Curso de Métodos Estadísticos y Analíticos de Datos Genómicos

Leonardo Collado Torres
lcollado@ibt.unam.mx y lcollado@wintermexico.com
Lic. en Ciencias Genómicas
www.lcg.unam.mx/~lcollado/

Winter Genomics (WG) e Instituto de Biotecnología (IBT) de la UNAM

21 de Enero de 2010

Bioconductor para Datos de Secuenciación Masiva II

- Una nueva compañía de servicios bioinformáticos con datos de secuenciación masiva.

Es la base oculta

- Creado por: Hervé Pages, Patrick Aboyoun y Michael Lawrence.
- `IRanges` es el paquete de bajo nivel que nos permite manejar este tipo de datos en R.
- Es muy útil para representar información a lo largo de posiciones en el genoma.
- Tiene una serie de funciones para hacer *operaciones en intervalos*.

Datos por intervalos

- Es muy útil para manejar información por posición en el genoma: el gen, su posición de inicio, de fin, la cadena, ...
- Aquí va un ejemplo con 3 genes. La forma clásica para almacenar esta info con R sería un data frame:

```
> start <- c(3, 7, 100)
> end <- c(5, 20, 200)
> chr <- c("chr1", "chr1", "chr2")
> strand <- c("+", "-", "+")
> df <- data.frame(chr = chr, strand = strand,
+     start = start, end = end)
> df
```

```
   chr strand start end
1 chr1      +     3   5
2 chr1      -     7  20
3 chr2      +   100 200
```

Datos por intervalos

- Pero con IRanges podemos hacerlo así:

```
> library(IRanges)
> RD <- RangedData(ranges = IRanges(start = start
+      end = end), strand = strand,
+      space = chr)
> RD
```

```
RangedData with 3 rows and 1 value column across 2
        space     ranges |      strand
  <character>  <IRanges> | <character>
1        chr1 [  3,   5] |           +
2        chr1 [  7,  20] |           -
3        chr2 [100, 200] |           +
```

- La diferencia radica en la habilidad de agrupar los datos por el espacio. En el ejemplo, por el cromosoma.

Datos por intervalos

```
> range(ranges(RD))
CompressedIRangesList of length 2
$chr1
IRanges of length 1
    start end width
[1]     3  20    18

$chr2
IRanges of length 1
    start end width
[1]   100 200   101
```

Operaciones

- Hay toda una gama y a continuación muestro algunas.[1]

```
> ir <- IRanges(c(1, 8, 14, 15, 19,
+     34, 40), width = c(12, 6, 6,
+     15, 6, 2, 7))
> strand <- rep(c("+", "-"), c(4,
+     3))
> rd <- RangedData(ranges = ir, strand = strand,
+     space = "chr1")
```

- Ya creado el objeto `RangedData` podemos usar funciones para accesar la información:

```
> start(rd)
```

```
[1]  1  8 14 15 19 34 40
```

```
> end(rd)
```

```
[1] 12 13 19 29 24 35 46
```

Operaciones

```
> width(rd)
[1] 12  6  6 15  6  2  7
```

[1] Siempre pueden checar `help(package = IRanges)`

Un subconjunto

```
> rd[2:5, ]
```

```
RangedData with 4 rows and 1 value column across
        space    ranges |      strand
  <character> <IRanges> | <character>
1        chr1  [ 8, 13] |           +
2        chr1  [14, 19] |           +
3        chr1  [15, 29] |           +
4        chr1  [19, 24] |           -
```

```
> ranges(rd[2:5, ])
```

Un subconjunto

```
SimpleRangesList of length 1
$chr1
IRanges of length 4
    start end width
[1]     8  13     6
[2]    14  19     6
[3]    15  29    15
[4]    19  24     6
```

Mover horizontalmente

```
> rd2 <- rd
> ranges(rd2) <- shift(ranges(rd2),
+     2)
> rd2[2:5, ]
```

```
RangedData with 4 rows and 1 value column across
       space    ranges |      strand
  <character> <IRanges> | <character>
1        chr1  [10, 15] |           +
2        chr1  [16, 21] |           +
3        chr1  [17, 31] |           +
4        chr1  [21, 26] |           -
```

O aumentar el tamaño

- ¿A alguien se le ocurre para qué quisieramos hacer esto?

```
> rd3 <- rd
> pos <- values(rd3)[, "strand"] ==
+     "+"
> ranges(rd3)[pos] <- resize(ranges(rd)[pos],
+     120)
> ranges(rd3)[!pos] <- resize(ranges(rd)[!pos],
+     120, start = FALSE)
> rd3[2:5, ]
```

O aumentar el tamaño

```
RangedData with 4 rows and 1 value column across
        space     ranges |      strand
  <character>  <IRanges> | <character>
1        chr1 [  8, 127] |           +
2        chr1 [ 14, 133] |           +
3        chr1 [ 15, 134] |           +
4        chr1 [-95,  24] |           -
```

Delimitar

```
> ranges(rd3) <- restrict(ranges(rd3),
+     1)
> rd3[2:5, ]
```

```
RangedData with 4 rows and 1 value column across
        space    ranges |      strand
  <character> <IRanges> | <character>
1        chr1 [ 8, 127] |           +
2        chr1 [14, 133] |           +
3        chr1 [15, 134] |           +
4        chr1 [ 1,  24] |           -
```

Funciones para resumir info

- Ya sea todo lo que está cubierto alguna vez, ninguna o donde no hay cambios.

```
> reduce(ranges(rd))
```

```
SimpleRangesList of length 1
$chr1
IRanges of length 3
    start end width
[1]     1  29    29
[2]    34  35     2
[3]    40  46     7
```

```
> gaps(ranges(rd))
```

Funciones para resumir info

```
SimpleRangesList of length 1
$chr1
IRanges of length 2
    start end width
[1]    30  33     4
[2]    36  39     4
```

```
> disjoin(ranges(rd))
```

```
SimpleRangesList of length 1
$chr1
IRanges of length 10
     start end width
[1]      1   7     7
[2]      8  12     5
[3]     13  13     1
[4]     14  14     1
```

Funciones para resumir info

```
[5]      15   18      4
[6]      19   19      1
[7]      20   24      5
[8]      25   29      5
[9]      34   35      2
[10]     40   46      7
```

Aún más interesante: sobrelapes y cobertura

```
> ol <- findOverlaps(ranges(rd),
+     reduce(ranges(rd)))
> as.matrix(ol)[1:3, ]
     query subject
[1,]     1       1
[2,]     2       1
[3,]     3       1
> cover <- coverage(ranges(rd))
> cover
```

Aún más interesante: sobrelapes y cobertura

```
SimpleRleList of length 1
$chr1
'integer' Rle of length 46 with 11 runs
  Lengths:  7 5 2 4 1 5 5 4 2 4 ...
  Values :  1 2 1 2 3 2 1 0 1 0 ...
```

Más info

- Para genomas es recomendable usar objetos tipo Rle (Run Length Encoding) porque son mucho más eficientes.
- `IRanges` también te permite generar *vistas*. Básicamente asocia una secuencia de ADN con un objeto *Ranges*.
- Referencia: `http://www.bioconductor.org/workshops/2009/SeattleNov09/IRanges/`

Intro

- Fueron creados por James Bullard y Steffen Durinck.
- El objetivo detrás de `GenomeGraphs` es poder visualizar tus datos rápidamente en la misma sesión de R en la que los estás analizando.
- `biomaRt` por otro lado te permite bajar información de una gama de bases de datos y tenerlos disponibles en R.

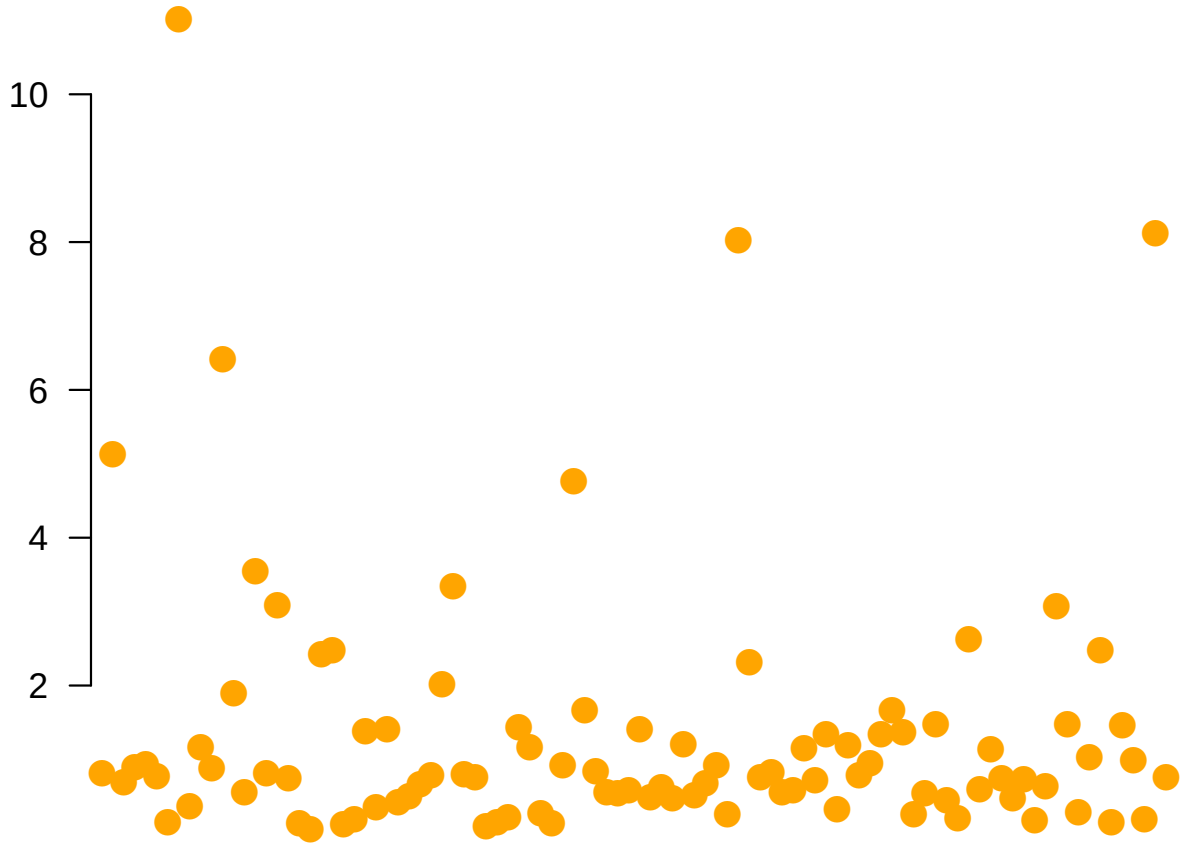
GenomeGraphs

Table 1: Overview of classes representing drawable genomic datasets

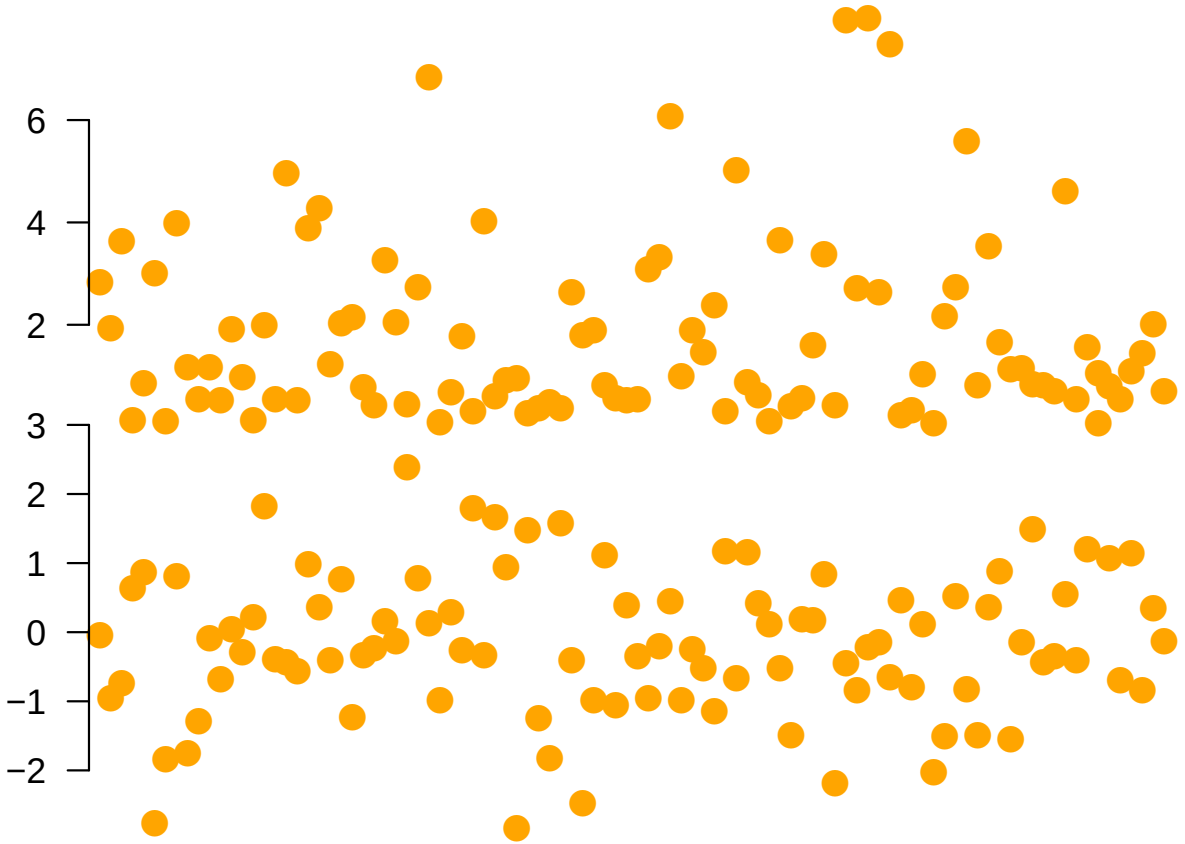
Class	Description
gdObject	the root class of the system, never directly instantiated
DisplayPars	class managing various plotting parameters
Gene	class representing a gene
GeneRegion	class defining a region of a chromosome, generally a set of genetic elements (genes)
Transcript	class defining a transcript
TranscriptRegion	class defining a region of a chromosome, generally a set of genetic elements (transcripts)
Ideogram	class representing an ideogram
Title	class to draw a title
Legend	class to draw a legend
GenomeAxis	class to draw an axis
AnnotationTrack	class used to represent custom annotation
Overlay	root class for overlays, never directly instantiated
RectangleOverlay	class to represent rectangular regions of interest
TextOverlay	class to draw text on plots
Segmentation	class to draw horizontal lines in various sets of data
GenericArray	class to draw data from microarrays.
ExonArray	class to draw data from exon microarrays.
GeneModel	class to draw custom gene models (intron-exon structures)
BaseTrack	class to draw arbitrary data at a given base
MappedRead	class to plot sequencing reads that are mapped to the genome

Figura 1: Tabla 1 Durinck y Bullard BMC 09

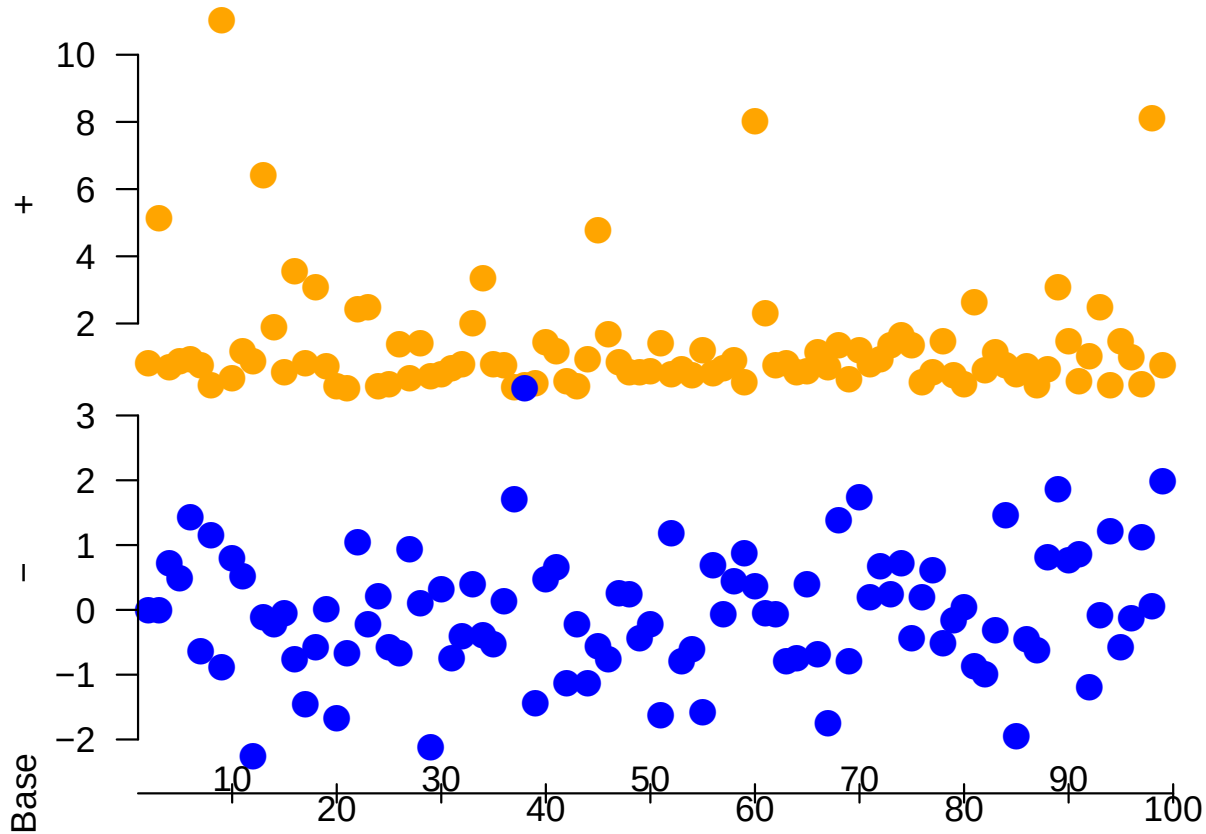
La más sencilla: makeBaseTrack



Ahora con 2 makeBaseTrack



Ahora con 2 makeBaseTrack y un makeGenomeAxis



Un ejemplo con biomaRt

- Encontremos los genes de *Bacillus subtilus* de la posición 12 mil a la 20 mil.
- Cargamos la base y ahora buscamos el nombre del cromosoma.

```
> bsub <- useMart("bacterial_mart_3",
+     dataset = "bac_6_gene")
> head(listAttributes(bsub))
```

- Luego obtenemos la info para los genes en la cadena positiva.

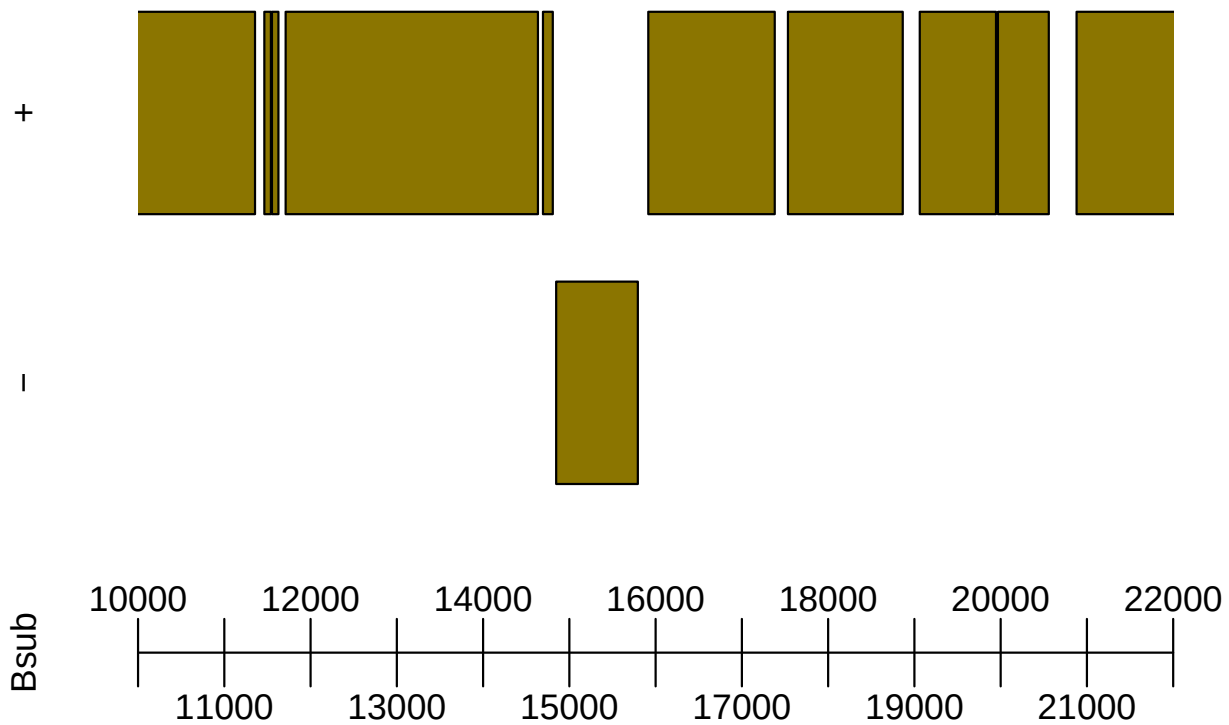
```
> pos <- makeGeneRegion(12000, 20000,
+     chromosome = "Chromosome",
+     strand = "+", biomart = bsub)
```

- Luego obtenemos la info para la cadena menos y graficamos usando `gdPlot`:

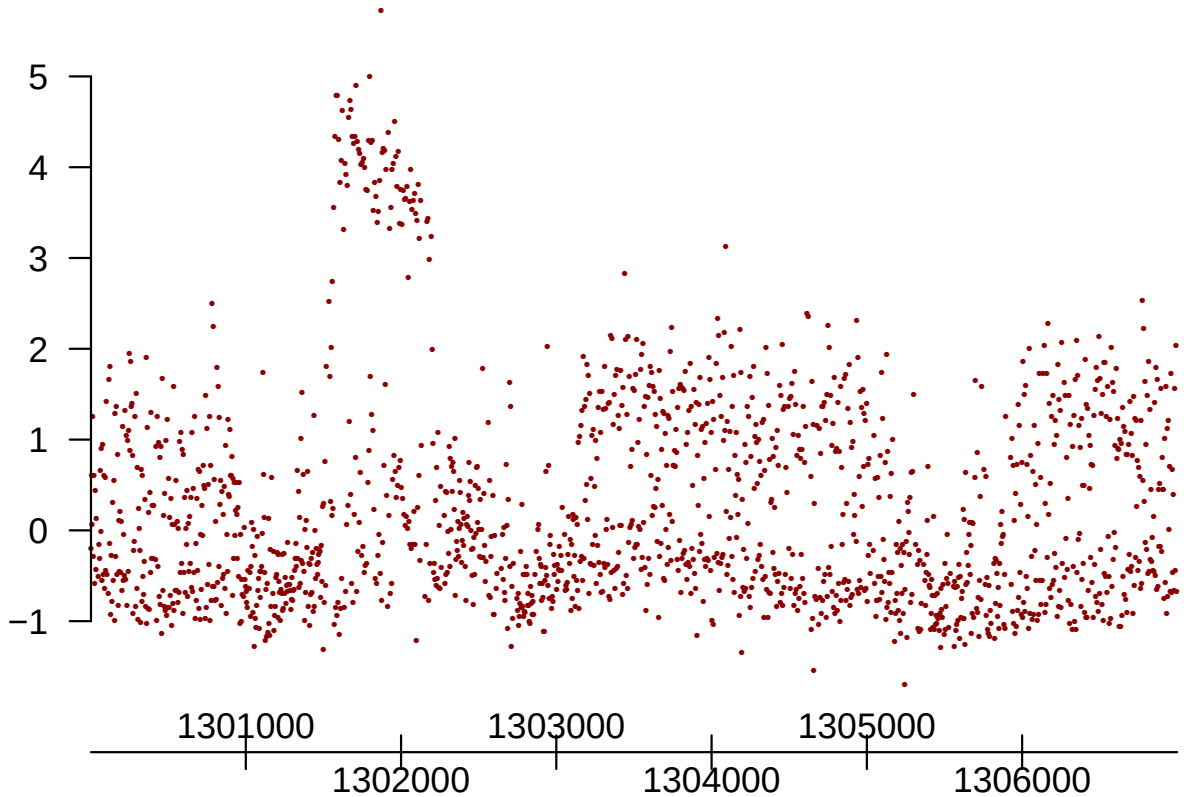
Un ejemplo con biomaRt

```
> neg <- makeGeneRegion(12000, 20000,
+     chromosome = "Chromosome",
+     strand = "-", biomart = bsub)
> gdPlot(list(`+` = pos, `-` = neg,
+     Bsub = makeGenomeAxis()))
```

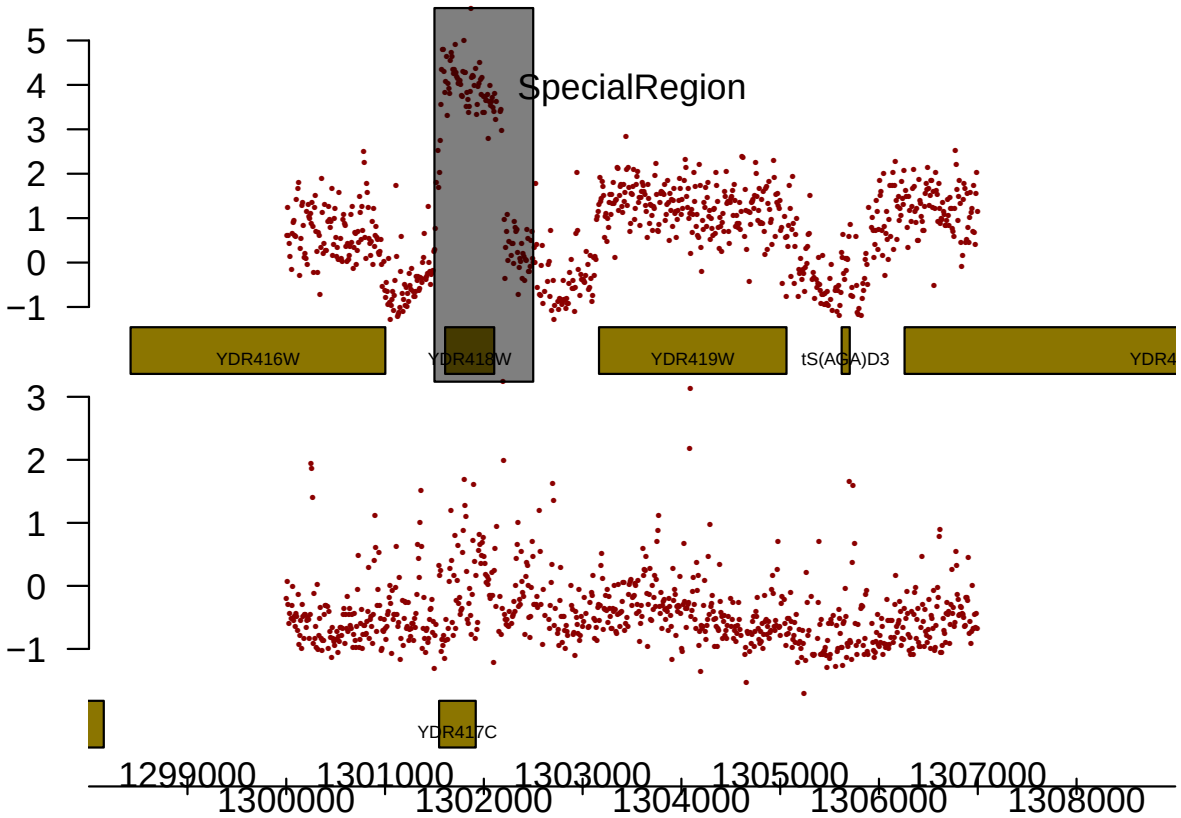
Obtenemos:


+
-
Bsub
10000
12000
14000
16000
18000
20000
22000
11000
13000
15000
17000
19000
21000

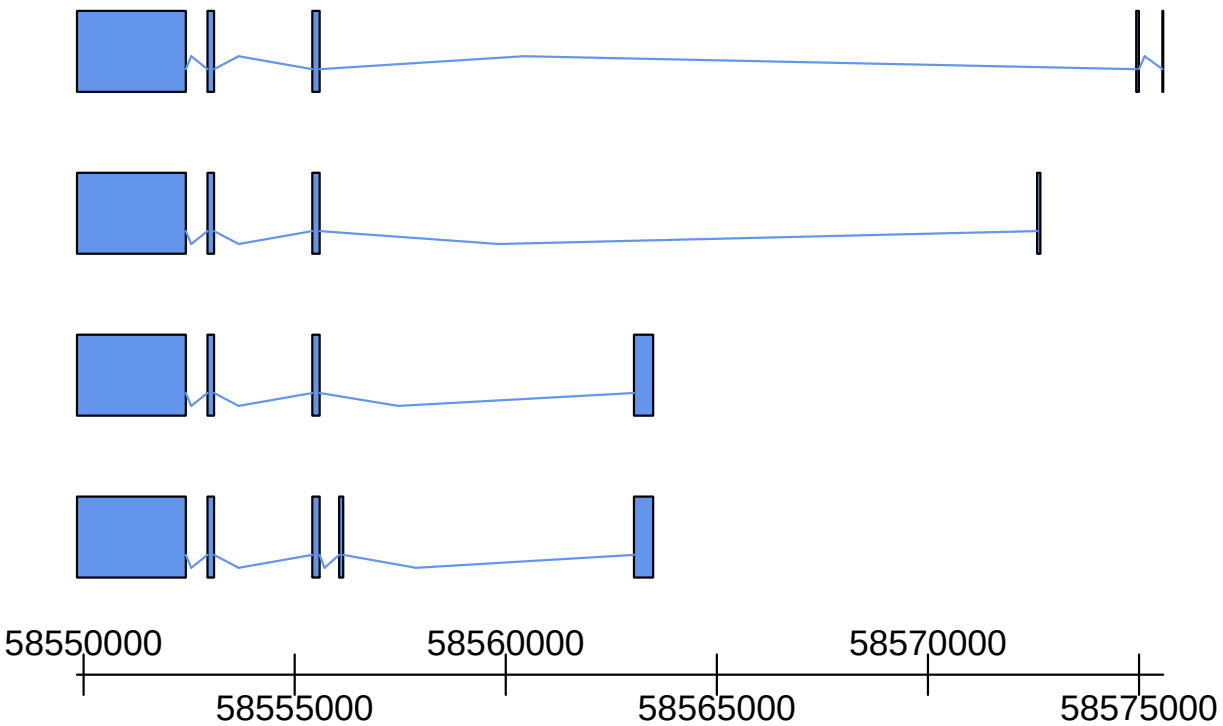
Datos de microarreglos - makeGenericArray



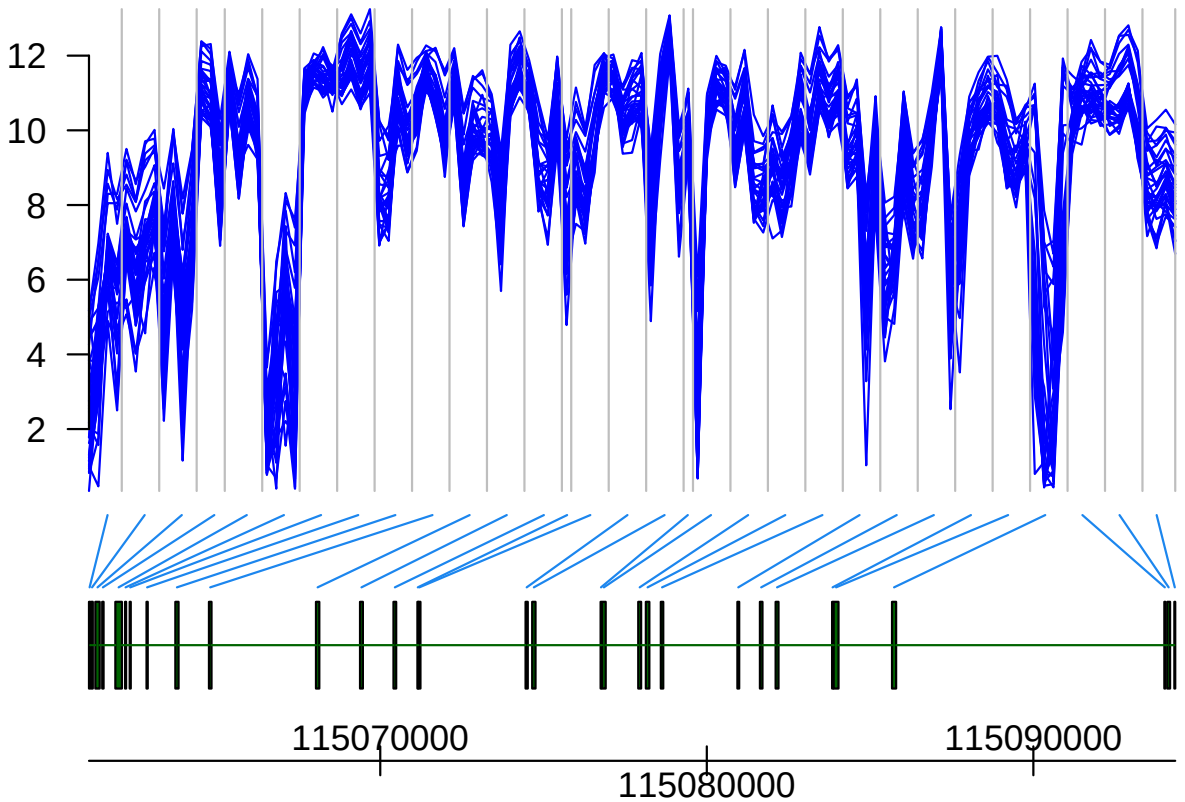
Más complicado



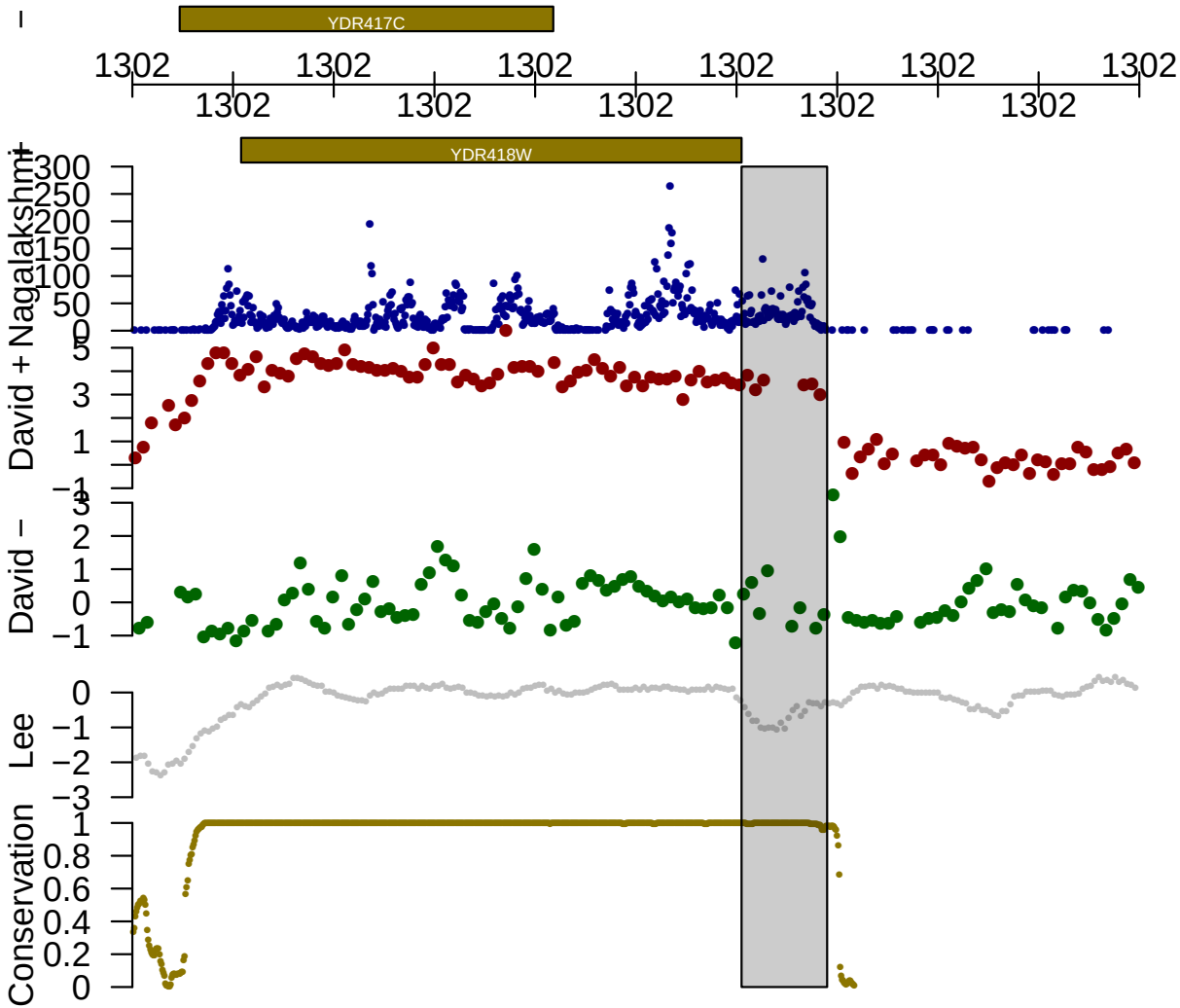
Modelos de genes eucariontes - makeTranscript



makeExonArray junto a makeGeneModel



Finalmente


YDR417C
1302
1302
1302
1302
1302
1302
1302
1302
1302
1302
1302
YDR418W
David + Nagalakshmi
300
250
200
150
100
50
0
David −
3
1
−1
3
2
1
0
−1
Lee
0
−1
−2
−3
Conservation
1
0.8
0.6
0.4
0.2
0

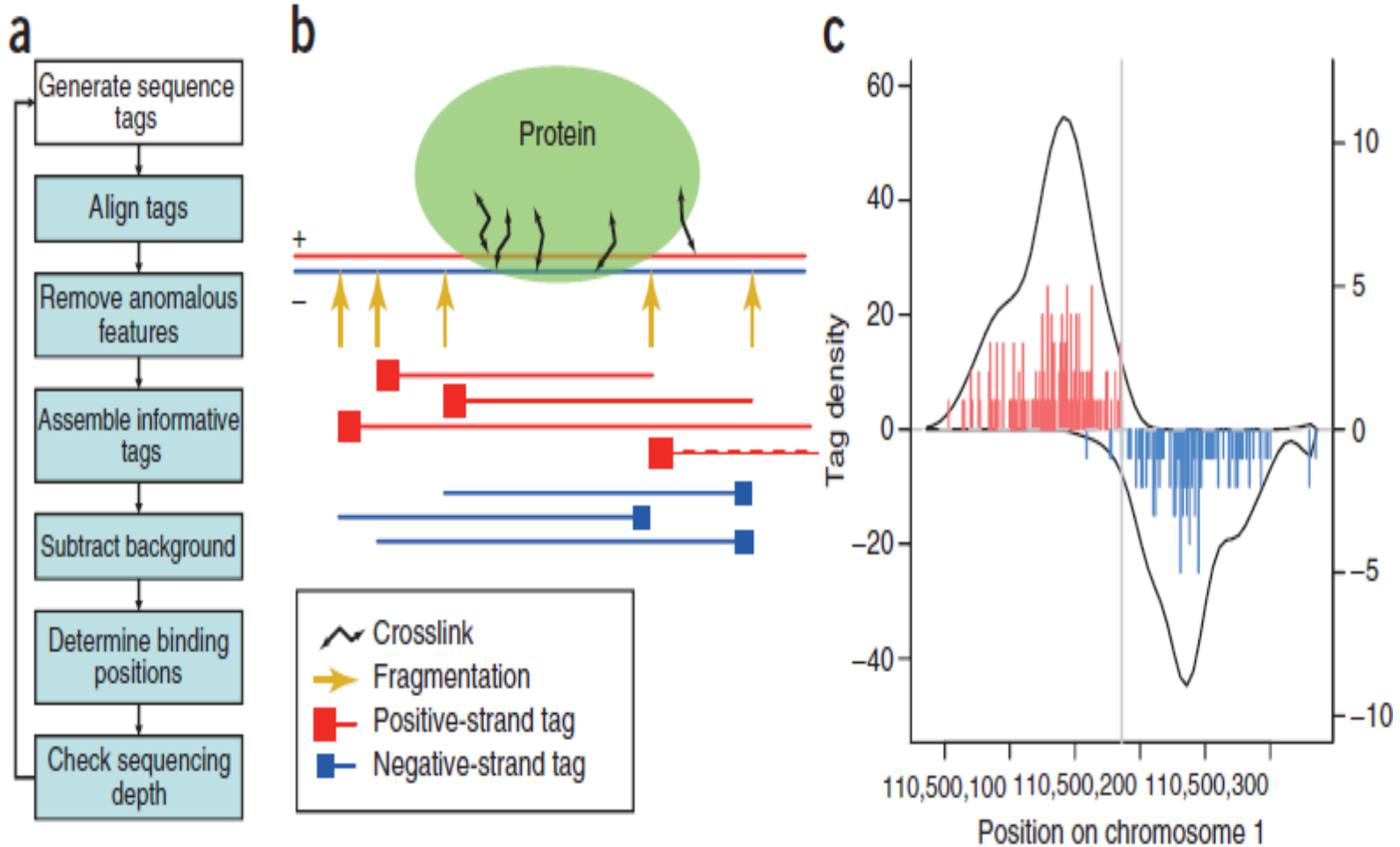
Más info

- biomaRt
- GenomeGraphs
- Artículo GenomeGraphs
- `http://www.bioconductor.org/packages/devel/bioc/html/GenomeGraphs.html`

Breve intro

- Fue diseñado para trabajar con datos de ChIP-seq.
- Entre otros, utiliza los paquetes IRanges, ShortRead y lattice.
- Creado por Deepayan Sarkar, Robert Gentleman, Michael Lawrence y Zizhen Yao

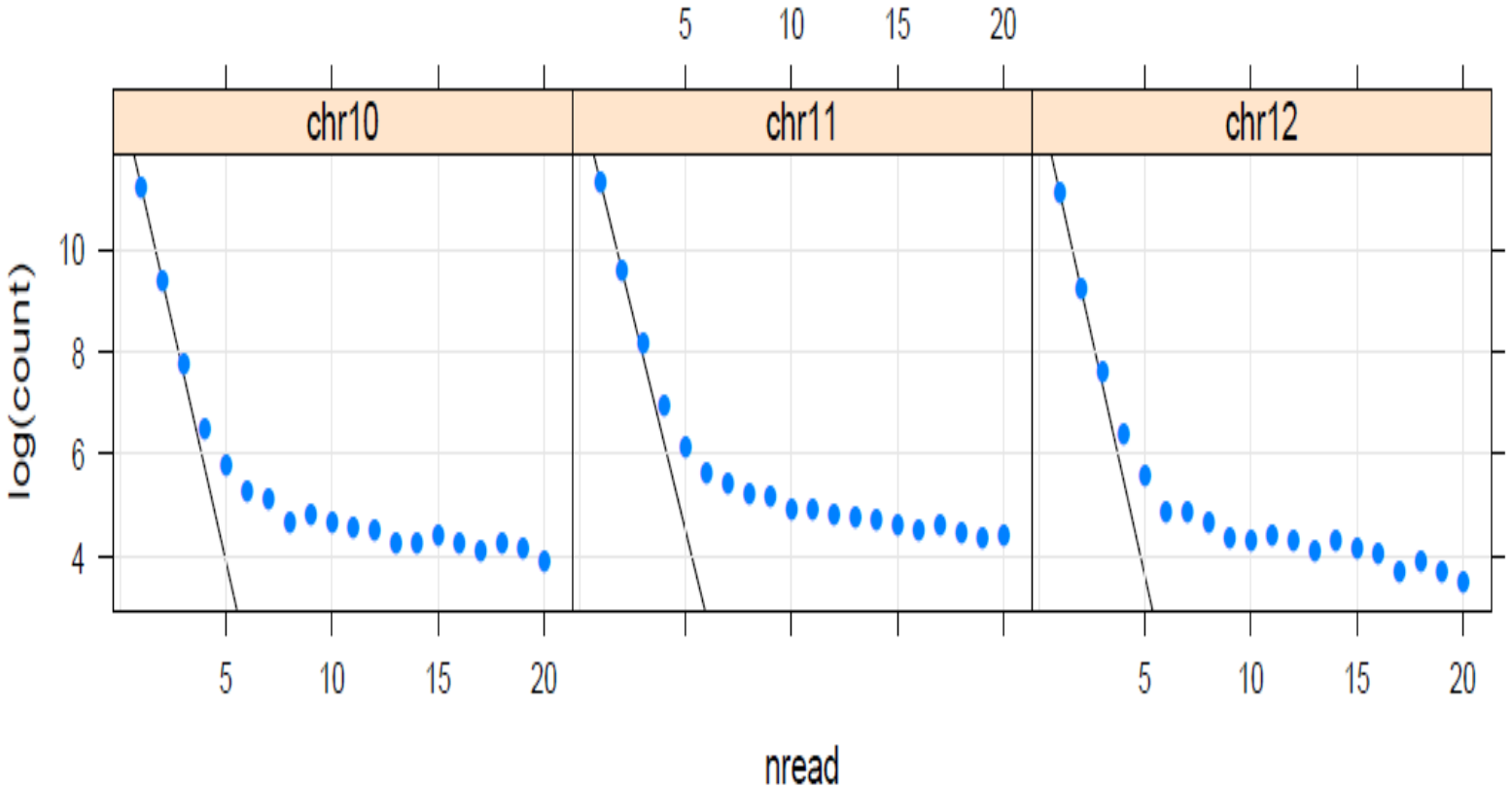
Profundidad de la islas



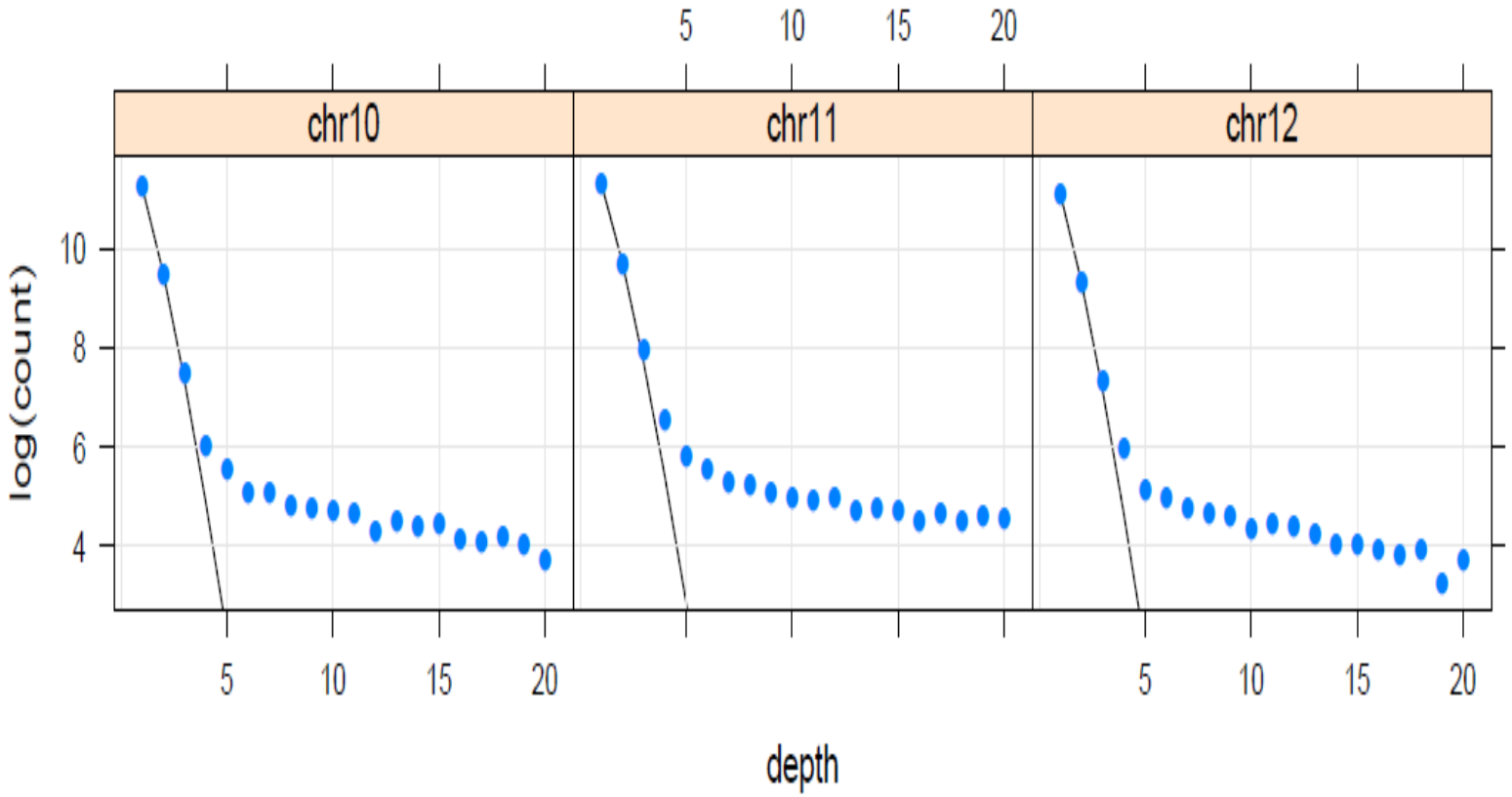
Buscamos...

- Islas y picos!
- Una isla es una región del genoma con cobertura contínua por nuestras secuencias.
- Un pico es una isla con altos valores de cobertura. Es decir, muchas secuencias.

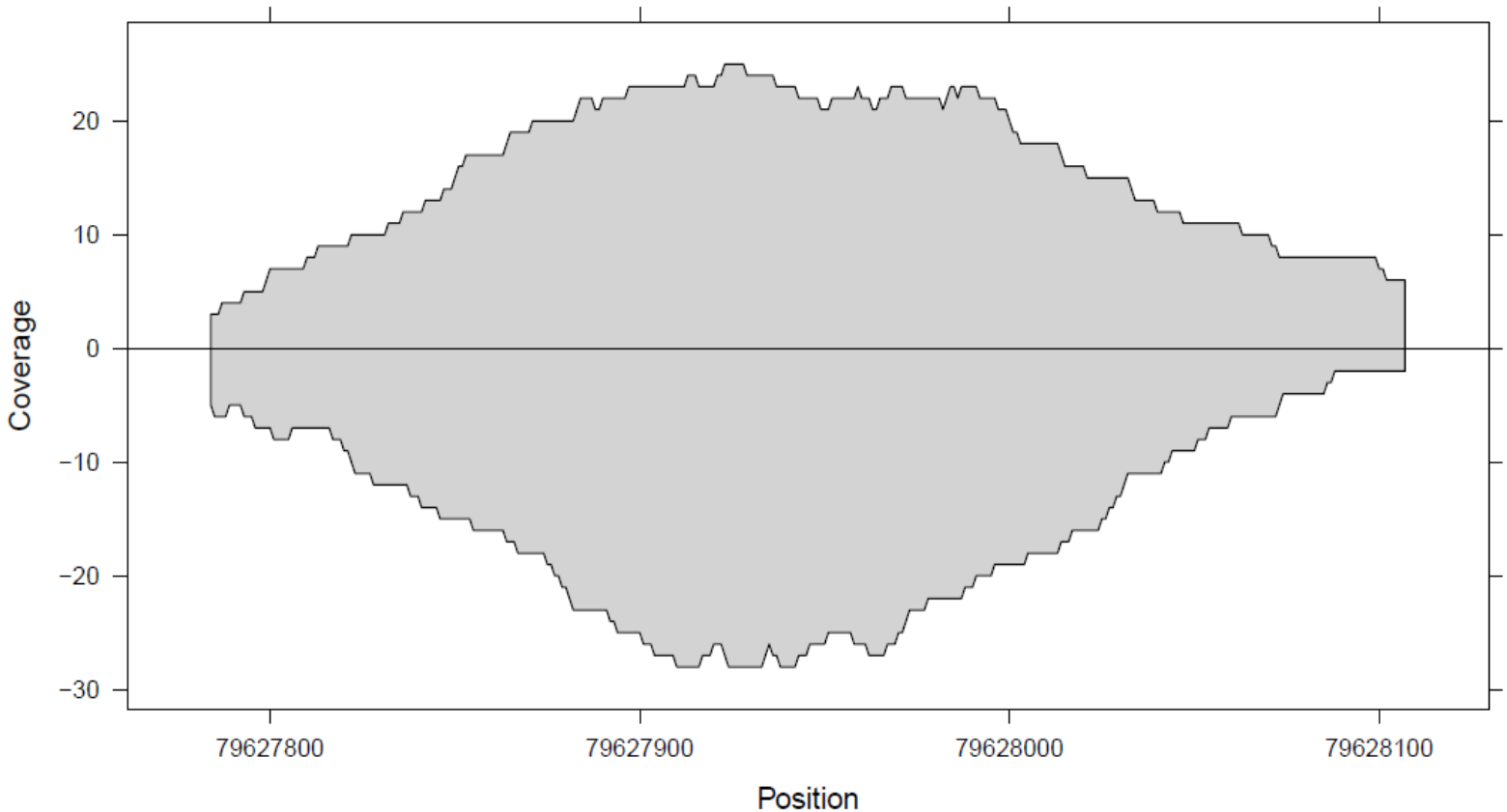
Frec. de n de secs por isla



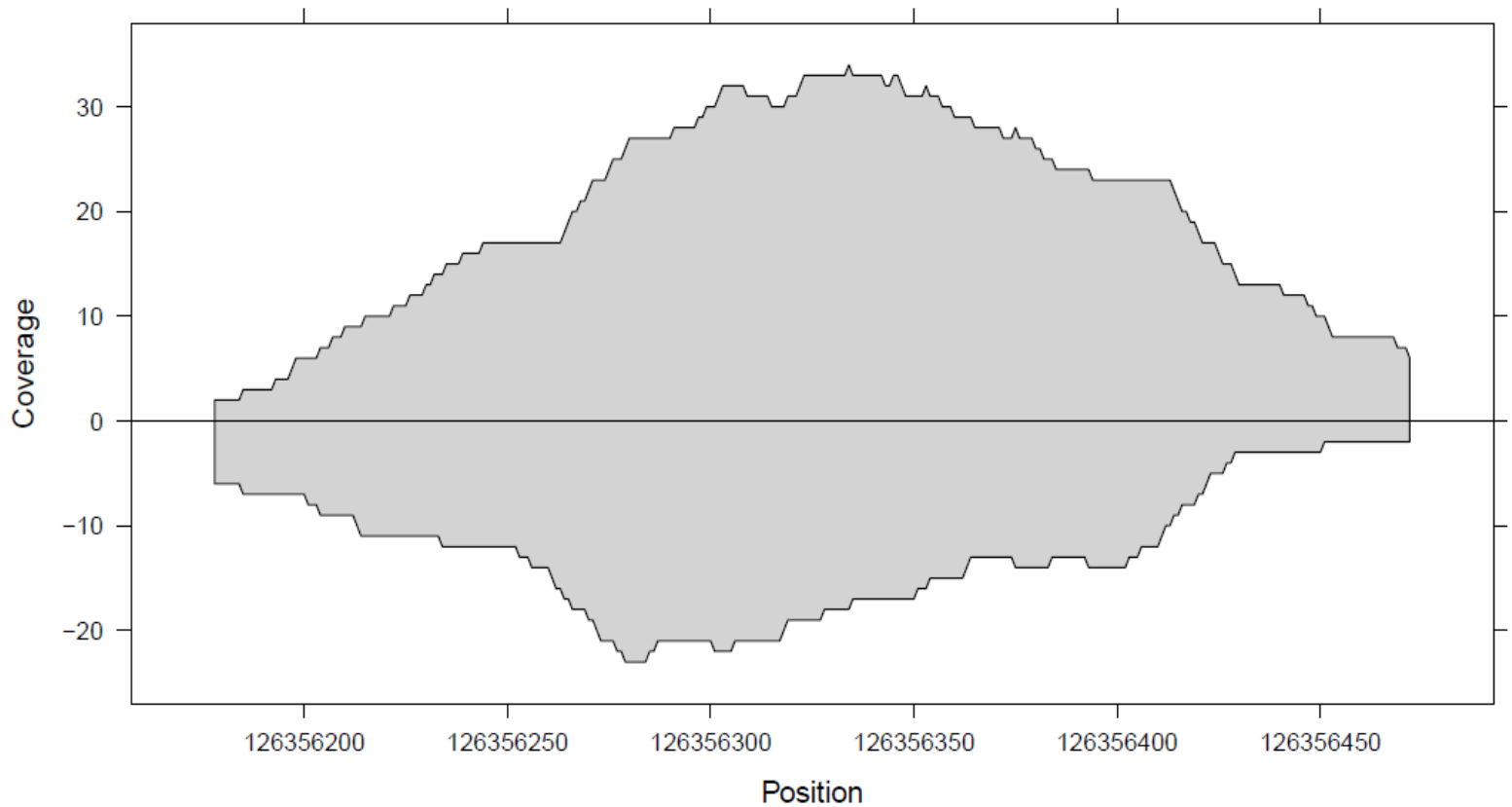
Profundidad de la islas



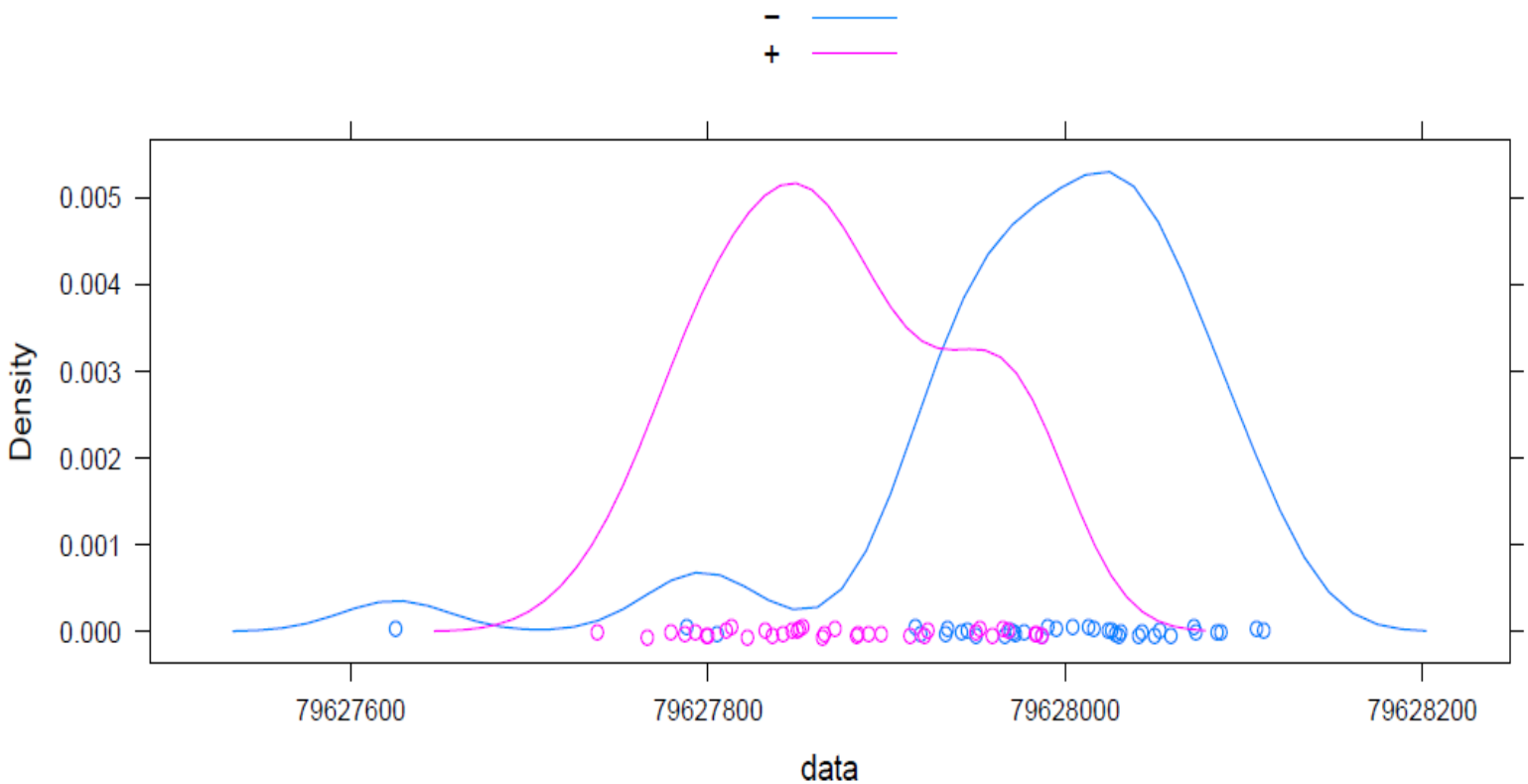
Coverageplot pico 1



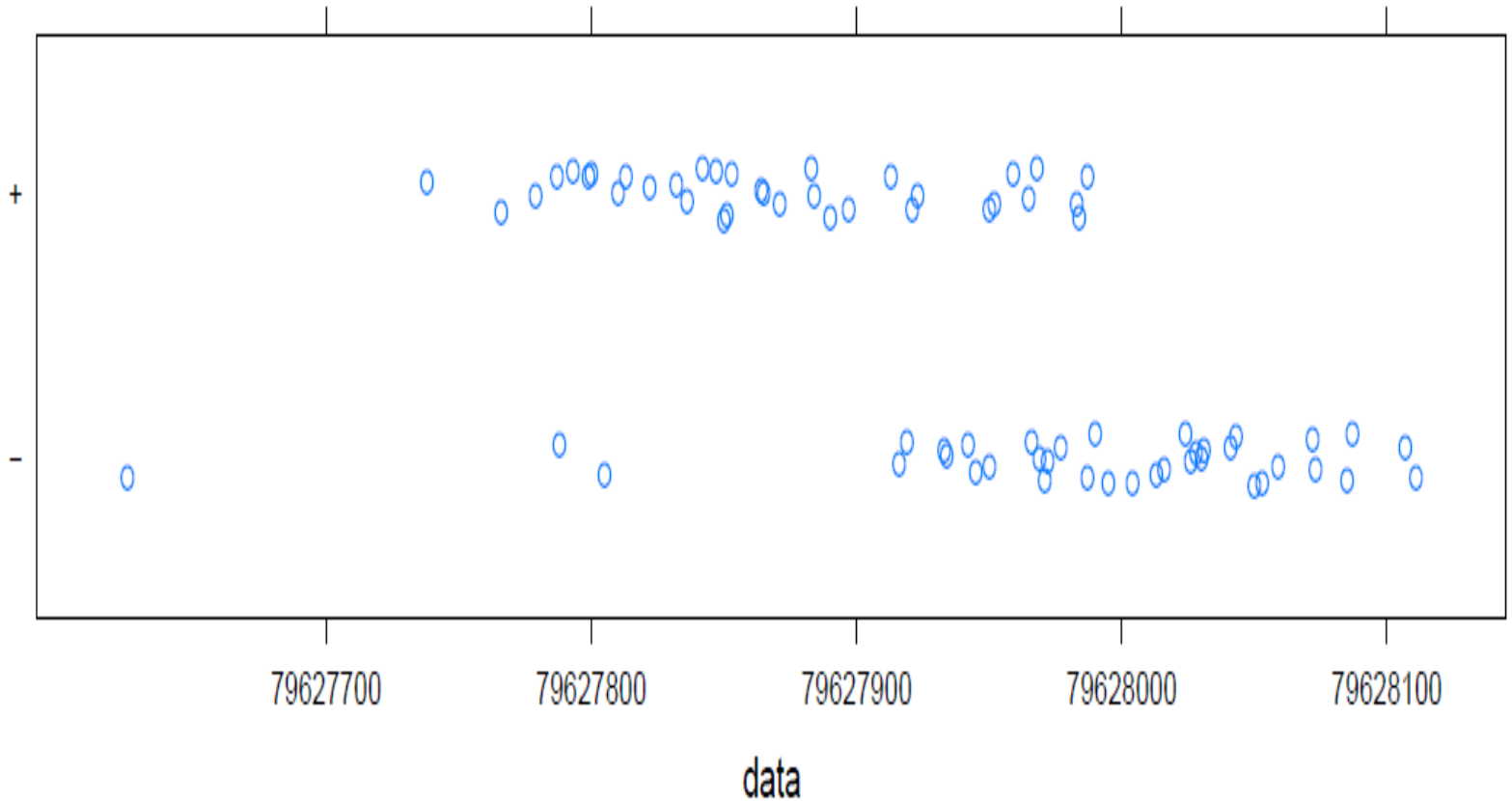
Coverageplot pico 2



Alternativamente graficamos la densidad



O por cadena



Más info

- Paquete chipseq
- Lab de chipseq en el BioC 2009
- Kharchenko et al, 2008

Otros paquetes que no vimos

- snpMatrix
- baySeq, DEGseq, edgeR
- ChIPpeakAnno, ChIPseqR
- Rsamtools
- Rolexa
- ChIPsim
- rtracklayer
- HilbertVis, HilbertVisGUI
- genomeIntervals
- Les recomendamos el curso `http://www.bioconductor.org/workshops/2009/SeattleNov09`

sessionInfo

Información de mi sesión:

```
> sessionInfo()

R version 2.10.0 (2009-10-26)
i386-pc-mingw32

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices
[4] utils     datasets  methods
[7] base

other attached packages:
[1] IRanges_1.4.9
```