CLASE VARIACIÓN GENÓMICA. Práctica

Daniel Rico y Maria Rigau

Máster Bioinformática 01/03/16

COPY NUMBER VARIANTS (CNVs)

Parte 1: Descarga de mapas de CNVs

1. Buscar el siguiente artículo:

Zarrei M, MacDonald JR, Merico D, et al. A copy number variation map of the human genome. Nat Rev Genet 2015;16:172–83.

http://www.nature.com/nrg/journal/v16/n3/full/nrg3871.html

2. Descargar excel

- 8. Supplementary information Table S10 (3.5 MB)

3. Abrimos RStudio

```
# Set working directory, nos metemos en la carpeta donde tenemos los objetos que
os hemos pasado
getwd()
```

```
## [1] "/home/mrigau/CLASES"
```

```
setwd("~/CLASES/")

# # Instalamos los paquetes de R
# Paquetes de CRAN
# install.packages( "gdata" )

# Paquetes de Bioconductor
# source("https://bioconductor.org/biocLite.R")
# biocLite("GenomicRanges")

# Cargamos las librerías
library( gdata )
library( GenomicRanges )


# Cargamos CNVs de Zarrei


# Especificamos que queremos la página 3, dónde hay ganancias y pérdidas
CNV_Zarrei <- read.xls ( "Stringent_map.xls", sheet = 3, header = TRUE )


# Si no hemos podido descargar el excel cargamos el objeto de R
# load("CNV_Zarrei.RData")
```

PREGUNTAS:

- ¿Qué clase de objeto es?
- ¿Qué dimensiones y qué estructura tiene?
- ¿Tienen nombre las columnas y las filas?

```
# Les damos nombres a las CNVs (junto "Z" de Zarrei, cromosoma, y coordenadas)
rownames( CNV_Zarrei ) <- paste( "Z", CNV_Zarrei$chr, CNV_Zarrei$start,
CNV_Zarrei$end, sep = "_" )

# Los chromosomas están guardados como "chr1", etc. Pero nosotros los queremos
guardar como 1, 2, 3... X, Y
# Esto lo hacemos con la función substr(x, start, stop)
CNV_Zarrei$chr <- as.factor(substr(CNV_Zarrei$chr, 4,
nchar(as.character(CNV_Zarrei$chr))))

CNV_Zarrei$chr <- ordered (CNV_Zarrei$chr, levels = c(1:22, "X", "Y"))
```

PREGUNTAS:

- ¿Cuántas CNVs hay de cada tipo?
- ¿Y separando por cromosomas?

```
# ¿Cuántas CNVs hay de cada tipo?
summary( CNV_Zarrei$type)
```

```
##      Gain Gain+Loss      Loss
##       348       794     10590
```

```
# Y separado por cromosomas?
head( table( CNV_Zarrei [ , c("chr", "type")]) )
```

```
##    type
## chr Gain Gain+Loss Loss
##   1   27        65  722
##   2   33        61  824
##   3   11        38  636
##   4   16        51  797
##   5   12        47  630
##   6   10        36  670
```

```
# Otras maneras que os enseñaron en clase para hacer lo mismo
# xtabs ( ~ chr + type, data = CNV_Zarrei)
# with ( CNV_Zarrei, table( chr, type ))
# table(CNV_Zarrei$chr, CNV_Zarrei$type)



# CREAMOS UN OBJETO GRanges
Zarrei_RANGES <- GRanges( seqnames = CNV_Zarrei$chr ,
                          ranges = IRanges( start = CNV_Zarrei$start,
                                            end = CNV_Zarrei$end,
                                            names = rownames(CNV_Zarrei) ),
                          type = CNV_Zarrei$type)


summary(width(Zarrei_RANGES)) # Los tamaños de las CNVs están en bp
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      56     589    1237   11650    4298 2167000
```

```
# Histograma del tamaño de las CNVs
hist(width(Zarrei_RANGES))
```

```
# Lo veremos mejor calculando el logaritmo del tamaño de las CNVs
hist(log10(width(Zarrei_RANGES)))
```

```
# Hacemos un boxplot separando los tamaños de las CNVs por cromosomas
lista <- by(log10(width(Zarrei_RANGES)), as.factor(seqnames(Zarrei_RANGES)), list)

str(lista)
```

```
## List of 24
##  $ 1 : num [1:814] 5.22 5.19 6.07 2.94 3 ...
##  $ 2 : num [1:918] 3.43 3.57 3.25 3.75 3.07 ...
##  $ 3 : num [1:685] 3.69 3.96 2.93 3.32 4.72 ...
##  $ 4 : num [1:864] 3.92 3.64 2.64 2.86 3.68 ...
##  $ 5 : num [1:689] 4.78 3.56 4.17 2.78 3.57 ...
##  $ 6 : num [1:716] 5.15 5.3 3.44 3.42 3.19 ...
##  $ 7 : num [1:738] 4.84 3.28 2 2.73 4.05 ...
##  $ 8 : num [1:628] 4.84 3.11 3.59 2.46 3.34 ...
##  $ 9 : num [1:503] 5.5 2.73 4.42 2.92 4.26 ...
##  $ 10: num [1:549] 4.05 3.55 2.56 3.6 2.92 ...
##  $ 11: num [1:516] 4.24 2.19 2.9 2.77 3.77 ...
##  $ 12: num [1:531] 4.69 3.76 2.55 3.88 3.51 ...
##  $ 13: num [1:410] 2.98 3.99 4.84 3.97 2.93 ...
##  $ 14: num [1:309] 5.13 6.03 4.37 4.16 4.93 ...
##  $ 15: num [1:316] 3.85 5.87 5.67 5.51 5.52 ...
##  $ 16: num [1:376] 4.39 2.73 3.2 2.83 3.87 ...
##  $ 17: num [1:426] 3.4 2.7 3.2 3.31 3.1 ...
##  $ 18: num [1:365] 4.98 4.04 2.65 3.83 3.22 ...
##  $ 19: num [1:362] 5.3 3.92 2.92 3.52 3.03 ...
##  $ 20: num [1:251] 3.43 3.94 2.75 2.61 2.66 ...
##  $ 21: num [1:186] 5.69 5.05 4.48 3.35 5.33 ...
##  $ 22: num [1:210] 5.62 4.72 5.31 3.09 2.8 ...
##  $ X : num [1:360] 3.87 3.05 2.99 2.66 2.96 ...
##  $ Y : num [1:10] 4.51 4.84 3.81 3.62 5.01 ...
##  - attr(*, "call")= language by.default(data = log10(width(Zarrei_RANGES)),
INDICES = as.factor(seqnames(Zarrei_RANGES)),      FUN = list)
##  - attr(*, "class")= chr "by"
```

```
boxplot(lista, outline = T, las = 2, varwidth= T ,
        main = "CNV size by chromosome",
        ylab = "log10 CNV size")
```

Parte 2: Descarga de genes codificantes para proteínas con BioMart

- Vamos a http://www.ensembl.org
- Nosotros queremos el genoma según el Build GRCh37, ya que el mapa de CNVs está hecho con esta anotación. Por lo tanto, tenemos que ir al apartado “View in archive site” (abajo a la derecha) para acceder a versiones más antiguas.

Actualmente, en Ensembl van por la versión 83 (Build GRCh38), y nosotros queremos la 75 (la última del Build GRCh37)

- Una vez en la ventana de los *Ensembl Archives* entramos a “Ensembl 75: Feb 2014 (GRCh37.p13)”
- Abrimos **BioMart**

¿Qué es Biomart?

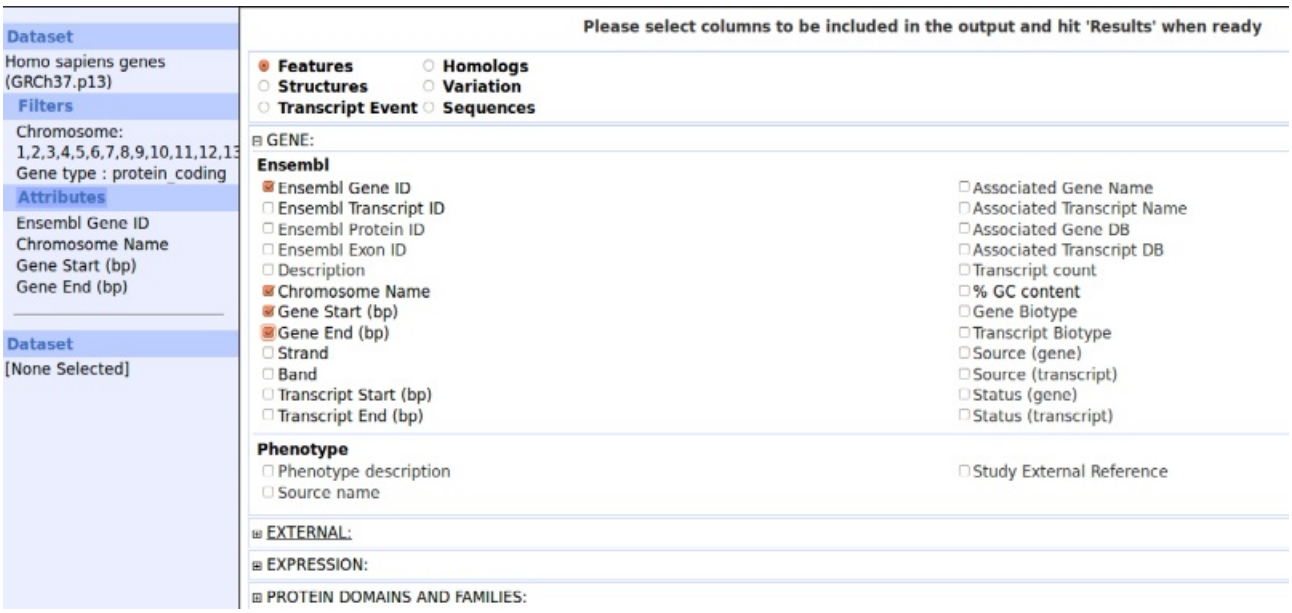
Data mining • BioMart is a search engine that can find multiple terms and put them into a table format. • Such as: mouse gene (IDs), chromosome and base pair position

Seleccionamos los filtros:

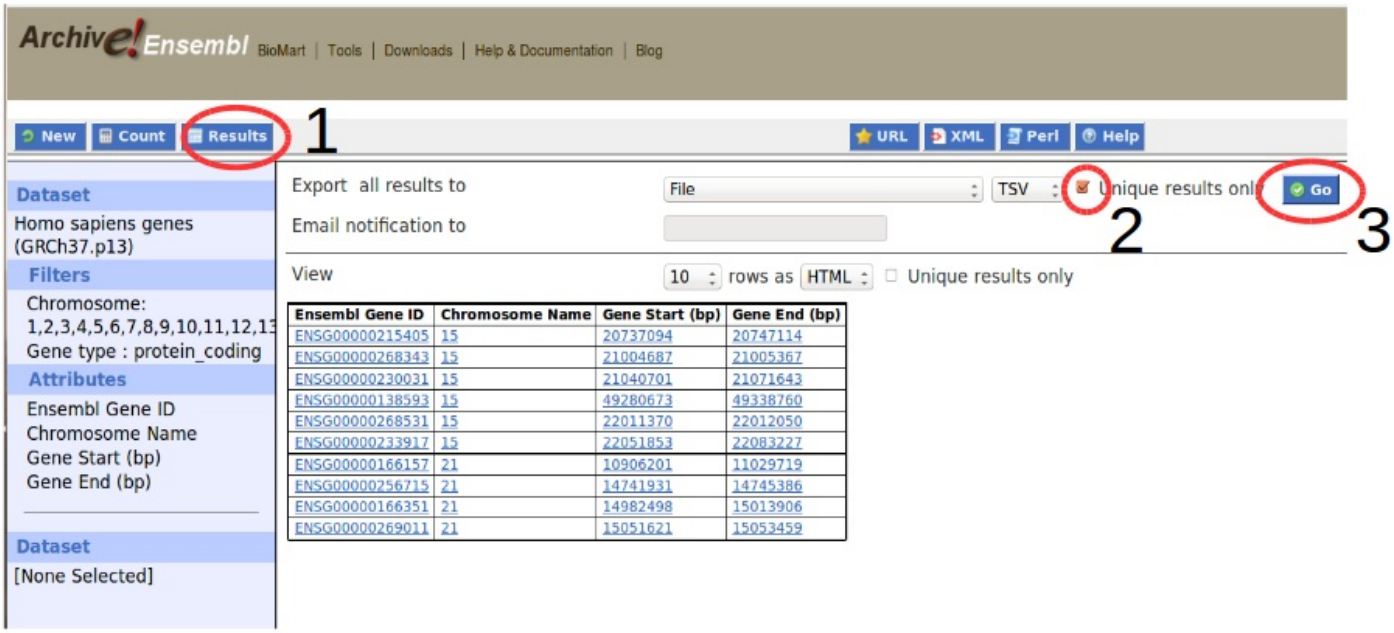
- Dataset
  Choose database –> Ensembl Genes 75

Choose dataset –> Homo sapiens genes

- Filters: Chromosome –> Select 1 to 22 Gene type –> Select “protein_coding”
- Attributes:



Exportamos los resultados



Volvemos a RStudio

```
# Abrimos la tabla creada con BioMart
GRCh37_protein_coding <- read.table( "~/CLASES/BioMart_protein_coding_75.txt", sep
= "\t", header = TRUE, stringsAsFactors = F )


# En caso de no haberla podido crear, cargar la siguiente tabla
# load( "GRCh37_protein_coding.RData")
```

```
# Aquí también podéis probar las funciones class(), head(), str(), dim(), etc.
para ver cómo es la tabla

# Cambiamos los nombres
colnames( GRCh37_protein_coding ) <- c( "Gene.ID", "Chromosome", "Start", "End")

# Damos nombre a las filas
rownames( GRCh37_protein_coding ) <-GRCh37_protein_coding$Gene.ID


# CREAMOS UN OBJETO GRanges
protein_coding_RANGES <- GRanges( seqnames = GRCh37_protein_coding$Chromosome ,
                                  ranges = IRanges( start =
GRCh37_protein_coding$Start,
                                                    end =
GRCh37_protein_coding$End,
                                                    names = rownames(
GRCh37_protein_coding ) ) )
```

En realidad, todo esto se puede hacer con un paquete de bioconductor

```
# source("https://bioconductor.org/biocLite.R")
# biocLite("biomaRt")

library( biomaRt )

listMarts(host = "feb2014.archive.ensembl.org")

ensembl=useMart(biomart="ENSEMBL_MART_ENSEMBL", host =
"feb2014.archive.ensembl.org")
ensembl = useDataset("hsapiens_gene_ensembl",mart=ensembl)

listFilters(ensembl)
listAttributes(ensembl)

filterOptions("biotype",ensembl)

# The getBM function has three arguments that need to be introduced:  filters,
attributes and values.
# Filters define a restriction on the query.  For example you want to restrict the
output to all genes located on the human X chromosome then the filter chromosome
name can be used with value 'X'. The listFilters function shows you all available
filters in the selected dataset

genes.with.id = getBM(attributes=c("ensembl_gene_id", "chromosome_name",
"start_position", "end_position"),
                      filters = c("chromosome_name", "biotype" ),
                      values = list(c(1:22), "protein_coding"),
                      mart= ensembl) # fuction to get  gene id's and gene name from
data base

colnames(genes.with.id)
colnames(genes.with.id) <- colnames(GRCh37_protein_coding)

rownames(genes.with.id) <- genes.with.id$Gene.ID

identical(GRCh37_protein_coding, genes.with.id)
```

Parte 3: Solapamiento de CNVs con genes que codifican para proteinas

Para encontrar solapamientos entre CNVs y genes utilizaremos el paquete GenomicRanges

```
overlaps <- findOverlaps(query = protein_coding_RANGES, subject = Zarrei_RANGES,
type = "within" )
```

PREGUNTAS:

- ¿De qué clase es el objeto *overlaps*?

Los genes que se encuentran dentro de CNVs los podemos obtener de la siguiente manera:

```
queryHits(overlaps)
# vector con los índices de los genes dentro de CNVs
```

Y las CNVs que están englobando estos genes las obtenemos con

```
subjectHits(overlaps)
# vector con los índices de las CNVs que engloban genes
```

- ¿Cuántos genes se encuentran en número de copia variable en algún individuo?
- ¿Cuántas CNVs engloban uno o más genes?

```
# Vamos a crear vectores con los genes y las cnvs que solapan
genes_within_cnvs_1 <- names(protein_coding_RANGES) [  queryHits(overlaps)  ]

cnvs_with_genes_1 <- names(Zarrei_RANGES) [  subjectHits(overlaps)  ]

# Añado un unique()
genes_within_cnvs <- names(protein_coding_RANGES) [ unique( queryHits(overlaps) )
]
cnvs_with_genes <- names(Zarrei_RANGES) [ unique( subjectHits(overlaps) ) ]

# ¿Qué diferencias hay entre los dos primeros objetos y los dos segundos (creados
con unique)?
# ¿A qué se deben?
# ¿Qué significan?
```

- ¿ De qué tipo son estas CNVs? (Esta información se puede sacar del objeto **CNV_Zarrei** y de **Zarrei_RANGES**)
- Hemos buscado solapamientos del tipo “within”. Qué otros tipos de solapamientos hay?
- Cuántos genes solapan con CNVs con el tipo de solapamiento que viene por defecto?

Parte 4: Características de los genes CNV

Ahora buscaremos qué proporción de genes afectados es esencial, tal y como han hecho en el artículo de Zarrei et al.

Cargamos listas de genes

```
objetos <- load( "ESSENTIAL_GENES.RData")
# visualizo qué objetos he cargado
objetos
```

```
## [1] "ESSENTIAL_GENES"
```

```
# ¿Cuántos genes CNV son esenciales?
sum(genes_within_cnvs %in% ESSENTIAL_GENES)
```

```
## [1] 102
```

```
# Creamos una tabla de contingencia
CNV_ESSENTIAL <- table(data.frame(CNV =names(protein_coding_RANGES) %in%
genes_within_cnvs,
                                 ESSENTIAL = names(protein_coding_RANGES) %in%
ESSENTIAL_GENES))

CNV_ESSENTIAL
```

```
##        ESSENTIAL
## CNV     FALSE  TRUE
##   FALSE 13813  4781
##   TRUE    734   102
```

Estas proporciones son diferentes de lo que esperaríamos por azar?

```
# Qué significan estas proporciones?
fisher.test(CNV_ESSENTIAL)
```

- Cómo interpreto este resultado?

Puedo hacer lo mismo con genes OMIM

```
load("OMIM_genes.RData")
CNV_OMIM <- table(data.frame(CNV =names(protein_coding_RANGES) %in%
genes_within_cnvs, OMIM = names(protein_coding_RANGES) %in% OMIM_genes))
fisher.test(CNV_OMIM)
```

- Qué nos indica este resultado?
- Es igual o diferente a lo que pasaba con los genes esenciales?
- Qué habrías esperado?