Ejercicios de R

Curso: Introducción a la Estadística y Probabilidades CM-274

Lecturas Importantes

1. Una guía de ciencia de datos con R.
   `http://www.analyticsvidhya.com/blog/2016/02/complete-tutorial-learn-data-science-scratch/`.

Preguntas

1. Sea una definición de la raiz cuadrada, definida por el método de Newton

```
> r =
+   function(x, eps = 1e-10) {
+     g = 1
+     while(abs(1 - g^2/x) > eps)
+       g = .5 * (g + x/g)
+     g
+   }
```

   pero esta función trabaja para escalares, pero no cuando se pasa un vector

```
>r(c(1,2))
[1] 1
Warning message:
In while (abs(1 - g^2/x) > eps) g = 0.5 * (g + x/g) :
  the condition has length > 1 and only the first element will be used
```

   Una manera de resolver este problema es dividir el cálculo en dos partes; una para calcular la raíz cuadrada para valores escalares y la otra usa un bucle sobre un vector

```
> s.r =
+   function(x, eps = 1e-10) {
+     g = 1
+     while(abs(1 - g^2/x) > eps)
+     g = .5 * (g + x/g)
+   g
+ }
```

```
>  r =
+   function(x, eps = 1e-10) {
+     ans = numeric(length(x))
+     for(i in seq(along = x))
+       ans[i] = s.r(x[i])
+   ans
+ }
```

Calculando ahora los valores

```
>  r(c(1,2))
```

La estrategia de usar bucles funciona, pero tienden a ser ineficientes debido a los cálculos que se llevan a cabo elemento a elemento. Una estrategia alternativa es llevar a cabo el cálculo de vectores en lugar de escalares. En este caso particular, podemos cambiar el cálculo para que funcione con vectores como sigue

- Cambia la inicialización de g de forma que sea un vector y no un escalar

  `g = rep(1, length = length(x))`

- Cambiar la prueba para que los cambios de **g** continuen hasta que todos los elementos de la respuesta se hayan calculado con una suficiente precisión.

  `while(any(abs(1 - g^2/x)) > eps))`

Esto continúa mejorando las aproximaciones de la raíz cuadrada hasta que todos ellos han alcanzado el nivel de exactitud. Lleva esto a la práctica implementando los cambios en `r` y probando la función resultante.

La estrategia de la sección anterior conlleva el cálculo de la raíz cuadrada incluso después de que las raíces cuadradas se han determinado para elementos de x. Estos cálculos adicionales pueden evitarse manteniendo un registro de los elementos de x cuyas raíces no se calculan con la precisión suficiente y sólo realizando los cálculos para esos elementos.

```
n.d = abs(1 - g^2/x)) > eps
```

Esto puede ser hecho como parte de la prueba

```
while(any((n.d = abs(1 - g^2/x))) > eps))
```

Dentro del bucle, los cambios pueden llevarse a cabo sólo en el subconjunto de **g** que necesita ser actualizado, es decir,`g[n.d]`. Las actualizaciones se llevan a cabo utilizando sólo los elementos correspondientes de **g** y x , es decir, `g[n.d]` y `x [n.d]`.

2. El siguiente programa que produce?

```
> f1 <- function(x ,k){
+   n <- length(x)
+   r <- NULL
+   for(i in 1:(n -k)){
+     if(all(x[i:i + k -1]==1))r <- c(r, i)
+   }
+   return(r)
+ }
```

   (a) Si realizamos un test

   ```
   > f1(c(1,0,0,1,1, 0, 1,1,1), 2)
   ```

   y produce los valores 3467. Es correcto el resultado?.

   (b) Utiliza la función `debug()` para utilizar `browse` y mostrar

   - Si el vector fue recibido correctamente
   - cuando colocamos n dos veces en `browse`. Qué sucede cuando colocamos n tres veces en `browse`. Explica.
   - Si $k = 2$, que significa y que produce lo siguiente

```
Browse[2] > x[i:i + k- 1]
Browse[2] > i:i + k- 1
Browse[2] > i
Browse[2] > k
```

donde se encuentra el error, del código inicial, si es que existe?.

3. Escribimos dos funciones `primero` y `ultimo`, que extrae un número específico de elementos desde el inicio y el final de un vector (en el orden que aparecen en el vector). Las funciones deben ser llamadas como siguen

```
primero(x , k)
ultimo(x, k)
```

donde x es el vector de valores que son extraidos y k especifica el número de elementos a extraer. Si el argumento k es omitido en una de las llamadas, debe tomar por valor por defecto 1.

   (a) Asumiendo que $k \leq$ `length(x)`, escribimos versiones (lo más simples) de las funciones dadas anteriormente.

   (b) Modifica las funciones (a) de manera que si $k >$ `length(x)` entonces estas funciones deberian retornar los valores en x.

   (c) Modifica las funciones (a) de manera que si $k >$ `length(x)` las funciones retornan los k valores, si no hay valores existentes estos deben ser `NA`.

4. Para $n > 2$, la densidad `chi-cuadrado` tiene un máximo valor. Escribe código R, que usa la función `optimise` para localizar el máximo de la densidad para un valor $n > 2$.

5. El siguiente código define una función para la clase `palette`

```
> palette <- function(r, g, b, max=1) {
+   p <- list(colours=cbind(r, g, b), max=max)
+   class(p) <- "palette"
+   p
+ }
```

Ejemplos de la función son mostrados a continuación

```
> palette(1, 0, 0)

$colours
     r g b
[1,] 1 0 0

$max
[1] 1

attr(,"class")
[1] "palette"
```

```
>  palette(0:3, 0, 0, max=3)

$colours
     r g b
[1,] 0 0 0
[2,] 1 0 0
```

```
[3,] 2 0 0
[4,] 3 0 0

$max
[1] 3

attr(,"class")
[1] "palette"
```

(a) Escribe una función que imprima los colores usando la función `rgb()`. Por ejemplo

```
palette(1, 0, 0)
[1] "#FF0000"

palette(0:3, 0, 0, max=3)
[1] "#000000" "#550000" "#AA0000" "#FF0000"
```

(b) Escribe una función que retorne un objeto conteniendos los colores seleccionados. Por ejemplo

```
palette(0:3, 0, 0, max=3)[1]
[1] "#000000"

 palette(0:3, 0, 0, max=3)[2:3]
[1] "#550000" "#AA0000"
```

El resultado de esta función es un objeto de `palette` que está siendo impreso por la función anterior.

6. (a) ¿ Qué produce los siguientes códigos y las propiedades que muestran

```
> f1 <- function(x = {y <- 1; 2}, y = 0) {
+   x + y
+ }
> f1()
```

(b) ¿ Por qué las siguientes dos invocaciones de `lapply` son equivalentes?

```
> trims <- c(0, 0.1, 0.2, 0.5)
> x <- rcauchy(100)
>
> lapply(trims, function(trim) mean(x, trim = trim))
> lapply(trims, mean, x = x)
```

(c) Considera el siguiente problema : Dado una matriz númerica X, determina el índice de la primera fila de números positivos que no contiene NA. Resuelve el problema usando `for` y la función `apply()`.

(d) ¿Cómo se determina el entorno desde el que se llama una función?.

7. Se puede crear un `array` de prueba de 3 dimensiones, de la siguiente manera

```
> p_Array <- array( sample( 1:60, 60, replace=F), dim=c(5,4,3) )
```

La expresión anterior produce un array $5 \times 4 \times 3$, que puede representado matemáticamente como

$$\{x_{i,j,k} : i = 1,2,\ldots,5; j = 1,2,3,4; k = 1,2,3\}$$

Además

```
> apply(p_Array, 3, tmpFn)
```

significa que el índice k es guardado en la respuesta y la función `tmpFn` es aplicado a las 3 matrices $\{x_{i,j,1} : 1 \leq i \leq 5; 1 \leq j \leq 4\}$, $\{x_{i,j,2} : 1 \leq i \leq 5; 1 \leq j \leq 4\}$ y $\{x_{i,j,3} : 1 \leq i \leq 5; 1 \leq j \leq 4\}$.
Similarmente

```
> apply(p_Array, c(3, 1), tmpFn)
```

significa que los índices i y k son guardados en las respuestas y la función `tmpFn` es aplicado a los 15 vectores

$\{x_{i,j,1} : 1 \leq j \leq 4\}$, $\{x_{i,j,2} : 1 \leq j \leq 4\}$, etc.

La expresión anterior, hace la misma operación, pero el formato de la respuesta es diferente: al usar `apply` de esta manera, siempre vale la pena escribir un pequeño ejemplo para comprobar que el formato de la salida de `apply` es como se espera.

(a) Escribe una función `p_Fn` que toma un sólo argumento, que es una array de dimensión 3. Si este array es notado por $\{x_{i,j,k} : i = 1, 2, \ldots, d_1; j = 1, 2, \ldots, d_2; k = 1, 2, \ldots, d_3\}$ entonces a función `tmpFn` retorna una lista de la matriz $\{w_{i,j,k}\}$ de orden $d_1 \times d_2 \times d_3$ y la matriz $\{z_{i,j}\}$ de orden $d_2 \times d_3$, donde

$$w_{i,j,k} = x_{i,j,k} - \min_{i=1}^{d_1} x_{i,j,k} \quad \text{y} \quad z_{j,k} = \min_{i=1}^{d_1} x_{i,j,k} - \max_{i=1}^{d_1} x_{i,j,k}$$

(b) Escribe una función `p_Fn2`, que retorna una matriz $\{z_{j,k}\}$ de orden $d_2 \times d_3$, donde

$$z_{j,k} = \sum_{i=1}^{d_1} x_{i,j,k}^k.$$

8. Un camino aleatorio simétrico empieza en el origen y es definido como sigue: Supongase que $X_1, X_2, \ldots$ son variables aleatorias idénticamente distribuidas independientes con la siguiente distribución

$$\begin{cases} +1 & \text{con probabilidad } 1/2 \\ -1 & \text{con probabilidad } 1/2 \end{cases}$$

Definimos la secuencia $\{S_n\}_{n \geq 0}$ como

$$\begin{aligned} S_0 &= 0 \\ S_n &= S_{n-1} + X_n, \quad \text{para} \quad n = 1, 2, \ldots \end{aligned}$$

Entonces $\{S_n\}_{n \geq 0}$ es un camino aleatorio simétrico empezando en el origen. La posición del camino aleatorio en el tiempo n es la suma de los previos pasos : $S_n = X_1 + \cdots X_n$.

(a) Escribe una función `rcamino(n)` que toma un argumento n y retorna un vector el cuál es una realización de $(S_0, S_1, \ldots, S_n)$ las primeras n posiciones de un camino aleatorio simétrico que empieza en el origen. El código siguiente

```
> sample( c(-1,1), n, replace=TRUE, prob=c(0.5,0.5) )
```

simula n pasos.

(b) Escribe una función `rcaminoPos(n)` que simula el hecho que un camino dura para una longitud de tiempo n y que devuelve la longitud de tiempo del camino que pasa por encima del eje X. Debes observar que un camino con longitud 6 y vértices en $0, 1, 0, -1, 0, 1, 0$ está 4 unidades de tiempo por encima del eje X y 2 unidades de tiempo por debajo del eje X).

9. El conjunto de datos `faithful` contiene la duración (en minutos) `eruptions` y el tiempo de espera hasta otra erupción `waiting`(en minutos) de para un geyser Old Faithful. Estamos interesados en conocer la relación que hay entre las dos variables.

   (a) Crea una variable factor `longitud` que es **t_erup1** si la erupción es menor que 3.2 minutos y **t_erup2** en otros casos.

   (b) Usa la función `bwplot` en el paquete `lattice`, para construir un gráfico (diagrama de cajas paralelos) de los tiempos de espera para las erupciones **t_erup1** y **t_erup2**.

   (c) Usa la función `densityplot` construye un gráfico (de densidades superpuestas) de los tiempos de espera para las erupciones **t_erup1** y **t_erup2**.

   En el problema anterior, se compararon los tiempos de espera de los géiseres Old Faithful para las erupciones **t_erup1** y **t_erup2** donde la variable `longitud` en el data frame `faithful` define la duración de la erupción.

   (d) Supongamos un data frame `dframe` que contiene una variable numérica `num.var` y un factor `factor.var`. Después de que el paquete `ggplot2` se halla cargado, entonces, los comandos de R

```
> ggplot(dframe, aes(x = num.var, color = factor.var))
> + geom_density()
```

   construirán estimaciones de densidades superpuestas de la variable `num.var` para cada valor del factor `factor.var`. Utiliza estos comandos para construir estimaciones de densidades superpuestas de los tiempos de espera de los géiseres con erupciones **t_erup1** y **t_erup2**.

   (e) Con un data frame `dframe` que contiene una variable numérica `num.var` y un factor `factor.var`, la sintaxis de `ggplot2`

```
> ggplot(dframe, aes(x = num.var, color = factor.var))
> + geom_boxplot()
```

   construirá caja de bloques paralelos de la variable `num.var` para cada valor del factor `factor.var`. Utiliza estos comandos para construir cajas de bloques paralelos de los tiempos de espera de los géiseres con erupciones **t_erup1** y **t_erup2**.

   Sugerencia: Revisa el siguiente ejemplo

```
> library(ggplot2)
>
> #datos de muestra
>
> dat <- data.frame(dens = c(rnorm(100), rnorm(100, 10, 5))
+                   , lines = rep(c("a", "b"), each = 100))
> #Plot.
> ggplot(dat, aes(x = dens, fill = lines)) + geom_density(alpha = 0.5)
```

10. Supongamos que se está interesado en mostrar 3 miembros de la familia de curvas beta, donde la densidad con parámetros a y b, denotados por `Beta(a, b)` es dado por

$$f(y) = \frac{1}{B(a,b)} y^{a-1}(1-y)^{b-1}, \quad 0 < y < 1.$$

   Se puede dibujar un sola densidad beta, con paramétros $a = 5$ y $b = 2$, usando la función `curve`:

```
> curve(dbeta(x, 5, 2), from=0, to=1)
```

   (a) Usa tres aplicaciones de la función `curve` para mostrar las densidades `Beta(2, 6)`, `Beta(4, 4)` y `Beta(6, 2)` en un sólo gráfico.

(b) Usa el siguiente comando de R, para colocar un título al gráfico de las ecuaciones de densidad beta

```
> title(expression(f(y)==frac(1,B(a,b))*y^{a-1}*(1-y)^{b-1}))
```

(c) Usa la función `text`, para etiquetar cada una de las curvas betas con sus correspondientes valores de los paramétros a y b.

(d) Redibuja el gráfico usando diferentes colores o tipos de líneas para las tres curvas de densidad.

11. Explica el siguiente código y el Teorema del Límite Centrak. La respuesta debe ser correctamente escrita y ordenada

```
> iter = 2000
> avg0 <- avg1 <- avg2 <- avg3 <- rep(0, iter)
> for (i in 1:iter) {
+     S = rexp(90)  # muestra desde la distribucion   exp(1)
+     avg0[i] = S[1]
+     avg1[i] = mean(S[1:3])
+     avg2[i] = mean(S[1:30])
+     avg3[i] = mean(S[1:90])
+ }
> SR = stack(list(`n=1` = avg0, `n=3` = avg1, `n=30` = avg2,
+     `n=90` = avg3))
> names(SR) = c("promedios", "n")
> ggplot(SR, aes(x = averages, y = ..density..)) + facet_grid(n ~
+     .) + geom_histogram() + scale_x_continuous(limits = c(0,
+     3))
```

12. Escribe una función llamada `listaFN` que toma un único argumento n e implementa el siguiente algoritmo

(a) Simula n números independientes, denotado por $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ desde la distribución normal estándar.

(b) Calcula la media $\bar{\mathbf{x}} = \sum_{j=1}^{n} x_j/n$.

(c) Si $\bar{\mathbf{x}} \geq 0$, entonces simula n números independientes, denotados por $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ desde la densidad exponencial con media $\bar{\mathbf{x}}$.

(d) Si $\bar{\mathbf{x}} < 0$, entonces simula n números independientes, denotados por $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ desde la densidad exponencial con media $-\bar{\mathbf{x}}$. Se coloca $\mathbf{y} = (y_1, y_2, \ldots, y_n) =$ **-z**.

(e) Calcula k que es el número j con $|y_j| > |x_j|$.

(f) Retorna la lista de **x, y** y k con nombres `xVec, yVec` y `count` respectivamente.

(g) Ejecuta las siguientes líneas y verifica el formato de las respuestas

```
> lapply( rep(10,4), listaFN )
> sapply( rep(10,4), listaFN )
```