Tema 5: Programación de funciones y subrutinas. Ejercicios resueltos.

1. Predecir el resultado de cada una de los conjuntos de expresiones siguientes. Comprobar después el resultado:

   a) `answer <- 0 ; for (j in 3:5) {answer <- j+answer}`

   b) `answer<- 10 ; for (j in 3:5) {answer <- j+answer}`

   c) `answer <- 10; for (j in 3:5) {answer <- j*answer}`

   Solución: `12, 22, 600`

2. Revisar la ayuda de la función `prod()`. Utilizarla para realizar el cálculo del último apartado del ejercicio anterior.

   Ayuda: `10*prod(3:5)`

3. Sumar todos los números del 1 al 100 de dos maneras: con un bucle (utilizando `for`) y utilizando la función `sum`.

   Ayuda:

   ```
   suma <-0; for (i in 1:100) suma <- suma + i
   sum(1:100) # también sum(seq(1,100))
   ```

4. Multiplicar todos los números del 1 al 10 de dos maneras diferentes: con un bucle (utilizando `for`) y utilizando la función `prod`.

   Ayuda:

   ```
   producto<-1; for (i in 1:10) producto<- producto * i
   prod(1:10)
   ```

5. Construir una función que convierta kilómetros en millas y otra que convierta euros en dólares. ¿Qué mejoras piensas que podrías incluir a estas funciones?

   Ayuda:

   ```
   Convierte.km.a.millas <- function(km){km*0.62}
   Convierte.euros.a.dolares <- function(euros){euros*1.45}
   ```

6. Construir una función `mi.orden` que ordene un vector de forma ascendente o descendente según un argumento que nos lo indique (añadir los avisos necesarios para que funcione siempre).

   Ayuda:

   ```
   mi.orden <- function(vector, ordenascen=T) {
        if(is.vector(vector)== FALSE || is.numeric(vector)==FALSE)
             stop("El argumento debe ser un vector numérico")
        sort(vector, decreasing= !ordenascen) }
   ```

7. Construir una función que calcule los estadísticos básicos y un histograma de una variable continua, y otra que haga lo mismo sobre una variable discreta. ¿Sería posible hacerlo en una única? Intentarlo.

   Ayuda:

```
mi.resultado<- function(variable, tipo_var){
   switch(tipo_var,
         {hist(variable); summary(variable)},
         {barplot(table(variable)); table(variable)},
         stop("Segundo arg. debe ser 1(var. cont) ó 2(var. disc)")
         )
}
```

8. Construir una función `mi.factorial` que calcule el factorial de todos los elementos de un vector (añadir los avisos necesarios para que funcione siempre).

   Ayuda:

```
mi.factorial <- function(vector){
   if(sum(vector<0))
         stop("Las componentes del vector deben ser positivas")
   factorial(vector)
   }
```

9. Reescribir la función `ttest` descrita en las trasparencias de la sesión 4 para que cuando el usuario lo pida pueda realizar un diagrama de cajas de los dos grupos que se comparan.

```
ttest <- function(y1, y2, test="dos-colas", alpha=0.05, graf.caja=F)
{
lista.variables<- list(y1,y2)
if(graf.caja==T) boxplot(lista.variables)
n1 <- length(y1); n2 <- length(y2) ; ngl<-n1+n2-2
s2 <- ((n1-1)*var(y1) + (n2-1)*var(y2))/ngl
tstat <- (mean(y1) - mean(y2))/sqrt(s2*(1/n1 + 1/n2))
area.cola <- switch(test,
"dos-colas" = 2 * (1 - pt(abs(tstat),ngl)),
inferior = pt(tstat,ngl),
superior = 1 - pt(tstat,ngl),
stop("el test debe ser dos-colas, inferior o superior" )
)
list(tstat=tstat, gl=ngl,
rechazar=if(!is.null(area.cola)) area.cola<alpha,
area.cola=area.cola)
}
```

10. Escribir una función que a partir de un `data.frame`, haga una descriptiva numérica y gráfica de todas las variables (diferenciando el tipo de variable, categórica o cuantitativa) y que además haga una gráfica de dispersión por pares (utilizar función `pairs`) para las variables continuas.

    Solución:

```
descriptiva<-function(df){
if(!is.data.frame(df)) stop("Se requiere un data frame")
aux<-numeric(length(df))
for(i in c(1:length(df)))
     {
     if(is.factor(df[[i]]))
          {
          y<-table(df[i])
          pie(table(df[i]),main=names(df)[i])
          }
     else
          {
          y<-summary(df[i])
          hist(df[[i]],main=names(df)[i]) ; boxplot(df[i])
          aux<-c(aux,i)
          }
     print(y)
     }
pairs(df[aux])
}
# comprobación de uso:
datos <- data.frame(anyos=c(1.3,0.4,1.1,2.3,3.1,1.3),
      tipo=c(2,3,3,1,3,1),edad=c(22,21,34,42,17,43),
      sexo=c("H","M","H","H","M","H"))
descriptiva(datosimp)
data(PlantGrowth)
# sólo hay una variable continua, no hará pairs
descriptiva(PlantGrowth)
data(iris)
descriptiva(iris)
```