

Práctica semana 2 bash & et al

algunos comandos generales ls

ls

ls ~/Desktop

algunos comandos generales free & friends

df -h

free

uname -a

algunos comandos generales find

```
find . -iname *.pdf
```

```
find . -iname .pdf -o -iname .txt
```

```
find . -size +6M
```

```
find -help
```

a directorio home

ls ~/

*ls ~/Doc**

crear directorio de trabajo

```
mkdir ~/Documents/linux-s2
```

a documentos

```
cd ~/Documents/linux-s2
```

de regreso

```
cd -
```

copiar archivos

```
cp ./array* ~/Documents/linux-s2
```

Parte 1

http://rous.mit.edu/index.php/Unix_commands_applied_to_bioinfo

VER / MANIPULAR CONTENIDO DE ARCHIVOS

```
head -n 5 arrayDat.txt
```

```
head -n 5 arrayAnnot.txt
```

print everything except the last line

```
head -n -1 arrayDat.txt
```

```
head -n -1 arrayAnnot.txt
```

get 1st column (i.e. probeID)

```
cut -f 1 arrayDat.txt
```

get 3rd column (i.e. geneSymbol)

```
cut -f 3 arrayAnnot.txt
```

get the first three columns

```
cut -f 1-3 arrayDat.txt
```

get the 1st and 3rd columns and redirect the output into a file (mapping probeID to geneSymbol)

```
cut -f 1,3 arrayAnnot.txt > probe2gene.txt
```

put each column in a separate tmp files

```
cut -f 1 arrayAnnot.txt > tmp1
```

```
cut -f 2 arrayAnnot.txt > tmp2
```

```
cut -f 3 arrayAnnot.txt > tmp3
```

merge lines in a new column order

```
paste tmp3 tmp1 tmp2 > arrayAnnotOrdered.txt
```


sort by 1st field (probeID)

sort probe2gene.txt

sort by the 2nd field (geneSymbol)

```
sort -k 2 probe2gene.txt
```

sort by geneSymbol in reverse order

```
sort -r -k 2 probe2gene.txt
```

sort by 2nd field (default lexicographic sort)

```
sort -k 2 arrayDat.txt
```

sort by 2nd field (numeric sort)

```
sort -n -k 2 arrayDat.txt
```

print number of lines, words and bytes

```
wc arrayDat.txt
```

print number of lines only

```
wc -l arrayDat.txt
```

print the number of fastq files in the current directory

```
ls *.fas | wc -l
```


print all distinct rows then count

```
uniq arrayAnnot.txt | wc -l
```

count distinct rows without considering the 1st field
(probeID)

```
uniq -f 1 arrayAnnot.txt |wc -l
```

report distinct gene symbols and count number of occurrences

```
uniq -f 1 -c arrayAnnot.txt
```

report repeated gene symbols

```
uniq -f 1 -d arrayAnnot.txt
```

report list of unique gene symbols

```
uniq -f 1 -u arrayAnnot.txt
```

print lines that match “chr” (lines with the words “chromosome” and/or “cytochrome” are matched).

```
grep chr arrayAnnot.txt
```

print lines that match “chr” as a whole word

```
grep -w chr arrayAnnot.txt
```

print lines that match “SS” (case sensitive)

```
grep SS arrayAnnot.txt
```


print lines that match “SS” (case insensitive, that is “SS” or “ss” or “Ss” or “sS”)

```
grep -i SS arrayAnnot.txt
```

print how many lines match the pattern “orf”

```
grep -c orf arrayAnnot.txt
```

print lines that do NOT match the pattern “orf”

```
grep -v orf arrayAnnot.txt
```

precede the matching line with the line number

```
grep -n orf arrayAnnot.txt
```

make a list with four gene symbols

```
cut -f 3 arrayAnnot.txt/head -n 5 > tmp
```

use list “tmp” to match lines in arrayAnnot.txt

```
grep -f tmp arrayAnnot.txt
```

join two files (use default field as key)

join arrayDat.txt arrayAnnot.txt

join arrayAnnot.txt arrayDat.txt

join two files (default separator is blank space)

join arrayDat.txt arrayAnnot.txt

note how joint fields are separated by blank spaces and default setting of cut do not yield the desired output

```
join arrayDat.txt arrayAnnot.txt | cut -f 6
```

note how specifying a blank space as delimiter for cut does not yield the desired output because the description field consists of multiple words

```
join arrayDat.txt arrayAnnot.txt | cut -d " " -f 6
```

join two files (separator is a tab)

```
join -t $'\t' arrayDat.txt arrayAnnot.txt
```

```
join -t $'\t' arrayDat.txt arrayAnnot.txt | cut -f 6
```

sort by key, then use the probeID as key (probeID is the 2nd and 3rd field in the files)

```
sort -k 2 probe2gene.txt > tmp1
```

```
sort -k 3 -t $'\t' arrayAnnot.txt > tmp2
```

```
join -1 2 -2 3 -t $'\t' tmp1 tmp2
```

specify which field to output for each file

```
join -1 2 -o '1.1 2.2 2.3 2.4 2.5' arrayAnnotOrdered.txt  
arrayDat.txt
```

```
join -1 2 -2 3 -t $'\t' -o '1.1 1.2 2.2' tmp1 tmp2
```

split the content into separate files of 50 lines each

```
split -l 50 arrayDat.txt
```

substitute the word “chromosome” with the string “chr”

```
sed s/chromosome/chr/g arrayAnnot.txt
```

print every line of the input file (missing condition)

```
awk '{print $0}' arrayDat.txt
```


print every line of the input file (default argument for print is \$0)

```
awk '{print}' arrayDat.txt
```

print 1st field only

awk '{print \$1}' arrayDat.txt

rearrange the fields, unseparated

```
awk '{print $1 $3 $2}' arrayDat.txt
```

rearrange the fields, separated by a blank space

```
awk '{print $1, $3, $2}' arrayDat.txt
```

rearrange the fields, separated by a tab

```
awk '{print $1, "\t", $3, "\t", $2}' arrayDat.txt
```

print the number of fields for each record

```
awk '{print NF}' arrayDat.txt
```

print the last field of each record

```
awk '{print $NF}' arrayDat.txt
```

print the last field -1 of each record

```
awk '{print $NF-1}' arrayDat.txt
```


print a string

```
awk 'BEGIN {print "Hello world!"}'
```

print the result of an arithmetic operation

```
awk 'BEGIN {print 2+3}'
```

```
awk 'BEGIN {print "2+3=" 2+3}'
```

print first 5 lines (default action statement is to print)

awk 'NR < 6' arrayDat.txt

print first line (i.e. column headers)

```
awk 'NR == 1' arrayDat.txt
```

print first 5 records, excluding headers (NR ==1)

awk 'NR > 1 && NR < 7' arrayDat.txt

print the total number of records

```
awk 'END {print NR}' arrayDat.txt
```

print sum of values for each gene

```
awk 'NR > 1 {s=0; for (i=2; i<=NF; i++) s=s+$i;  
print s}' arrayDat.txt
```

print mean of values for each gene

```
awk 'NR > 1 {s=0; n=NF-1; for (i=2; i<=NF; i++)  
s=s+$i; s=s/n; print s}' arrayDat.txt
```


print the lines that match the string “orf”

```
awk '/orf/' arrayAnnot.txt
```

print the probe IDs whose annotation contain the string
“orf”

```
awk '/orf/ {print $1}' arrayAnnot.txt
```

print number of lines matching “orf” (emulates grep -c)

```
awk '/orf/ {n++}; END {print n+0}' arrayAnnot.txt
```

```
awk '/orf/ {n++}; END {print n}' arrayAnnot.txt
```

algunos ejercicios

los datos de dengue todos desorganizados

cut para sacar las tres primeras columnas

```
cut -f 1,2,3 dengue_referencias_30.10.12_D1.csv #  
cambiando el delimitador  
cut -d, -f 1,2,3 dengue_referencias_30.10.12_D1.csv
```

PARTE 2

counting number of sequences in a fasta file:

```
grep -c "^>" DNAdata.fas
```


add something to end of all header lines:

```
sed 's/>.*/&_WHATEVERYOUWANT/' DNAdata.fas  
> modif.fas
```

clean up a fasta file so only first column of the header is outputted:

echo don't clean up a fasta file so only first column of the header is outputted

```
awk '{print $1}' DNAdata.fas > output.fas
```

To extract ids, just use the following:

```
grep -o -E “^>+” DNAdata.fas | tr -d “>”
```

A useful step is to linearize your sequences (i.e. remove the sequence wrapping).

```
sed -e 's/(>.*// ' file.fasta | tr -d "\"" | tr -d "n" | sed -e 's//#/' | tr "#" "\n" | sed -e '/^$/d'
```

Remove duplicated sequences.

```
sed -e '/^>/s/$/@/' -e 's/^>/##/' file.fasta | tr -d '\n' |  
tr "#" "\n" | tr "@" "\t" | sort -u -t $'\t' -f -k 2,2 | sed  
-e 's/^>/' -e 's/\t/\n/'
```

the song remains the same

```
sed -e '/^>/s/$/@/' -e 's/^>/#/' file.fasta |  
tr -d '\n' | tr "#" "\n" | tr "@" "\t" |  
sort -u -t " " -f -k 2,2 |  
sed -e 's/^>/' -e 's/ \t/\n/'
```

parte 3 sed en extracción

```
cat texto.txt
```

```
cat texto.txt | sed -n -e 's/.(.)</PMID>.*//p'
```

```
cat texto.txt | sed -n -e 's/.(.)</PMID>.*//p' -e  
's/.(.)</ArticleTitle>.*//p' -e  
's/.(.)</AbstractText>.*//p'
```

```
cat texto.txt | sed -n -e '///{s/.>(.)<.*//;h}' -e  
'///{s/.>(.)<.*//;H}' -e '///{s/.>(.)<.*//;H;g;s/\n/  
/g;p}'
```


history

history

a archivo

```
history > historia01.txt
```

```
sed -i 's/ / /g' historia01.txt
```

```
sed -i 's/^ //g' historia01.txt
```

```
cut -d' ' -f 2- historia01.txt
```

o mas fácil

history -w Historia.txt

TAREA PARA LA CASA crear dos archivos

labels con código de acceso año país ordenado por

país

año