Ciclo de clases en bioinformática: Principios de R

Leonardo Collado Torres
lcollado@ibt.unam.mx
Licenciado en Ciencias Genómicas
www.lcg.unam.mx/~lcollado/

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

Octubre - Noviembre, 2009

Introducción a R

Introducción a R

Ciclo de clases en bioinformática

El plan:

1. Principios de R con *Leo*
2. UNIX con *Vero*
3. Perl con *Leti*
4. Bases de datos con *Vero*
5. Java con *Blanca*[1]

Bienvenidos al reto!!!

[1] Por confirmar :)

Principios de R

Objetivos:

1. Aprender a usar R como calculadora
2. Leer datos en R
3. Hacer gráficas usando R. . . adiós Excel
4. Saber encontrar ayuda para R
5. Manejo básico de datos en R
6. Sentirse cómodo con la sintáxis de R
7. Iniciar la familirización con la bioinformática[2]

[2] Ciclos, condicionales, tipos de datos

Principios de R

No vamos a ver:

1. La parte estadística de R
2. Gráficas avanzadas
3. Bioconductor

Raíces

- R es, de alguna forma, el hijo de S creado por `Bell labs`. En realidad es una *implementación* de S, tal como S-PLUS.
- Fue creado por Ross Ihaka y Robert Gentleman.
- Es un lenguaje interpretado y *vive* en el momento de la interpretación.
  - ¿Eso que quiere decir?

Info sobre R

- Es útil como un ambiente de programación
  1. Gráficas
  2. Estadística (bueno para números)
  3. Herramientas para datos masivos biológicos (genómicos) en Bioconductor
- Sigue un ciclo de 6 meses: siempre hay versión estable y en desarrollo.
  - ¿Cúal es la versión estable más reciente?
- Es multi plataforma: Windows, Max y Linux/Unix.
- El sitio principal es el del *Comprehensive R Archive Network* mejor conocido como CRAN: `http://cran.r-project.org`

Instalación de R

Para ahora ya todos lo deben haber instalado. . .

1. Entrar a CRAN
2. Dar click en R Binaries
3. Escogan su plataforma, por ejemplo Windows
4. En caso de tener la opción, escogan `base`
5. Bajen el instalador y sigan sus instrucciones :)

En general

La idea es que ustedes:

- Bajen la presentación y el archivo `.R` asociado a su compu
- Sigan la presentación[3]
- Para los códigos, copien y péguenlos en vez de escribirlos.[4]

Esto con el fin de hacer la clase más dinámica. En general hago preguntas para checar que no los haya perdido en el camino... También pregunten si tienen dudas!!!

[3] Las hago de tal forma que en un futuro pueden servirles de referencia.

[4] Toma menos tiempo y es mejor que entiendan el código a que se la pasen escribiendo todo la clase.

dotProject

- Voy a usar el dotProject para subir las presentaciones y archivos `.R`
- En cada clase les voy a dar ejercicios[5]. Compartan sus preguntas vía el foro del curso (en dotProject). Otros tal vez tienen la misma pregunta.
- Les voy a responder solo vía el foro :)[6]
- Una semana después de la clase, voy a subir un archivo con las respuestas esperadas. Revísenlos!
- Al final, si quieren aprender a usar R tienen que practicar, y practicar, y practicar, y prac. . .

[5] TAREA!! jeje :)

[6] Es para aprendar a usar el foro y compartir allí las preguntas/respuestas.

Abrir R

Para abrir R hay varias opciones dependiendo de su plataforma[7]:

- En Windows, den doble click en el ícono de R. O si quieren algo más *rudo* escriban R en la consola de comandos.
- En Mac, ya sea que le den doble click al ícono o que escriban R en una ventana de terminal.
- En Linux/Unix, escriban R en la terminal.

¿Ya lo abrieron?

. . .

Una opción avanzada es usar el GNU Emacs o el XEmacs. Tendrán que instalar el ESS.

[7] O si tienen alguna variable de ambiente especial. . .

Mensaje inicial

Al abrir R sale un mensaje con información.

- ¿Qué versión de R tienes instalada?
- ¿Cómo citarías a R?
- ¿Cómo ves la lista de contribuidores?
- Hay una mención especial para alguien, ¿quién es y por qué?

Para salir de R

- Si ahora quieren salirse, tienen que usar la función q de quit. Ya sea que la usen sin ningún *argumento*:

  ```
  > q()
  ```

- O especificando que no quieren guardar nada:

  ```
  > q("no")
  ```

La función básica

Hay muchas formas, incluyendo `Google`!

- La función básica es help:

  ```
  > help("LoQueBuscas")
  ```

- Se puede abreviar así:

  ```
  > `?`(LoQueBuscas)
  ```

- Busquen la ayuda de `q`. ¿`q` es un alias de qué función?
- ¿Qué pasa si escriben lo siguiente?

  ```
  > Help("quit")
  ```

En el navegador

- Les recomiendo que usen help.start para que las páginas de ayuda es abran en su navegador

```
> help.start()
> `?`(q)
```

Buscando funciones

- `help` es para cuando conocen el nombre exacto de la función. Si intuyen el nombre de la función, podemos buscar funciones usando apropos:

```
> apropos("quit")
```

```
[1] "quit"
```

- ¿Cúantas funciones salen si buscan `save` con `apropos`?

Argumentos

- Si ya conocen la función pero quieren recordar los argumentos de esta, usen args:

```
> args(apropos)

function (what, where = FALSE, ignore.case = TRUE, mode = "any")
NULL
```

- Noten que muestra los valores *default* de los argumentos.

En la red

- Otra opción para encontrar ayuda es vía el sitio de R usando RSiteSearch

  ```
  > RSiteSearch("help")
  ```

- Es un poco más tedioso, pero busca en una lista de emails de ayuda.

En la lista de mails

- Otra opción es que entren directamente (se suscriban) a la lista de mails de ayuda:
  `https://stat.ethz.ch/mailman/listinfo/r-help`
- En general les van a pedir información sobre su sesión.
- ¿Qué función usarían?

```
> apropos("session")

[1] "sessionData"
[2] "sessionInfo"
[3] "setSessionTimeLimit"
```

Cálculos :)

R es muy útil como calculadora, aunque no es de botones.

- Una suma simple:

```
> 2 + 3
```

```
[1] 5
```

- Si se fijan, imprime un uno entre corchetes. Es la línea 1 de la salida.
- ¿Cómo harían 2 por 3?
- División:

```
> 2/3
```

```
[1] 0.6666667
```

- Noten que redondea al 7mo dígito. Si lo requieren, se puede cambiar :)

Cálculos II

- Exponentes:

```
> 2^3
```

```
[1] 8
```

- Logaritmo:

```
> log(3, base = 2)
```

```
[1] 1.584963
```

- Raíz cuadrada:

```
> sqrt(2)
```

```
[1] 1.414214
```

- Pi:

```
> pi
```

```
[1] 3.141593
```

- Encuentren el área de un círculo de radio 84.5901 cm

Cálculos III

- El área es de 22479.62 cm2. ¿Tuvieron que usar paréntesis?
- ¿Por qué la siguiente expresión nos da 10?

```
> 1e+06/1e+05

[1] 10
```

Abundan!

- En la computación - informática - bioinformática, se usan mucho los condicionales.
- Por ejemplo: ¿es 3 mayor a 2?
- ¿Alguna idea de como hacer esto en R?

Solución rápida

- Es como si lo escribieran en su cuaderno:

```
> 3 > 2
```

```
[1] TRUE
```

- R te va a decir si es falso o verdadero :)
- Expliquenme porque sale `TRUE` en la siguiente expresión:

```
> 3 > 2 & 3 < 4
```

```
[1] TRUE
```

Asignar valores

- En cualquier lenguaje generalmente guardamos valores en un objeto[8].
- Asignemos un 2 a x, y luego usemos x:

```
> x <- 2
> x > 2
```

```
[1] FALSE
```

```
> x <= 2 & x^2 < 5
```

```
[1] TRUE
```

[8] O variable, en R son objetos

if y else

- En R usamos el condicional if para *controlar el flujo*:

```
> if (x < 3) {
+     print("hola")
+ }
```

```
[1] "hola"
```

- Muchas veces tenemos dos opciones por lo que usamos el else:

```
> if (x < 2) {
+     print("hola")
+ } else {
+     print("boo")
+ }
```

```
[1] "boo"
```

if y else

- Para 3 o más opciones hay que usar varios if y else:

```
> if (x < 2) {
+     print("hola")
+ } else if (x < 1) {
+     print("boo")
+ } else {
+     print("loteria")
+ }

[1] "loteria"
```

Un ejercicio de lógica

- ¿Cúal es el resultado de este código?

```
> x <- 1.2
> if (x > 2) {
+     print("opcion1")
+ } else if (x < 0) {
+     print("opcion2")
+ } else if (x^2 < 3) {
+     print("opcion3")
+ } else {
+     print("no se")
+ }
```

ifelse

- Una opción sencilla para 2 opciones, y similar a Excel, es la función ifelse:

```
> x <- 5
> ifelse(x/5 == 1, "Verdadero", "Falso")

[1] "Verdadero"
```

Para repetir

- Muchas veces hay que repetir un cálculo, o una función para un rango de valores.
- Para hacerlo, necesitamos un ciclo, que es otra forma de *control de flujo*.
- En R hay varias opciones: for, while, repeat, apply

Secuencias

- Pero antes veamos como crear una secuencia de números.
- Podemos concatenar dos o más números:

```
> c(2, 3)
[1] 2 3
> c(2, 7, 3)
[1] 2 7 3
```

- Si tenemos números seguidos, podemos aprovechar un atajo:

```
> 1:4
[1] 1 2 3 4
> 6:3
[1] 6 5 4 3
```

Secuencias

```
> c(1:3, 8, 0:-3)
[1]  1  2  3  8  0 -1 -2 -3
```

- Estamos creando un *vector atómico* de tipo entero.

Funciones relacionadas

- Otras veces queremos hacer secuencias. Por ejemplo, 10, 20, 30, . . . , hasta 100. Para eso usamos seq:

```
> seq(10, 100, by = 10)
```

```
 [1]  10  20  30  40  50  60  70  80  90
[10] 100
```

- O luego queremos repetir un número varias veces, y para ahorrar tiempo usamos rep:

```
> rep(2, 3)
```

```
[1] 2 2 2
```

- Creen un vector con los siguientes números: 1 al 3, 8, 0 a menos 2, repitan 4 veces el 5, y terminen con los la tabla del 5 hasta 10.

Solución

- ¿Sencillo?

```
> c(1:3, 8, 0:-2, rep(5, 4), seq(0,
+     50, by = 5))

 [1]  1  2  3  8  0 -1 -2  5  5  5  5  0
[13]  5 10 15 20 25 30 35 40 45 50
```

El ciclo for

- Probablemente es el tipo de ciclo más usado.
- La idea es hacer algo para cada elemento de una secuencia, como un *for each*.
- Un ejemplo :) Para cada número entre 1 y 10, si su raíz cuadrada es mayor a 2.5 imprimimos `sip`, de lo contrario imprimimos un `nop`.

```
> for (i in 1:10) {
+     if (sqrt(i) > 2.5) {
+         print("sip")
+     }
+     else {
+         print("nop")
+     }
+ }
```

El ciclo for

```
[1] "nop"
[1] "nop"
[1] "nop"
[1] "nop"
[1] "nop"
[1] "nop"
[1] "sip"
[1] "sip"
[1] "sip"
[1] "sip"
```

Guardando

- Generalmente queremos guardar los resultados de nuestro ciclo en un objeto. La forma sencilla es usar la concatenación, pero para hacerlo debemos *inicializar* el objeto.
- Usemos un `for` para encontrar los cuadrados de 1 hasta 10.

```
> res <- NULL
> for (i in 1:10) {
+     res <- c(res, i^2)
+ }
> res
```

```
 [1]   1   4   9  16  25  36  49  64  81
[10] 100
```

Guardando

- Ahora pueden hacer el siguiente ejercicio: Usando solo sumas y ciclos for, multipliquen por 10 los números enteros entre 1 y 10.
- Van a necesitar dos ciclos `for` y dos objetos donde guarden el resultado. Uno de los objetos lo van a inicializar como nulo y otro como 0.

Solución

- Cada objeto (res y temp en mi caso) lo vamos a usar para guardar el resultado de un ciclo for.

```
> res <- NULL
> for (i in 1:10) {
+     temp <- 0
+     for (j in 1:10) {
+         temp <- temp + i
+     }
+     res <- c(res, temp)
+ }
> res
 [1]  10  20  30  40  50  60  70  80  90
[10] 100
```

Guardando su sesión

- Muchas veces necesitan parar de trabajar, apagar su compu, y seguir después.
- Las funciones save.image y load son muy útiles para esto :)

```
> save.image(file = "archivo.Rdata")
> load(file = "archivo.Rdata")
```

Guardando objetos

- Otras veces solo quieren guardar un objeto. Para eso usaremos save:

```
> save(objeto, file = "archivo.Rda")
```

- O varios objetos:

```
> save(objeto1, objeto2, file = "archivo.Rda")
```

- Luego, cuando sigan trabajando lo(s) leen con:

```
> load(file = "archivo.Rda")
```

Historia

- Si quieren guardar la historia de los comandos que usaron es con savehistory:

```
> savehistory(file = "archivo.Rhistory")
```

- Y luego la cargan con `loadhistory`:

```
> loadhistory(file = "archivo.Rhistory")
```

Una gráfica

Entrando a la parte final ...

- Vamos a hacer una gráfica sencilla.
- Vamos a graficar en el eje X los números del 1 al 10, y en el eje Y sus cuadrados.

Creemos los objetos

- Primero creemos un objeto x y otro y con estos valores:

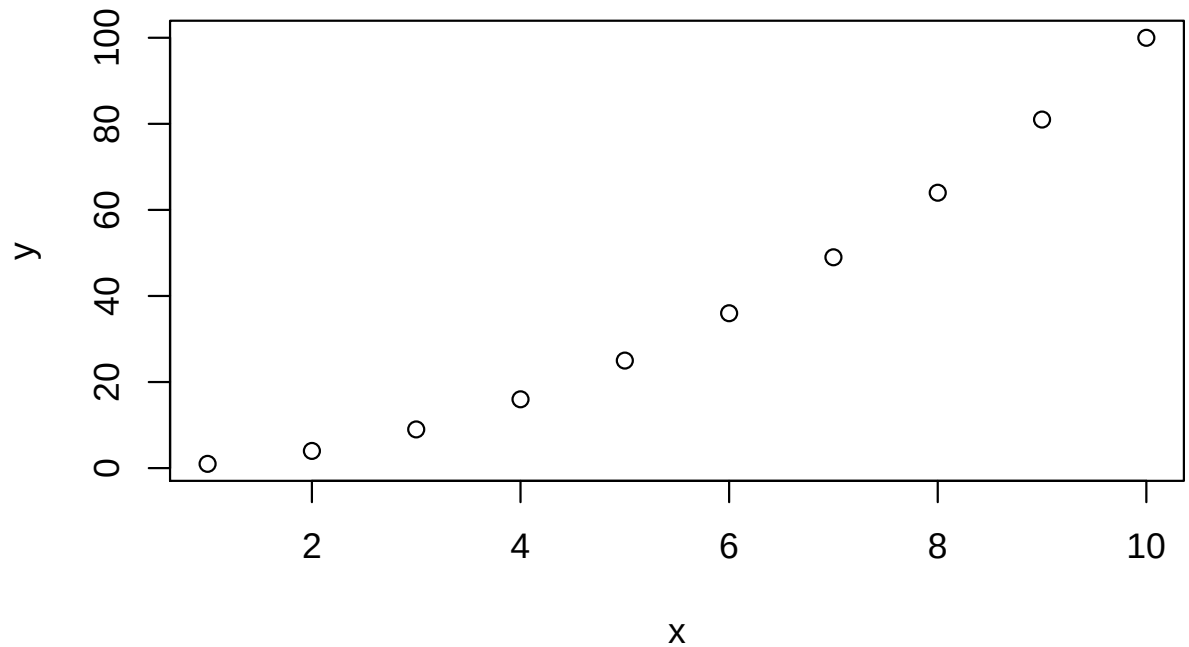
```
> x <- 1:10
> y <- x^2
```

- Me estoy ahorrando un for al encontrar el cuadrado directamente. Es parte de la regla de reciclaje de R :)

La gráfica

Luego es solo cosa de usar la función plot

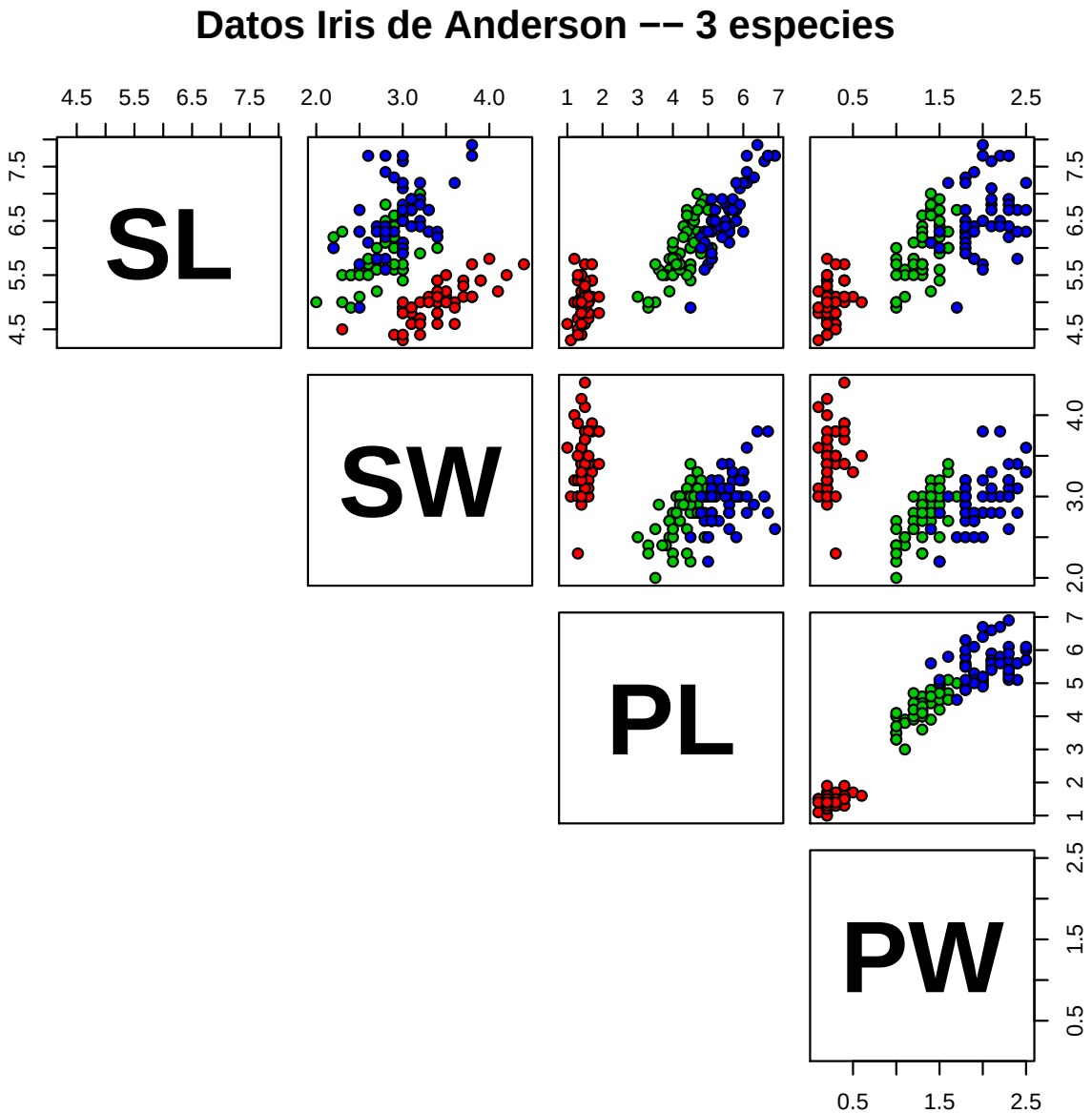
```
> plot(x, y)
```



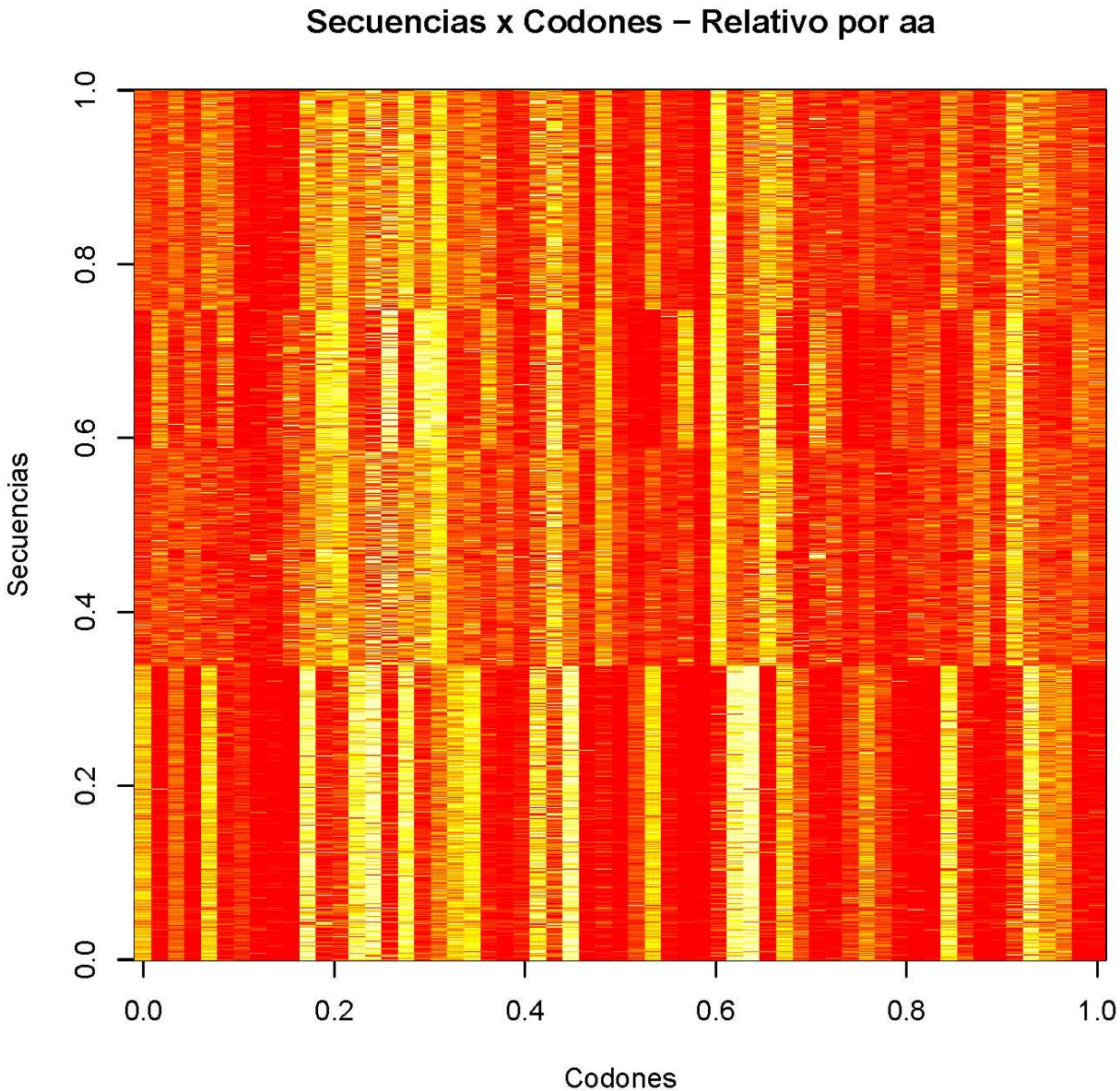
Para picarlos :)

- Las siguientes gráficas son ejemplos de lo que puede llegar a hacer con R

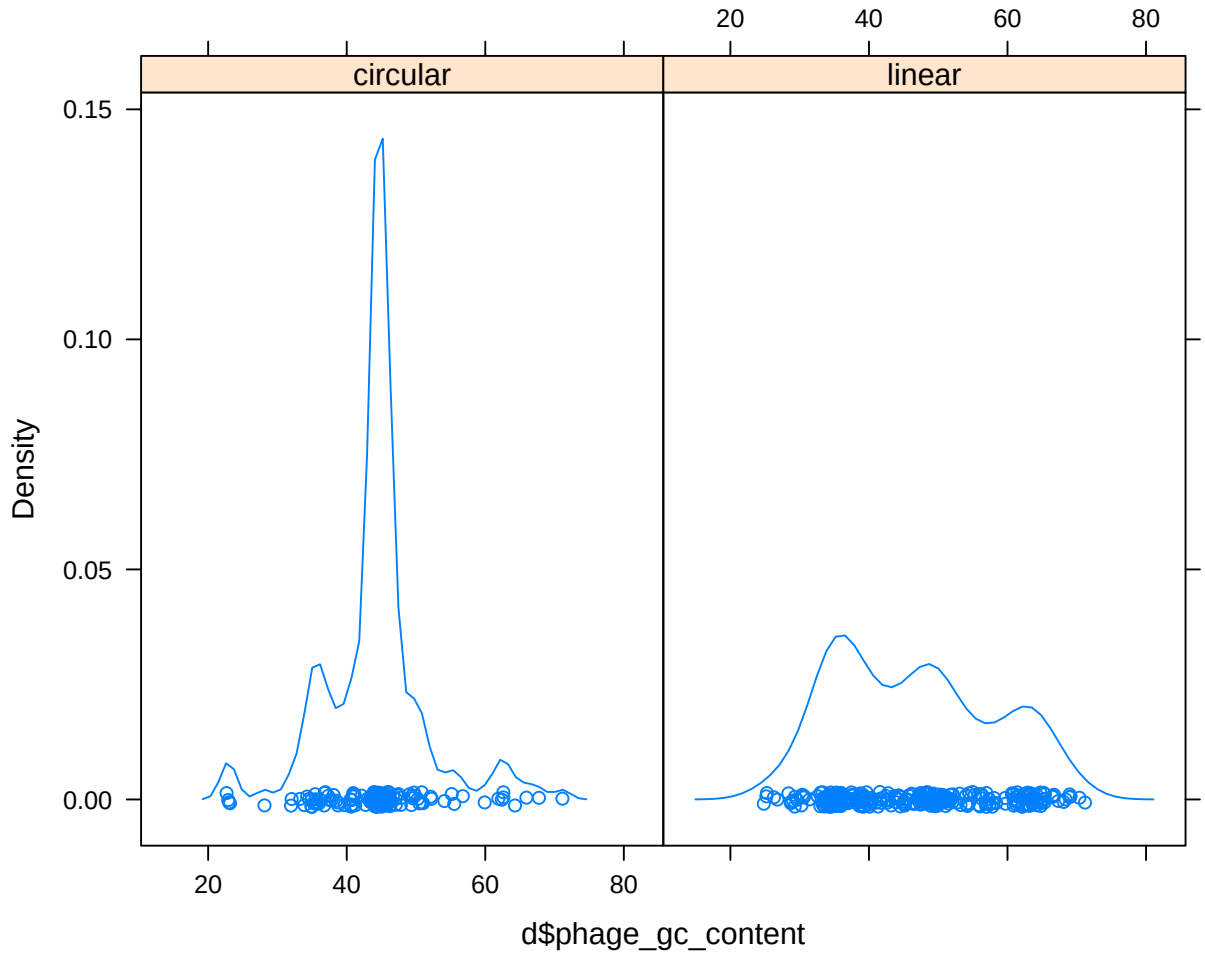
Un scatterplot de los datos Iris



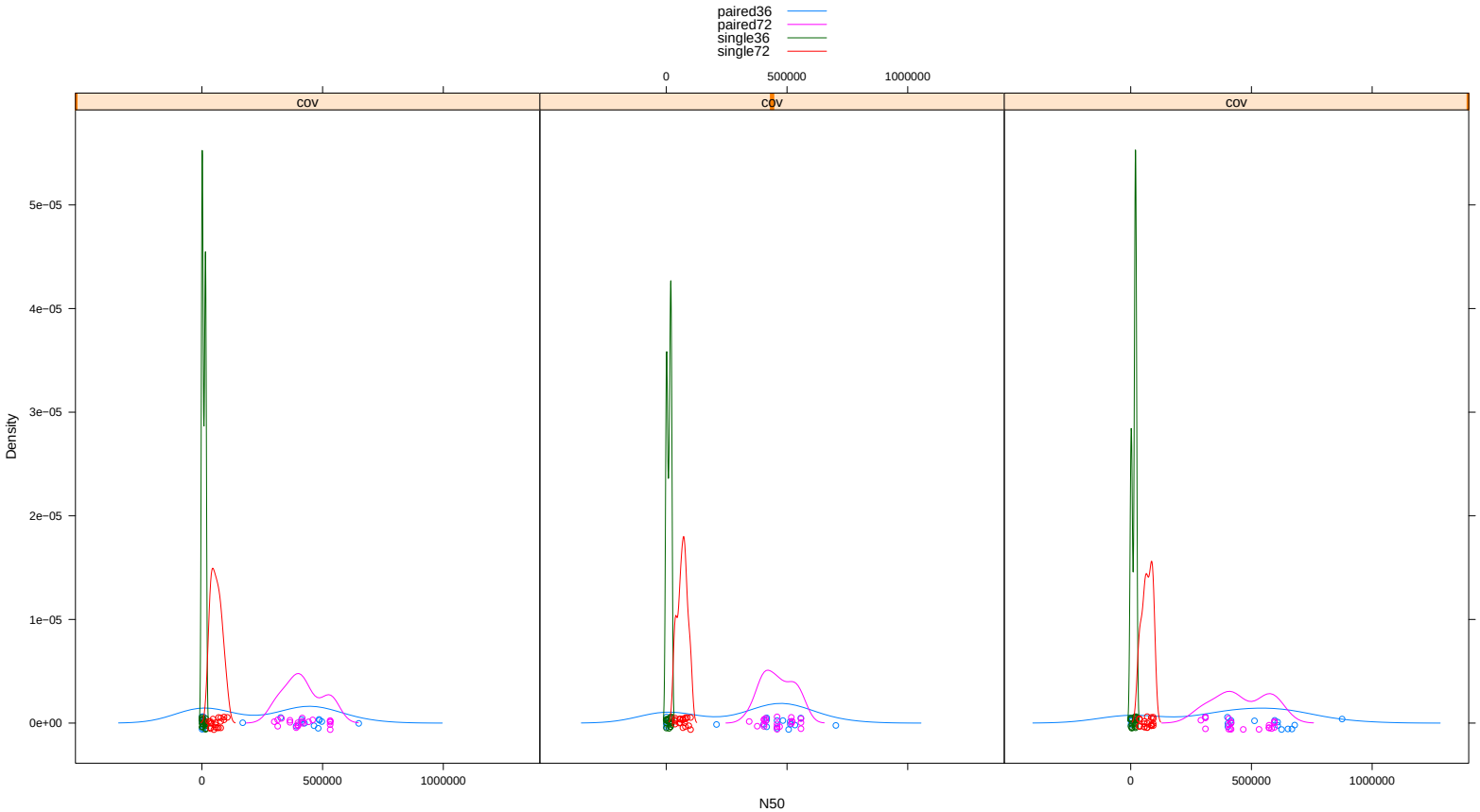
Una matriz de datos gigante



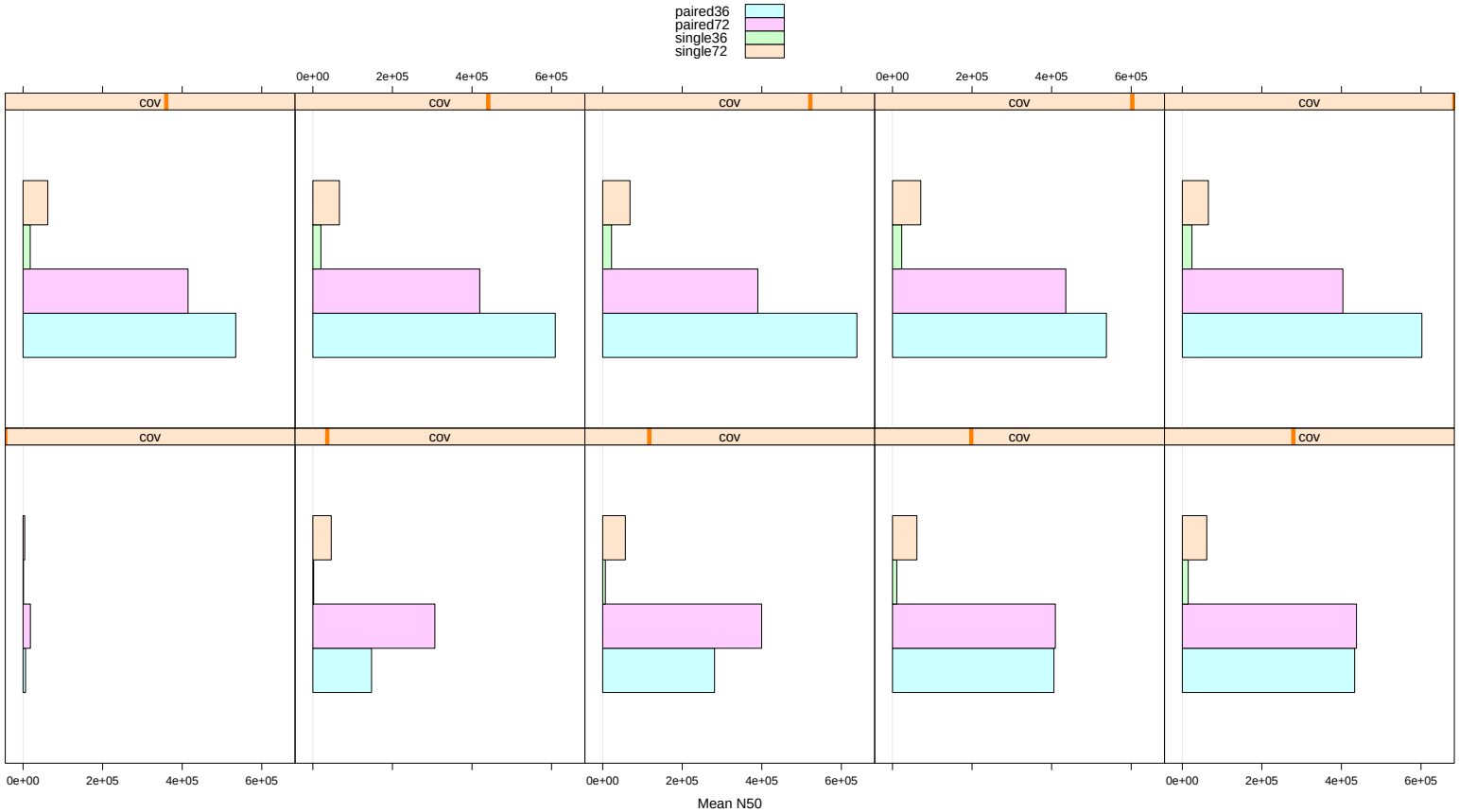
Una gráfica con lattice



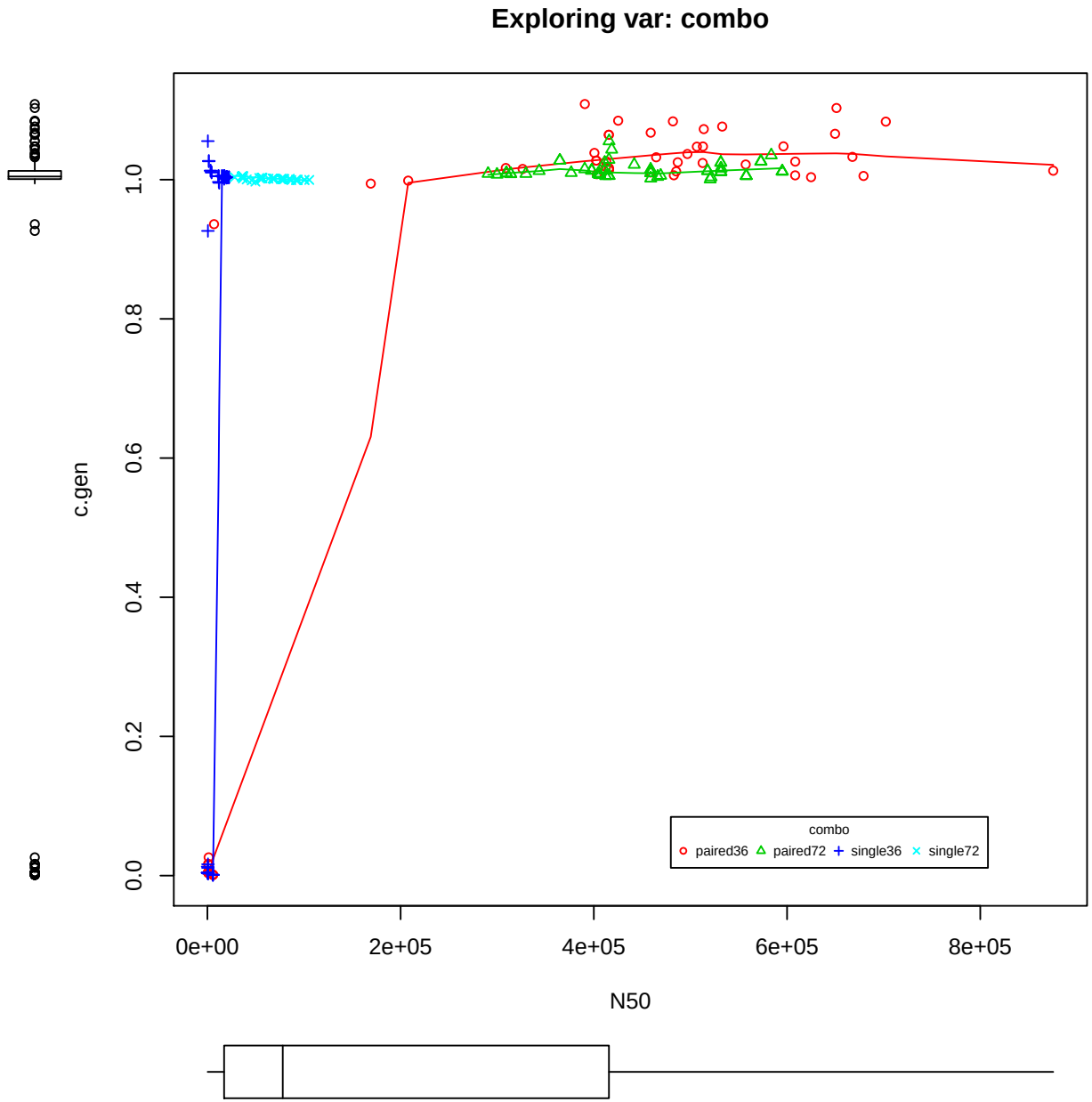
Otra gráfica con lattice



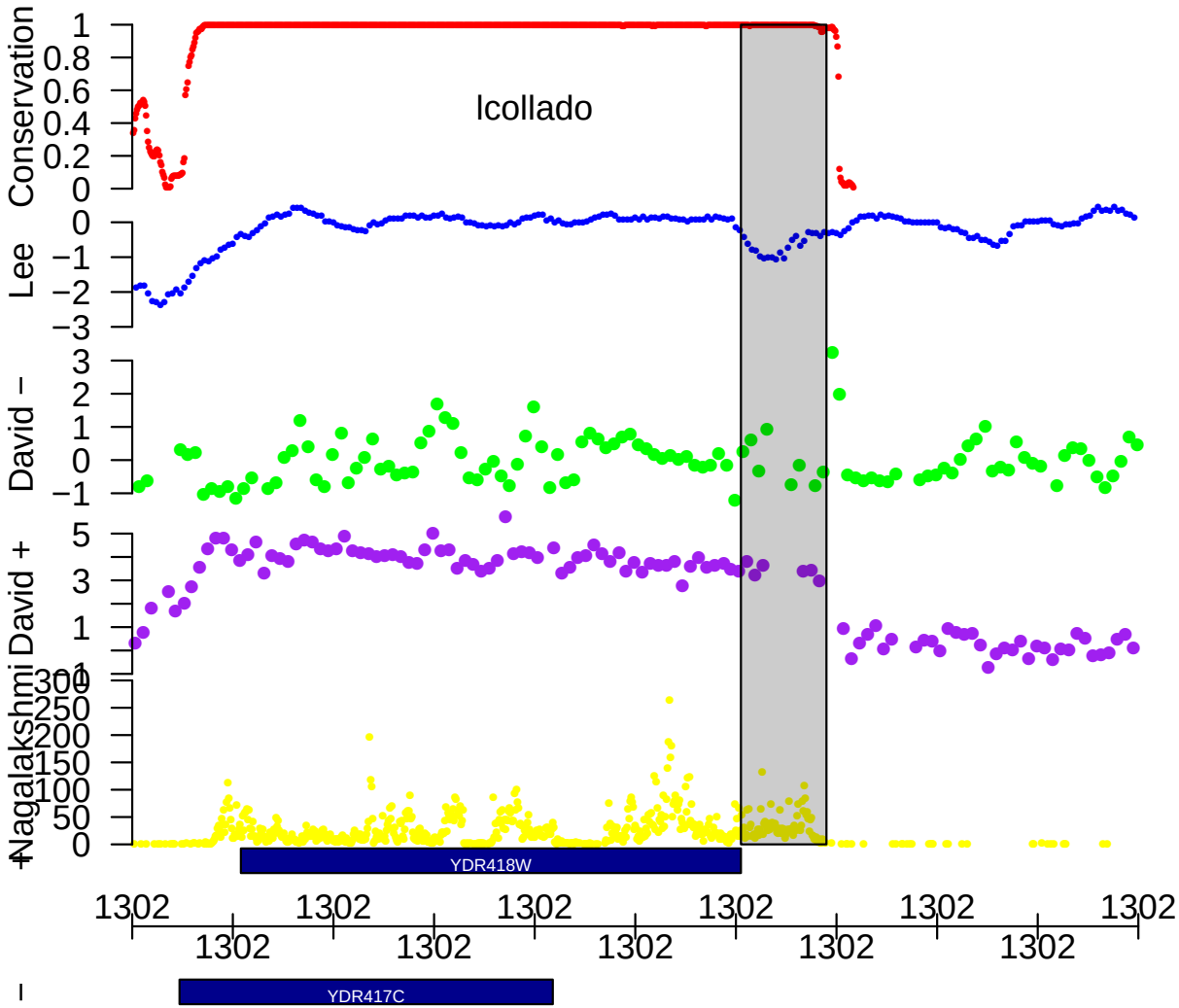
Barras con lattice



Una gráfica con plotrix



Una gráfica con GenomeGraphs



Ejercicios I

Usen los números 2, 5, 4, 10 y 8 para:

- Almacenarlos en un vector de datos x
- Encuentren el cuadrado de cada número.
- Substraigan 3 de cada número.
- Substraigan 5 de cada número y luego encuentren su raíz.

Ejercicios II

Encuentren:

- Las fracciones de 1/1 hasta 1/10 usando enteros. Usen los dos puntos :)
- Los años pares desde 1964 hasta 2008.
- Los múltiplos de 25 desde 1000 hasta 0 en ese orden.

Ejercicios III

Conocemos el tamaño de los genomas de 10 bacteríofagos. Sus tamaños en mbs son: `233.2 180.5 280.3 244.8 252.4 178.2 211.2 196.2 176.8 185.7` Almacenen esta información en un vector y encuentren:

- La suma total de los genomas usando un ciclo for.
- Repitan el paso anterior usando la función sum.
- El tamaño promedio de los 10 genomas.
- Repitan el paso anterior usando la función mean.

Pronto veremos

- Más sobre los tipos de objetos en R
- Aprenderemos a leer datos de tablas en R
- Veremos más gráficas básicas

sessionInfo

Información de mi sesión:

```
> sessionInfo()

R version 2.10.0 Under development (unstable) (2009-07-21 r48968)
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
```