Nombre y Apellido: Juan Carlos García Estupiñán

Actividad 1.- Manipulación y formateo de archivos: Formato BED

Objetivo

El objetivo de esta actividad es que el estudiante aprenda a manipular y formatear archivos empleando diferentes comandos de Linux aprendidos a lo largo de las sesiones. Específicamente, se trabajará con el formato BED (*Browser Extensible Data*) que se utiliza ampliamente en bioinformática para almacenar regiones genómicas como coordenadas y anotaciones asociadas. Este formato se caracteriza por presentar los datos en forma de columnas separadas por espacios o tabuladores.

Parte I: Obtención de los datos.

Los datos con los cuales va a trabajar hacen referencia a una serie de regiones de interés detectadas en un tipo de células inmunitarias, las células B, en humanos. Para ello, se han realizado dos réplicas del experimento, obteniendo dos archivos llamados human_coordinates_1.bed y human_coordinates_2.bed. Estos dos archivos están disponibles en la propia actividad propuesta en el campus virtual:

- Actividades/Portafolio de pruebas aplicativas/Prueba aplicativa 1/human_coordinates_1.bed
- Actividades/Portafolio de pruebas aplicativas/Prueba aplicativa 1/human_coordinates_2.bed

Acceda a la ruta anterior y descárguese los datos en su entorno de trabajo. Visualice las 5 primeras líneas de cada uno de los archivos. Incluya el código empleado para realizarlo junto a una captura de pantalla (0,5 pts)

- **Script: 1head.sh**

```
File  Edit  View  Search  Terminal  Help
#!/bin/bash

## $1 $2 representarán los datos crudos de human coordinates en ../data/raw/
## EJECUCIÓN ##
## Empezaremos viendo las 5 primeras líneas de cada archivo
echo ">>> Vemos las 5 primeras líneas de los arcivos."
head -n 5 $1 $2
```

- **Stdout: 1head.sh**

```
File  Edit  View  Search  Terminal  Help
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$ ./1head.sh ../data/raw/human_coordinates_{1..2}.bed
>>> Vemos las 5 primeras líneas de los arcivos.
==> ../data/raw/human_coordinates_1.bed <==
chr3    62722434        62722633
chr3    62796034        62796233
chr3    62796234        62796433
chr3    62796434        62796633
chr3    62815834        62816033

==> ../data/raw/human_coordinates_2.bed <==
chr3    62722434        62722633
chr3    62796034        62796233
chr3    62796234        62796433
chr3    62796434        62796633
chr3    62815834        62816033
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$
```

Seguidamente, responda a cada una de las preguntas que se le indican, adicionando siempre una captura de pantalla con los comandos empleados y la respuesta obtenida por la salida estándar.

- **¿Cuántas líneas presenta cada uno de los archivos descargados? (0,5 pts)**
  - **Script: 2nrow.sh**

```
File  Edit  View  Search  Terminal  Help
1  #!/bin/bash
   1
   2 ## $1 $2 representan human coordinates en ../data/raw/
   3 ## EJECUCIÓN ##
   4 ## Contamos las líneas de cada uno de los archivos
   5 echo ">>> Número de líneas de $1"
   6 cat -n $1 | wc -l
   7
   8 ## En el siguiente hay un problema con el salto de línea, con lo cual,
   9 ## usaremos otra combinación de comandos.
  10 ## Vemos la enumeración de las líneas | la última fila | el número de esta
  11 echo ">>> Número de líneas de $2"
  12 cat -n $2  | tail -1  | cut -f1
~
```

  - **Stdout: 2nrow.sh**

```
File  Edit  View  Search  Terminal  Help
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$ ./2nrow.sh ../data/raw/human_coordinates_{1..2}.bed
>>> Número de líneas de ../data/raw/human_coordinates_1.bed
1907
>>> Número de líneas de ../data/raw/human_coordinates_2.bed
  1910
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$
```

- **¿Cuántas columnas presenta cada uno de los archivos descargados? (1 pts)**
  - **Script: 3ncol.sh**

```
File  Edit  View  Search  Terminal  Help
#!/bin/bash

## $1 y $2 representan human coordinates de ../data/raw/
## EJECUCIÓN ##
## Usamos awk, donde NF es el number of fields
echo "Vemos el número de columnas de $1"
awk -F "\t" '{print NF}' $1 | uniq

echo "Vemos el número de columnas de $2"
awk -F "\t" '{print NF}' $2 | uniq S
```

- **Stdout: 3ncol.sh**

```
File  Edit  View  Search  Terminal  Help
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$ ./3ncol.sh ../data/raw/human_coordinates_{1..2}.bed
Vemos el número de columnas de ../data/raw/human_coordinates_1.bed
3
Vemos el número de columnas de ../data/raw/human_coordinates_2.bed
3
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$
```

- **¿Tenemos representación de todos los cromosomas humanos en ambos archivos? (1 pts)**
  - **Script: 4lookchr.sh**

```
File  Edit  View  Search  Terminal  Help
#!/bin/bash


## EJECUCIÓN ##
## ¿Hay representación de todos los cromosonas humanos en ambos los archivos?.
## Vemos las listas de los cromosomas, el nº de líneas es < 23. Vemos en esta
## que falta el par 19 y los sexuales.
read -p "Inserte el nº de cromosomas de su especie: " nchr_esp ## Humanos
lookchr(){
    echo ">>> ¿Están todos los cromosomas en $1"
    echo ">> Cromosomas del archivo $1"
    cut -f1 $1 | sort -k1.4 -n | uniq
    echo ">> Nº de líneas de $1, si es < a $nchr_esp, faltan chr"
    cut -f1 $1 | sort -k1.4 -n | uniq | wc -l
}

read -p "Inserte la ruta de los datos crudos 1: " raw1
read -p "Inserta la ruta de los datos crudos 2: " raw2

lookchr $raw1
lookchr $raw2

```

  - **Stdout: 4lookchr.sh parte 1**

```
File  Edit  View  Search  Terminal  Help
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$ ./4lookchr.sh
Inserte el nº de cromosomas de su especie: 23
Inserte la ruta de los datos crudos 1: ../data/raw/human_coordinates_1.bed
Inserta la ruta de los datos crudos 2: ../data/raw/human_coordinates_2.bed
>>> ¿Están todos los cromosomas en ../data/raw/human_coordinates_1.bed
>> Cromosomas del archivo ../data/raw/human_coordinates_1.bed
chr1
chr2
chr3
chr4
chr5
chr6
chr7
chr8
chr9
chr10
chr11
chr12
chr13
chr14
chr15
chr16
chr17
chr18
chr20
chr21
chr22
>> Nº de líneas de ../data/raw/human_coordinates_1.bed, si es < a 23 , faltan chr
21
```

- **Stdout: 4lookchr.sh parte 2**

```
>>> ¿Están todos los cromosomas en ../data/raw/human_coordinates_2.bed
>> Cromosomas del archivo ../data/raw/human_coordinates_2.bed
chr1
chr2
chr3
chr4
chr5
chr6
chr7
chr8
chr9
chr10
chr11
chr12
chr13
chr14
chr15
chr16
chr17
chr18
chr20
chr21
chr22
>> Nº de líneas de ../data/raw/human_coordinates_2.bed, si es < a 23 , faltan chr
21
```

Al ser réplicas experimentales esperaríamos que ambos archivos fueran idénticos. **Para comprobarlo, primero ordene los dos archivos por el nombre del cromosoma (determinado en la primera columna). Seguidamente, compárelos para mostrar qué regiones son distintas entre ambos. Adjunte una captura de pantalla con los comandos empleados que muestren cuántas y qué regiones son distintas entre ambos archivos (2 pts)**

- **Script: 5diff.sh**

```
File  Edit  View  Search  Terminal  Help
#!/bin/bash

## $1 y $2 datos crudos human coordinates ../data/raw/

## VARIABLES ##
read -p "Inserte la ruta de los datos ordenados 1: " sort1
read -p "Inserte la ruta de los datos ordenados 2: " sort2
read -p "Inserte la ruta de las diferencias: " diff

## EJECUCIÓN ##
## Ordenamos los datos crudos de menor a mayor, cromosomas (nº) y coordenadas.
echo ">>> Ordenamos ambos datos y guardamos los resultados en un nuevo archivo"
sort -k1.4 -k 2 -n $1 > $sort1
sort -k1.4 -k 2 -n $2 > $sort2

echo ">>> En caso de diferencias ¿Cuántas y que regiones son diferentes?"
## Primero preguntamos si difieren
echo ">> ¿Difieren?"
diff $sort1 $sort2 -q
## Lo siguientes ya es ver cuáles presentan en ese caso diferencias, si no
## hay diferencias no habría ningún output.
echo ">> ¿Dónde?"
diff $sort1 $sort2 > $diff
cat $diff ## Ver las diferencias
## Eliminamos los datos ordenados para no ocupar excesivo espacio. En caso de
## fallo en los siguientes scripts, con volver a correr este se restaura lo
## importante!!!.
rm $sort1 $sort2

~
```

- **Stdout: 5diff.sh**

```
File  Edit  View  Search  Terminal  Help
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$ ./5diff.sh ../data/raw/human_coordinates_{1..2}.bed
Inserte la ruta de los datos ordenados 1: ../data/processed/sort1.bed
Inserte la ruta de los datos ordenados 2: ../data/processed/sort2.bed
Inserte la ruta de las diferencias: ../data/processed/diff.bed
>>> Ordenamos ambos datos y guardamos los resultados en un nuevo archivo
>>> En caso de diferencias ¿Cuántas y que regiones son diferentes?
>> ¿Difieren?
Files ../data/processed/sort1.bed and ../data/processed/sort2.bed differ
>> ¿Dónde?
317a318
> chr1  204073115       204127743
620a622
> chr6  31164337        31170682
1654a1657
> chr17 42313412        42388540
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$
```

Una vez identificadas estas regiones, las debe seleccionar y guardarlas en un archivo nuevo. Ojo solo tiene que guardar las tres columnas, cromosoma, coordenada de inicio y coordenada de fin de cada una de las regiones detectadas. Visualice las primeras líneas de este archivo creado. Incluya una captura de pantalla que muestre el código empleado (1 pts)

- **Script: 6diff_processed.sh**

```
File  Edit  View  Search  Terminal  Help
#!/bin/bash

## VARIABLES ##
read -p "Inserte la ruta de las diferencias: " diff
read -p "Inserte la ruta del archivo temporal: " temporary_file

## EJECUCIÓN ##
## Filtramos los valores que nos interesa. Un truco que
## podemos hacer es crear un archivo temporal y el resultado
## procesado de las diferencias sobrescribirlo con mv en
## archivo $diff original. Evitamos excesivos archivos.
grep "> chr" $diff > $temporary_file
mv $temporary_file $diff

## Eliminamos los valores "> " que estorban
## Con sed -i modificamos automáticamente los datos de las
## diferencias procesadas, a partir de los patrones y reemp_
## plazos establecidos.
sed -i "s/> chr/chr/" $diff

echo ">>> Vemos las diferencias procesadas"
cat $diff

~
```

- **Stdout: 6diff_processed.sh**

```
File  Edit  View  Search  Terminal  Help
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$ ./6diff_processed.sh
Inserte la ruta de las diferencias: ../data/processed/diff.bed
Inserte la ruta del archivo temporal: ../data/processed/temporary_file.txt
>>> Vemos las diferencias procesadas
chr1    204073115       204127743
chr6    31164337        31170682
chr17   42313412        42388540
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$
```

Ahora va a transformar el formato de estas coordenadas genómicas almacenadas. Para ello, debe sustituir el primer tabulador por dos puntos y el segundo por un guion; de forma que las coordenadas presenten la siguiente estructura: chr:inicio-fin. Fíjese en el ejemplo:

- Formato inicial: chr6 20978845 20979044

- Formato final: chr6:20978845-20979044

Incluya una captura de pantalla con el código empleado visualizando el cambio de formato de las regiones (1,5 pts)

- **Script: 7format_change.sh**

```
File  Edit  View  Search  Terminal  Help
#/bin/bash

## VARIABLES ##
read -p "Inserte la ruta de las diferencias: " diff
## Formato para buscar en el Genenome Browser:
read -p "Ruta para archivo de Genome Browser: " genome_browser

## EJECUCIÓN ##
## Cambiamos el formato anterior
awk '{print $1 ":"  $2 "-" $NF }' $diff > $genome_browser

echo ">>> Visualizamos los datos que usaremos en Genome Browser:"
cat $genome_browser
## Eliminamos las diferencias:
rm $diff
~
```

- **Stdout: 7format_change.sh**

```
File  Edit  View  Search  Terminal  Help
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$ ./7format_change.sh
>>> Visualizamos los datos que usaremos en Genome Browser:
chr1:204073115-204127743
chr6:31164337-31170682
chr17:42313412-42388540
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$
```

Una vez que tenga las regiones seleccionadas con el formato correcto, las deberá caracterizar e identificar para conocer qué genes alberga en su interior. Para ello, deberá acceder al siguiente navegador genómico alojado por la Universidad de California, Santa Cruz: https://genome.ucsc.edu/. Una vez allí, se situará en el menú denominado "**Genomes**" (parte superior derecha) y seleccionará el *assembly* actual y de referencia del genoma del ser humano denominado **Human GRCh38/hg38**.

¿A qué nos referimos cuando hablamos del *Human Genome Assembly*?. (0,5 pts)

- El ensamblaje genómico (genome assembly) se refiere al proceso de poner secuencias de nucleótidos en el orden correcto. En el caso de Human Genome Assembly, se referiría al esamblaje genómico humano (https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/genome-assembly).

Al dar click en él, se abrirá un sitio web interactivo donde podrá pegar cada una de las regiones detectadas para identificar qué genes se encuentran en dichas coordenadas genómicas. **Adjunte una captura de pantalla (como la que se muestra a continuación) para cada una de las regiones encontradas previamente donde se visualice la región y el o los genes que se encuentran en ella (1 pts)**

Pegue aquí la región junto con sus coordenadas y de click al botón "**go**" para buscar dicha región de interés

Genes que se encuentran dentro de la región

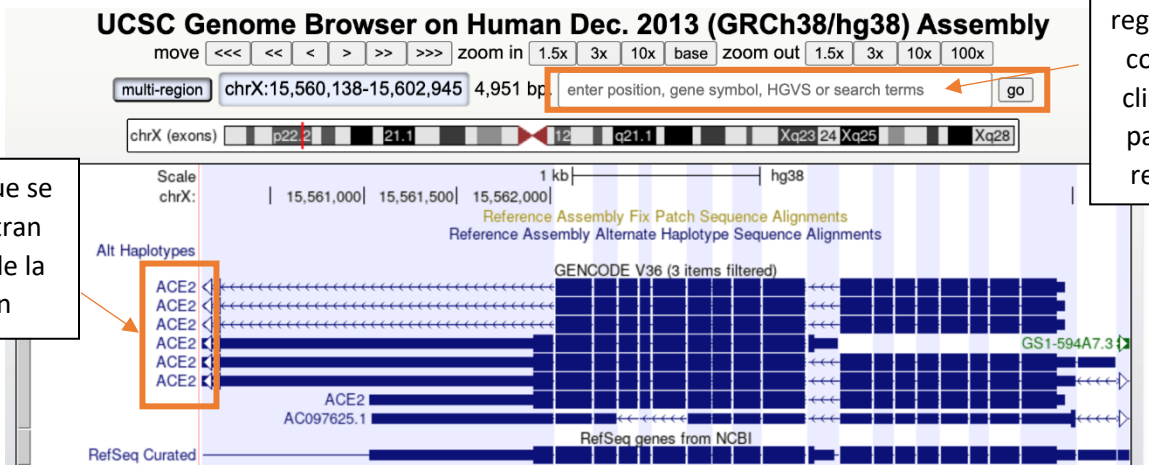


Figura 1. Vista del UCSC Genome Browser.

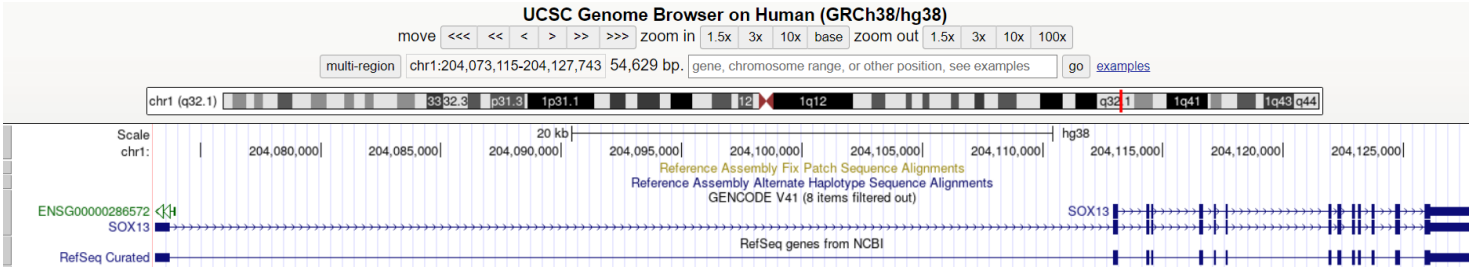


Figura 2. Vista del USC Genome Browser: chr1:204073115-204127743.

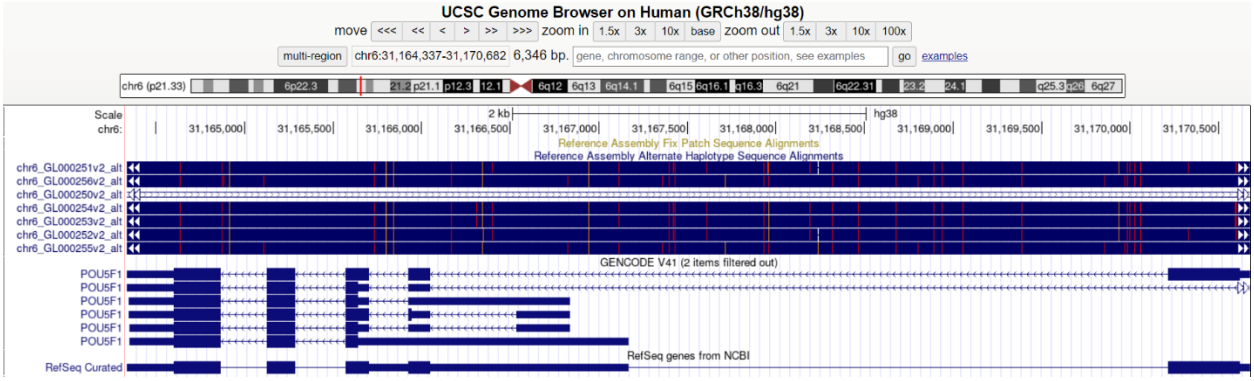


Figura 3. Vista del USC Genome Browser: chr6:31164337-31170682.

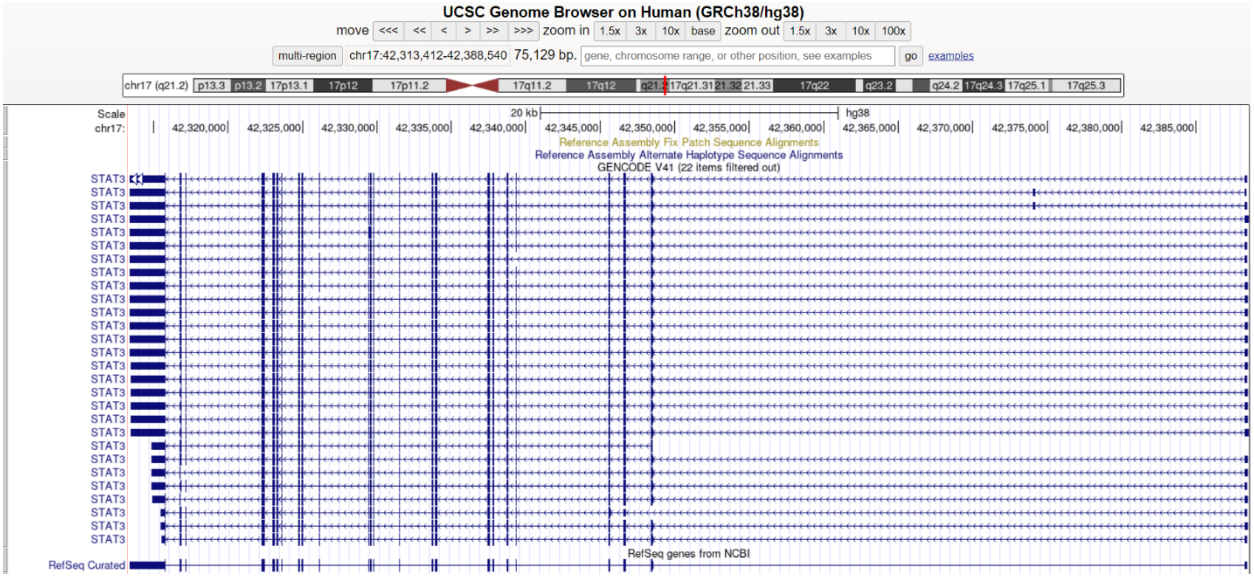


Figura 4. Vista del USC Genome Browser: chr17:42313412-42388540

Finalmente, cree un archivo final, donde incluya en la primera columna las regiones identificadas previamente con el formato, *cromosoma:inicio-fin* y una segunda columna con el nombre del gen que ha detectado en cada una de ellas. Visualice las primeras líneas del archivo creado. Incluya una captura de pantalla con el código empleado (1pts).

- **Script: 8coord_genes.sh**

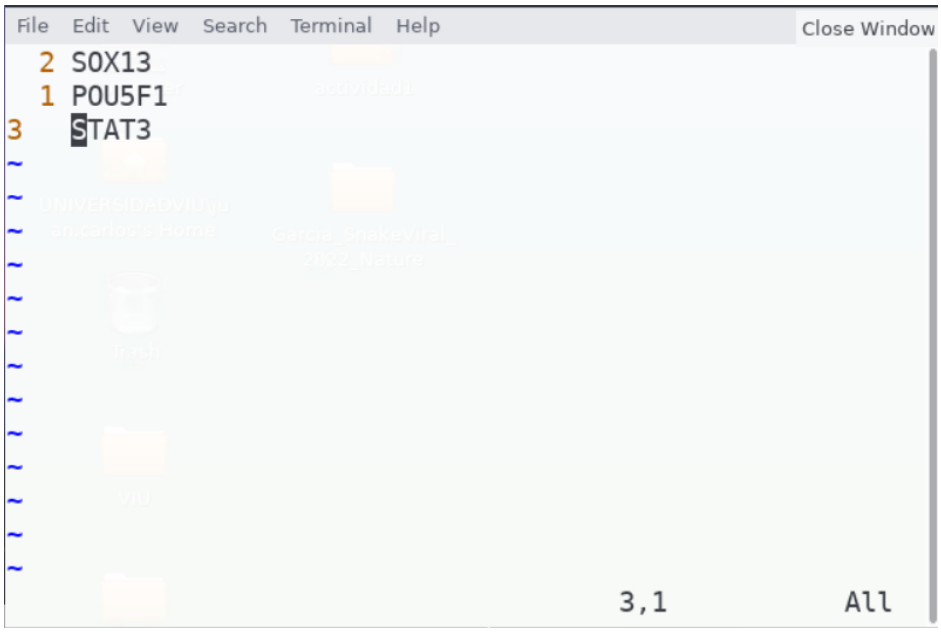
```
File  Edit  View  Search  Terminal  Help
#!/bin/bash

## VARIABLES ##
read -p "Ruta de Genome Browser: " genome_browser
read -p "Ruta para los genes: " genes
read -p "Ruta de los coordenadas y sus genes: " coordgenes

## EJECUCIÓN ##
## Copiamos los genes. Podría abrir un vim directamete en $genome_browser
## y añadir los genes tabulando. pero en caso de más genes y coordenadas
## veo más práctico copiar una lista de los genes y pegarla a las coordenadas.
vim $genes
## Por lo visto tengo un problema con los saltos de línea ($), viendo con
## cat -E genome_browser se ve lo siguiente:
## $hr1:204073115-204127743
## $hr6:31164337-31170682
## chr17:42313412-42388540$
## Con sed y el uso de esta expresión regular logro solucionar el problema. Se
## podría usar dos2unix como vimos en clase también.
sed -r 's/\r$//' -i $genome_browser # ahora $ están todos al final

## Juntamos $genome_browser y $genes finalmente
paste $genome_browser $genes > $coordgenes
echo ">>> Cromosomas, cooordenadas y los genes"
cat $coordgenes
## Podemos asegurarnos del todo con cat -T, ^I muestra las tabulaciones.
echo ">>> Miramos los lugares en concreto donde hay tabulador:"
cat -T $coordgenes
```

- **Stdout: 8coord._genes.sh**




```
File  Edit  View  Search  Terminal  Help
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$ ./8coord_genes.sh
Ruta de Genome Browser: ../data/processed/genome_browser.bed
Ruta para los genes: ../data/processed/genes.txt
Ruta de los coordenadas y sus genes: ../data/processed/coordgenes.txt
>>> Cromosomas, cooordenadas y los genes
chr1:204073115-204127743        SOX13
chr6:31164337-31170682  POU5F1
chr17:42313412-42388540 STAT3
>>> Miramos los lugares en concreto donde hay tabulador:
chr1:204073115-204127743^ISOX13
chr6:31164337-31170682^IPOU5F1
chr17:42313412-42388540^ISTAT3
[UNIVERSIDADVIU\juan.carlos@a-3uv58hx3etnvo code]$
```

En caso de querer ver el código utilizado puede pinchar este enlace:
https://github.com/Juankkar/Programacion_Shell_Scripting_VIU/tree/main/actividad1.