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

Graficando con R

Ejercicios I

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

- Almacenarlos en un vector de datos x

```
> x <- c(2, 5, 4, 10, 8)
```

- Encuentren el cuadrado de cada número.

```
> x^2
```

```
[1]   4  25  16 100  64
```

- Substraigan 3 de cada número.

```
> x - 3
```

```
[1] -1  2  1  7  5
```

- Substraigan 5 de cada número y luego encuentren su raíz.

```
> sqrt(x - 5)
```

```
[1]       NaN 0.000000       NaN 2.236068
[5] 1.732051
```

- ¿Qué es un NaN?

Ejercicios II

Encuentren:

- Las fracciones de 1/1 hasta 1/10 usando enteros. Usen los dos puntos :)

```
> 1/1:10
```

```
 [1] 1.0000000 0.5000000 0.3333333
 [4] 0.2500000 0.2000000 0.1666667
 [7] 0.1428571 0.1250000 0.1111111
[10] 0.1000000
```

- Los años pares desde 1964 hasta 2008.

```
> seq(1964, 2008, by = 2)
```

```
 [1] 1964 1966 1968 1970 1972 1974 1976
 [8] 1978 1980 1982 1984 1986 1988 1990
[15] 1992 1994 1996 1998 2000 2002 2004
[22] 2006 2008
```

Ejercicios II

- Los múltiplos de 25 desde 1000 hasta 0 en ese orden.

```
> rev(seq(0, 1000, by = 25))
```

```
 [1] 1000  975  950  925  900  875  850
 [8]  825  800  775  750  725  700  675
[15]  650  625  600  575  550  525  500
[22]  475  450  425  400  375  350  325
[29]  300  275  250  225  200  175  150
[36]  125  100   75   50   25    0
```

- ¿Qué hace la función rev?

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.

```
> fagos <- c(33.2, 180.5, 280.3,
+     244.8, 252.4, 178.2, 211.2,
+     196.2, 176.8, 185.7)
> suma <- 0
> for (i in 1:length(fagos)) {
+     suma <- suma + fagos[i]
+ }
> suma
```

```
[1] 1939.3
```

Ejercicios III

- Uso la función length en vez de poner 10 para no equivocarme, lo cual puede pasar cuando tienes muchos datos.
- Repitan el paso anterior usando la función sum.

```
> sum(fagos)
```

```
[1] 1939.3
```

- El tamaño promedio de los 10 genomas.

```
> sum(fagos)/length(fagos)
```

```
[1] 193.93
```

- Repitan el paso anterior usando la función mean.

```
> mean(fagos)
```

```
[1] 193.93
```

- ¿Cómo encontrarían la mediana? Acuérdense de usar apropos!!

Ejercicios III

- Respuesta:

```
> apropos("median")
[1] "median"          "median.default"
> median(fagos)
[1] 190.95
```

- Alternativamente[1]:

```
> a <- sort(fagos)
> mean(a[5:6])
[1] 190.95
```

[1] Aunque no se las recomiendo.

De varios tipos

- Hasta ahorita hemos creado vectores númericos, y usamos todos sus valores.
- Sin embargo hay varios tipos:
  1. De enteros: 1, 2, 5, 10
  2. Números: 1.2, 2.5, 5.3
  3. Lógicos: `FALSE, TRUE, FALSE`
  4. De caracteres: *hola, leo, jaja*
- A diferencia de otros lenguajes, en R[2] no hay que especificar formalmente de que tipo es.

[2] Generalmente.

Algunos ejemplos

- ¿Qué tipo de vector es x, y, z, w y k?

```
> x <- c(3.5, 8, 10)
> y <- c(100:1, -2)
> z <- c(TRUE, FALSE, NA)
> w <- c("TRUE", "FALSE")
> k <- c("hola", 5.6, TRUE)
```

- Si luego tienen dudas, pueden usar la función class

```
> class(w)

[1] "character"
```

Nombres

- Muchas veces queremos usar nombres para las posiciones dentro de un vector, en vez de números.
- Digamos que queremos hacer un vector (con los nombres) con los tamaños de genoma para E. coli MG1655 (4.6 mb), Geobacter WCH70 (3.54 mb) y Helicobacter pylori P12 (1.71mb).
- Hay que:
  1. Crear un vector con los datos
  2. Al vector de nombres asociados asignarle los nombres en el mismo orden. Para eso usamos la función names:

```
> bacterias <- c(4600000, 3540000,
+     1710000)
> bacterias
[1] 4600000 3540000 1710000
```

Nombres

```
> names(bacterias) <- c("E. coli MG1655",
+     "Geobacter WCH70", "Helicobacter pylori P12
> bacterias
```

```
          E. coli MG1655
                 4600000
         Geobacter WCH70
                 3540000
Helicobacter pylori P12
                 1710000
```

- ¿Qué creen que pasa si le asignamos menos nombres que el número de elementos del vector (bacterias tiene 3)?

```
> names(bacterias) <- c("E. coli MG1655",
+     "Geobacter WCH70")
> bacterias
```

Nombres

```
E. coli MG1655 Geobacter WCH70
       4600000          3540000
          <NA>
       1710000
```

- ¿Qué es una NA?

Accesando a posiciones específicas de un vector

- No siempre que tenemos un vector queremos usar todos los datos dentro de él. Por eso necesitamos saber accesar posiciones específicas.
- Hay varias formas de hacerlo:
  1. Por medio de índices (números). Recuerden que en R todo empieza en 1 y no en 0.
  2. Por nombres, si los hay.
  3. Por vectores lógicos.
- La más sencilla es por índices. Para eso usamos los corchetes, por ejemplo, para la posición 7:

```
> x <- 10:1
> x[7]
```

```
[1] 4
```

Accesando a posiciones específicas de un vector

- Voy a asignarle como nombres las primeras 10 letras al vector x.

```
> names(x) <- letters[1:10]
```

- Ahora obtengamos el valor de x para la letra c. Noten que uso comillas dobles.

```
> x["c"]
```

```
c
8
```

- Otra opción para usar los nombres es con una variable:

```
> y <- c("a", "d")
> x[y]
```

```
 a  d
10  7
```

Accesando a posiciones específicas de un vector

- Si queremos los valores de x tal que sean mayores a 5 usamos un vector lógico:

```
> x[x > 5]
```

```
 a  b  c  d  e
10  9  8  7  6
```

- La expresión $x > 5$ nos crea el vector lógico, por eso también podemos usar una variable:

```
> x > 5
```

```
    a     b     c     d     e     f
 TRUE  TRUE  TRUE  TRUE  TRUE FALSE
    g     h     i     j
FALSE FALSE FALSE FALSE
```

Accesando a posiciones específicas de un vector

```
> z <- x > 3 & x < 8
> x[z]
d e f g
7 6 5 4
```

Pequeños ejercicios

Revisemos el uso de vectores:

- Creen un vector *a* con los números enteros desde 0 hasta -25.
- Al vector de nombres asociado al objeto *a*, asignele las letras de la 1 a la 26.[3].
- ¿Cuáles son los valores en las posiciones 5, 10 y 25?
- ¿Cuáles son los valores para las letras z, f y l?
- ¿Cuáles son los valores para las posiciones pares? Usen un vector lógico *rep*etido.

[3] En realidad deberían ser 26, R usa el alfabeto inglés

Solución:

- Para ahorrar líneas de código no uso variables intermedias, aunque es válido usarlas :)
- Noten como uso la función rep para recuperar las posiciones pares.

```
> a <- 0:-25
> names(a) <- letters[1:26]
> a[c(5, 10, 25)]
```

```
  e   j   y
 -4  -9 -24
```

```
> a[c("z", "f", "l")]
```

```
  z   f   l
-25  -5 -11
```

```
> a[rep(c(FALSE, TRUE), 13)]
```

Solución:

```
 b  d  f  h  j  l  n  p  r  t
-1 -3 -5 -7 -9 -11 -13 -15 -17 -19
 v  x  z
-21 -23 -25
```

- ¿Cómo le harían para recuperar todas las posiciones menos la 5, 10 y 25?

Vector de índices negativo

- Una forma sería creando un vector de índices excluyendo esas posiciones:

```
> pos <- c(1:4, 6:9, 11:24, 26)
> a[pos]
```

```
 a  b  c  d  f  g  h  i  k  l
 0 -1 -2 -3 -5 -6 -7 -8 -10 -11
 m  n  o  p  q  r  s  t  u  v
-12 -13 -14 -15 -16 -17 -18 -19 -20 -21
 w  x  z
-22 -23 -25
```

- Sin embargo, la forma más sencilla es con el signo menos, osea un vector de índices negativo:

```
> -c(5, 10, 25)
```

```
[1]  -5 -10 -25
```

Vector de índices negativo

```
> a[-c(5, 10, 25)]
  a   b   c   d   f   g   h   i   k   l
  0  -1  -2  -3  -5  -6  -7  -8 -10 -11
  m   n   o   p   q   r   s   t   u   v
-12 -13 -14 -15 -16 -17 -18 -19 -20 -21
  w   x   z
-22 -23 -25
```

Plot despacio

Veamos más despacio y detallado como usar plot

- Generalmente tendrán un vector de datos numérico. Por ejemplo, el caso de los 10 fagos más grandes.

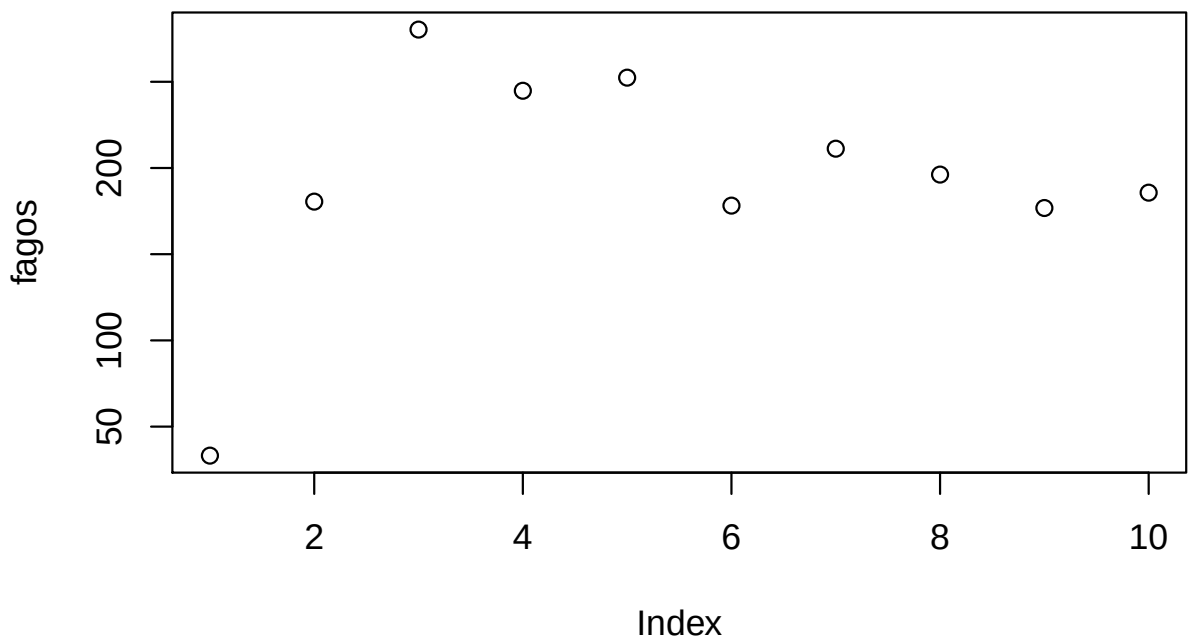
```
> fagos
```

```
 [1]  33.2 180.5 280.3 244.8 252.4 178.2
[7] 211.2 196.2 176.8 185.7
```

- La gráfica más básica seria pasar ese vector a la función plot.

```
> plot(fagos)
```

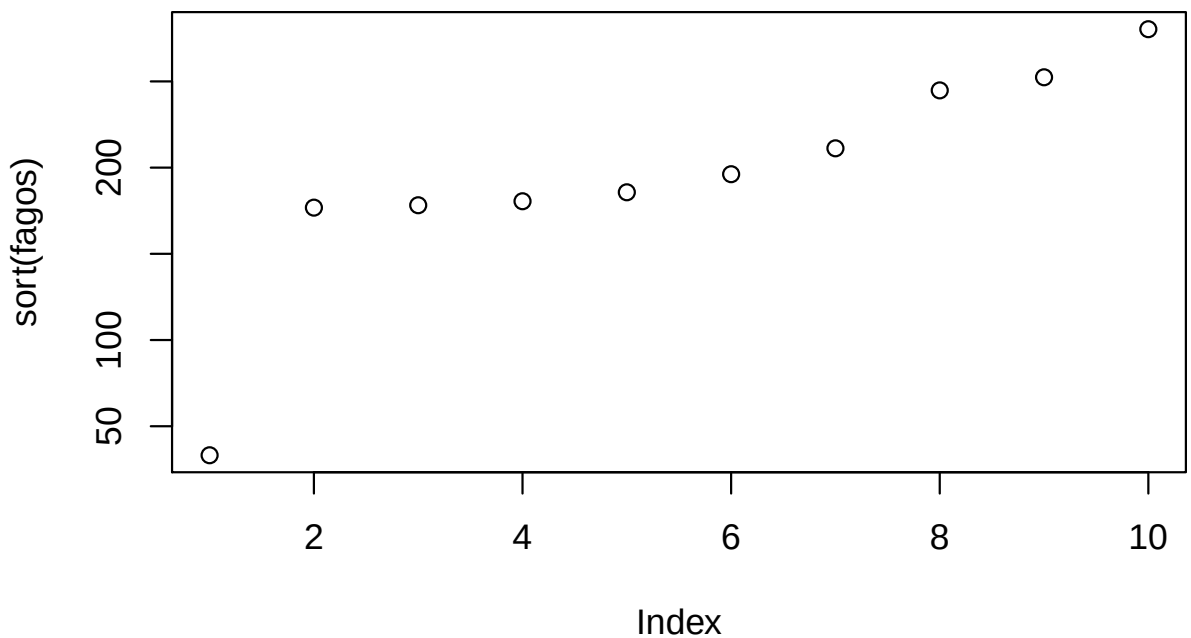
Plot despacio



Index

Si se fijan, nuestro vector está en el eje Y y los grafica en el orden que están en el vector usando el index.
Si los ordenamos, se nota inmediatamente la diferencia:

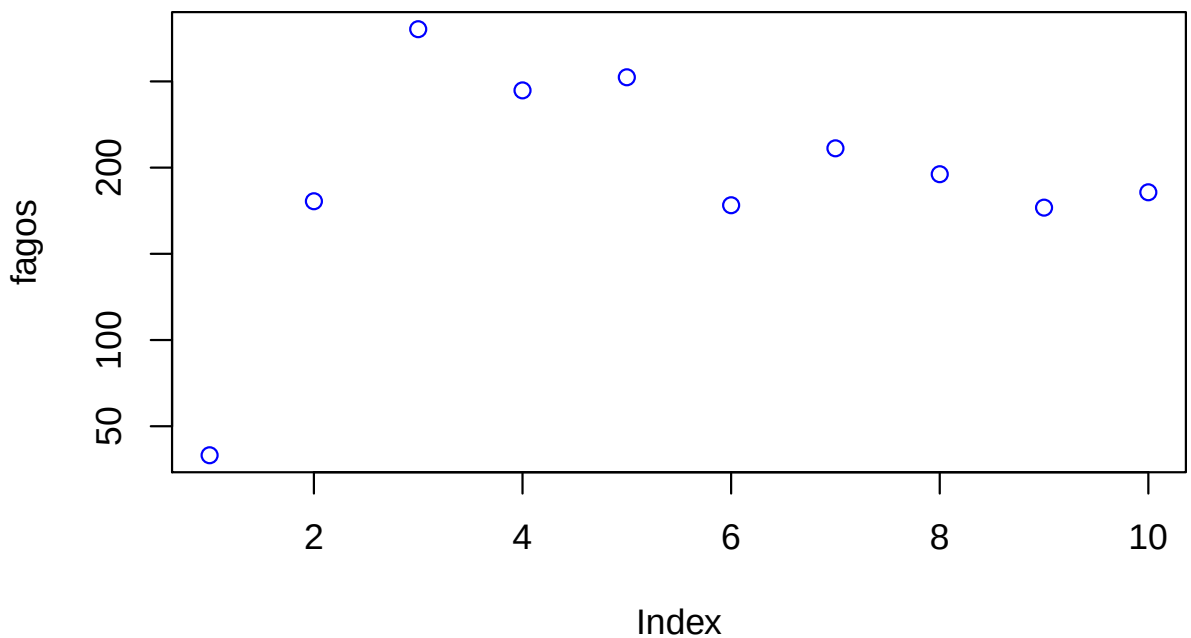
```
> plot(sort(fagos))
```



Col

Automáticamente pone en el eje *Y* el *nombre* del vector, pone los puntos en color negro como bolitas vacías. Pongamoslas en azul usando col:)

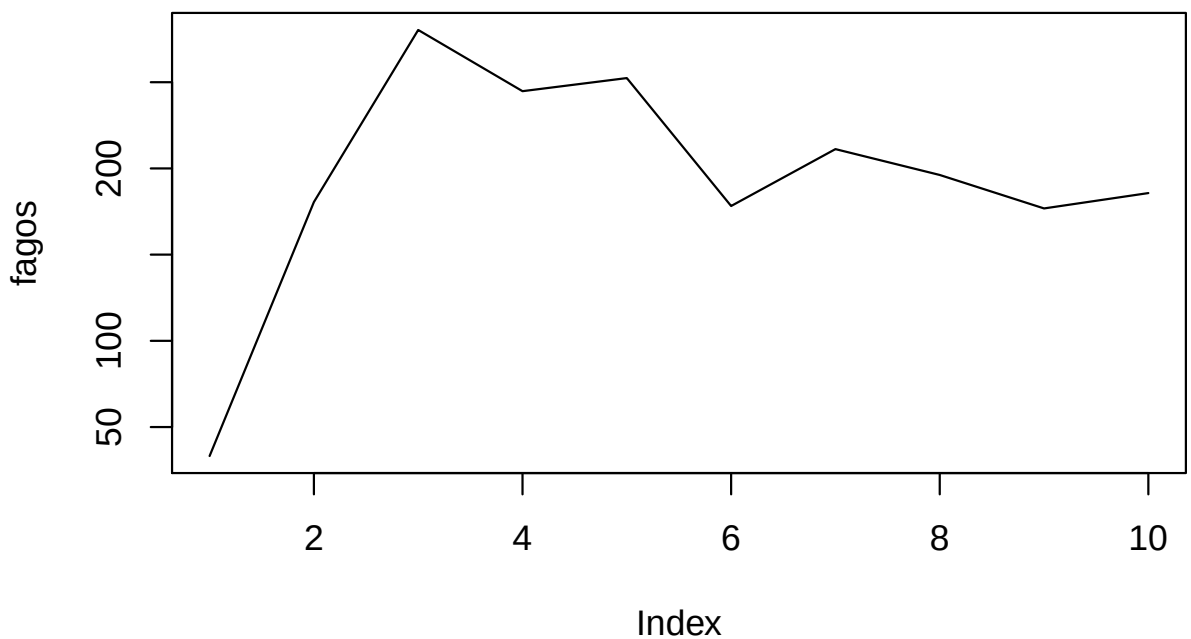
```
> plot(fagos, col = "blue")
```



Type I

A veces preferimos ver una línea que los puntos, para eso usamos el argumento type:

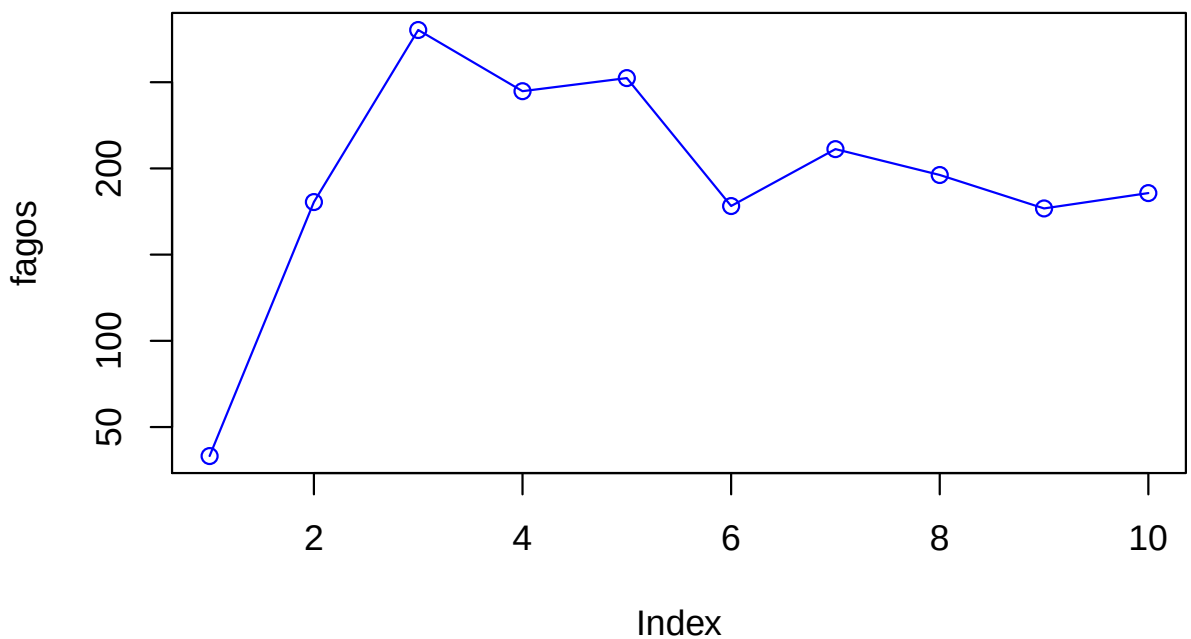
```
> plot(fagos, type = "l")
```



Type o

O mejor aún, una línea con los puntos como bolitas, todo en color azul:

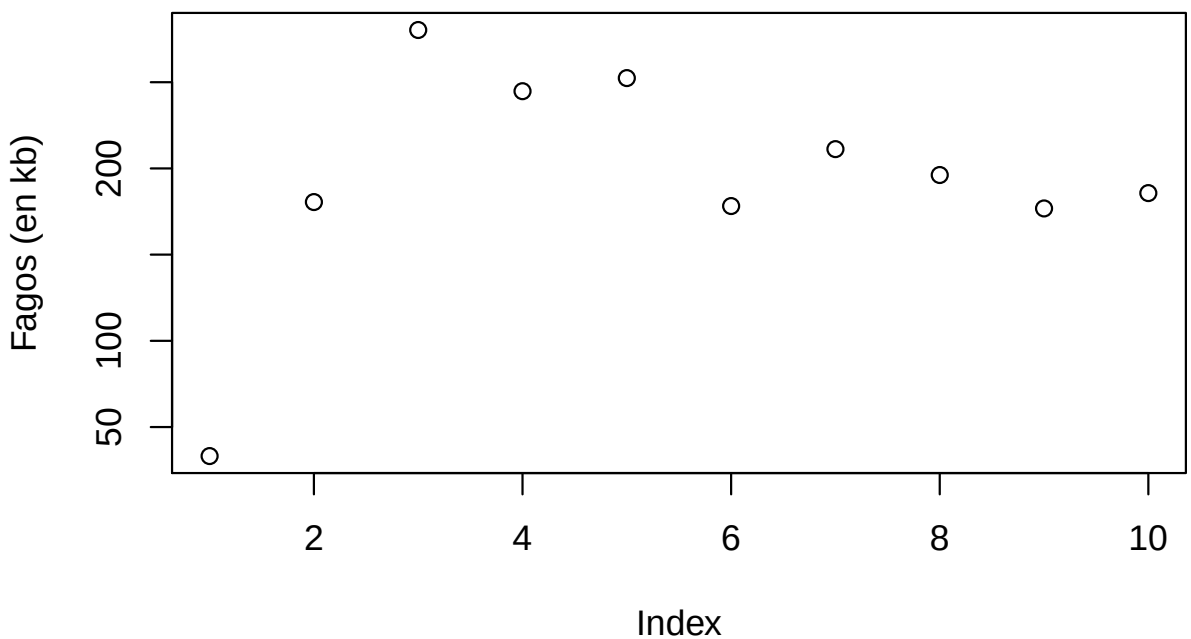
```
> plot(fagos, type = "o", col = "blue")
```



Ylab

Sabemos que tenemos en `fagos` los tamaños de los 10 más grandes en kb. Usemos ylab para reflejar esto:

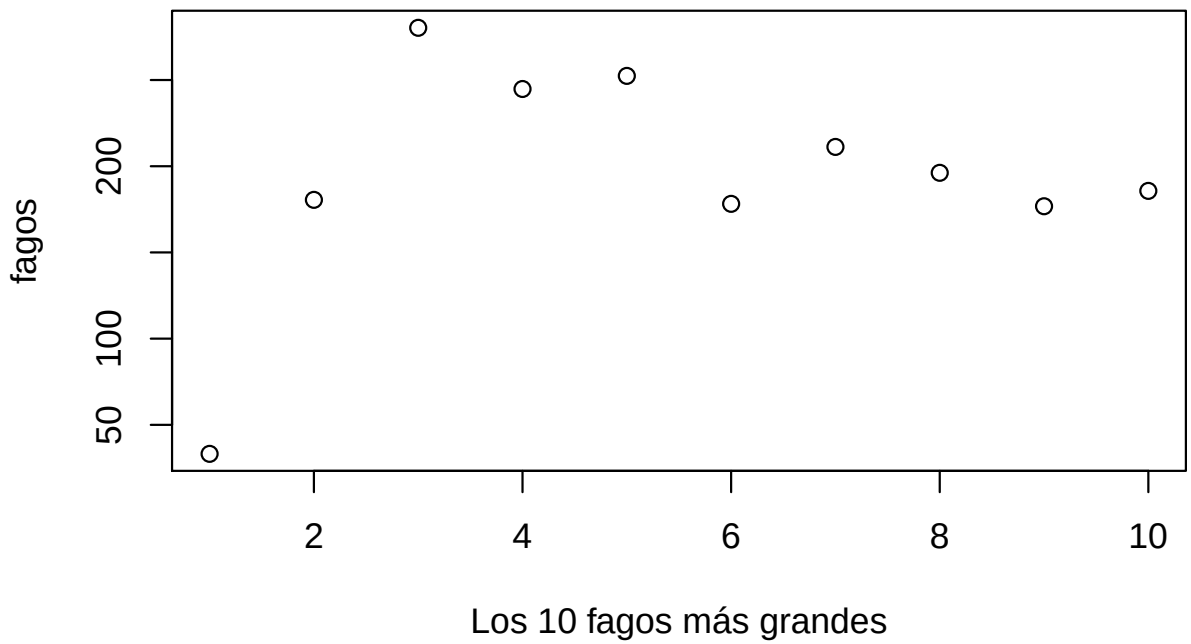
```
> plot(fagos, ylab = "Fagos (en kb)")
```



Xlab

Ya no queremos al Índex", así que usemos xlab:

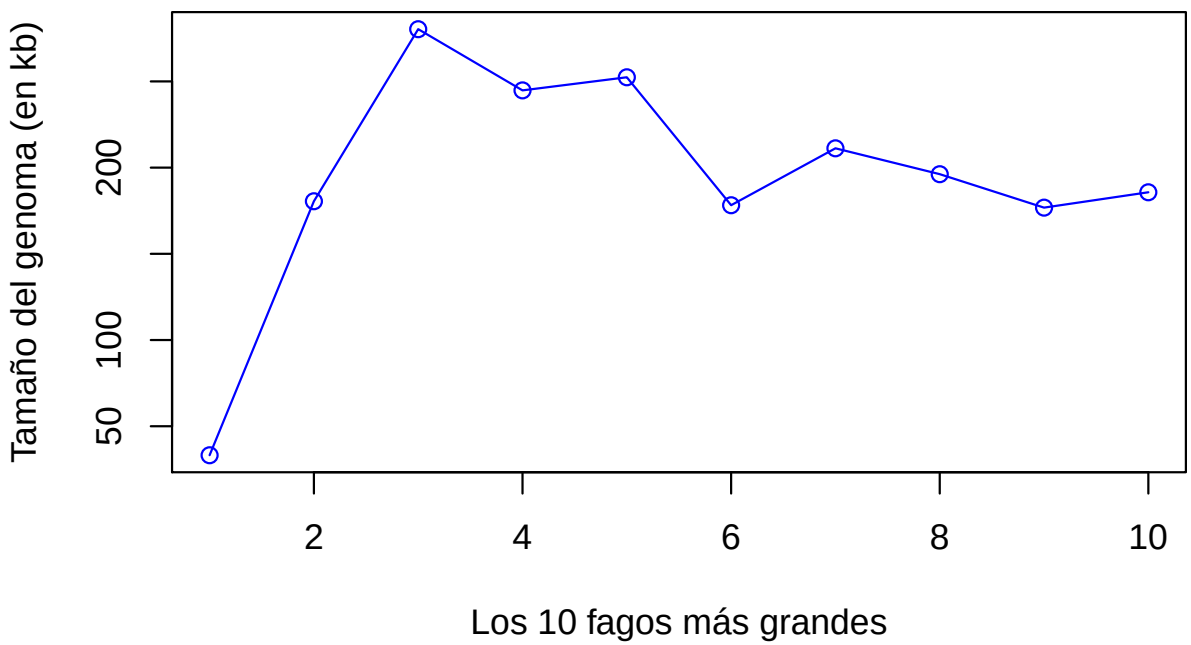
```
> plot(fagos, xlab = "Los 10 fagos más grandes")
```



Juntando

Si juntamos lo que ya vimos podemos hacer la siguiente gráfica:

```
> plot(fagos, col = "blue", type = "o",
+      ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes")
```

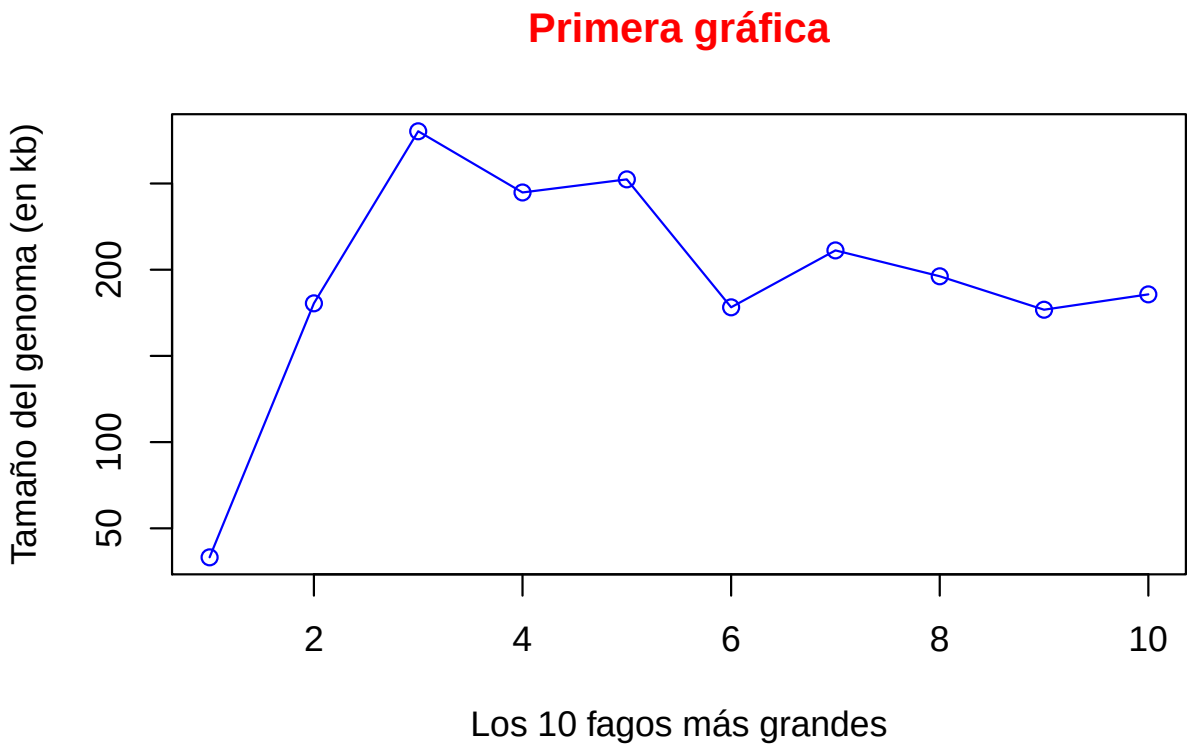


Título

Queremos agregarle un título en color rojo, por lo que usaremos main y col.main:

```
> plot(fagos, col = "blue", type = "o",
+      ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes",
+      main = "Primera gráfica", col.main = "red")
```

Título

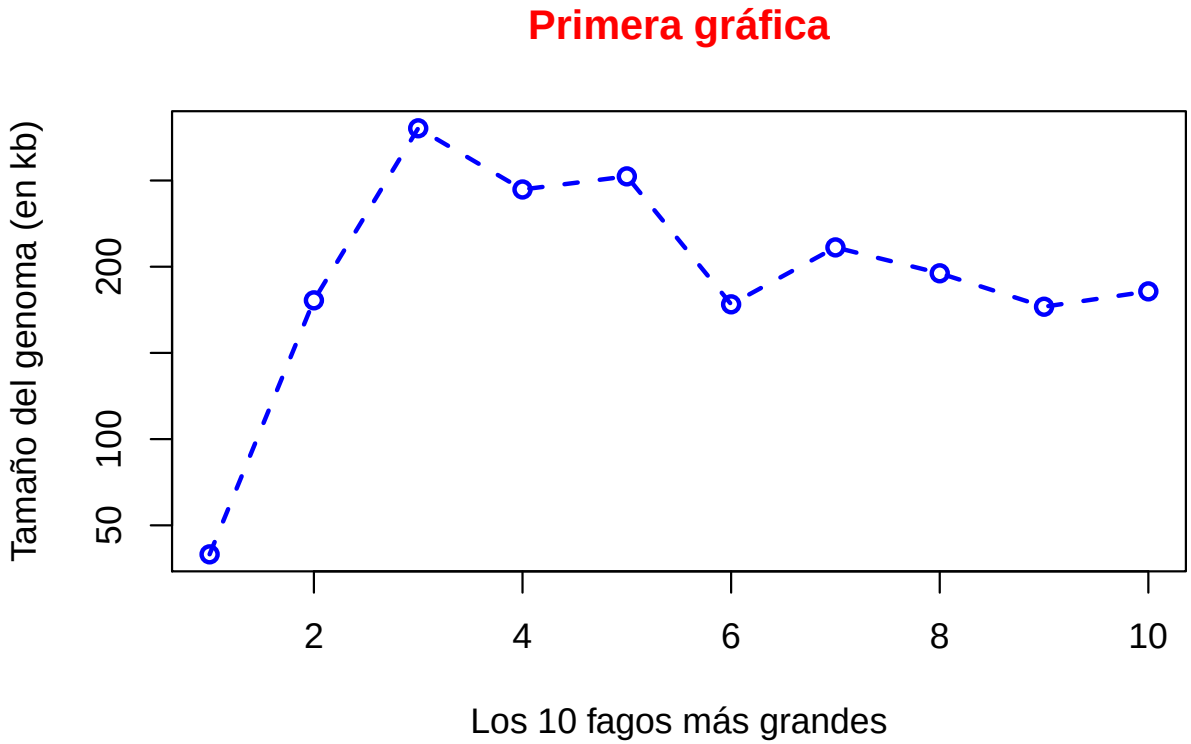

Primera gráfica
Tamaño del genoma (en kb)
50
100
200
2
4
6
8
10
Los 10 fagos más grandes

Lty y Lwd

Si queremos cambiar el tipo de línea tenemos que usar lty. Para el grosor de la línea es con lwd:

```
> plot(fagos, col = "blue", type = "o",
+      ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes",
+      main = "Primera gráfica", col.main = "red",
+      lty = 2, lwd = 2)
```

Lty y Lwd


Primera gráfica
Tamaño del genoma (en kb)
50
100
200
2
4
6
8
10
Los 10 fagos más grandes

2 líneas

- Ahora, a parte de `fagos` vamos a crear el objeto *fagos2* con la siguiente info:

  ```
  > fagos2 <- seq(25, 250, by = 25)
  ```

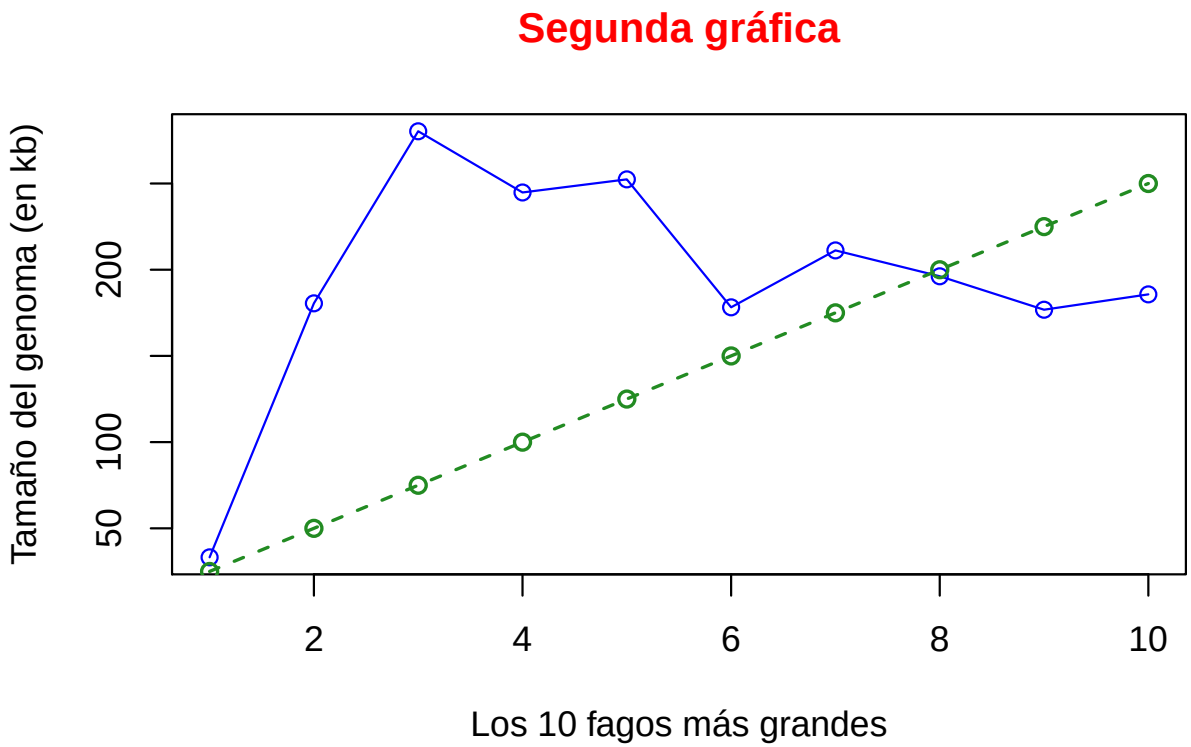
- Queremos crear una gráfica con las dos líneas. Para eso vamos a usar la función lines.
- plot es una función de alto nivel, es decir, crea un espacio gráfico cuando la usas. lines es de bajo nivel[4], por lo que hay que usar `plot` antes de `lines`.

[4] No crea un espacio gráfico

Lines

```
> plot(fagos, col = "blue", type = "o",
+      ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes",
+      main = "Segunda gráfica", col.main = "red")
> lines(fagos2, col = "forest green",
+      type = "o", lty = 2, lwd = 1.5)
```

Lines

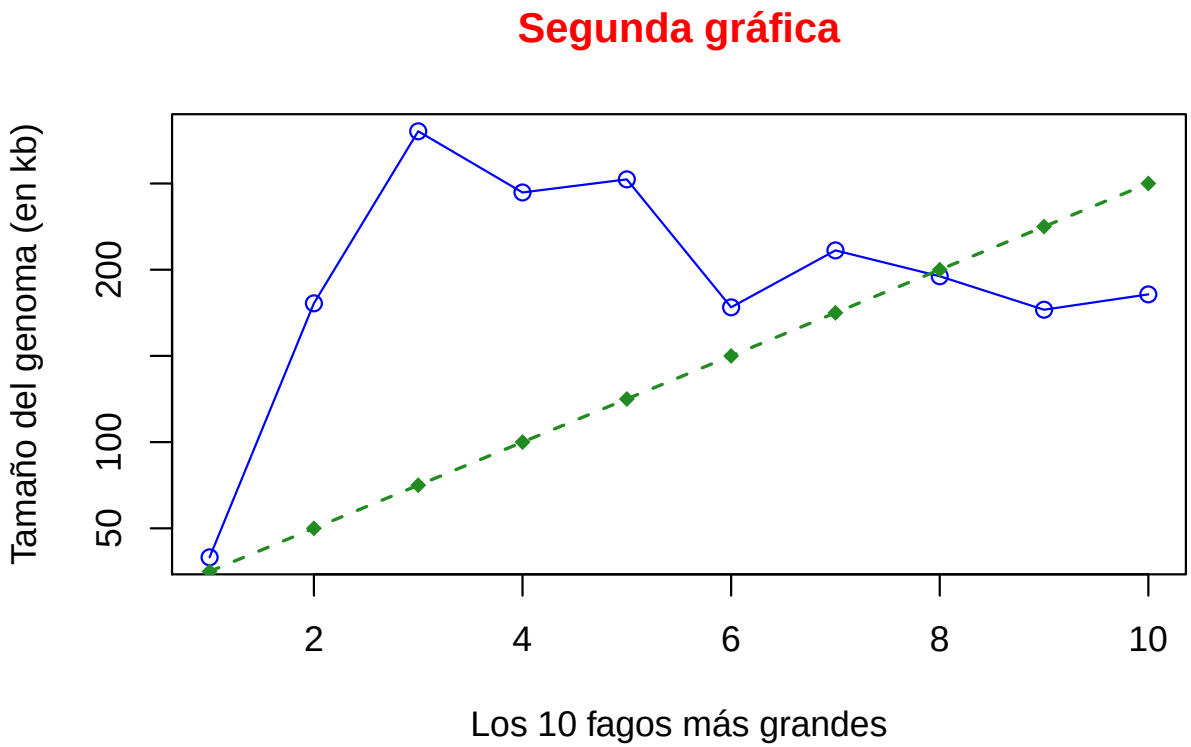

Segunda gráfica
Tamaño del genoma (en kb)
50
100
200
2
4
6
8
10
Los 10 fagos más grandes

Pch

Queremos que se diferencien aún más las dos líneas. Para eso usaremos el argumento pch:

```
> plot(fagos, col = "blue", type = "o",
+      ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes",
+      main = "Segunda gráfica", col.main = "red")
> lines(fagos2, col = "forest green",
+      type = "o", lty = 2, lwd = 1.5,
+      pch = 18)
```

Pch

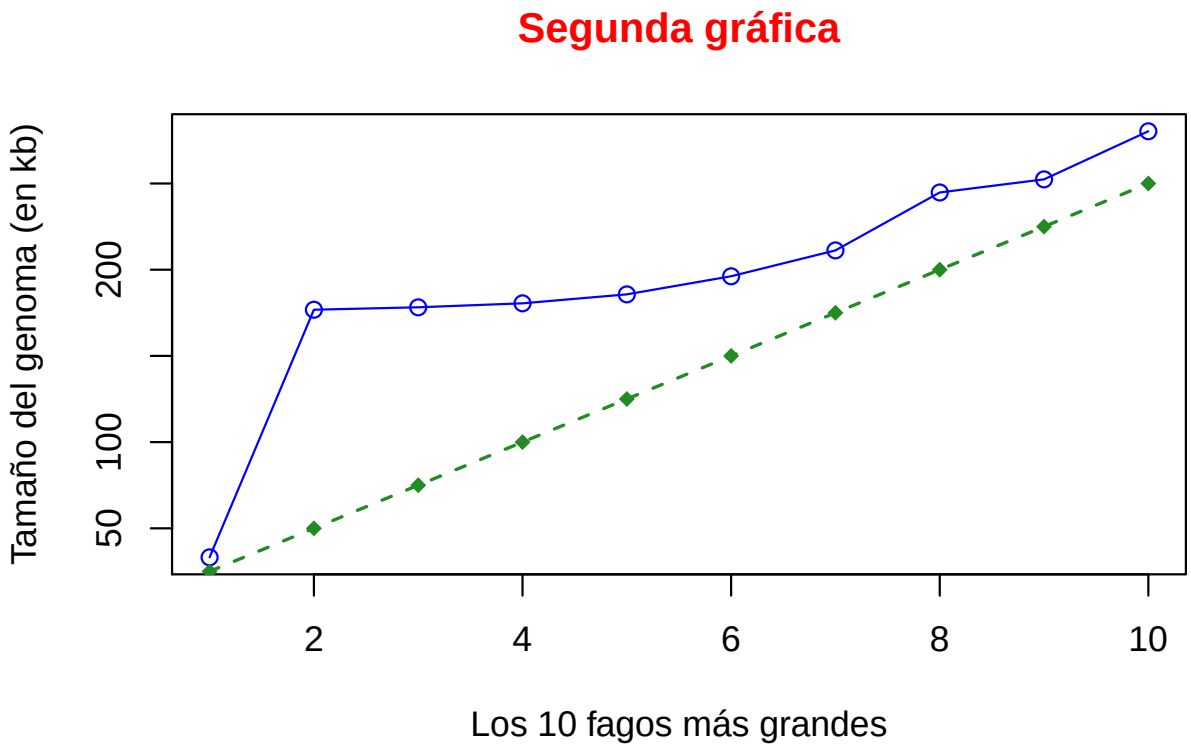

Segunda gráfica
Tamaño del genoma (en kb)
50
100
200
2
4
6
8
10
Los 10 fagos más grandes

Lineal?

Una diagonal, como en el caso de `fagos2`, puede ser útil para ver si nuestros datos tienen algún comportamiento lineal.

```
> plot(sort(fagos), col = "blue",
+      type = "o", ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes",
+      main = "Segunda gráfica", col.main = "red")
> lines(fagos2, col = "forest green",
+      type = "o", lty = 2, lwd = 1.5,
+      pch = 18)
```

Lineal?


Segunda gráfica
Tamaño del genoma (en kb)
50
100
200
2
4
6
8
10
Los 10 fagos más grandes

Visualizando valores de corte

- Digamos que en este caso estamos seleccionando fagos por tamaño y que nuestro valor de corte es 200 kb.
- Queremos que se vea esta línea en nuestra gráfica. Para eso hay que usar la función de bajo nivel abline:

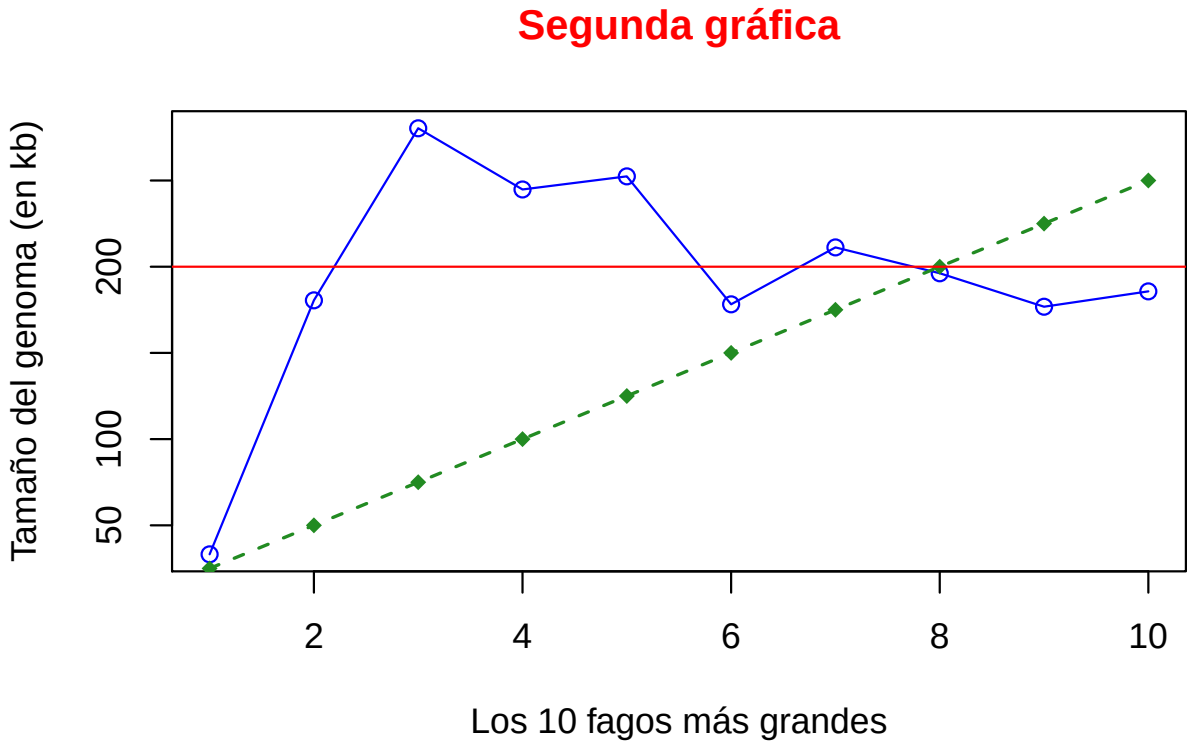
```
> args(abline)

function (a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,
    coef = NULL, untf = FALSE, ...)
NULL
```

Usando abline

```
> plot(fagos, col = "blue", type = "o",
+      ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes",
+      main = "Segunda gráfica", col.main = "red")
> lines(fagos2, col = "forest green",
+      type = "o", lty = 2, lwd = 1.5,
+      pch = 18)
> abline(a = 200, b = 0, col = "red")
```

Usando abline

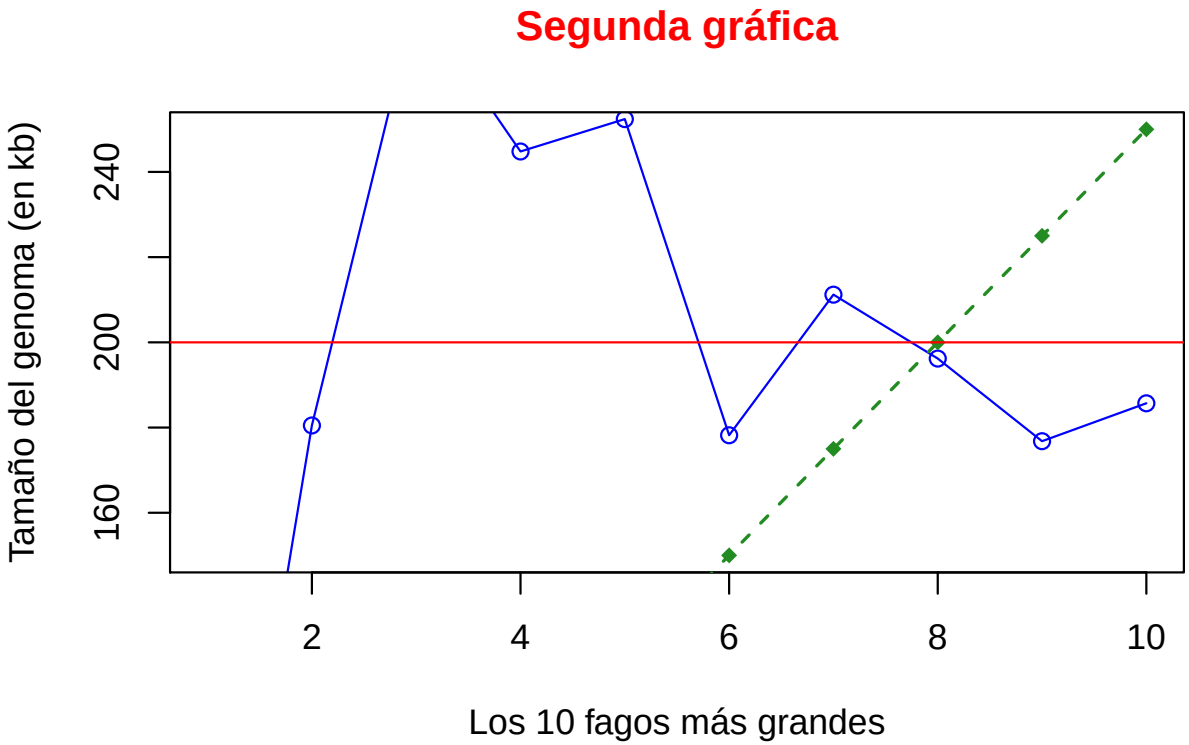


Ylim

¿Y si queremos hacer un acercamiento (zoom)? Usemos el argumento ylim[5] en la función `plot`:

```
> plot(fagos, col = "blue", type = "o",
+      ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes",
+      main = "Segunda gráfica", col.main = "red",
+      ylim = c(150, 250))
> lines(fagos2, col = "forest green",
+      type = "o", lty = 2, lwd = 1.5,
+      pch = 18)
> abline(a = 200, b = 0, col = "red")
```

Ylim



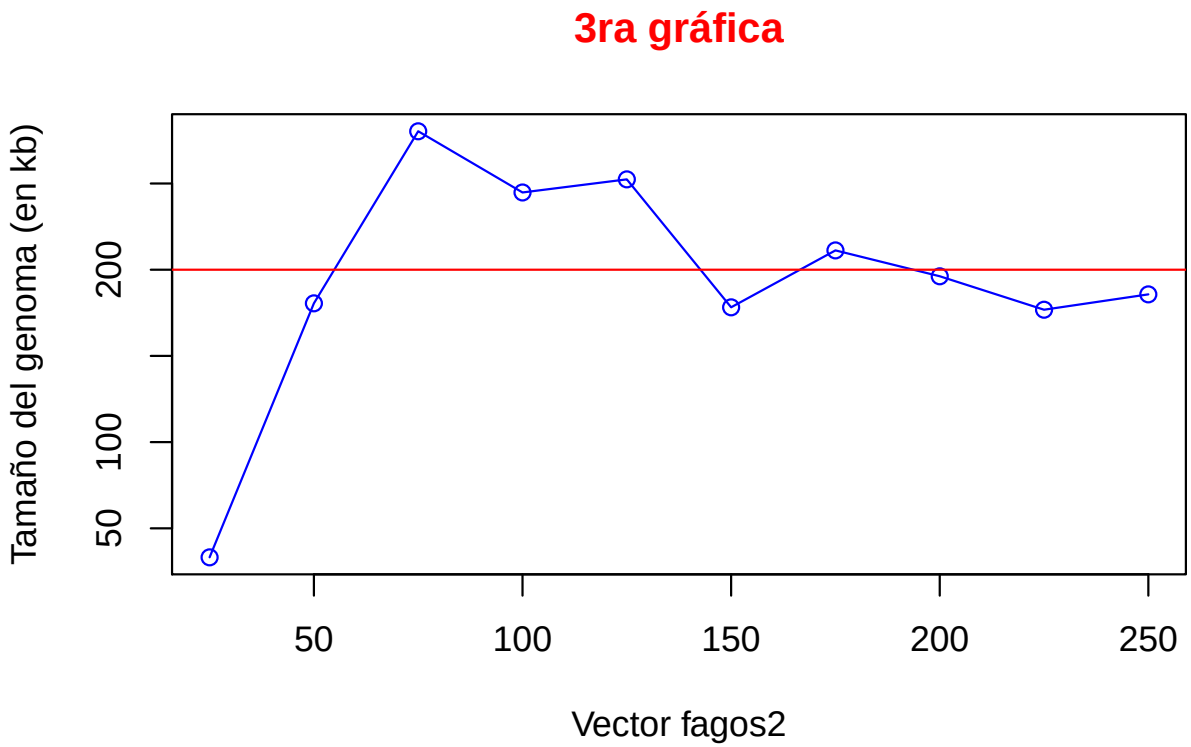
[5] Para el eje x sería xlim.

Uno vs el otro

Cuando tienen dos vectores, otra opción es usar uno para el eje x y otro para el eje y en ese orden.

```
> plot(fagos2, fagos, col = "blue",
+      type = "o", ylab = "Tamaño del genoma (en kb)",
+      xlab = "Vector fagos2", main = "3ra gráfica",
+      col.main = "red")
> abline(a = 200, b = 0, col = "red")
```

Uno vs el otro



titulo

- Muy seguido pocos datos, sobre todo del tipo categórico, y queremos visualizarlos.
- Muchas personas deciden usar gráficas de "pie", sin embargo, no son recomendables porque:

  ```
  > `?`(pie)
  ```

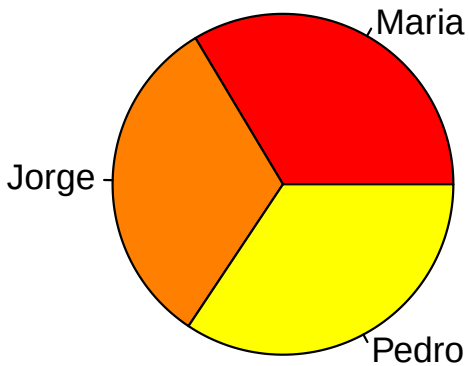
- ```
  Pie charts are a very bad way of displaying
  information. The eye is good at judging linear
  measures and bad at judging relative areas. A
  bar chart or dot chart is a preferable way of
  displaying this type of data
  ```
  ```

Un pie

Como a muchos les gustan, veamos rápido como hacer una:

```
> pub <- c(44, 42, 45)
> names(pub) <- c("Maria", "Jorge",
+     "Pedro")
> pie(pub, col = heat.colors(3))
```

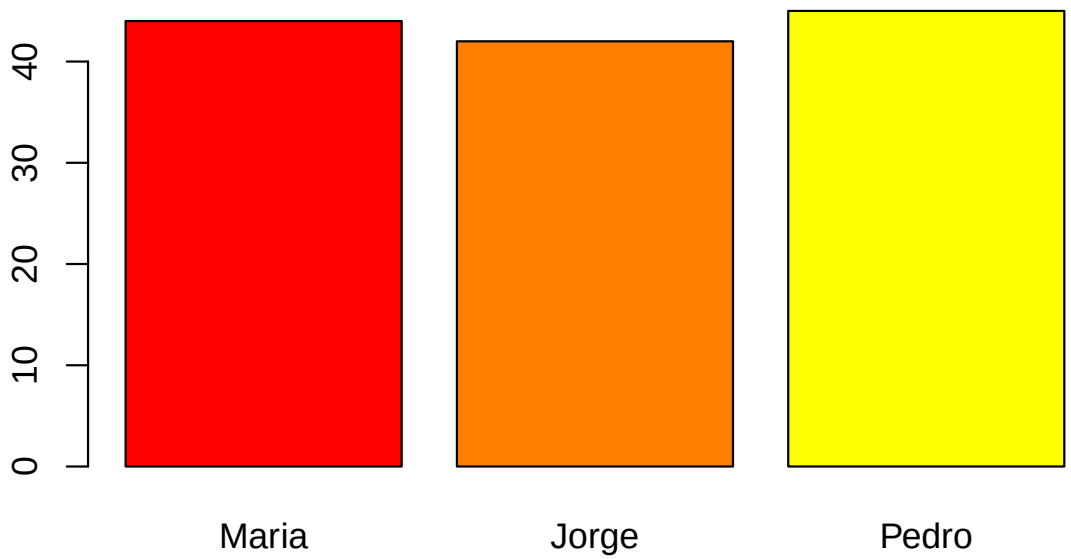
Un pie



Barras

Lo mejor es hacer una gráfica de barras.
Noten que uso la función heat.colors para obtener los 3 colores.

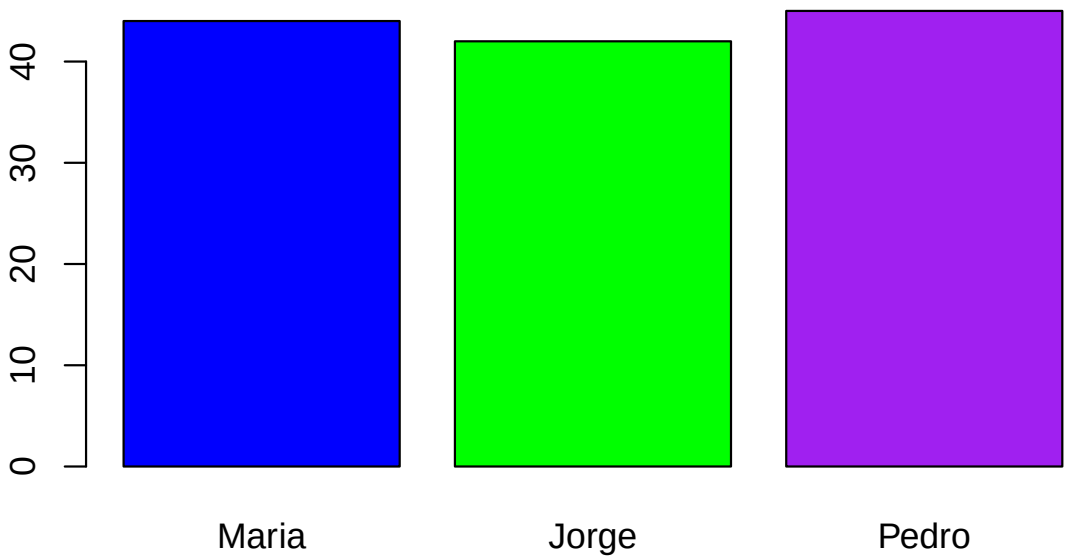
```
> barplot(pub, col = heat.colors(3))
```



Otros colores

En vez de `heat.colors` puedo darle un vector de colores.

```
> barplot(pub, col = c("blue", "green",
+     "purple"))
```

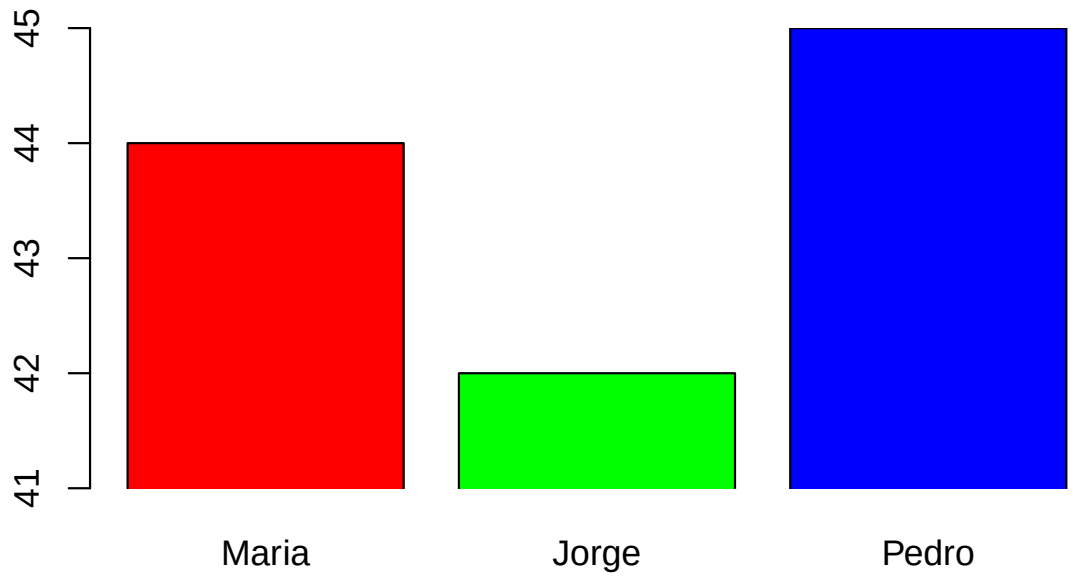


Con *zoom*

¿Cuál es la mejor gráfica?

Ahora uso rainbow en vez de `heat.colors`.

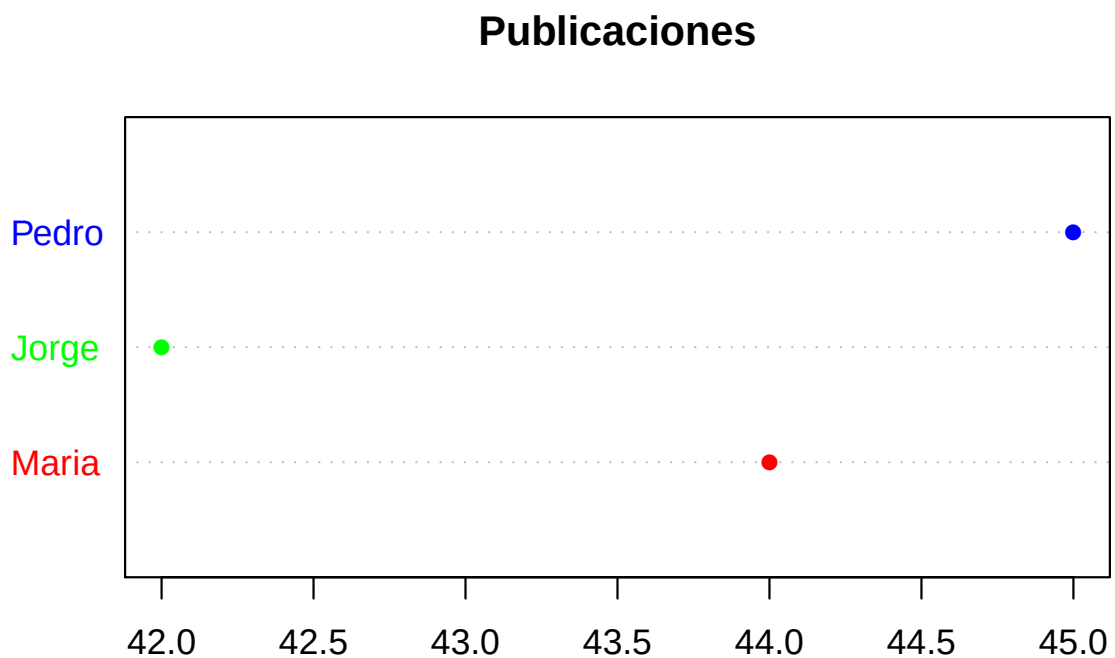
```
> barplot(pub, col = rainbow(3),
+         ylim = c(41, 45), xpd = FALSE)
```



Dotchart

Otra opción es hacer una *dotchart*.
Cuando tienes pocos datos son útiles:

```
> dotchart(pub, main = "Publicaciones",
+          col = rainbow(3), pch = 16)
```

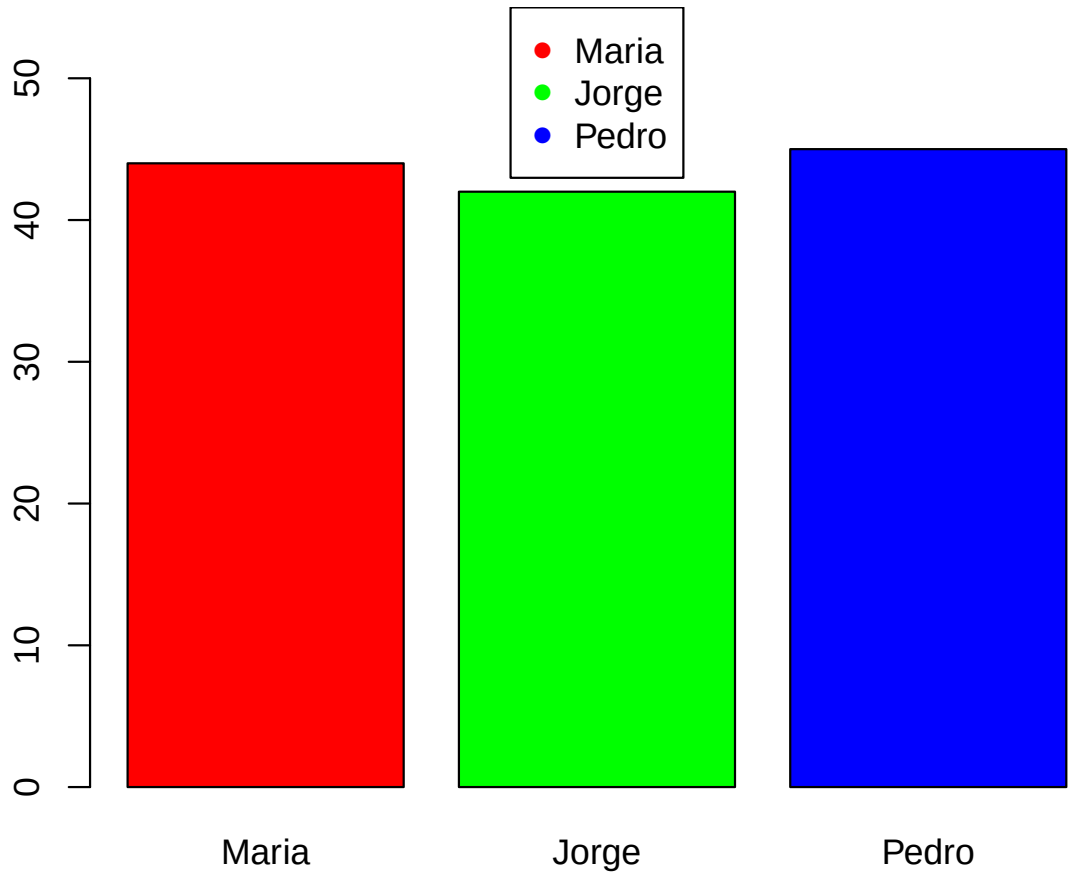


Legend

Muchas veces queremos agregar la *legend* de los datos. Para eso usamos la función legend:

```
> barplot(pub, col = rainbow(3),
+     ylim = c(0, 55))
> legend("top", names(pub), col = rainbow(3),
+     pch = 16)
```

Legend

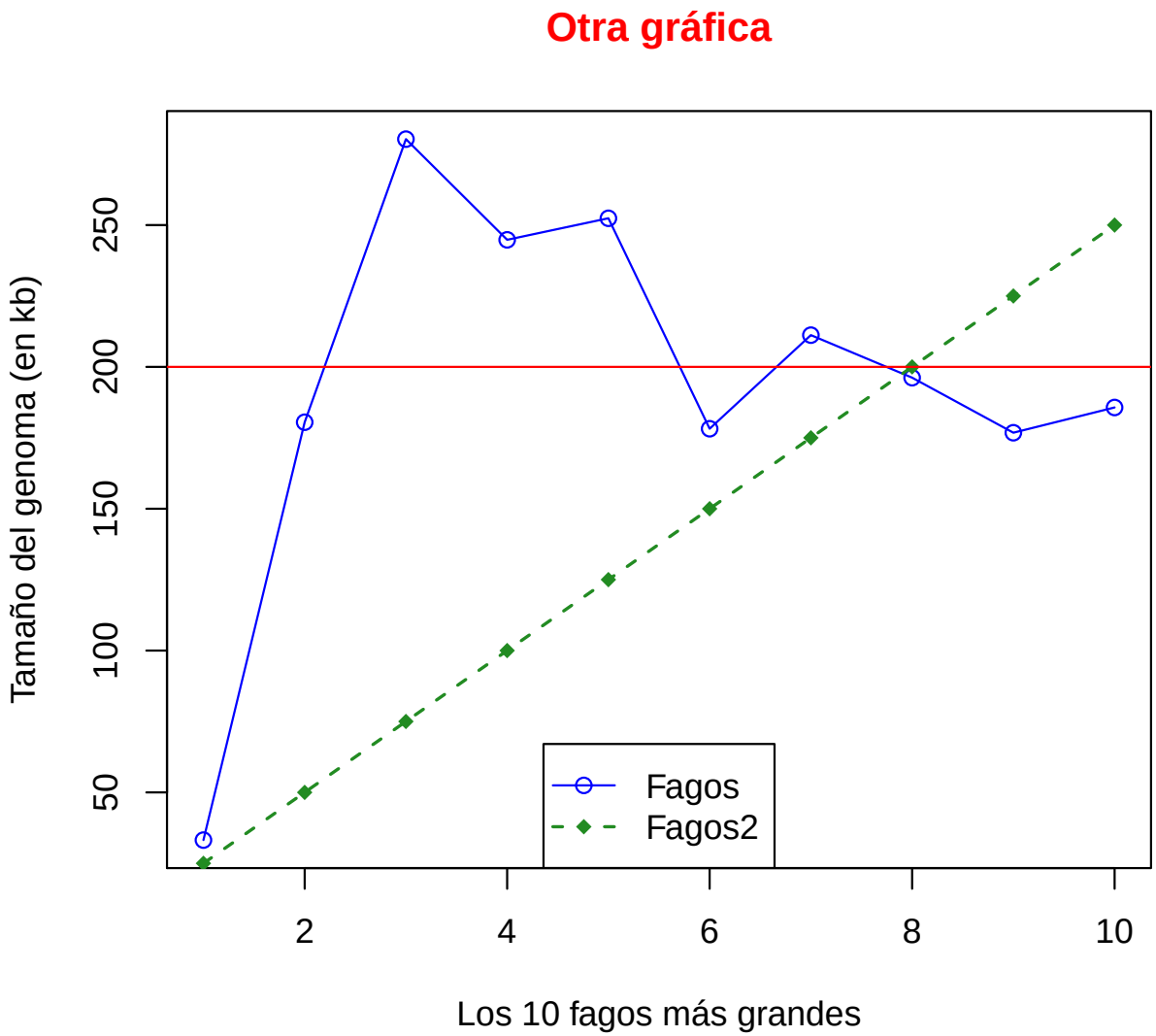


Otra legend

En el caso de nuestra gráfica de líneas, hay que especificar más argumentos para que salga bien la *legend*.

```
> plot(fagos, col = "blue", type = "o",
+      ylab = "Tamaño del genoma (en kb)",
+      xlab = "Los 10 fagos más grandes",
+      main = "Otra gráfica", col.main = "red")
> lines(fagos2, col = "forest green",
+      type = "o", lty = 2, lwd = 1.5,
+      pch = 18)
> abline(a = 200, b = 0, col = "red")
> legend("bottom", c("Fagos", "Fagos2"),
+      col = c("blue", "forest green"),
+      lty = c(1, 2), pch = c(1, 18),
+      lwd = c(1, 1.5))
```

Otra legend


Otra gráfica
Tamaño del genoma (en kb)
50
100
150
200
250
2
4
6
8
10
Los 10 fagos más grandes
Fagos
Fagos2

Ejercicios I

- ¿Qué hace `lty`?
- Si quiero cuatro colores con la función `rainbow`, ¿cómo los obtengo?
- Si una línea la quiero 3 veces más gruesa, ¿cómo lo hago?
- ¿Qué argumento uso para reducir el tamaño de un punto? Chequen la ayuda de `par`.

Ejercicios II

- Creen los siguientes vectores:

```
> x <- runif(100)
> y <- rnorm(100)
```

- Hagan una gráfica donde veamos esos puntos como líneas. Usen diferentes colores, diferentes puntos, diferente ancho de línea y diferente tipo de línea.
- Nombren el eje Y: *Valor*.
- Nombren el eje X: *Posición*.
- Pongan su usuario del dotProject como título.
- Agreguen una línea roja en $Y = 1$.
- Agreguen la *legend*. Para que no se sobrelape con los puntos, usen el argumento `ylim` en la función `plot`.

Ejercicios III

- Creen el siguiente vector:

  ```
  > islas <- sort(islands, decreasing = TRUE)[1:8]
  ```

- Hagan una gráfica de pie usando los colores de la función `topo.colors`.
- Hagan una gráfica de barras usando los colores de la función `rainbow`. Pónganle un título.
- Los nombres no salen completos, así que usen $las = 2$ y el argumento cex. . . para que se vean los nombres.
- Agreguen la *legend* en la parte superior del lado derecho. Los colores tienen que ser los mismos.

Pronto veremos

- Matrices y data frames en R
- Aprenderemos a leer datos de tablas en R
- Veremos como guardar una gráfica en formato pdf, png y jpeg.
- Veremos más gráficas

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
```