

Ejercicios de R

Curso: Introducción a la Estadística y Probabilidades CM-274

Lecturas Importantes

1. Zev Chonoles tiene una lista importantes de Links que todo estudiantes de matemáticas o de CC tiene que conocer: <http://math.uchicago.edu/~chonoles/links/>,
2. *La guia de Estilo de Google para R*, una guia hecha por la comunidad de Google que usa R: <http://google-styleguide.googlecode.com/svn/trunk/Rguide.xml>.

Preguntas

1. ¿Que hacen los siguientes códigos ?. Corregir si es necesario. Explica que hace cada función, su propósito de los argumentos en la llamada de la función y que tipo de estructura es producida en cada llamada de función. Muestra ejemplos.

```
(a) > f1 <- function(d) {  
+   n <- nrow(d)  
+   dd <- cbind(d, 1:n)  
+   cm <- apply(dd[-n,], 1, f2)  
+   i <- which.min(cm[2,])  
+   j <- cm[1,i]  
+   return(c(d[i,j], i, j))  
+ }  
  
>  
> f2 <- function(x) {  
+   lx <- length(x)  
+   i <- x[lx]  
+   j <- which.min(x[(i+1):(lx-1)])  
+   k <- i+j  
+   return(c(k, x[k]))  
+ }  
  
(b) > n <- 1000  
>   u <- runif(n)  
>   x <- u^(1/3)  
>   hist(x, prob = TRUE, main = bquote(f(x)==3*x^2))  
>   y <- seq(0, 1, .01)  
>   lines(y, 3*y^2)  
  
(c) > n <- 1000  
>   p <- 0.25  
>   u <- runif(n)  
>   k <- ceiling(log(1-u) / log(1-p)) - 1  
>  
>   # Es mas eficiente, por que?  
>   k <- floor(log(u) / log(1-p))
```

```
(d) > n <- 1000
> a <- 3
> b <- 2
> u <- rgamma(n, shape=a, rate=1)
> v <- rgamma(n, shape=b, rate=1)
> x <- u / (u + v)
>
> q <- qbeta(ppoints(n), a, b)
> qqplot(q, x, cex=0.25, xlab="Beta(3, 2)", ylab="Muestra")
> abline(0, 1)
```

```
(e) > f1 <- function(n, mu, Sigma) {
+     d <- length(mu)
+     S <- svd(Sigma)
+     R <- S$u %*% diag(sqrt(S$d)) %*% t(S$v)
+     Z <- matrix(rnorm(n*d), nrow=n, ncol=d)
+     X <- Z %*% R + matrix(mu, n, d, byrow=TRUE)
+     X
+ }
```

```
(f) > l <- 2
> t <- 3
> u <- 100
> pp <- numeric(10000)
> for (i in 1:10000) {
+     N <- rpois(1, l * u)
+     Un <- runif(N, 0, u)
+     Sn <- sort(Un)
+     n <- min(which(Sn > t))
+     pp[i] <- n - 1
+ }
>
> pp <- replicate(10000, expr = {
+     N <- rpois(1, l * u)
+     Un <- runif(N, 0, u)
+     Sn <- sort(Un)
+     n <- min(which(Sn > t))
+     n - 1 })
>
> c(mean(pp), var(pp))
```

2. (a) La función

```
> f <-function(x,y){
+   if(y > 0)
+     y *sin(x)
+   else
+     x*sin(y)
+ }
```

no soporta el **reciclado**. Explica como puedes modificar la función para que si pueda soportarlo.

- (b) i. Encuentra expresiones en R para encontrar el epsilon de la máquina. https://en.wikipedia.org/wiki/Machine_epsilon.
- ii. Reproduce el siguiente código fuente en R, para mostrar la siguiente tabla de probabilidad de la distribución estándar normal. Explica el uso de la función `outer()`.

```

> id <- 0:4
> dn <- seq(0, .8, by = .2)
> p = outer(id, dn, function(x,y) pnorm(x + y))
> dimnames(p) = list(z = id, "Primer lugar decimal de z " = dn)
> p = round(p, 5)

```

- (c) i. Dada una matriz numérica X , determinar el índice de la primera fila cuyos elementos son todos números positivos (y que no contienen valores NA). Resuelve usando la función `apply` y usando un bucle `for`.
- ii. Escribe una función llamada `nesimo.na (x,n)` que toma un vector x y retorna
- el índice de la n -ésima valor NA que ocurre en x o
 - NA si hay menos de n valores NA en el vector x .

3. (a) Considere la siguiente curiosidad

$$\begin{aligned}
 8 \times 8 + 13 &= 77 \\
 8 \times 88 + 13 &= 717 \\
 8 \times 888 + 13 &= 7117 \\
 8 \times 8888 + 13 &= 71117 \\
 8 \times 8888 + 13 &= 711117 \\
 &\text{etc.}
 \end{aligned}$$

Escribe código R, que verifica las 10 primeras ecuaciones, imprimiendo los resultados, usando la fórmula dada

- usando el bucle `for`.
- usando una expresión vectorizada.

(b) Escribe código R que calcula el valor de la función

$$f(x,y) = \begin{cases} \sqrt{x} + \sin(y), & x \geq 0 \\ y + \cos(x), & \text{en otros casos.} \end{cases}$$

4. Explica el juego llamado morra <http://www.frontier.net/~grifftoe/morra.html>, a través de la explicación de las funciones utilizadas, así como el uso de los argumentos en la llamada de función. Muestra ejemplos que muestren las estructuras producidas por esas funciones.

```

> # Ejemplo del uso del simplex
> library(boot)
> A1 <- rbind(c(-2, 1, 1), c(4, -1, 3))
> b1 <- c(1, 3)
> a <- c(2, 2, 3)
> simplex(a = a, A1 = A1, b1 = b1, maxi = TRUE)
> detach(package:boot)
>
>
>
>
> resolver.juego <- function(A) {
+   min.A <- min(A)
+   A <- A - min.A
+   max.A <- max(A)
+   A <- A / max(A)
+   m <- nrow(A)

```

```

+     n <- ncol(A)
+     it <- n^3
+     a <- c(rep(0, m), 1)
+     A1 <- -cbind(t(A), rep(-1, n))
+     b1 <- rep(0, n)
+     A3 <- t(as.matrix(c(rep(1, m), 0)))
+     b3 <- 1
+     sx <- simplex(a=a, A1=A1, b1=b1, A3=A3, b3=b3,
+                   maxi=TRUE, n.iter=it)

+     a <- c(rep(0, n), 1)
+     A1 <- cbind(A, rep(-1, m))
+     b1 <- rep(0, m)
+     A3 <- t(as.matrix(c(rep(1, n), 0)))
+     b3 <- 1
+     sy <- simplex(a=a, A1=A1, b1=b1, A3=A3, b3=b3,
+                   maxi=FALSE, n.iter=it)

+     soln <- list("A" = A * max.A + min.A,
+                  "x" = sx$soln[1:m],
+                  "y" = sy$soln[1:n],
+                  "v" = sx$soln[m+1] * max.A + min.A)
+     soln
+   }
+ }
+
+ A <- matrix(c( 0,-2,-2,3,0,0,4,0,0,
+               2,0,0,0,-3,-3,4,0,0,
+               2,0,0,3,0,0,0,-4,-4,
+               -3,0,-3,0,4,0,0,5,0,
+               0,3,0,-4,0,-4,0,5,0,
+               0,3,0,0,4,0,-5,0,-5,
+               -4,-4,0,0,0,5,0,0,6,
+               0,0,4,-5,-5,0,0,0,6,
+               0,0,4,0,0,5,-6,-6,0), 9, 9)

+ library(boot)
+ s <- resolver.juego(A)
+ round(cbind(s$x, s$y), 7)

```

5. Se dice que un número primo es *gemelo* al par ordenado (x, y) , tal que $y = x + 2$. Construye una lista de los números primos gemelos menores que 100.
6. (a) Sea X el número de "unos" obtenidos en 12 lanzamientos de un dado. Entonces X tiene una distribución Binomial ($n = 12, p = 1/3$). Calcule una tabla de probabilidades binomiales para $x = 0, 1, \dots, 12$ por dos métodos:
 - i. Usando la fórmula para la densidad:

$$P(X = K) = \binom{n}{k} p^k (1 - p)^{n-k}$$

y aritmética en R. Usa $0:12$ para la secuencia de x valores y la función `choose` para calcular los coeficientes binomiales $\binom{n}{k}$.

- ii. Usando la función `dbinom` de R y comparar tus resultados con ambos métodos.

- (b) Sea X el número de "unos" obtenidos en 12 lanzamientos de un dado. Entonces X tiene una distribución Binomial ($n = 12, p = 1/3$). Calcule el CDF para $x = 0, 1, \dots, 12$ por dos métodos:
- Usando la función `cumsum` y el resultado del ejercicio anterior.
 - Con el uso de la función `pbinom`. ¿Qué es $P(X > 7)$?
7. (a) Escribe una función llamada `norma` que calcula la norma Euclídea de un vector numérico. La norma Euclídea de un vector $x = (x_1, \dots, x_n)$ es

$$\|x\| = \sqrt{\sum_{i=1}^n x_i^2}.$$

Usa operaciones vectorizadas para calcular la suma. Prueba esta función sobre los vectores $(0, 0, 0, 1)$ y $(2, 5, 2, 4)$ para verificar que el resultado de la función es correcto.

- (b) Construye una matriz con 10 filas y 2 columnas conteniendo data, de la siguiente manera

```
> x = matrix(rnorm(20), 10, 2)
```

Esta es una muestra aleatoria de 10 observaciones de una distribución normal bivariada estándar. Utiliza la función `apply` y la función `norma` del ejercicio anterior para cada una de esas 10 observaciones.

8. El modelo de Regresión Lineal Simple se ajusta a una respuesta y_i mediante una función lineal de una variable predictor x_i .

$$\hat{y}_i = a + bx_i \text{ para } (i = 1, \dots, n).$$

Por lo general, los mínimos cuadrados son utilizados para estimar los parámetros desconocidos a y b , pero a veces se utiliza la menor desviación absoluta. Esto requiere la elección de a y b a fin de minimizar

$$Q(a, b) = \sum_{i=1}^n |y_i - \hat{y}_i|.$$

- Implementa una función que calcule $Q(a, b)$. Debes definir una función de un solo argumento el cual es un vector cuyos primer elemento es a y el segundo elemento b .
 - Explica como usa R la función `optim` para obtener el mejor ajuste de valores de a y b .
9. (a) Escribe dos funciones en R: una que toma una función como argumento de entrada y otra que devuelve una función como salida.
- (b) Escribe una función en R, que toma un entero n como argumento y entonces usa un bucle `for` para generar n números aleatorios usando `runif`. Evalúa la performance de tu función usando la función `system.time`. Por ejemplo calcula la diferencia en el tiempo que toma tu función en retornar 10 millones de números y una llamada directa a `runif`.
- (c) Usa la función `choose` que calcula los coeficientes binomiales para determinar las probabilidades para todas las posibles salidas del siguiente código.

```
> adivina5 <- function(n, fSalida, Fecha_sorteo = "1952-1-17"){
+   if(dia_semana(as.Date(Fecha_sorteo)) == "Lunes")
+     Fecha_sorteo <- as.Date(Fecha_sorteo) +
+       c(0, cumsum(rep(c(3, 4), ceiling((n - 1)/2),
+         length.out = n - 1)))
+   else
+     Fecha_sorteo <- as.Date(Fecha_sorteo) +
+       c(0, cumsum(rep(c(4, 3), ceiling((n - 1)/2),
+         length.out = n - 1)))
+ }
```

```

+ Maquina <- sample(c("A", "B"), n, replace = TRUE)
+ Conjunto <- sample(1:10, n, replace = TRUE)
+ Dia <- substr(dia_semana(Fecha_sorteo), start = 1, stop = 1)
+ info <- cbind(as.character(Fecha_sorteo), Maquina, Conjunto, Dia)
+ A <- replicate(n*10, sample.int(40, 5))
+ for(i in 1:n) {
+   cat(info[i, ], "\n", file = fSalida, append = TRUE)
+   write(A[, (10*i - 9):(10*i)], file = fSalida, ncolumns = 5, append = TRUE)
+ }
+ }

```

10. (a) Fijar cada uno de los siguientes error y corregirlos

```

> mtcars[-1:4, ]
> mtcars[mtcars$cyl<= 5]
> mtcars[mtcars$cyl== 4 | 6, ]

```

(b) Por qué `x <- 1:5; x[NA]` produce 5 valores NA?.

(c) Por qué `mtcars[1:20]` retorna un error. Cómo difiere del similar `mtcars[1:20,]`?

(d) Qué hace `df [is.na (df)] <- 0`? Como funciona?.