

Outline

Lectures

Introducción

Command line

Conocer los alrededores

Conectando comandos

Algunos comandos útiles

Una encuesta rápida

Data science es ...

- ... acerca de datos
- ... utilizar el método científico
- ... soportar la toma de decisiones
- ... aumentar la inteligencia

El científico de datos

- No es acerca de R o python únicamente
- No es acerca de *machine learning* únicamente
- El científico de datos *crea producto de datos*.

Productos de datos

- Es un sistema tecno-social que procesa datos de sobre un sistema generador de los mismos para aumentar la inteligencia o capacidades de acción de un agente externo al sistema.

Disminución del *dolor*

Cualquier barrera, impedancia entre tu cerebro y los datos hará tu trabajo menos efectivo y te causará mucho dolor.

Tener las herramientas adecuadas, saber que esas herramientas existen y conocer cómo y cuándo utilizarlas es lo que distingue a una persona experta.

Filosóficamente: *Dataísmo*

- Believing without evidence is always morally wrong, Francisco Mejía Uribe, Aeon (2018)
- The Philosophy of Data, David Brooks, The New York Times (2013)
- Augmenting Human Intellect: A Conceptual Framework Douglas C. Engelbart (1962)

Filosóficamente: *Dataísmo*

*A **Dataist** is someone that trusts Big Data (above) and computer algorithms, in preference to human knowledge and wisdom.*

***Homo Deus: A Brief History of Tomorrow.** Yuval Noah Harare (2017)*

La computadora

Las computadoras sólo hacen cuatro cosas:

- Ejecutan programas
- Almacenan datos
- Se comunican entre sí para hacer las cosas recién mencionadas.
- Interactúan con nosotros.
 - La interacción puede ser gráfica (como están acostumbrados) conocida también como **GUI** (*Graphical User Interface*) vía el ratón u otro periférico, o desde la línea de comandos, llamada como **CLI** (*Command Line Interface*).

¿Por qué?

En muchas ocasiones, se verán en la necesidad de responder muy rápido y en una etapa muy temprana del proceso de **big data** . Las peticiones regularmente serán cosas muy sencillas, como estadística univariable y es aquí donde es posible responder con las herramientas mágicas de UNIX.

Línea de comandos

La línea de comandos es lo que estará entre nosotros y la computadora casi todo el tiempo en este curso. De hecho, una lectura obligada (no me hagan que la deje de tarea es *In the beginning... was de command line* de [Neal Stephenson](#), el escritor de [Criptonomicon](#). La [CLI](#) es otro programa más de la computadora y su función es ejecutar otros comandos. El más popular es `bash`, que es un acrónimo de [Bourne again shell](#). Aunque en esta clase también usaremos `zsh`.

Archivos y directorios

La computadora guarda la información de una manera ordenada. El sistema encargado de esto es el 'file system'. Básicamente es un árbol de información (aunque hay varios tipos de 'file systems' que pueden utilizar modificaciones a esta estructura de datos, lo que voy a decir aplica desde su punto de vista como usuarios) que guarda los datos en una abstracción que llamamos **archivos** y ordena los archivos en **carpetas** o **directorios**, los cuales a su vez pueden contener otros **directorios**.

Muchos de los comandos del **CLI** o **shell** tienen que ver con la manipulación del **file system**.

Navegación en la terminal

Moverse rápidamente en la **CLI** es de vital importancia. Teclea en tu *terminal*

Anita lava la tina

Y ahora intenta lo siguiente:

Ctrl + a Inicio de la línea

Ctrl + e Fin de la línea

Ctrl + r Buscar hacia atrás

- Elimina el *flechita arriba*

Ctrl + b / Alt + b Mueve el cursor hacia atrás

Ctrl + f / Alt + f Mueve el cursor hacia adelante

Ctrl + k Elimina el resto de la línea (en realidad corta y pone en el búfer circular)

Ctrl + y Pega la último del búfer.

Alt + y Recorre el búfer circular.

Ctrl + d Cierra la terminal

Ctrl + z Manda a *background* el programa que se está ejecutando

¿Dónde estoy?

`pwd` Imprime el nombre del *directorio* actual

`cd ..` Cambia el directorio un nivel arriba (a el directorio "padre"¹)

`cd ~ / cd` Cambia el directorio `$HOME` (tu directorio)

¹Lo siento, terminología ... ¿Alguna sugerencia?

¿Qué hay en mi directorio (*folder*) ?

`ls` Lista los contenidos (archivos y directorios) en el directorio actual, pero no los archivos *ocultos*.

`ls -l` Lista los contenidos en formato *largo* (`-l`), muestra el tamaño de los archivos, fecha de último cambio y permisos

`tree` Lista los contenidos en el directorio actual y todos los sub-directorios en una estructura de *árbol*

`tree -L 2` Límite la expansión del *árbol* a dos niveles

`tree -hs` Muestra los archivos `shows file sizes (-s)` in human-readable format (`-h`)

¿Qué hay en mi archivo?

`head -n10 $f` Muestra el principio (*head*) del archivo, `-n` especifica el número de líneas (10).

`tail -n10 $f` Muestra la final (*tail*) del archivo.

`tail -n10 $f | watch -n1` Muestra la parte final del archivo cada segundo (usando `watch`)

`tail -f -n10 $f "sigue"(follows)` (`-f`) la parte final del archivo, cada vez que hay cambios

- Útil cuando estás ejecutando un programa que guarda información a un archivo, por ejemplo)

`wc $f` Cuenta las palabras, caracteres y líneas de un archivo

¿Dónde está mi archivo?

`find -name «lost_file_name»type f` Encuentra el archivo por nombre

`find -name «lost_dir_name»type d` Encuentra directorios por nombre

La línea de comandos permite usar comodines (*wildcards*) o expresiones regulares (*regular expressions*) para encontrar archivos:

Listar todos los archivos que tienen una extensión txt

```
ls *.txt
```

Listar todos los archivos que contienen a en el nombre y extensión txt

```
ls *a*.txt
```

Listar todos los archivos que tienen 5 caracteres en el nombre:

```
ls ??????.txt
```

Entubando

| (*pipe*) Entuba la salida de un comando al siguiente comando.

```
ls -la | wc -l
```

```
# grep "busca y selecciona" cadenas o patrones (lo v  
seq 50 | grep 3
```

Redireccionando *hacia*

>, » Redirecciona la salida de los comandos a un sumidero (e.g. un archivo, o la pantalla o la impresora).

```
ls >> prueba.dat
```

```
seq 10 > numeros.txt
```

Redireccionando *desde*

< Redirecciona desde el archivo

```
sort < prueba.dat # A la línea de comandos acomoda c  
sort < prueba.dat > prueba_sort.dat # Guardar el sor
```

Ejecución condicional

`&&` es un AND, sólo ejecuta el comando que sigue a `&&` si el primero es exitoso.

```
ls && echo "Hola"
```

```
ls && echo "Hola"
```

seq

Genera secuencias de números

```
seq 5
```

```
seq inicio step final
```

```
seq 1 2 10
```

Usando otro separador (-s) que no sea es caracter de espacio

```
seq -s '|' 10
```

Agregando *padding*

```
seq -w 1 10
```

Cambia, reemplaza o borra caracteres del stdin al stdout

```
echo "Hola_mi_nombre_es_Adolfo_De_Unánue" | tr '[:u
```

```
echo "Hola_mi_nombre_es_Adolfo_De_Unánue" | tr -d ' '
```

```
echo "Hola_mi_nombre_es_Adolfo_De_Unánue" | tr -s ' '
```


- `wc` significa *word count*
 - Cuenta palabras, renglones, bytes, etc.
- En nuestro caso nos interesa la bandera `-l` la cual sirve para contar líneas.

```
seq 30 | grep 3 | wc -l
```

head, tail

- `head` y `tail` sirven para explorar visualmente las primeras diez (*default*) o las últimas diez (*default*) renglones del archivo, respectivamente.

```
"" > cd data  
> head UFO-Dic-2014.tsv  
> tail -3 UFO-Dic-2014.tsv ""
```

cat

cat concatena archivos y/o imprime al stdout

```
echo 'Hola mundo' >> test
```

```
echo 'Adios mundo cruel' >> test
```

```
cat test
```

```
rm test
```

También podemos concatenar archivos

```
cat UFO-Nov-2014.tsv UFO-Dic-2014.tsv > UFO-Nov-Dic-2014.tsv  
wc -l UFO-Nov-Dic-2014.tsv
```

En el siguiente ejemplo redireccionamos al stdin el archivo como entrada del `wc -l` sin generar un nuevo proceso

```
< numeros.txt wc -l
```

split

- `split` hace la función contraria de `cat`, divide archivos.
- Puede hacerlo por tamaño (bytes, `-b`) o por líneas (`-l`).

```
split -l 500 UFO-Nov-Dic-2014.tsv  
wc -l UFO-Nov-Dic-2014.tsv
```

cut

- Con cut podemos dividir el archivo pero por columnas.
- Donde columnas puede estar definido como campo (-f, -d), carácter (-c) o bytes (-b).
- En este curso nos interesa partir por campo.

Creemos unos datos de prueba

```
echo "Adolfo|1978|Físico" >> prueba.psv
echo "Patty|1984|Abogada" >> prueba.psv
```

Ejecuta los siguientes ejemplos, ¿Cuál es la diferencia?

```
cut -d| -f1 prueba.psv
cut -d| -f1,3 prueba.psv
cut -d| -f1-3 prueba.psv
```

uniq

- `uniq` Identifica aquellos renglones consecutivos que son iguales.
- `uniq` puede contar (`-c`), eliminar (`-u`), imprimir sólo las duplicadas (`-d`), etc.

sort

- `sort` Ordena el archivo, es muy poderoso, puede ordenar por columnas (`-k`), usar ordenamiento numérico (`-g`, `-h`, `-n`), mes (`-M`), random (`-r`) etc.

```
sort -t " , " -k 2 UFO-Nov-Dic-2014.tsv
```

uniq y sort

- Combinados podemos tener un group by:

Group by por estado y fecha

```
cat UFO-Dic-2014.tsv |\
  cut -d$'\t' -f1,3 |\
  tr '\t' ' ' |\
  cut -d' ' -f1,3 |\
  sort -k 2 -k 1 |\
  uniq -c |\
  sort -n -r -k 1 |\
  head
```