Seminar III: R/Bioconductor
August-December 2009

Leonardo Collado Torres
Bachelor in Genomic Sciences (LCG),
UNAM, Cuernavaca, Mexico
lcollado@lcg.unam.mx

http://www.lcg.unam.mx/~lcollado/

August 21, 2009

Assistants: Alejandro Reyes areyes@lcg.unam.mx, José Reyes jreyes@lcg.unam.mx and Víctor Moreno jmoreno@lcg.unam.mx

Note: Questions through the forum please. Those who are not from the sixth LCG generation send us an email so we can register you on the forum.

Abstract

Expected solutions to the first set of exercises.

1 Review

1. Why does the following expression show a warning? This is part of what rule?

> c(2, 3) + c(4, 5, 7)

> "Because the 2nd vector's length is not a multiple of the first one"

[1] "Because the 2nd vector's length is not a multiple of the first one"
"and vice versa. It's due to the recycling rule."

2. For all the prime numbers between 1 and 10, calculate its square root. What is the sum, median and mean?

```r
prime <- c(2, 3, 5, 7)
> sqrt(prime)
[1] 1.414214 1.732051 2.236068 2.645751
> sum(prime)
[1] 17
> sum(sqrt(prime))
[1] 8.028084
> median(prime)
[1] 4
> median(sqrt(prime))
[1] 1.984059
> mean(prime)
[1] 4.25
> mean(sqrt(prime))
[1] 2.007021
```
2 Plots

- Read the following file into R:
  
  ```
  ftp://ftp.ebi.ac.uk/pub/databases/genome_reviews/gr2species_phage.txt
  ```

  and make the following plots with your username on the title. Check whether using a log10 scale on the y axis helps.

  ```r
  > phage <- read.delim(file.path("ftp://ftp.ebi.ac.uk/pub/databases/genome_reviews/gr2species_phage.txt"), header = F)
  ```

  1. Sort the genome sizes (column 2) and plot them in a line with increasing values.

  ```r
  > plot(sort(phage[, 2]), type = "l",
  +       col = "blue", main = "lcollado")
  ```

  ![Plot](image)

  ```r
  > plot(sort(log10(phage[, 2])), type = "l",
  +       col = "blue", main = "lcollado")
  ```

  > print("You can say that using log10 does help on this case")

  [1] "You can say that using log10 does help on this case"

  \(^1\)Look for the useful function for this case
2. Plot a histogram with a density line for the same data.

```r
> hist(phage[, 2], col = "light blue",
+       prob = T)
> lines(density(phage[, 2]), col = "red")
```

3. Plot a boxplot for the differences between contiguous sorted genomes. Meaning, 2nd smallest - smallest, 3rd smallest - 2nd smallest, etc.\(^2\)

\(^2\)You might want to use `apropos` searching for diff...
> contig <- diff(sort(phage[, 2]))
> boxplot(contig, col = "forest green")

> contig <- diff(sort(log10(phage[, + 2])))
> boxplot(contig, col = "forest green")
> print("Boxplot without log10 was more useful")

[1] "Boxplot without log10 was more useful"
4. Make a barplot showing the 10 biggest genomes. Include the names on the x axis and every bar has to have a different color and/or density.

```r
> top <- sort(phage[, 2], decreasing = T)[1:10]
> names <- NULL
> for (i in 1:10) {
+   names <- c(names, phage[which(phage[, 2] == top[i]), 1])
+ }
> barplot(top, col = rainbow(10),
+   names.arg = phage[names, 1],
+   cex.names = 0.5, las = 2, main = "lcollado")

> barplot(log10(top), col = rainbow(10),
+   names.arg = phage[names, 1],
+   cex.names = 0.5, las = 2, main = "lcollado")
> print("Using log10 has almost no effect")
[1] "Using log10 has almost no effect"
```

3 They have to be readable
4 The `which` function might be useful.
3 Apply functions

1. What is the mean genome size for every type of replicon (column 4)? You have an atomic vector and a factor so use . . .

   ```r
tapply(phage[, 2], phage[, 4], mean)
   ```

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Segment L</th>
<th>Segment M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>253726.3</td>
<td>106813.8</td>
</tr>
<tr>
<td></td>
<td>106813.8</td>
<td>106813.8</td>
</tr>
<tr>
<td>Segment S</td>
<td>106813.8</td>
<td></td>
</tr>
</tbody>
</table>

2. Create a matrix `mat` with 10 rows and 10 columns and 100 random uniform values from 1 to 10. Create your own function and apply it to every row so that every row will now sum 1 in your new matrix `mat2`.

   ```r
   mat <- matrix(runif(100, 1, 10), 10, 10)
   mat2 <- t(sapply(1:10, function(x) {
     mat[x, ]/sum(mat[x, ])
   })),
   apply(mat2, 1, sum)
   ```

   `[1] 1 1 1 1 1 1 1 1 1 1`
3. Using the same matrix `mat`, make the matrix `mat3` with row sums equal to 1 using built in R matrix functions. `mat2` and `mat3` should be the same.

```r
> mat3 <- mat/rowSums(mat)
> rowSums(mat3)

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