write (Save)
:wq - Write and quit
:q - Quit, fails if unsaved
:q! - Quit, even if unsaved
```

Movement

```
$ - Jump to end of line
^ - Jump to start of line
h - Move left
j - Move down
k - Move up
l - Move right
```

Modes

```
ESC - Return to normal mode
i - Insert at cursor position
a - Insert after cursor position
o - Insert on line below cursor
v - Enter visual mode
ctrl+v - Enter visual mode (vertical)
V - Enter visual mode (full lines)
```

Text editors in the command line

bash scripts

- plain text file which contains a series of commands
- Anything you can run normally on the command line can be put into a script and it will do exactly the same thing

```
#!/bin/sh

pwd
ls

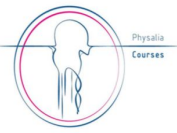
COURSE="introduction to GWAS"
echo $COURSE
```

Always starts with the “**shebang**” (which shell to use)

... followed by commands

Running bash scripts `$~ ./<my_script>`

Connect to our instance on **Amazon Web Services (AWS)**



AWS: on-demand cloud computing platform

```
filippo@filippo-LB:~$ cd Dropbox/cursos/laval2019/  
filippo@filippo-LB:~/Dropbox/cursos/laval2019$ ssh -i GWAS2019.pem ubuntu@ec2-54-190-27-29.us-west-2.compute.amazonaws.com
```

We will set up your connection now.

Windows first, then Mac / Linux.