notas del grep sobre los archivos gtf

- utilicé el archivo de **hits únicos** que se obtuvieron a partir de la tabla de Raúl como la lista para hacer el `grep`
- hice un pequeño script para hacer el grep y guardar un archivo con los resultados para cada gtf
  - esto lo hice de esta forma por que el argumento que se le pasa a `xargs` debe usarse dos veces, ya busqué y hay manera de hacerlo pero cuando lo probé aún no lo sabía
- para paralelizar el proceso utilicé `xargs` con el parámetro `-P` seguido del número de cores que le quería asignar al proceso el comando completo quedó algo así

```
$ ls -1 gtfs | xargs -P20 -I % ./xargs_grep_on_gtfs.sh %
```

- dónde `ls -1` provee la lista de los archivos sobre los cuales se va a buscar
- `xargs -P20 -I %` indica que se usarán 20 nucleos de procesamiento, y el argumento -I es necesario no se por qué, de otra forma `xargs` solo se ejecuta una vez y termina

por otro lado el script `xargs_grep_on_gtfs.sh` lo que hace es ejecutar `grep -f hits_unicos_raul.txt ./gtfs$1 > xargs_results/$1` dónde `$1` representa cada archivo gtf

TODO sustituir el script por la sintaxis corrrecta de `xargs` (véase el ejemplo mas abajo de como hacerlo)

- Con los archivos que resultaron de correr lo anterior, renombre aquellos archivos que no se encontraban vacios usando el siguiente comando en la carpeta xargs_results

```
$ wc -l * | grep GCF | egrep -v " 0 " | egrep -o "GCF.+" | xargs -I % mv % %.match
```

- los archivos con match los copio en una carpeta nueva llamada `only_matches_gtfs`
- los siguientes comandos con sus expresiones regulares pueden usarse como referencia para construir las 3 listas necesarias para comparar genes, transcritos y proteinas
  - `/usr/bin/ls -1 | xargs -I % sh -c " echo % && egrep -o '; protein_id[^;]+' %"`
  - `/usr/bin/ls -1 | xargs -I % sh -c " echo % && egrep -o '; transcript_id[^;]+' %"`
  - `/usr/bin/ls -1 | xargs -I % sh -c " echo % && egrep -o 'gene_id[^;]+' %"` **esta es la manera de usar mas de una vez el *placeholder* de xargs**
- un ejemplo de la modificación de lo anterior para ya generar las listas es:

```
/usr/bin/ls -1 | xargs -I % egrep -o "; protein_id[^;]+" % > ../tablas_matches/proteinas.txt
```

- correr las variantes de genes y transcritos da como resultado los tres archivos necesarios para construir la tabla que se va a necesitar

como nota de interés correr estos programas dieron como resultados tres archivos con el mismo número de lineas cada uno, por lo que la tabla resultante de la unión de estos no tendrá celdas vacias

- junté los archivos resultantes lado a lado con R, pero basícamente podría hacerse en excel o libreoffice y luego utilicé este comando para dejarla un poco mas presentable.

```
sed 's/;//g' tabla_redundante.tsv | sed 's/gene_id //' | sed 's/protein_id //' | sed 's/transcript_id //' | sed 's/" /"/g' > tabla_sin_header_ni_tags.tsv
```

por la forma en que la hice en R todavía tenía un header que se lo quité a mano, también olvidé quitarle las comillas así que usé algo como esto seguramente:

```
sed -i 's/"//g'  tabla_sin_header_ni_tags.tsv
```

- revisé cual era la redundancia los archivos de genes, proteinas y transcritos (en ese orden) con el siguiente comando que me permite conocer el número de elementos unicos en cada uno de los archivos

```
 /usr/bin/ls -1 *txt | xargs -I % sh -c "cat % | sort | uniq | wc -l"
673 # resultado de genes
1814 # resultado de proteinas
1815 # resultado de transcritos?
```

- en cambio aplicando la misma lógica para la tabla de tres columnas se obtiene lo siguiente:
  - primero construyo una tabla únicamente con elementos únicos usando

```
cat tabla_sin_header_ni_tags.tsv | sort | uniq > tabla_chida_no_redundante.tsv
```

aplicar `wc -l` sobre esa tabla da cómo resultado **1816** combinaciones únicas

TODO Aqui falta por ver cuales son las proteinas que no se están encontrando en los archivos GTF considerando que el archivo de hits únicos tiene 1932 entradas! y sólo se están recuperando 1814

05 Oct 22

- para intentar resolver la falta de matches intento quitar las versiones de los *accession numbers* de los hits

```
sed 's/\..//' hits_unicos_raul.txt > hits_unicos_sin_version.txt
```

- modifico el programa de xargs_grep_on_gtfs.sh para crear la **v2** del mismo que incluye una función para insertar el nombre del archivo de origen a cada archivo de resultados esta modificación también inclute como archivo de entrada el archivo de hits sin versiones específicas.
    - vuelvo a ejecutar:

```
$ ls -1 gtfs | xargs -P30 -I % ./xargs_grep_on_gtfs_v2.sh %
```

- nuevamente, marco los archivos con match usando:

```
$ cd xargs_results
$ wc -l * | grep GCF | egrep -v " 0 " | egrep -o "GCF.+" | xargs -I % mv % %.match
```

- creo el directorio **only_match2** y copio los resultados con terminación **.match**

```
cd ..
mkdir only_match2
cp xargs_results/*match only-match2
mkdir tablas_match2
```

obtengo las listas por separado de genes, transcritos, proteinas y origenes con

```
/usr/bin/ls -1 | xargs -I % egrep -o "; protein_id[^;]+" % > ../tablas_match2/proteinas.txt
/usr/bin/ls -1 | xargs -I % egrep -o "; transcript_id[^;]+" % > ../tablas_match2/transcripts.txt
/usr/bin/ls -1 | xargs -I % egrep -o "gene_id[^;]+" % > ../tablas_match2/genes.txt
/usr/bin/ls -1 | xargs -I % egrep -o "; G.+.gtf$" % > ../tablas_match2/origin.txt
```

- para crear la tabla en R hice lo siguiente:

abro una terminal de R con `R`

```
# leer archivos
gen <- readLines("genes.txt")
tx <- readLines("transcripts.txt")
px <- readLines("proteinas.txt")
ori <- readLines("origin.txt")


# armado de data frame
df <- data.frame(gen, px, tx, ori)

# eliminado de espacios en blanco

df$gen <- trimws(df$gen)
df$px <- trimws(df$px)
df$tx <- trimws(df$tx)
df$ori <- trimws(df$ori)

# eliminado de strings incecesarias

df$ori <- gsub(".gtf","",df$ori)
df$ori <- gsub("; ","",df$ori)
df$tx <- gsub("; transcript_id ","",df$tx)
df$px <- gsub("; protein_id ","",df$px)
df$gen <- gsub("gene_id ","",df$gen)
df$gen <-gsub("\"","",df$gen)
df$px <-gsub("\"","",df$px)
df$tx <-gsub("\"","",df$tx)

# escribir tabla

write.table(df, "tabla_matches_redundantes.tsv", sep="\t", col.names=F, row.names=F, quote=F)



$ /usr/bin/ls -1 *txt | xargs -I % sh -c "cat % | sort | uniq | wc -l"
689 # genes unicos
265 # organismos de origen
1847 # proteinas unicas
1848 # transcrutos unicos
```

- finalmente ejecuto los siguiente para generar la tabla final, el resultado es mejor que la primera vez pero sigue siendo mas pequeña de lo esperado de acuerdo al numero de hits unicos

```
cat tabla_matches_redundantes.tsv | sort | uniq > tabla_simplificada_final.tsv
```

- por ultimo cargué la tabla en R y generé el numero de elementos unicos por especie, veáse el archivo con un nombre parecido a lo que acabo de escribir en la carpeta de las tablas

```
table(table(df$V4))
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 24 25 26 27 28 47 59
 4 31 22 23 40 35 26 23 15 11  9  5  4  2  1  3  1  2  1  1  1  1  1  1  1  1
```

- el organismo con mayor número de proteinas coincidentes con HAS o CHIS tuvo 59 hits el cual está identificado cómo: *GCF_902806645.1_cgigas_uk_roslin_v1_genomic*
- el número de transcritos por genes fue el siguiente:

```
table(table(df$V2))
```

```
   1    2
1839    8
```

- lo que quiere decir que sólo 8 proteinas estaban representadas por mas de un transcrito (2 fué el unico valor distinto a 1)