Ejercicios de R

Curso: Introducción a la Estadística y Probabilidades CM-274

Lecturas Importantes

1. Notas de Sebastian Raschka

   `http://www.kdnuggets.com/2016/05/implement-machine-learning-algorithms-scratch.html` para implementar algoritmos en Machine Learning desde cero.

2. Artículo de Ari Lamstein `http://www.kdnuggets.com/2016/11/data-science-101-good-at-r.html` de como empezar a programar en R para ciencia de datos.

Preguntas

1. Responde las siguientes preguntas

   (a) Usa funciones de *R* para reproducir la siguiente tabla de probabilidad de la distribución normal estándar.

   ```
   > #<  codigo >

   > p
      Lugar del primer decimal z
   z    0       0.2    0.4     0.6     0.8
   0 0.50000 0.57926 0.65542 0.72575 0.78814
   1 0.84134 0.88493 0.91924 0.94520 0.96407
   2 0.97725 0.98610 0.99180 0.99534 0.99744
   3 0.99865 0.99931 0.99966 0.99984 0.99993
   4 0.99997 0.99999 0.99999 1.00000 1.00000
   ```

   (b) Usa la aproximación de Riemann

   $$I_{LR}(a,b) = h\sum_{i=1}^{n}\phi(x_i)$$

   para computar la integral

   $$\int_0^2 \phi(x)dx$$

   donde $\phi(\cdot)$ es la función densidad de la distribución normal estándar y asumamos conocida (usa la función `dnorm`). El código no debe contener bucles.

2. Explica el siguiente código así como se utiliza en la `Ley de los grandes números` usando la librería `ggplot2`. Debes explicar en que consiste la Ley de los Grandes números y el caso que se utiliza en el código.

```
> n <- 10000
> media <- cumsum(rnorm(n))/(1:n)
> g <- ggplot(data.frame(x = 1:n, y = media), aes(x = x, y = y))
> g <- g + geom_hline(yintercept = 0) + geom_line(size = 2)
> g <- g + labs(x = "Numero de obs", y = "media cumulativa")
> g
```

3. 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).

4. Dado un vector $(x_1, x_2, \ldots x_n)$, la autocorrelación muestral es definida como

$$r_k = \frac{\sum_{i=k+1}^{n} (x_i - \overline{x})(x_{i-k} - \overline{x})}{\sum_{i=1}^{n} (x_i - \overline{x})^2}$$

   (a) Escribe una función `tmpFn(xVec)` que toma un sólo argumento `xVec` que es un vector y retorna una lista de dos valores $r1$ y r_2. En particular encuentra los valores de r_1 y r_2 para el vector $(2, 5, 8, \ldots, 53, 56)$.

   (b) Generaliza la función de manera que toma dos argumentos: el vector `xVec` y el entero k que está entre 1 y $n - 1$ donde n es la longitud de `xVec`. La función debe retornar un vector de los valores $(r_0 = 1, r_1, \ldots, r_n)$.

5. 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.

6. Supongamos

$$A = \begin{pmatrix} 1 & 1 & 3 \\ 5 & 2 & 6 \\ -2 & -1 & -3 \end{pmatrix}$$

(a) Verifica que $A^3 = 0$, donde 0 es una matriz 3×3 con entradas igual a 0.

(b) Reemplaza la tercera columna de A por la suma de la segunda y tercera columna.

7. Crea una matriz 6×10 de enteros aleatorios, desde $1, 2, \ldots, 10$ ejecutando los dos línea de código

```
> set.seed(75)
> aMat <- matrix(sample(10, size = 60, replace = T), nr = 6)
```

(a) Encuentra el número de entradas en cada fila que son mayores que 4.

(b) ¿ Cuantas filas contienen dos ocurrencias del número 7?.

(c) Encuentra esos pares de columnas, cuyo total es mayor que 75. La respuesta debe ser una matriz de dos columnas; así por ejemplo, la fila $(1, 2)$ en la matriz de salida que la suma de columnas 1 y 2 en la matriz es mayor que 75.

8. Escribe una función que toma dos argumentos n y k que son números enteros positivos y que retorna una matriz $n \times n$

$$\begin{pmatrix} k & 1 & 0 & 0 & \ldots & 0 & 0 \\ 1 & k & 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & k & 1 & \ldots & 0 & 0 \\ 0 & 0 & 1 & k & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & k & 1 \\ 0 & 0 & 0 & 0 & \ldots & 1 & k \end{pmatrix}$$

9. (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?.

10. Ejecute las siguientes líneas para crear dos vectores de números aleatorios enteros los cuáles son escogidos con reemplazamiento, desde los enteros $0, 1, \ldots, 999$. Ambos tienen longitud 250.

```
> set.seed(50)
> xVec <- sample(0:999, 250, replace=T)
> yVec <- sample(0:999, 250, replace=T)
```

Supongamos que $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ denota el vector `xVec` y $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ denota el vector `yVec`.

(a) Crea el vector $(y_2 - y_1, \ldots, y_n - y_{n-1})$.

(b) Crea el vector $\left(\frac{\sin(y_1)}{\cos(x_2)}, \frac{\sin(y_2)}{\cos(x_3)}, \ldots, \frac{\sin(y_{n-1})}{\cos(x_n)}\right)$.

(c) Crea el vector $(x_1 + 2x_2 - x_3, x_2 + 2x_3 - x_4, \ldots, x_{n-2} + 2x_{n-1} - x_n)$.

(d) Calcula $\sum_{i=1}^{n-1} \frac{e^{-x_i+1}}{x_i + 10}$.

Usando los vectores `xVec` y `yVec` y las funciones `sort, order, mean, sqrt, sum` y `abs`.

(e) Selecciona los valores en `yVec` que son mayores que 600.

(f) ¿ Cuáles son las posiciones de los índices de los valores mayores que 600.

(g) ¿ Cuáles son los valores de `xVec` que corresponde a los valores en `yVec` mayores que 600.

(h) Crea el vector $(|x_1 - \bar{\mathbf{x}}|^{1/2}, (|x_2 - \bar{\mathbf{x}}|^{1/2}, \ldots, (|x_n - \bar{\mathbf{x}}|^{1/2}$, donde $\bar{\mathbf{x}}$ denota la media del vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.

(i) Ordena los números en el vector `xVec` en el order de valores crecientes de `yVec`.

(j) Selecciona los elementos en `yVec` con índices $1, 4, 7, 10, 13, \ldots$

11. (a) Supongamos que $x_0 = 1$ y $x_1 = 2$ y

$$x_j = x_{j-1} + \frac{2}{x_{j-1}}, \quad \text{para} \quad j = 1, 2, \ldots$$

Escribe una función que tome como argumento un único n y retorne los $n - 1$ valores de la secuencia $(x_j)_{j \geq 0}$, es decir los valores $x_0, x1, x_2, \ldots, x_{n-2}$.

(b) Escribe una función que toma como único argumento `vector1` que es un vector. La función debe retornar

$$\sum_{j=1}^{n} e^j$$

donde n es la longitud de `vector1`.

12. (a) Considere los siguientes dos métodos para simular distribuciones discretas con valores $0, 1, 2, 3, 4, 5$, con valores de probabilidad $0.2, 0.3, 0.1, 0.15, 0.05, 0.2$.
El primer método es el de **inversión**

```
> probs <- c(0.2, 0.3, 0.1, 0.15, 0.05, 0.2)
> randiscrete1 <- function(n, probs) {
+   cumprobs <- cumsum(probs)
+   numero1 <- function() {
+     x <- runif(1)
+     N <- sum(x > cumprobs)
+     N
+  }
+   replicate(n, numero1())
+ }
```

y el segundo método es el de rechazo(rejection)

```
> randiscrete2 <- function(n, probs) {
+   numero1 <- function() {
+     repeat{
+       U <- runif(2, min=c(-0.5, 0), max=c(length(probs) - 0.5, max(probs)))
+       if (U[2] < probs[round(U[1]) + 1]) break
+       }
+     return(round(U[1]))
+   }
+ replicate(n, numero1())
+ }
```

Ejecuta ambas funciones, usando $n = 100, 1000$ y 10000. Usa `system.time()` para determinar que método es más rápido.

(b) Repite el ejercicio anterior sobre una distribución sobre los enteros $\{0, 1, 2, \ldots, 99\}$, definida por

```
> set.seed(91626)
> probs <- runif(100)
> probs <- probs / sum(probs)
```

13. Explica la salida de los siguientes códigos

(a)
```
> X = runif(200)
> Y = runif(200)
> W = rnorm(300, sd = 0.2)
> Z = rnorm(300, sd = 0.2)
> Q = runif(300) * 2 - 1
> P = runif(300) * 2 - 1
> qplot(X, Y, xlab = "$X_1$", ylab = "$X_2$")
> qplot(W, Z, xlab = "$X_1$", ylab = "$X_2$")
```

(b)
```
> qplot(X, X + runif(200)/10, xlab = "$X_1$", ylab = "$X_2$")
> qplot(X, -X + runif(200)/10, xlab = "$X_1$", ylab = "$X_2$")
```

(c)
```
> qplot(cos(seq(0, 2 * pi, length = 30)), sin(seq(0,
+     2 * pi, length = 30)), xlab = "$X_1$", ylab = "$X_2$")
> qplot(Q[(Q^2 + P^2) < 1], P[(Q^2 + P^2) < 1], xlab = "$X_1$",
+     ylab = "$X_2$")
```

14. (a) Escribe una función que evalua polinomios de la forma

$$P(x) = c_n x^{n-1} + c_{n-1} x^{n-2} + \cdots + c_2 x + c_1$$

La función debe tomar x y un vector con los coeficientes del polinomio como argumento y debe de retornar el valor del polinomio evaluado. La función debe llamarse `directpoly()`

(b) Se puede evaluar un polinomio en x podemos utilizar la `regla de Horner`:

- Colocamos $a_n \leftarrow c_n$.
- Para $i = n-1, \ldots, 1$ colocamos $a_i = a_{i+1} x + c_i$.
- Retorna a_1 (Este es el valor de $P(x)$).

Escribe una función en R, con argumentos x y un vector con los coeficientes del polinomio y retorna el valor del polinomio evaluado en x. La función debe devolver valores apropiados cuando x es un vector. La función se debe llamar `horner()`.

(c) Prueba los siguientes códigos

```
> system.time(directpoly(x=seq(-10, 10, length=5000000),c(1, -2, 2, 3, 4, 6, 7)))
> system.time(horner(x=seq(-10, 10, length=5000000),  c(1, -2, 2, 3, 4, 6, 7)))
```

(d) ¿ Qué ocurre en la comparación cuando el número de coeficientes del polinomio es muy pequeño?. Prueba con el polinomio

$$P(x) = 2x^2 + 17x - 3$$

15. Soluciona la ecuación de diferencias: $x_n = rx_{n-1}(1 - x_{n-1})$ con el valor inicial x_1:

(a) Escribe una función `dif_ecuacion(inicio, erre, n_cont)` que retorna el vector $(x_1, \ldots, x_n)$, donde $x_k = rx_{k-1}(1 - x_{k-1})$ donde `n_cont` denota n, `inicio` denota x_1 y `erre` denota r. Prueba la función que has escrito:

- Para $r = 2$ y $0 < x < 1$, se debe conseguir que $x_n \to 0.5$ cuando $n \to \infty$.
- Prueba

```
> tmp <-dif_ecuacion(inicio = 0.95, erre = 2.99, n_cont = 500)
```

Ahora vuelve a la consola y escribe

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

También prueba

```
> plot(tmp[300 : 500], type="l")
```

(b) Escribe una función que determina el número de iteraciones necesarias para conseguir $|x_n - x_{n-1}| < 0.02$. Esta función tiene dos argumentos: `inicio` y `erre` (Para `inicio = 0.95` y `erre = 2.99`, la respuesta debe dar 84).

16. Estudia el siguiente código

```
> library(scatterplot3d)
> R = expand.grid(0:12, 0:12)
> R[, 3] = 12 - R[, 1] - R[, 2]
> names(R) = c("x1", "x2", "x3")
> for (i in 1:nrow(R)) {
+     v = R[i, 1:3]
+     if (sum(v) == 12 & all(v >= 0)) {
+         R$p[i] = dmultinom(v, prob = c(0.5, 0.3, 0.2))
+     } else R$p[i] = 0
+ }
> par(cex.main = 1.5, cex.axis = 1.2, cex.lab = 2, cex.symbols = 2,
+     col.axis = "blue", col.grid = "lightblue")
> scatterplot3d(x = R$x1, y = R$x2, z = R$p, type = "h",
+     lwd = 10, pch = " ", xlab = "$x_1$", ylab = "$x_2$",
+     zlab = "$p_{X_1,X_2,X_3}(x_1,x_2,1-x_1-x_2)$",
+     color = grey(1/2))
> title("...")
```