The Unix Shell

bioinformatics@age.mpg.de

https://mpg-age-bioinformatics.github.io

References:
https://github.com/tracykteal/shell-genomics-edamame
http://www.datacarpentry.org/lessons/#genomics-workshop
https://software-carpentry.org/lessons/

Outline

- What is the shell?
- Geeks and repetitive tasks
- Resources
- How to access the sell
- Starting with the shell
- Arguments
- The Unix directory file structure
- Examining the contents of other directories
- Shortcut: Tab Completion
- Full vs. Relative Paths
- Symbolic links
- Saving time with wild cards
- Command history
- Examining files
- Searching files
- Redirection
- Creating, moving, copying, and removing
- Writing files
- Running programs
- Running scripts
- Permissions
- Loops
- Logical statements
- Shell files
- Others

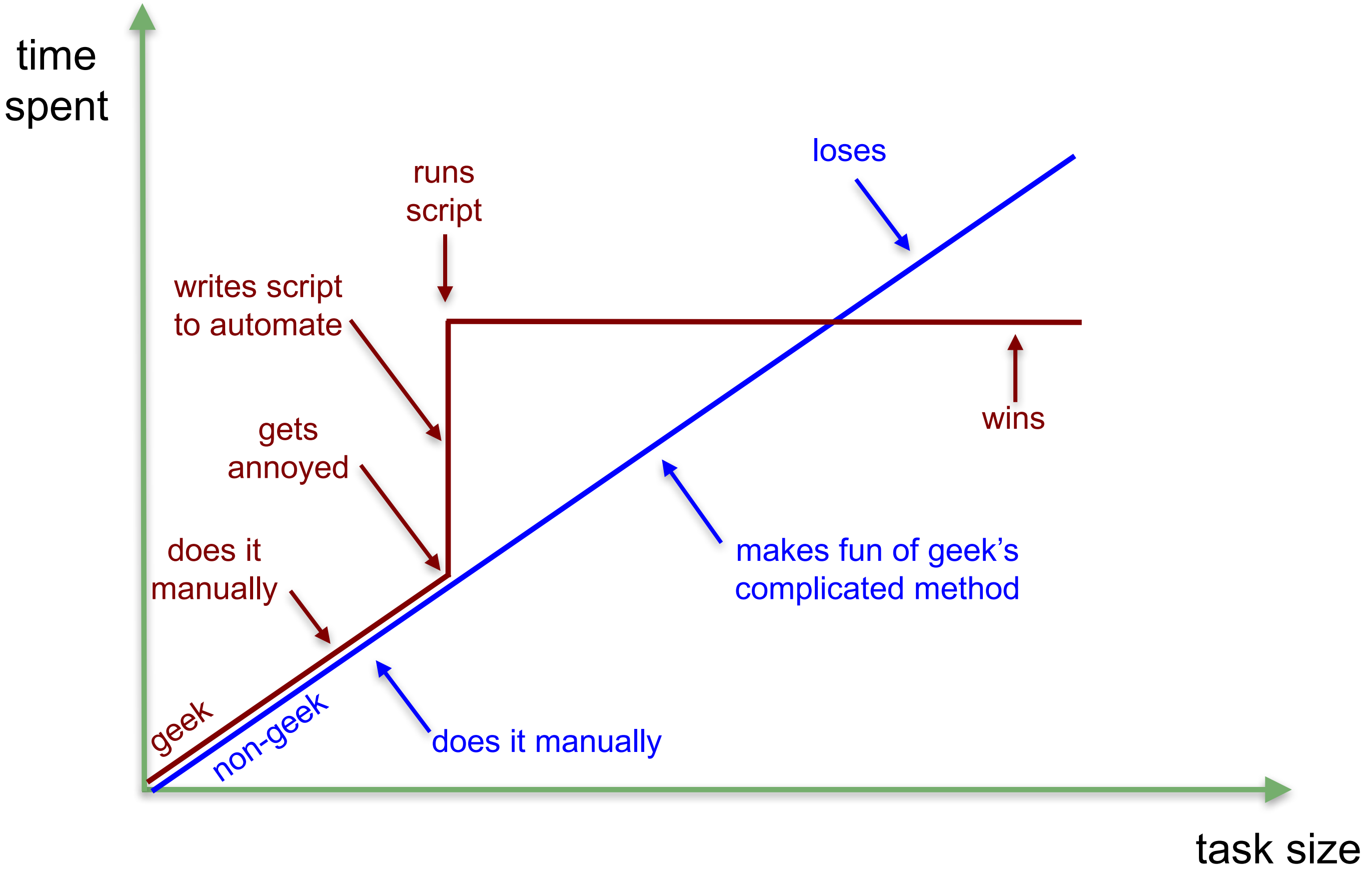
What is the shell?

The shell is a program that presents a command line interface which allows you to ***control your computer using commands entered with a keyboard*** instead of controlling graphical user interfaces (GUIs) with a mouse/keyboard combination.

There are many reasons to learn about the shell.

- ***For most bioinformatics tools, you have to use the shell.*** There is no graphical interface. If you want to work in metagenomics or genomics you're going to need to use the shell.
- The shell gives you power. The command line gives you the ***power to do your work more efficiently and more quickly***. When you need to do things tens to hundreds of times, knowing how to use the shell is transformative.
- ***To use remote computers or cloud computing, you need to use the shell.***

Geeks and repetitive tasks



Resources

Shell cheat sheets:

http://fosswire.com/post/2007/08/unixlinux-command-cheat-sheet/
https://files.fosswire.com/2007/08/fwunixref.pdf
https://github.com/swcarpentry/boot-camps/blob/master/shell/shell_cheatsheet.md

Explain shell - a web site where you can see what the different components of a shell command are doing.

http://explainshell.com
http://www.commandlinefu.com

Carpentry:

https://software-carpentry.org/lessons/
http://www.datacarpentry.org/lessons/#genomics-workshop

Tutorials:

http://tldp.org/LDP/Bash-Beginners-Guide/html/index.html
http://guide.bash.academy

Interactive learning:

https://www.codecademy.com/learn/learn-the-command-line

How to access the shell

The shell is already available on Mac and Linux.

For Windows, you'll have to download a separate program.

Mac

On Mac the shell is available through Terminal

Applications -> Utilities -> Terminal

Go ahead and drag the Terminal application to your Dock for easy access.

Windows

For Windows, we're going to be using gitbash.

Download and install http://msysgit.github.io on your computer.

Open up the program.

Starting with the shell

Enter the following commands:

```
git clone https://github.com/mpg-age-bioinformatics/shell-genomics-edamame.git
```

This command will grab all of the data needed for this workshop from the internet. (We're not going to talk about git right now, but it's a tool for doing version control.)

```
cd shell-genomics-edamame
```

`cd` stands for 'change directory’

```
ls
```

`ls` stands for ‘list‘ and it lists the contents of a directory

Starting with the shell

```
cd edamame-data
ls -F
```

Anything with a "/" after it is a directory.

Things with a "*" after them are programs.

It there's nothing there it's a file.

```
ls -l
```

`ls -l` gives a lot more information too, such as the size of the file:

<file permissions> <number of links> <owner> <group> <size> <month> <day> <time> <filename>

Arguments

Most programs take additional arguments that control their exact behavior. For example, `-F` and `-l` are arguments to `ls`.

```
man ls
```

This will open the manual page for `ls`. Use the space key to go forward and b to go backwards. When you are done reading, just hit `q` to quit.

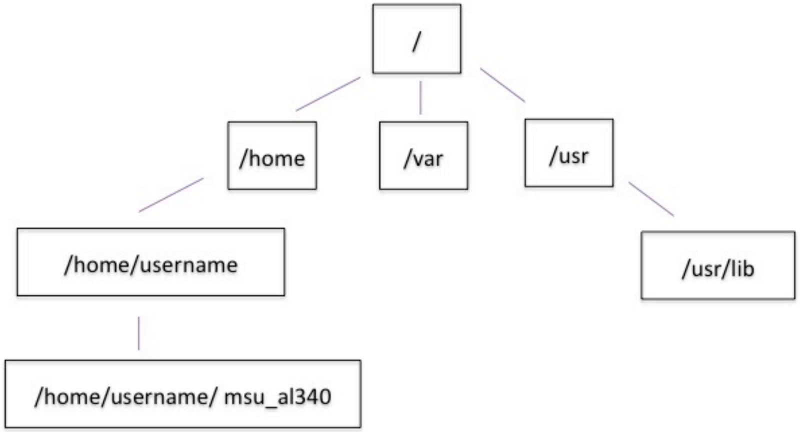
Alternative:

```
ls ––help
```

Programs that are run from the shell can get extremely complicated. To see an example, open up the manual page for the `find` program. No one can possibly learn all of these arguments, of course. So you will probably find yourself referring back to the manual page frequently.

The Unix directory file structure (a.k.a. where am I?)

When we started we first did something like go to the folder of our username. Then we opened `'tutorial-shell-genomics'` then `'data’`.



This is called a hierarchical file system structure, like an upside down tree with root (/) at the base that looks like this. That (/) at the base is often also called the 'top' level.

When you are working at your computer or log in to a remote computer, you are on one of the branches of that tree, your home directory (/home/username).

The Unix directory file structure (a.k.a. where am I?)

```
whoami
```

This tells you your user name.

Alternative:

```
echo $USER
```

`USER` is a variable which can be accessed with `$USER` or `${USER}`.

```
J=Jorge
echo ${J}
```

Spaces …

```
J=“Jorge Boucas”
echo ${J}
```

The Unix directory file structure (a.k.a. where am I?)

```
cd
```

This puts you in your home directory.

Now using `cd` and `ls`, go in to the 'tutorial-shell-genomics' directory and list its contents.

Let's also **check to see where we are**. Sometimes when we're wandering around in the file system, it's easy to lose track of where we are and get lost.

```
pwd
```

This stands for 'print working directory'. The directory you're currently working in.

To go 'back up a level' we need to use `..` ie.

```
cd ..
```

Exercise

Move around in the `'hidden'` directory and try to find the file `'youfoundit.txt’`.

Examining the contents of other directories

```
cd
ls shell-genomics-edamame
```

This will list the contents of the `edamame-data` directory without you having to navigate there.

```
cd
cd shell-genomics-edamame/edamame-data/hidden
```

This will make you jump directly to `hidden` without having to go through the intermediate directory.

Exercise

Try finding the `'anotherfile.txt'` file without changing directories.

Shortcut: Tab Completion

```
 cd
cd s<tab>
```

The shell will fill in the rest of the directory name for `shell-genomics-edamame`.

Now go to `shell-genomics-edamame/edamame-data/MiSeq`

```
ls F3D<tab><tab>
```

When you hit the first tab, nothing happens. The reason is that there are multiple files in the directory which start with `F3D`.

Tab completion can also fill in the names of programs. For example, enter `e<tab><tab>`. You will see the name of every program that starts with an `e`. One of those is `echo`. If you enter `ec<tab>` you will see that tab completion works.

Full vs. Relative Paths

```
cd
pwd
```

The output is the full name of your home directory. This tells you that you are in a directory called `username`, which sits inside a directory called `home` which sits inside the very top directory in the hierarchy. The very top of the hierarchy is a directory called `/` which is usually referred to as the *root directory*.

```
cd /home/<username>/shell-genomics-edamame/edamame-data/hidden
```

This is the full path approach.

```
cd
cd edamame-data/shell/hidden
```

This is the relative path approach.

Symbolic links

```
cd
ln -s cd /home/<username>/shell-genomics-edamame/edamame-data/hidden hid
```

This will create a symbolic link called `hid` which will take you to the folder `/home/<username>/shell-genomics-edamame/edamame-data/hidden`.

Check out the content of your home folder:

```
ls -l ~/
```

Check out the contents of `hid`:

```
ls hid
```

Remove a symbolic link:

```
unlink hid
```

Exercise

List the contents of the `/bin` directory. Do you see anything familiar in there?

Our data set: FASTQ files

We did an experiment and want to look at the bacterial communities of **mice in two treatments** using 16S sequencing. We have **10 mice in one treatment and 9 in another** treatment. We also sequenced a **Mock community**, so we can check the quality of our data. So, we have **20 samples** all together and we've done **paired-end MiSeq sequencing**.

We get our data back from the sequencing center as **FASTQ files**, and we stick them all in a folder called MiSeq. This data is actually the data we're going to use for several sections of the course, and it's data generated by Pat Schloss.

We want to be able to look at these files and do some things with them.

Saving time with shortcuts

```
cd; cd shell-genomics-edamame; cd edamame-data
```

`;` is just another way of doing a line break.

Now try:

```
ls ~
```

This prints the contents of your home directory, without you having to type the full path.

The shortcut `..` always refers to the directory above your current directory. Thus:

```
ls ..
```

prints the contents of the `/home/<username>/shell-genomics-edamame`. You can chain these together, so:

```
ls ../../
```

prints the contents of `/home/<username>` which is your home directory.

Saving time with wild cards

Navigate to the `~/shell-genomics-edamame/edamame-data/MiSeq` directory.

The `*` character is a shortcut for "everything". Thus, if you enter `ls *`, you will see all of the contents of a given directory. Now try this command:

```
ls *fastq
```

This lists every file that ends with a `fastq`. This command:

```
ls /usr/bin/*.sh
```

lists every file in `/usr/bin` that ends in the characters `.sh`.

Saving time with wild cards

We have paired end sequencing, so for every sample we have two files. If we want to just see the list of the files for the forward direction sequencing we can use:

```
ls *R1*fastq
```

What happens if you do `R1*fastq`?

Exercise

Do each of the following using a single `ls` command without navigating to a different directory.

1. List all of the files in `/bin` that start with the letter 'c'
2. List all of the files in `/bin` that contain the letter 'a'
3. List all of the files in `/bin` that end with the letter 'o'

BONUS: List all of the files in '/bin' that contain the letter 'a' or 'c'

Command History

The ***up/down arrow*** takes you backwards/forwards in the command history.

^-C will cancel the command you are writing, and give you a fresh prompt.

^-R will do a reverse-search through your command history.

You can also review your recent commands with the `history` command:

```
history
```

You can reuse one of these commands directly by referring to the number of that command.

If your history looked like this:

```
   259  ls *
   260  ls /usr/bin/*.sh
   261  ls *R1*fastq
```

then you could repeat command #260 by simply entering:

```
!260
```

Exercise

Find the line number in your history for the last exercise (listing files in /bin) and reissue that command.

Examining Files

The easiest way to examine a file is to just print out all of the contents using the program `cat`. Enter the following command:

```
cat F3D0_S188_L001_R1_001.fastq
```

Short Exercises

1. Print out the contents of the `~/shell-genomics-edamame/edamame-data/MiSeq/stability.files` file. What does this file contain?

2. Without changing directories, (you should still be in `edamame-data`), use one short command to print the contents of all of the files in the `/home/<username>/shell-genomics-edamame/edamame-data/MiSeq` directory.

Examining Files

`cat` is a terrific program, but when the file is really big, it can be annoying to use. The program, `less`, is useful for this case. Enter the following command:

```
less F3D0_S188_L001_R1_001.fastq
```

Some commands in `less`:

Key	Action
“space”	to go forward
“b”	to go backward
“g”	to go to the beginning
“G”	to go to the end
“q”	to quit

`less` also gives you a way of searching through files. Just hit the "/" key to begin a search. Enter the name of the word you would like to search for and hit enter. It will jump to the next location where that word is found. Try searching the `dictionary.txt` file for the word "cat".

Examining Files

If you hit "/" then "enter", `less` will just repeat the previous search. **`less` searches from the current location and works its way forward.** If you are at the end of the file and search for the word "cat", `less` will not find it. You need to go to the beginning of the file and search.

For instance, let's search for the sequence `1101:14341` in our file.

You can see that we go right to that sequence and can see what it looks like.

Remember, the `man` program actually uses `less` internally and therefore uses the same commands, so you can search documentation using "/" as well!

Examining Files

The commands are `head` and `tail` and they just let you look at the beginning and end of a file respectively.

```
less F3D0_Shead F3D0_S188_L001_R1_001.fastq
tail F3D0_S188_L001_R1_001.fastq188_L001_R1_001.fastq
```

The `-n` option to either of these commands can be used to print the first or last `n` lines of a file. To print the first/last line of the file use:

```
 less –n 1 F3D0_Shead F3D0_S188_L001_R1_001.fastq
tail –n 1 F3D0_S188_L001_R1_001.fastq188_L001_R1_001.fastq
```

Searching files

`grep` is a command-line utility for searching plain-text data sets for lines matching a string or regular expression.

Let's search for that sequence 1101:14341 in the F3D0_S188_L001_R1_001.fastq file.

```
grep 1101:14341 F3D0_S188_L001_R1_001.fastq
```

What if we wanted all four lines, the whole part of that FASTQ sequence, back instead.

```
grep –A 3 1101:14341 F3D0_S188_L001_R1_001.fastq
```

The `–A` flag stands for "after match" so it's returning the line that matches plus the three after it.

The `–B` flag returns that number of lines before the match.

Exercise

Search for the sequence 'TTATCCGGATTTATTGGGTTTAAAGGGT' in the F3D0_S188_L001_R1_001.fastq file and in the output have the sequence name and the sequence. e.g.

```
@M00967:43:000000000-A3JHG:1:2114:11799:28499 1:N:0:188
TACGGAGGATGCGAGCGTTATCCGGATTTATTGGGTTTAAAGGGTGCGTAGGCGGGATGCAG
```

Search for that sequence in all the FASTQ files.

Redirection

All those sequences just went whizzing by with grep. How can we capture them?

The redirection command for putting something in a file is `>`

```
grep -B 2 TTATCCGGATTTATTGGGTTTAAAGGGT * > good-data.txt
ls
```

`>>` does the same as `>` but appending the ouput to file in case it already exists.

Redirection

The `|` takes the output that scrolling by on the terminal and then can run it through another command.

```
grep TTATCCGGATTTATTGGGTTTAAAGGGT * | less
```

Now we can use the arrows to scroll up and down and use `q` to get out.

`wc` stands for `word count`. It counts the number of lines or characters.

```
grep TTATCCGGATTTATTGGGTTTAAAGGGT * | wc
```

That tells us the number of lines, words and characters in the file. If we just want the number of lines, we can use the `-l` flag for `lines`.

```
grep TTATCCGGATTTATTGGGTTTAAAGGGT * | wc -l
```

Creating, moving, copying, and removing

Navigate to the `data` directory. Lets copy the file using the `cp` command.

```
cp stability.files stability.files_backup
```

The `mkdir` command is used to make a directory.

```
mkdir backup
```

We can move files around using the command `mv`.

```
mv stability.files_backup backup/
```

This moves `stability.files_backup` into the directory `backup/` or the full path would be `~/edamame-data/shell/MiSeq/backup`

Creating, moving, copying, and removing

The `mv` command is also how you rename files.

```
mv stability.files stability.files_IMPORTANT
```

The `rm` file removes the file. Be careful with this command. It doesn't just nicely put the files in the Trash. They're really gone.

```
rm backup/stability.files_backup
```

By default, ` rm `, will NOT delete directories. You can tell ` rm ` to delete a directory using the `-r` option. Let's delete that `new` directory we just made. Enter the following command:

```
rm -r new
```

Exercise

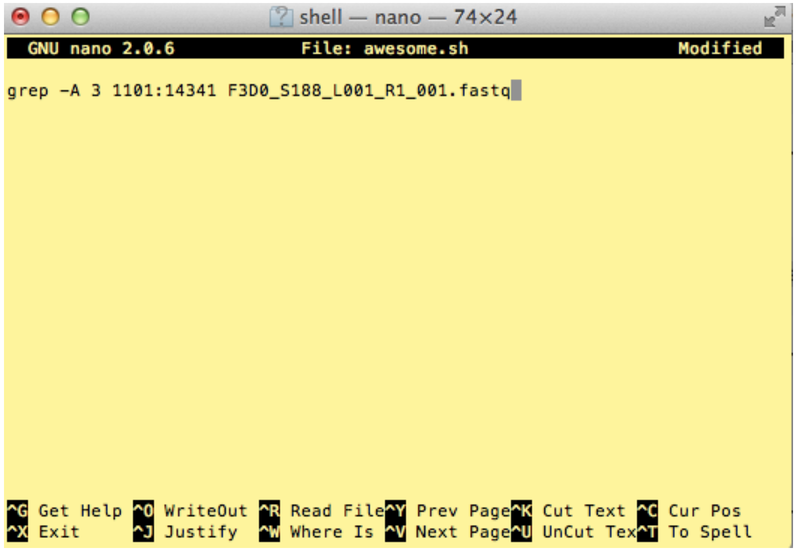
Do the following:

1. Rename the `` `stability.files_IMPORTANT` `` file to `` `stability.files` ``.
2. Create a directory in the `` `MiSeq` `` directory called `` `new` ``
3. Then, copy the `` `stability.files` `` file into `` `new` ``

Writing files

To write in files, we're going to use the program `nano`. We're going to create a file that contains the favorite grep command so you can remember it for later. We'll name this file `'awesome.sh’`.

```
nano awesome.sh
```



```
  GNU nano 2.0.6              File: awesome.sh                     Modified

grep -A 3 1101:14341 F3D0_S188_L001_R1_001.fastq
```

Type `Ctrl-X`. It will ask if you want to save it. Type `y` for yes. Then it asks if you want that file name. Hit 'Enter'.

Exercise

Open 'awesome.sh' and add "echo AWESOME!" after the grep command and save the file.

We're going to come back and use this file in just a bit.

Running programs

Commands like `ls`, `rm`, `echo`, and `cd` are just ordinary programs on the computer. A program is just a file that you can *execute*. The program `which` tells you the location of a particular program. For example:

```
which ls
which find
```

When we enter a program name and hit enter, there are a few standard places that the shell automatically looks. Enter the command:

```
echo $PATH
```

Running programs

Navigate to the `edamame-data` directory and list the contents. Try to run the program `hello.sh` by entering:

```
hello.sh
```

You should get an error saying that hello.sh cannot be found. That is because the directory `/home/username/edamame-data/shell` is not in the `PATH`. You can run the `hello.sh` program by entering:

```
./hello.sh
```

Remember that `.` is a shortcut for the current working directory. You can run `hello.sh` equally well by specifying one of the following options:

```
/home/<username>/shell-genomics-edamame/edamame-data/hello.sh
~/shell-genomics-edamame/edamame-data/hello.sh
```

Running scripts

Go in to the 'MiSeq' directory where we created 'awesome.sh' before.

It's a command, so we should just be able to run it. Give it a try.

```
./awesome.sh
```

To do that we have to make it 'executable'. We do this by changing its mode. The command for that is `chmod` - change mode. We're going to change the mode of this file, so that it's executable and the computer knows it's OK to run it as a program.

```
chmod +x awesome.sh
```

Try running it again:

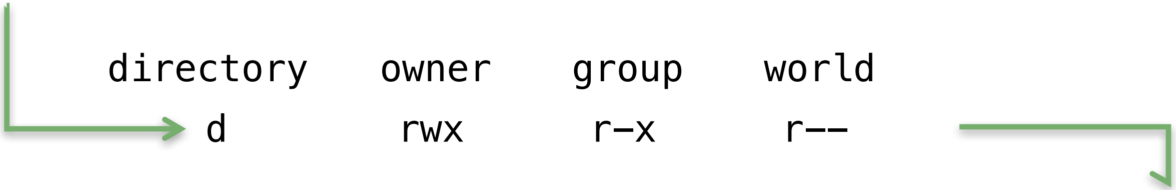
```
./awesome.sh
```

Alternative, change permissions while changing mode:

```
chmod 755 awesome.sh ; ./awesome.sh
```

Permissions

```
-rwxr-xr--   1 amrood   users 1024  Nov 2 00:10  myfile
drwxr-xr--   1 amrood   users 1024  Nov 2 00:10  mydir
```



```
directory   owner   group   world
    d        rwx     r-x     r--
```

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

Loops

Syntax:

```
for      in                              ;
    do

done
```

Example:

```
for name in jorge franziska rafael daniel ;
    do
    echo ${name}
done
```

Alternative:

```
items_variable="jorge franziska rafael daniel"
for item in ${items_variable} ;
    do
    echo ${item}
done
```

Loops

Syntax:

```
while           ;
    do

done
```

Example:

```
while read line ;
    do
    echo ${line}
done < file.txt
```

Logical statements

Syntax:

```
if [                ];
    then


else;


fi
```

Example:

```
J=Jorge

if [ ${J} == "Jorge" ] ;
    then
    echo ${J}

else;
    echo "This is not Jorge"

fi
```

Shell files

From `man bash`:

```
/bin/bash
       The bash executable
/etc/profile
       The systemwide initialization file, executed for login shells
~/.bash_profile
       The personal initialization file, executed for login shells
~/.bashrc
       The individual per-interactive-shell startup file
~/.bash_logout
       The individual login shell cleanup file, executed when a login
shell exits
~/.inputrc
       Individual readline initialization file
```

Example `~/.bash_profile`

```
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
export PATH=~/soft/bin:$PATH
source /software/Modules/modules.rc
export PS1="\[\033[01;32m\]\h\[\033[00m\]:\[\033[01;34m\]\$CurDir\$\[\033[00m\] "
```

Others

Disk usage of a folder. Example, your home folder:

```
cd ~ ; du -sh .
```

Connecting to a remote server over a Secure Shell (ssh) – ssh username@remote.adress :

```
ssh JBoucas@amalia.age.mpg.de
```

Copying files over ssh to your home folder on a remote server:

```
scp file.txt UName@ServerAddress:~/
```

Copying files over ssh from your home folder on a remoter server:

```
scp UName@ServerAddress:~/file.txt .
```

Both `scp` will only allow you to copy files (not directories) unless you use the `-r` argument for `recursively`.

END

bioinformatics@age.mpg.de

https://mpg-age-bioinformatics.github.io