GUÍA TEORICO PRÁCTICA 5

Boxplots y algunas simulaciones

Ejercicio 1:

a) Las siguientes instrucciones permiten verificar los valores de la tabla11 para una $\chi 2$ con 1 grado de libertad (**Interpretación del Boxplot** notas de clases teóricas).

```
> ene <- 1   # grados de libertad
> # mediana cuartil inf. cuartil sup.
qchisq(c(0.5, 0.25, 0.75), ene)
[1] 0.4549364 0.1015310 1.3233037
> auxi1 <- qchisq(c(0.5, 0.25, 0.75), ene)

> # distancia intercuartil
> diff(auxi1[c(2, 3)])
[1] 1.221773
> #Vallas
auxi1[3] + 1.5 * diff(auxi1[c(2, 3)])
[1] 3.155963
> auxi1[2] - 1.5 * diff(auxi1[c(2, 3)])
[1] -1.731128
> auxi2 <- auxi1[3] + 1.5 * diff(auxi1[c(2, 3)])

> # Porcentaje fuera de la valla superior
100 * (1 - pchisq(auxi2, ene))
[1] 7.565006

> # Porcentaje fuera de la valla inferior
100 * (pchisq(auxi1[2] - 1.5 * diff(auxi1[c(2, 3)]),
ene))
[1] 0

> # Límites dados por μ ± 1.96 σ
> ene + 1.96 * sqrt(2 * ene)
[1] 3.771859
> ene - 1.96 * sqrt(2 * ene)
[1] -1.771859

> # Porcentaje fuera de los límites μ ± 1.96 σ
> auxi3 <- ene + 1.96 * sqrt(2 * ene)
> auxi4 <- ene - 1.96 * sqrt(2 * ene)
> 100 * (pchisq(auxi4, ene) + (1 - pchisq(auxi3, ene)))
[1] 5.212169
```

b) Verifique los valores que presentan las tablas 10 y 11 de las notas de las clases teóricas. Sugerencia: puede seguir los pasos dados en a)

Ejercicio 2:

Iteración incondicional ("ciclo")

> for (i in vec) expresión

Aquí "i" es una variable "muda" que se crea en forma temporaria. Toma el valor vec[1] y ejecuta la "expresión" , luego toma el valor vec[2] y ejecuta la "expresión" nuevamente. De esta manera "expresión" es ejecutada tantas veces como length(vec).

a) Ejecute las siguientes instrucciones

```
> x <- rnorm(10)
> y <- numeric(length(x))
> for(i in 1:length(x)) y[i] <- sqrt(x[i])
```

esto realiza lo mismo que y <- sqrt(x)

Proposiciones condicionales if - then - else

La sintaxis básica de una proposición condicional es

> if (condición) expresión

La "condición" tiene que ser alguna expresión que devuelva un valor lógico (T ó F). Por ejemplo la "condición" puede ser "x > 2 " donde x es un escalar.

La "expresión" es un comando único ó un grupo de comandos encerrados entre llaves. Cuando R encuentra esta proposición primero evalúa la "condición" y si condición=T ejecuta la "expresión" si condición=F no hace nada.

b) Genere 10 valores pseudo Normales y calcule cuantos valores son mayores que: 0 y cuantos valores son mayores que 1.
i) Utilice una iteración y la proposición if (condición).
ii) No utilice iteraciones.

Observación: ya hemos visto que al programar en R hay muchas maneras de realizar una misma tarea. No todas son igualmente eficientes. Un principio que

debería seguir es tratar de minimizar las expresiones y en consecuencia los ciclos.

La sintaxis básica de la *proposición if* puede ser **extendida** con "*else*":

```
> if(condición) expresión1
> else  expresión2
```

Cuando R encuentra estas proposiciones, primero evalúa "condición". Si condición=T luego ejecuta "expresión1" y si condición = F ejecuta "expresión2".

Si hay múltiples *proposiciones-if* , cada "else" se asocia al “if” más reciente: Por ejemplo

```
> if(condición1) expresión1
> else if(condición2) expresión2
> else if(condición3) expresión3
> else expresión4
```

En este ejemplo, si condición1=T luego R ejecutará expresión1, en caso contrario procederá a evaluar la condición2. R ejecutará la expresión2 si resulta condición2 = T, en caso contrario evaluará la condición3. Si la condición3 es evaluada, R ejecutará la expresión3 si condición3 =T, si no lo es ejecutará la expresión4.

Ejercicio 3: Verifique mediante un estudio de simulación para muestras de tamaño 5 de una Gaussiana estándar que aproximadamente el

- 67% de las muestras no tienen valores fuera de los puntos de corte (vallas internas)
- 24% de las muestras tienen un outlier
- 9% de las muestras tienen 2 outliers.

Utilice por lo menos 1000 muestras de tamaño 5.

La siguiente función permite realizar la simulación anterior.

```
outli.sim=function(n,m)
#n=numero de simulaciones
#m=tamaño de la muestra
```

```
{
   corte<-0
   out1<-0
   out2<-0

 for(i in 1:n)
   {x<-rnorm(m)

   q<-quantile(x,probs=c(0.25,0.75),na.rm=T)

   dq<-q[2]-q[1]
   sumout<-sum((x<(q[1]-1.5*dq))|(x>(q[2]+1.5*dq)))

    if(sumout==1) out1<-out1+1
    if(sumout==2) out2<-out2+1
    if(sumout==0) corte<-corte+1

}
out1<-out1/n
out2<-out2/n
corte<-corte/n

out<-c(out1,out2,corte)
names(out)<-c("% de muestras con 1 outlier"," % de muestras
   con 2 outliers",
      "% de muestras sin outliers")
out

}
```

Fin de la función

a) Analice lo que realiza esta función con pocas replicaciones.

b) Elimine los ciclos y las asignaciones condicionales de la simulación anterior utilizando operaciones matriciales. Realice las simulaciones pedidas.

Ejercicio 4: Repita la simulación del ejercicio 4 para muestras de tamaño 5 de la distribución *t* con 5, 10 y 20 grados de libertad.

Ejercicio 5: Repita la simulación del ejercicio 4 para muestras de tamaño 5 de la distribución Chi-cuadrado con 5 y 20 grados de libertad.

Ejercicio 6: Para el conjunto de datos analizado en la práctica anterior `faitfhul$eruptions,` estudie las siguientes instrucciones de **R**

```
with(faithful, {
    par(mfrow=c(2,1))
    boxplot(eruptions, horizontal =TRUE)
    plot(density(eruptions, bw = 0.15))
    rug(eruptions)
    rug(jitter(eruptions, amount = 0.01), side = 3,
col = "light blue")
})
```

¿Qué realizan las funciones **`with`**, **`rug`** y **`jitter`**?