Objetos de Bioconductor para datos de expresión

Class	Bioinformática
Created	@Feb 24, 2021 9:22 AM
Materials	
Profesor	Leonardo Collado
Reviewed	☐
Type	Lecture

GenomicRanges Package

Información de genes dentro de un objeto SummarizedExperiments

An Introduction to the GenomicRanges Package

Marc Carlson, Patrick Aboyoun, Hervé Pagès, and Martin Morgan

October 27, 2020

Contents

1 Introduction

The *GenomicRanges* package serves as the foundation for representing genomic locations within the Bioconductor project. In the Bioconductor package hierarchy, it builds upon the *IRanges* (infrastructure) package and provides support for the *BSgenome* (infrastructure), *Rsamtools* (I/O), *ShortRead* (I/O & QA), *rtracklayer* (I/O), *GenomicFeatures* (infrastructure), *GenomicAlignments* (sequence reads), *VariantAnnotation* (called variants), and many other Bioconductor packages.

This package lays a foundation for genomic analysis by introducing three classes (*GRanges*, *GPos*, and *GRangesList*), which are used to represent genomic ranges, genomic positions, and groups of genomic ranges. This vignette focuses on the *GRanges* and *GRangesList* classes and their associated methods.

The *GenomicRanges* package is available at https://bioconductor.org and can be installed via `BiocManager::install`:

```
if (!require("BiocManager"))
    install.packages("BiocManager")
BiocManager::install("GenomicRanges")
```

A package only needs to be installed once. Load the package into an R session with

```
library(GenomicRanges)
```

Experimentando con *SummarizedExperiments*

```
## Lets build our first SummarizedExperiment object
library("SummarizedExperiment")
?SummarizedExperiment

## De los ejemplos en la ayuda oficial

## Creamos los datos para nuestro objeto de tipo SummarizedExperiment
## para 200 genes a lo largo de 6 muestras
nrows <- 200
```

```
ncols <- 6
## Números al azar de cuentas
set.seed(20210223)
counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)
## Información de nuestros genes
rowRanges <- GRanges(
  rep(c("chr1", "chr2"), c(50, 150)),
  IRanges(floor(runif(200, 1e5, 1e6)), width = 100),
  strand = sample(c("+", "-"), 200, TRUE),
  feature_id = sprintf("ID%03d", 1:200)
)
names(rowRanges) <- paste0("gene_", seq_len(length(rowRanges)))
## Información de nuestras muestras
colData <- DataFrame(
  Treatment = rep(c("ChIP", "Input"), 3),
  row.names = LETTERS[1:6]
)
## Juntamos ahora toda la información en un solo objeto de R
rse <- SummarizedExperiment(
  assays = SimpleList(counts = counts),
  rowRanges = rowRanges,
  colData = colData
)

## Exploremos el objeto resultante
rse
```

>output<

```
> rseclass: RangedSummarizedExperiment
#Genes (Renglones) vs Muestras (Columnas)
dim: 200 6
#Información opcional como la descripción del experimento
metadata(0):
#Nombre de las tablas
assays(1): counts
rownames(200): gene_1 gene_2 ... gene_199 gene_200
rowData names(1): feature_id
colnames(6): A B ... E F
colData names(1): Treatment
```

```
## Información de los genes en un objeto de Bioconductor
> rowRanges(rse)

GRanges object with 200 ranges and 1 metadata column:
      seqnames        ranges strand |  feature_id
         <Rle>     <IRanges>  <Rle> | <character>
gene_1     chr1 286235-286334      + |       ID001
```

```
gene_2     chr1 586770-586869      - |       ID002
gene_3     chr1 577897-577996      + |       ID003
gene_4     chr1 494350-494449      + |       ID004
gene_5     chr1 686692-686791      - |       ID005
...      ...           ...    ... .          ...
gene_196     chr2 804998-805097      - |       ID196
gene_197     chr2 177462-177561      - |       ID197
gene_198     chr2 649993-650092      - |       ID198
gene_199     chr2 275940-276039      - |       ID199
gene_200     chr2 487418-487517      + |       ID200
-------
seqinfo: 2 sequences from an unspecified genome; no seqlengths

## Número de genes y muestras
dim(rse)

## IDs de nuestros genes y muestras
dimnames(rse)

## Nombres de tablas de cuentas que tenemos (RPKM, CPM, counts, logcounts, etc)
assayNames(rse)

## El inicio de nuestra tabla de cuentas
head(assay(rse))

## Tabla con información de los genes
rowData(rse) # es idéntico a 'mcols(rowRanges(rse))'
```

```
#Para saber el número de cromosomas
#Como un unique
> seqlevels(rse)[1] "chr1" "chr2"
> unique(as.vector(seqnames(rowRanges(rse))))[1] "chr1" "chr2"
```

Ejercicio

¿Qué es lo que pasa en esos dos comandos?

```
## Comando 1
# Se toman los genes 1 y 2 en todas las muestras
# Se asegura de que haga el subconjunto adecuado en todas las tablas
> rse[1:2, ]
class: RangedSummarizedExperiment
dim: 2 6
metadata(0):
assays(1): counts
rownames(2): gene_1 gene_2
rowData names(1): feature_id
```

```
colnames(6): A B ... E F
colData names(1): Treatment

> head(assay(rse[1:2, ]))
          A        B         C        D         E        F
gene_1 2577.960 8526.615 2226.3070 3615.897 1723.8851 3267.954
gene_2 7793.183 3462.579  478.2716 7688.384  295.2813 2698.921

-------------------------------------------

## Comando 2
# Se accede a las muestras A,D,F de todos los genes
# Es posible debido a que en el objeto rse tenemos nombres de todas las muestras

> rse[, c("A", "D", "F")]
class: RangedSummarizedExperiment
dim: 200 3
metadata(0):
assays(1): counts
rownames(200): gene_1 gene_2 ... gene_199 gene_200
rowData names(1): feature_id
colnames(3): A D F
colData names(1): Treatment

> head(assay(rse[, c("A", "D", "F")]))
          A         D        F
gene_1 2577.960 3615.8967 3267.954
gene_2 7793.183 7688.3839 2698.921
gene_3 9571.769 9916.0076 6880.067
gene_4 4641.969  258.5218 8737.586
gene_5 6436.758 3588.1194 7290.890
gene_6 6845.704 9750.8286 2192.060
> which(colnames(rse)%in%c("A", "D", "F"))[1] 1 4 6
> rse$Treatment[1] "ChIP"  "Input" "ChIP"  "Input" "ChIP"  "Input"
```

```
#Tras bambalinas
which(colnames(rse)%in%c("A","D","F"))

# Salen todos los valores de la columna treatment que está definido en la tabla de colData
rse$Treatment
```

Usando iSEE

Visualizar interactivamente objetos SummarizedExperiment

```
## Explora el objeto rse de forma interactiva
library("iSEE")
iSEE::iSEE(rse)
```

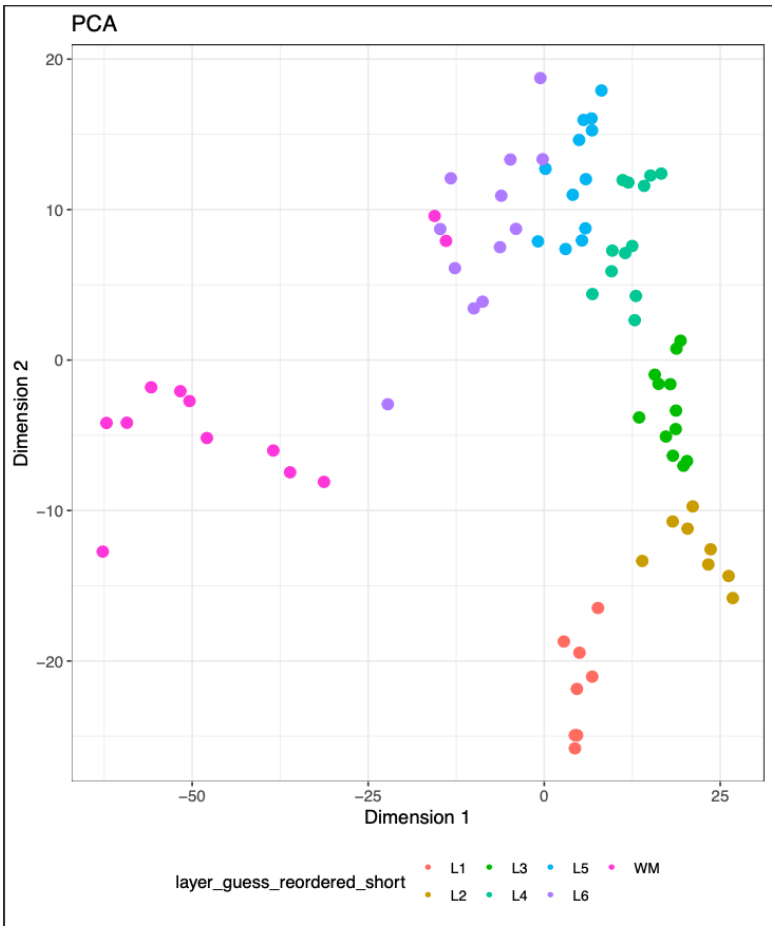
Permite descargar el código de R o la figura directamente para usarla en los papers.

shinyapps.io

Deploy your Shiny applications on the Web in minutes
Deploying your Shiny applications could not be easier. You don't need to own a server or know how to configure a firewall

https://www.shinyapps.io/

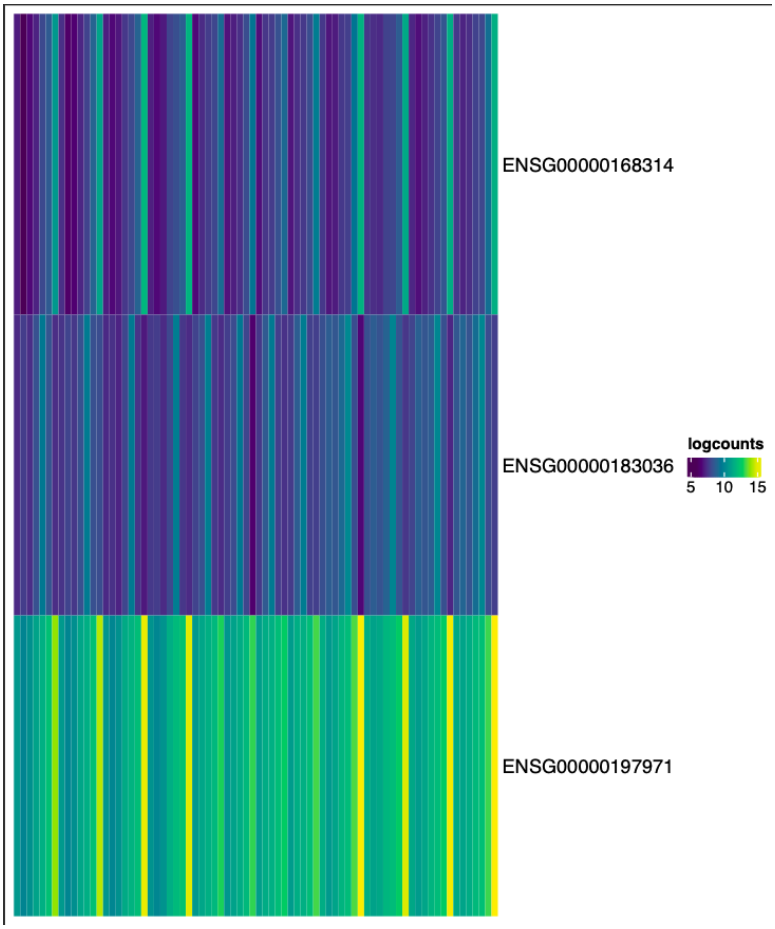
Ejercicio



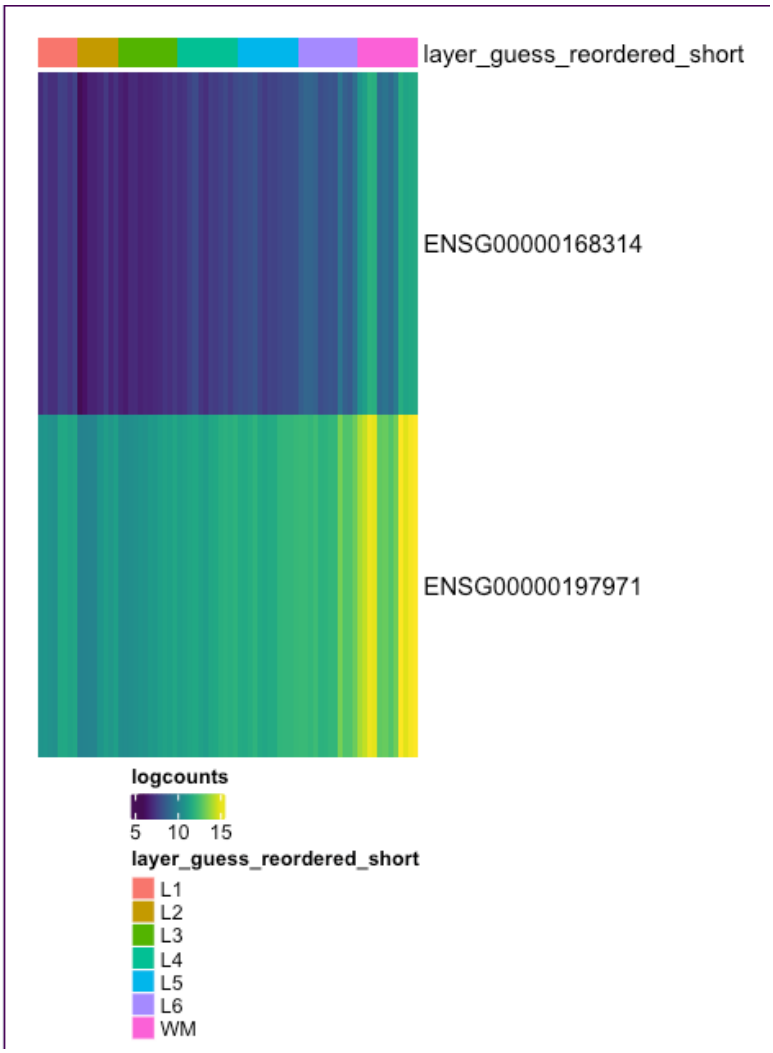
A partir del análisis de relación entre los genes:

- ENSG00000168314 MOBP
- ENSG00000183036 PCP4
- ENSG00000197971 MBP

El heatmap indica que los genes MOBP y MBP son más similares entre sí, en comparación a PCP4.



Esta gráfica indica que ambos genes tienen más presencia en las capas WM, L6 y un poco en L1.


layer_guess_reordered_short
ENSG00000168314
ENSG00000197971
logcounts
5
10
15
layer_guess_reordered_short
L1
L2
L3
L4
L5
L6
WM