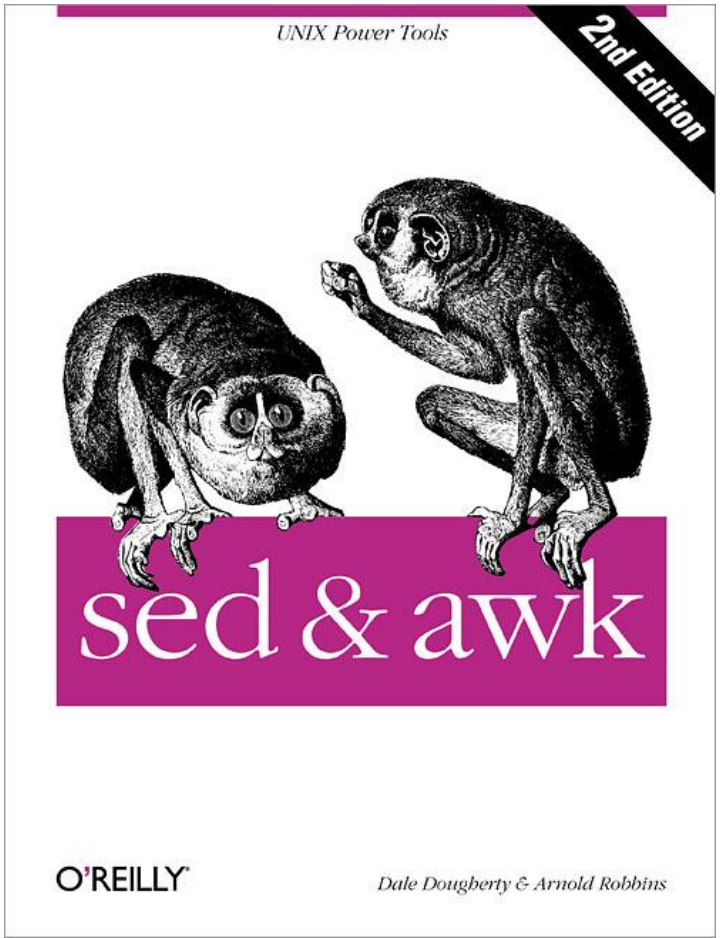
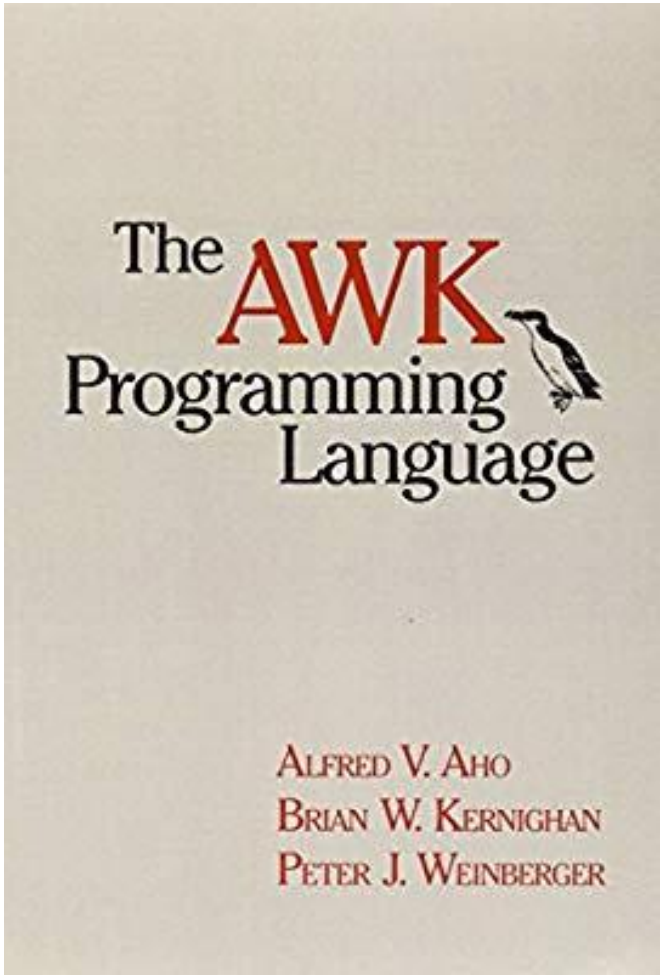
sed (stream editor)

awk (Aho,Weinberger y Kernighan)

Octubre 2025,

Fernando Alvarez





Standard input (stdin)
Standard output (stdout)

- Redireccionando el stdin y stdout
- **grep**
- g/re/p (global/regular expression/print)

- Versión extendida
- egrep
- grep -E

Expresiones regulares (1)

Una expresión regular es un patrón para buscar coincidencias con cadenas de caracteres.

La potencia de las expresiones regulares radica en su capacidad de incluir alternativas y repeticiones en la elaboración del patrón. Estas particularidades son elaboradas utilizando caracteres especiales, que no son interpretados literalmente, sino de una manera especifica.

- Algunas ***regex*** muy usadas **.** *un punto*: Cualquier carácter, excepto una nueva línea
- [ACG] : grupo de caracteres alternativos en una posición
- * (asterisco) cero o mas ocurrencias de la regex que le precede
- **^** inicio de linea

  $ final del linea

  ^$ linea vacia

- [^A-Z] sombrerito invierte la seleccion, son no mayusculas
- [1-9][0-9] número de dos digitos
- [0-9][0-9]* numero de 1 o mas digitos

Expresiones regulares

Otros elementos útiles para el manejo de texto

\n, \t, $, ^, “ ”,*, [a-z]

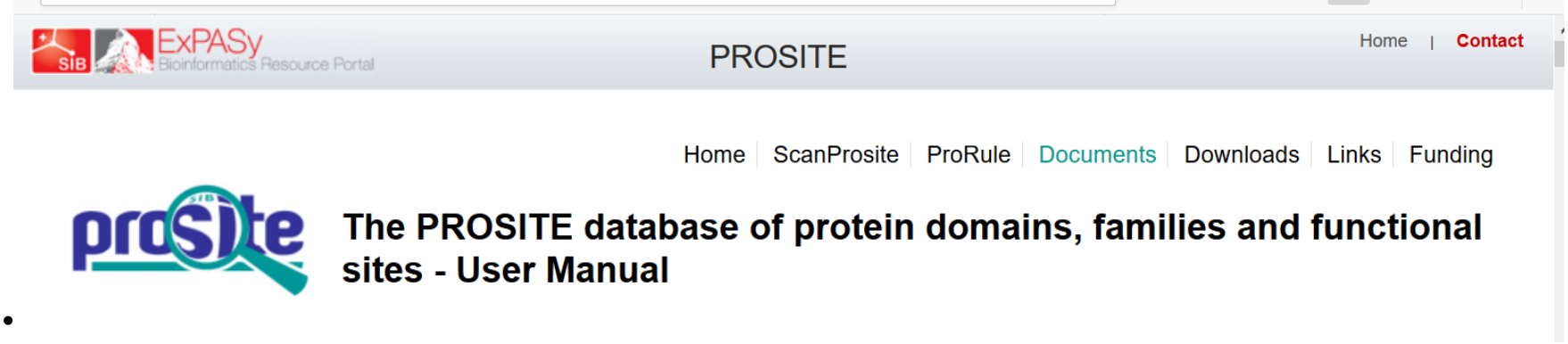
- \n salto de línea
- \t tabulador
- $ fin de línea
- ^ principio de línea
- “ “ espacio
- * todos los caracteres, repetidos n veces
- [a-z] todas las letras minúsculas una vez
- [0-9] todos los números una vez
- [ACGhstr] cualquiera de los caracteres dentro de los paréntesis rectos

Expresiones regulares (2)

las expresiones regulares de grep, sed y awk son similares, pero no identicas

Expresiones de usadas frecuentementes

[Xx]\{5\} cinco apariciones de X o x
[Xx]\{5,15\} entre cinco y 15 apariciones de X o x
^[A-Za-z][^]* primera palabra de un renglon
[A-Za-z][^]*$ ultima palabra de un renglon



-
-

The PA (PAttern) lines contains the definition of a PROSITE pattern. The patterns are described using the following conventions: The standard IUPAC one-letter codes for the amino acids are used.

- The symbol 'x' is used for a position where any amino acid is accepted.
- Ambiguities are indicated by listing the acceptable amino acids for a given position, between square parentheses '[ ]'. For example: [ALT] stands for Ala or Leu or Thr.
- Ambiguities are also indicated by listing between a pair of curly brackets '{ }' the amino acids that are not accepted at a given position. For example: {AM} stands for any amino acid except Ala and Met.
- Each element in a pattern is separated from its neighbor by a '-'.
- Repetition of an element of the pattern can be indicated by following that element with a numerical value or a numerical range between parenthesis. Examples: x(3) corresponds to x-x-x, x(2,4) corresponds to x-x or x-x-x or x-x-x-x.
- When a pattern is restricted to either the N- or C-terminal of a sequence, that pattern either starts with a '<' symbol or respectively ends with a '>' symbol. In some rare cases (*e.g.* PS00267 or PS00539), '>' can also occur inside square brackets for the C-terminal element. 'F-[GSTV]-P-R-L-[G>]' means that either 'F-[GSTV]-P-R-L-G' or 'F-[GSTV]-P-R-L>' are considered.
- A period ends the pattern.

- Examples: PA [AC]-x-V-x(4)-{ED}.

- This pattern is translated as: [Ala or Cys]-any-Val-any-any-any-any-{any but Glu or Asp}

Dedos de cinc Cis_2His_2

El grupo de plegamiento tipo Cis_2His_2 es la clase de dedos de cinc

Estos dominios adoptan un pliegue ββα y tienen el siguiente motivo de secuencia aminoacídica:

X_2-Cis-$X_{2,4}$-Cis-X_{12}-His-$X_{3,4,5}$-His

Crear series de números

Comando seq

seq 1 30

seq 1 3 30

sed, a stream editor

Invocando sed

sed /patron/comando archivo.dat

cat archivo.data | ……| sed /comando/ comando

pipeline

Comandos frecuentes

s sustituir
d eliminar

w escribir hacia un archivo
r leer desde un archivo

Direcciones, patrones y comandos sed

sed -e /patron1/comando1 -e /patron2/comando2 archivo.dat
sed /patron1/comando1 ; /patron2/comando2 archivo.dat
sed direccion1 ,direccion2 comando1 ; /patron2/comando2 archivo.dat

patron1 y patron2 son expresiones regulares

direccion1 y direccion2 se refieren al números de línea

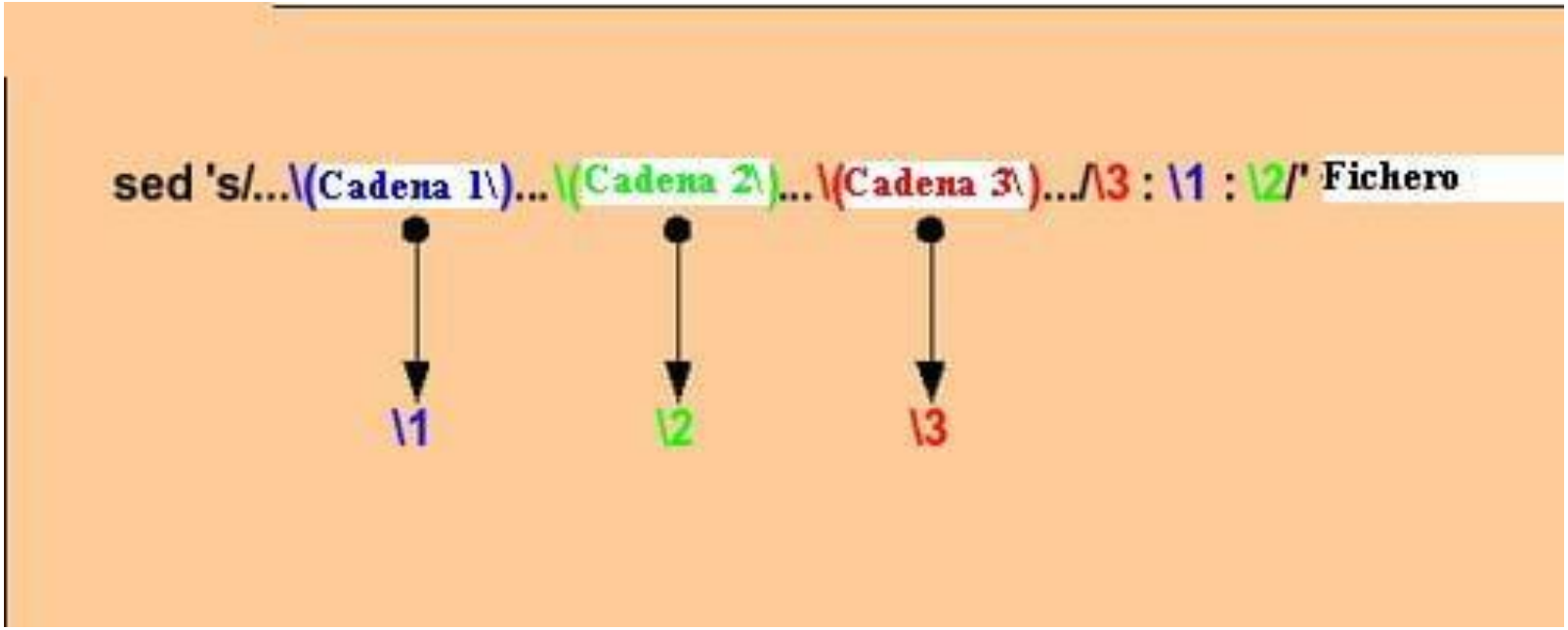
Ejemplos

sed /[Hh]ola/d file.dat **#elimina la líneas que contienen la palabra Hola u hola**

sed 3,40d file.dat **#elimina el segmento que va de la linea 3 a la 40**

sed /[Hh]ola/,50s/mundo/gente/ file.dat **#sustituye mundo por gente entre la línea que contiene "Hola" hasta la línea 50**

Colocando una regex dentro de una variable



- Uso posible: hacer una sustitucion en un contexto variable sin perder ese contexto
- (ejemplo cambiar locus tag en dm28c.genbank.gtf)

Ejercicios sed

- 1- transformar el archivo fastq en fasta
- 2- transformar un archivo fasta de varias líneas por secuencia en uno que tenga 1 solo línea en cada secuencia (usar archivo augustus.aa)
- Sugerencia usar primero el comando ***tr*** para eliminar los saltos de línea y luego sed

- 3- transformar el archivo obtenido anteriormente en un archivo multifasta que tenga 100 nucleótidos por línea

awk

invocando awk

awk ‘{comando}’ archivo.dat

Ejemplos

awk ‘{print $1}’ archivo.dat

awk ‘{print $3+4}’ archivo.dat

Podemos escribir un script en awk

awk -f script archivo.dat

awk puede un comando dentro de un pipeline de flujo

cat archivo.dat | awk ‘ {print $1,$2}’

Lectura sugerida:
http://www.sromero.org/wiki/linux/aplicaciones/uso_de_awk

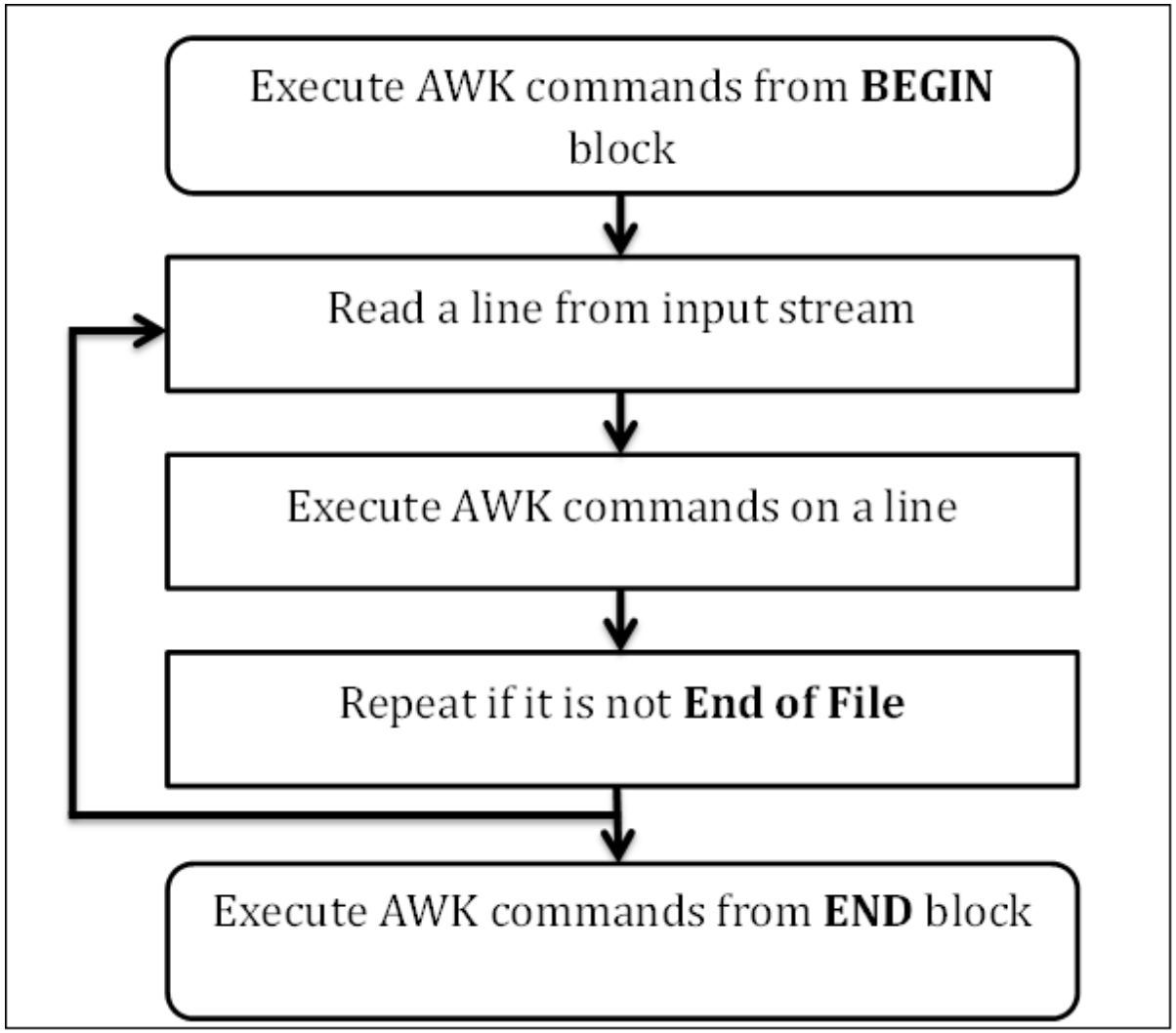
Forma genérica de awk

```
awk   'BEGIN {comandos,definiciones}     #opcional
/expreción regular/  &&  expresión lógica  # filtros opcionales
{comando1
commando2
.
comandoN
}
 END {comandos}'     archivo.data
```

```
awk   'BEGIN {print "Hola ejemplo"; FS=;}     #opcional
$3>70   # filtros opcionales
{print $1
promedio=promedio+$3
contador=contador+1
}
 END {print suma, contador, contador/suma}'     archivo.data
```

Estructura de un programa *awk*

El cuerpo de awk es un loop con un ciclo por renglón



Variables de interes predefinidas

-F separador de campos (por defecto espacio) definido afuera

FS separador de campos (por defecto espacio) definido en BEGIN

NR numero de records (renglones)

NF numero de campos (Number of Fields)

$1, $2, $NF (campo 1, campo2, campo final)

Se pueden definir variables en cualquier momento,
Bloque inicial, o en el cuerpo

Operadores Logicos

- < Menor que
- <= Menor o igual a
- > Mayor que
- >= Mayor o igual a
- != Diferente de
- == Igual a

Operadores && y ||
Para testar mas de una condicion, simultaneamente,
usamos: && - es AND y
|| es OR

Awk Arithmetic Opertors

Operator	Description
-	Subtraction
*	Multiplication
/	Division
%	Modulo Division

Operator	Description
+	Positivate the number
–	Negate the number
++	AutoIncrement
—	AutoDecrement

Operator	Description
=	Assignment
+=	Shortcut addition assignment
-=	Shortcut subtraction assignment
*=	Shortcut multiplication assignment
/=	Shortcut division assignment
%=	Shortcut modulo division assignment

Bucle for

- for (*initialization*; *condition*; *increment*) *body*

- *Ejemplo sumando amino ácidos*
- *freq_count.pl pep 1 Scerevisae.aa c*

- freq_count.pl pep 1 Scerevisae.aa c|awk -F, '{K=0;for (i=1;i<=NF;i++) K+=$i; { print K,NF}}'

ejercicios

- 1- Usando el archivo de anotacion PII.gff, imprimir las lineas de los genes cuyo largo sea mayor a 4000 nucletidos
- 2- Lo mismo que en 1, pero exclusivamente para los genes ubicado en el contig de nombre tig00000011
- 3- calcular la identidad promedio de los HSP de blast (archivo salidablast, formato tabular) cuyo largo de alineamiento sea superior a 500
- 4- Usando SOLO awk contar cuantos genes ribosomales hay en el archivo de anotacion PII.gff y calcular su largo promedio

Soluciones

- 1- sed 's/^@ER/>ER/' file.fastq |grep -A1 ^\>|sed '/\-\-/d‘
- 2- tr "\n" " " <augustus.aa |sed -e 's/>/\n>/g' |sed -e 's/>g\([^][^]*\)/>g\1\n/' -e 's/ //g‘ >1.fas
- 3- sed -e 's/\([A-Z]\{1,100\}\)/\1\n/g' 1.fas
- 4- sed -n '/g42.t1/,/>/p' augustus.aa