GUÍA Nro. 5

Ejercicio 1: Recalcule todos los valores que muestra la tabla 3-2 del UREDA (pág 64).

Sugerencia: las siguientes instrucciones calculan los valores de la tabla para una $\chi 2$ con 1 grado de libertad.

```
> 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
```

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 S-PLUS 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 S-plus 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 S-PLUS 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 S-PLUS ejecutará expresión1,
en caso contrario procederá a evaluar la condición2. S-PLUS ejecutará la expresión2 si resulta condición2=T, en caso contrario evaluará la condición3. Si la condición3 is evaluatada, S-PLUS 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) La función anterior es muy lenta. Escriba otra función eliminando al máximo posible los ciclos y las asignaciones condicionales. 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: En el estudio de las 10 ciudades más grandes de 16 paises (clases teóricas) se llegó a la conclusión que la transformación raíz cuadrada era muy débil y logaritmo era un poco fuerte (aunque satisfactoria). Estas observaciones sugieren que una transformación intermedia, como por ejemplo la raíz cuarta, podría también ser efectiva para estabilizar la dispersión de los 16 lotes.

a) ¿Por qué decimos que la raíz cuarta es una transformación intermedia entre le raíz cuadrada y el logaritmo?

b) ¿Qué pendiente en el gráfico de dispersión-nivel corresponde a p=1/4

c) Aplique la transformación raíz cuadrada a los datos y determine el efecto en los lotes graficando las distancias intercuartos versus las medianas para la nueva escala.

d) Construya boxplots paralelos, ordenados por la mediana, para los datos transformados por logaritmo y por la raíz cuarta. ¿Cuál de las dos transformaciones parece más efectiva para estabilizar la dispersión?

e) Por cuadrados mínimos obtuvimos una recta con pendiente 0.69. Repita la parte d) para la transformación de potencias sugerida por esta pendiente.

Ejercicio 7:

a) Lea los datos que están en el objeto **`ships`** de la librería **`MASS`** y vea en el **`help`** de qué se tratan.

```
> library(MASS)
> v1<-ships$incidents
> v2<-ships$type
> v1
 [1]  0  0  3  4  6 18  0 11 39 29 58 53 12 44  0 18
[17]  1  1  0  1  6  2  0  1  0  0  0  0  2 11  0  4
[33]  0  0  7  7  5 12  0  1
> v2
 [1] A A A A A A A A B B B B B B B C C C C C C C C
[25] D D D D D D D D E E E E E E E E
```

b) Sumo **uno** a todos los valores de la variable incidentes para eliminar los ceros.

```
> v11 <- v1+1
```

PASOS PARA CONTRUIR EL GRAFICO DE DISPERSION VS. NIVEL

c) Vea qué produce la función **split.** Utilice el **help.**

```
> data.class(v11)
[1] "numeric"

> aaa1 <- split(v11,v2)
> data.class(aaa1)
[1] "named"  ("list", en S-plus 2000)
> is.list (aaa1)
[1] T
```

d) Cálculo de la mediana y cuartiles superior e inferior de la cantidad de incidentes por cada tipo de barco (A, B, C, D, E). ¿Qué devuelve la función **sapply**? ¿Qué obtiene con **> sapply(aaa1,mean)** ?

```
> sapply(aaa1,quantile)
         A     B    C    D  E
[1,]  1.00  1.00 1.00  1.0  1
[2,]  1.00 17.50 1.75  1.0  1
[3,]  4.50 35.00 2.00  1.0  4
[4,]  8.25 47.25 2.25  3.5  8
[5,] 19.00 59.00 7.00 12.0 13


> dq <- sapply(aaa1,quantile)[4,] -
        sapply(aaa1,quantile)[2,]
> medianas<- sapply(aaa1,quantile)[3,]

> logdq <-log10(dq)
> logmedianas <- log10(medianas)
> plot(logmedianas,logdq)
```

¿Que transformaciones de la escalera de Tukey le parecen adecuadas para los datos en estudio, de acuerdo con el gráfico de dispersión nivel?

d) Compare los boxplots de los datos originales con los de los transformados por logaritmo y raiz cuadrada:

```
> lv11<-log10(v11)
> sqrtv11<-sqrt(v11)

> aaa<-split(v1,v2)
> laaa<-split(lv11,v2)
> sqraaa<-split(sqrtv11,v2)

> par(mfrow=c(3,1))

> boxplot(aaa[sort.list(sapply(aaa,median))],
+ main="accidentes segun tipo de barco",
+ sub="datos crudos")

> boxplot(laaa[sort.list(sapply(laaa,median))],
+ sub="datos transformados por log10")
```

```
> boxplot(sqraaa[sort.list(sapply(sqraaa,median))],
+ sub="datos transformados por sqrt")
```

d) Ajuste de una recta por cuadrados mínimos al gráfico de dispersión-nivel

```
> lm(logdq~logmedianas)
```

e) Pruebe con la transformación sugerida por el ajuste de cuadrados minimos para estabilizar la dispersion de los incidentes en los distintos tipos de barcos y compare.

Ejercicio 8: Un gráfico de dispersión-nivel puede construirse utilizando las medias muestrales para el nivel de los lotes y el desvío estándar muestral para las dispersiones. La relación entre la pendiente y la potencia se mantiene.
a) Calcule la media y los desvíos necesarios para la construcción del gráfico de dispersión-nivel para los datos de las grandes ciudades.

b) Construya el gráfico de dispersión-nivel modificado. ¿Cuál es la potencia sugerida?

c) ¿Cuáles son las ventajas o desventajas de utilizar esta modificación?