Trabajo Práctico 5: Introducción al lenguaje R- continuación

7. Programación de funciones

Hemos visto cómo utilizar scripts para ejecutar varios comandos. Veremos en lo que sigue cómo escribir funciones de manera de facilitar los pasos en un procedimiento.

7.1 Instrucciones de control de flujo

R incluye las instrucciones de control de flujo habituales de los lenguajes de programación. Estas incluyen (la sintaxis puede obtenerse mediante **`?Control`**):

```
if
if else
for
while
repeat
break
next
```

Muchas de estas instrucciones requieren la evaluación de una declaración lógica y estas declaraciones pueden ser expresadas mediante operadores lógicos

Operador	Significado
==	Igual a
!=	No igual a
<, <=	Menor que, menor o igual a
>, >=	Mayor que, mayor o igual a
&&	Y lógico (AND)
||	O lógico (OR)

Veamos algunos ejemplos.

Función **`if()`**:

```
> x <- rnorm(1)
> if(x > 2) print("Este valor es mayor que el 97.72 percentil")
```

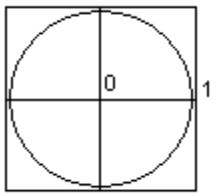
Si la expresión, que es el argumento de la función, es verdadera se ejecuta la instrucción que está escrita a continuación. Es decir, si el valor contenido en la variable **x** es mayor a 2, aparece en pantalla el texto que figura entre comillas como argumento de la función **print()**:

Función **`for():`** En el ejemplo siguiente indicamos primero que **`x`** y **`z`** son vectores. Luego los generamos de longitud 100 con contenido determinado por las asignaciones que

se encuentran entrellaves. Las llaves {} son necesarias pues se debe evaluar más de una expresión.

```
> n <- 50
>   x <-  vector( ) # inicializamos el vector
>   z <-  vector( ) # inicializamos el vector
>   for(i in 1:100) {
+   x[i] <- mean(rexp(n, rate = .5))
+   z[i] <- (x[i] - 2)/sqrt(2/n)
+   }
```

Función **`while():`** Construimos un estimador de Monte Carlo (MC) del número π.



El área del cuadrado es 4 y el del círculo inscripto es π (ver figura anterior). Por lo tanto, si generamos puntos al azar en el cuadrado de vértices (1,1), (1,-1), (-1,1) y (-1,-1), la probabilidad que uno de ellos caiga dentro del círculo con centro en el origen (0,0) y radio 1 es $p = \pi/4$. En el código escrito más abajo, **`n`** es la cantidad de puntos generados y **`s`** la cantidad de observaciones contenidas en el círculo unitario. Luego, **`s/n`** es la estimación por MC de la probabilidad *p* y **`4*s/n`** es la estimación por MC de π. Se detiene el procedimiento cuando estamos a lo sumo a una distancia de .001 del verdadero valor. Como se trata de una estimación por MC, tanto la estimación como la cantidad de pasos serán diferentes cada vez que se ejecuta el código.

```
eps <- 1; s <- 0; n <- 0 # initializa los valores
> while(eps > .001) {
+   n <- n + 1
+   x <- runif(1,-1,1)
+   y <- runif(1,-1,1)
+   if(x^2 + y^2 < 1) s <- s + 1
+   pihat <- 4*s/n
+   eps = abs(pihat - pi)
+   }
>   pihat # esta es nuestra estimación
[1] 3.141104
>  n # estos son la cantidad de pasos que se realizaron
[1] 326
```

7.2 Formato Función

Tal vez uno de los aspectos más poderosos del lenguaje R es que permite que el/la usuario/a escriba sus propias funciones. Muchos cálculos pueden incorporarse a una única función. A diferencia de los scripts, no se guardan las variables intermedias en el espacio de

trabajo. Además las funciones permiten que los valores de entrada puedan ser cambiados durante la ejecución de la función.

El formato general para crear una función es

```
nombrefun <- function(arg1, arg2, ...) { código de R }
```

En la instrucción anterior, **nombrefun** es cualquier nombre de objeto permitido y **arg1**, **arg2**, ... son los argumentos de la función. Como en cualquier función de R , pueden tener asignados valores por defecto. Cuando se escribe una función, es guardada en el espacio de trabajo como objeto "function".

Veamos un ejemplo simple de una función escrita por el usuario. Hemos agregado comentarios y espacio para destacar el primer **if** para facilitar la lectura de la función:

```
> f1 <- function(a, b) {
+   # esta función devuelve el máximo entre dos escalares
+   # y, en caso de igualdad, una declaración de que son iguales.
+   if(is.numeric(c(a,b))) {
+            if(a < b) return(b)
+            if(a > b) return(a)
+   else print("Los valores son iguales")
+   }
+   else print("No se permiten caracteres")
+   }
```

La función **f1** toma dos valores y devuelve una de varias posibilidades. Observe cómo funciona: antes de comparar los valores de **a** y **b** , primero determina si ambos son numéricos. La función **is.numeric()** devuelve **TRUE** si el argumento es un número real o un entero y **FALSE** en otro caso. Si esta condición es verdadera, se comparan los valores. En caso contrario da un mensaje de advertencia. Utilicemos la función con distintos argumentos:

```
> f1(2,3)
[1] 3
> f1(pi,exp(1))
[1] 3.141593
> f1(2,2)
[1] "Los valores son iguales"
> f1 ("Juan", "Laura")
[1] "No se permiten caracteres"
```

El objeto "function" **f1** permanecerá en el directorio de trabajo hasta que se lo remueva.

Si se escribe el nombre de la función sin los paréntesis

```
> f1
function(a, b) {
  # esta función devuelve el máximo entre dos escalares
  # y la declaración de que son iguales.
```

```
  if(is.numeric(c(a,b))) {
  if(a < b) return(b)
  if(a > b) return(a)
  else print("Los valores son iguales")
  }
  else print("No se permiten caracteres.")
  }
>f1
```

aparecen en la pantalla los comandos que la definen. Para modificarla se pueden usar los comandos **`fix()`** o **`edit()`** que vimos en la Sección 3.3.

Otro ejemplo (de *The New S Language*, by Becker/Chambers/Wilks, Chapman and Hall, London, 1988) . La fórmula siguiente se utiliza para calcular la cuota mensual (*c*) de una hipoteca:

$$c = M \frac{i/1200}{1-(1+i/1200)^{-12p}},$$

donde M es el monto otorgado, i es la tasa de interés y p es el plazo del préstamo. A continuación, se incluye el código de la función que calcula la cuota mensual:

```
 > hipoteca <- function(Monto = 100000, interés = 6, plazo = 30) {
+   cuota <- Monto*interés/1200/(1-(1+interés/1200)^(-12*plazo))
+   return(round(cuota, digits = 2))
+   }
```

Esta función toma tres argumentos, pero les hemos dado valores por defecto. Luego si evaluamos la función sin argumentos, calculará la cuota mensual para un préstamo de $100.000 con un interés de 6% anual y con unplazo de 30 años. La función **`round()`** se utiliza para obtener el resultado con dos decimales.

```
> hipoteca()
[1] 599.55

> hipoteca(200000,5.5) # usamos el plazo de 30 años por defecto
[1] 1135.58

> hipteca(plazo=15)    #error en el nombre de la función
Error: couldn't find function "hipteca"

> hipoteca(plazo=15)
[1] 843.86
>
```

7.4 Ejercicios

1. Escriba una función que por defecto genere $n = 20$ observaciones de una normal estándar y que también permita elegir el tamaño, la media y la varianza.

2. Escriba una función que
- genere una muestra con $n = 36$ observaciones de una normal con $\mu = 20$ $\sigma = 3$
- seleccione $m = 250$ muestras aleatorias (cada una de tamaño 20) con reemplazo de la muestra anterior
- calcule la media para cada una de esas muestras
- muestre el histograma de las 250 medias muestrales

3. Reescriba la función anterior de manera que los valores de n, m, μ y σ puedan ser elegidos cada vez que se ejecuta la función.

8. Métodos Estadísticos

R incluye una gran cantidad de métodos estadísticos y tests de hipótesis. Veremos los más utilizados.

8.1 Tests de t para una y dos muestras

La función principal que realiza este tipo de tests es **`t.test()`**. Produce tests de hipótesis e intervalos de confianza basados en el distribución *t*. Su sintaxis es:

```
Uso:

   t.test(x, y = NULL, alternative = c("two.sided", "less",
   "greater"), mu = 0, paired = FALSE, var.equal = FALSE, conf.level
   = 0.95)

Argumentos:

x, y: vectores numéricos que contienen los datos. Si y no está dado
        se realiza un test t para una muestra.

alternative: una cadena de caracteres especificando la hipótesis
     alternativa: `"two.sided"' (bilateral, valor por defecto),
     `"greater"'(mayor) o `"less"' (menor). Se puede especificar por
     la primera letra.


mu: un número especificanto el valor de la hip. nula (media ó
      diferencia de medias). El valor por defecto es 0.

paired: un valor lógico indicando si se quiere realizar un test t-
     apareado (por defecto se calcula el test t para muestras
     independientes).
```

var.equal: variable lógica utilizada sólo en el caso del test para muestras independientes. Si es `TRUE', se calcula la varianza pesada para estimar la varianza. Si es 'FALSE' (valor por defecto), se utiliza la sugerencia de Welsch para los grados de libertad utilizados.

conf.level: nivel de confianza del intervalo estimado para la media adecuado a la hip. alternativa especificada (por defecto es 95%).

(para el test con muestras independientes), variable lógica. Si es `TRUE', se calcula la varianza pesada para estimar la varianza. Si es 'FALSE' (valor por defecto), se utiliza la sugerencia de Welsh para los grados de libertad utilizados.

conf.level: nivel de confianza del intervalo (por defecto es del 95%) estimado para la media adecuada a la hip. alternativa especificada.

Este es el estadístico del test utilizado para el caso de dos muestras independientes:

$$t = \frac{(\bar{X}_1 - \bar{X}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}.$$

Ejemplo: Planteamos la hipótesis alternativa bilateral de que la altura media de los "black cherry trees" (cerezos) es distinta de 70 pies, utilizando el conjunto de datos **trees**

```
> data(trees)
> t.test(trees$Height, mu = 70)

        One Sample t-test

data: trees$Height
t = 5.2429, df = 30, p-value = 1.173e-05
alternative hypothesis: true mean is not equal to 70
95 percent confidence interval:
73.6628 78.3372
sample estimates:
mean of x
         76
```

Se rechaza la hipótesis nula.

Ejemplo: Los siguientes datos fueron obtenidos en http://cran.us.r-project.org/doc/contrib/Verzani-SimpleR.pdf y corresponden al tiempo de recuperación (en días) de 10 pacientes que toman una nueva droga y 10 pacientes que toman placebo. Queremos decidir si el tiempo medio de recuperación para pacientes que toman la droga es menor que para los que toman placebo. Los datos son:

Con droga: 15, 10, 13, 7, 9, 8, 21, 9, 14, 8

Placebo: 15, 14, 12, 8, 14, 7, 16, 10, 15, 12

Supondremos que las varianzas son iguales, aunque este supuesto debe validarse previamente. En R, este análisis se realiza así:

```
>  drug <- c(15, 10, 13, 7, 9, 8, 21, 9, 14, 8)
>  plac <- c(15, 14, 12, 8, 14, 7, 16, 10, 15, 12)
>  t.test(drug, plac, alternative = "less", var.equal = T)

            Two Sample t-test

     data:  drug and plac
     t = -0.5331, df = 18, p-value = 0.3002
     alternative hypothesis: true difference in means is less than 0
     95 percent confidence interval:
          -Inf 2.027436
     sample estimates:
     mean of x mean of y
          11.4      12.3

      >
```

No hay evidencia a nivel 0.05 para decidir que el tiempo medio de recuperación es diferente.

8.2 Regresión Lineal

Para ajustar un modelo lineal por cuadrados mínimos a un conjunto de datos se utiliza la función **lm()**. Esta función se puede utilizar para regresión lineal simple, regresión lineal múltiple y regresiones polinomiales.

Ejemplos:

```
> lm(y ~ x)                       # regresión lineal simple (RLS)
> lm(y ~ x1 + x2)                 # ajuste de un plano
> lm(y ~ x – 1)                   # RLS por el origen
> lm(y ~ x + I(x^2))              # modelo cuadrático
> lm(y ~ x + I(x^2) + I(x^3))   # modelo cúbico
```

En el primer ejemplo el modelo especificado por la fórmula **y ~ x** implica el ajuste entre **y** como variable dependiente o de respuesta y **x** como independiente o predictora. Para los ejemplos de las regresiones polinomiales, la función **I()** se usa para indicar a R que trate al argumento entre paréntesis "como si fuera" una variable (y no calcule esa cantidad).

La función **`lm()`**crea un objeto (*linear model object*) del que se puede extraer rica información.

Ejemplo: consideremos nuevamente el conjunto de datos **trees.** Ahora realizaremos un ajuste lineal utilizando la altura, **Height,** como variable independiente y el logaritmo natural del volumen, **log(Volume),** como variable dependiente. Hemos visto un diagrama de dispersión de estas dos variables en el Capítulo 4.

Estimaremos por cuadrados mínimos la recta de regresión que perrmite modelar log(Volume) como función de Height,

```
> attach(trees)
> ajuste <- lm(log(Volume) ~ Height)
```

El objeto **ajuste** es un objeto: *linear model*. Para ver qué contiene escribimos:

```
> attributes(ajuste)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"

>
```

Del objeto **ajuste**, podemos extraer por ejemplo los residuos del modelo contenidos en la variable **ajuste$residuals**.

Para obtener los estimadores de la pendiente y la ordenada al origen escribimos **ajuste** en la consola de comandos de R

```
> ajuste

Call:
lm(formula = log(Volume) ~ Height)

Coefficients:
(Intercept)       Height
   -0.79652      0.05354
```

El modelo de regresión ajustado tiene ordenada al origen (intercept) igual a -0.79652 y una pendiente (slope) 0.05354. Se puede obtener más información sobre el ajuste utilizando la función **summary()**:

```
> summary(ajuste)

Call:
lm(formula = log(Volume) ~ Height)

Residuals:
     Min       1Q   Median       3Q      Max
-0.66691 -0.26539 -0.06555  0.42608  0.58689
```

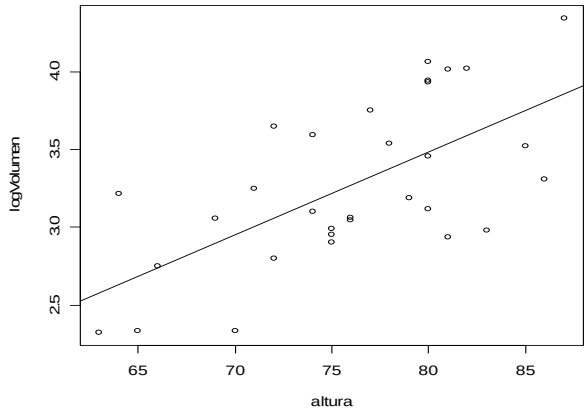
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.79652    0.89053  -0.894    0.378
Height       0.05354    0.01168   4.585 8.03e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4076 on 29 degrees of freedom
Multiple R-Squared: 0.4203,     Adjusted R-squared: 0.4003
F-statistic: 21.02 on 1 and 29 DF,  p-value: 8.026e-05
```

En la salida anterior obtenemos, entre otras cosas, estadísticos de los residuos, errores estándar de los estimadores y estadísticos de tests para los parámetros del modelo. Una llamada a **`anova(ajuste)`** mostrará la tabla de ANOVA para el modelo de regresión.

Finalmente podemos agregar la recta de regresión al diagrama de dispersión.

```
> logVolumen<-log(Volume)
> altura <- Height
> plot(altura, logVolumen)
> abline(ajuste) # también abline(-0.79652, 0.05354)
```



8.3 Análisis de la Varianza (ANOVA)

El ajuste de un modelo de Análisis de la Varianza es muy similar al visto para Regresión Lineal. Esto es natural, ya que ambos son modelos lineales. La forma más simple de ajustar un modelo ANOVA es utilizando la función **`aov()`** y el el tipo de modelo ANOVA se especifica en la fórmula. He aquí algunos ejemplos:

```
> aov(x ~ a) #  ANOVA de un factor
> aov(x ~ a + b) # ANOVA de dos factores sin interacciones
> aov(x ~ a + b + a:b) # ANOVA de dos factores con interacciones
> aov(x ~ a*b) # igual que el anterior
```

En las declaraciones anteriores, la variable **x** es continua y contiene *todas* las mediciones de un experimento de ANOVA. Las variables **a** y **b** representan los factores – contienen los niveles de los factores experimentales. Los niveles de un factor en R pueden ser *numéricos* (por ej. 1, 2, 3,…) o *categóricos* (por ej. bajo, mediano, alto, …), pero las variables deben guardarse como " *variables factor*". Veremos cómo se hace ésto.

8.3.1 Variables Factor

Como ejemplo de un experimento de ANOVA, consideremos el siguiente ejemplo sobre la fuerza de tres tipos de componentes de goma; se midió la fuerza de tensión (en kilos por cm cuadrado) para cada tipo en cuatro trozos de igual tamaño:

Tipo	*A*	*B*	*C*
Fuerza (kg/cm^2)	**180,185,176,175**	**179,190,185,188**	**197,200,199,194**

En R:

```
> Fuerza <- c(180,185,176,175,179,190,185,188,197,200,199,194)
> Tipo <- c(rep("A",4), rep("B",4), rep("C",4))
```

La variable **Tipo** especifica el tipo de goma y la variable **Fuerza** es la fuerza de tensión. En este momento, para R **Tipo** es una variable caracter (character). Queremos indicar que esas letras representan en realidad niveles de un factor en un experimento, para ello utilizamos el comando **factor():**

```
> Tipo <- factor(Tipo)
> Tipo
 [1] A A A A B B B B C C C C
Levels: A B C
```

Note que después de los valores que contiene la variable **Tipo,** aparece una lista con los niveles (**Levels)**. Para acceder directamente a los niveles en una variable de tipo factor directamente, se escribe:

```
> levels(Tipo)
[1] "A" "B" "C"
```

Con los datos en este formato es fácil realizar cálculos, con los datos contenidos en la variable **Fuerza** en los subrgupos determinados por los niveles de **Tipo**. Por ejemplo podemos calcular la media muestral en cada subgrupo:

```
> tapply(Fuerza,Tipo,mean)
    A     B     C
179.0 185.5 197.5
```

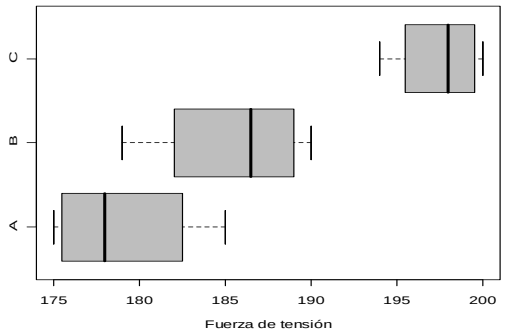
La función **tapply()** crea una *tabla* (*table*) con los valores resultantes de una función (en este caso **mean**) *aplicada* (*applied*) a los subgrupos definidos por el segundo argumento de tipo factor.

Para calcular las varianzas muestrales:

```
tapply(Fuerza,Tipo,var)
       A        B        C
20.66667 23.00000  7.00000
```

Podemos obtener boxplots paralelos especificando la relación en la función **boxplot()**:

```
> boxplot(Fuerza ~ Tipo, horizontal = T, xlab="Fuerza de tensión",
+                                                    col = "gray")
```



8.3.2 La Tabla de ANOVA

Para ajustar un modelo de ANOVA, especificamos el modelo de un factor en la función **aov()**:

```
> anova.fit <- aov(Fuerza ~ Tipo)
>   summary(anova.fit)
            Df Sum Sq Mean Sq F value    Pr(>F)
Tipo         2 704.67  352.33  20.862 0.0004175 ***
Residuals    9 152.00   16.89
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

8.4 Tests Chi-cuadrado

En R se pueden realizar tests de hipótesis para datos de conteo que utilizan el estadístico Chi-cuadrado de Pearson. Estos incluyen tests de bondad de ajuste y tests para tablas de contingencia. Se realizan utilizando la función **chisq.test()**. La sintaxis básica para esta función es (vea **?chisq.test** para información más detallada):

```
chisq.test(x, y = NULL, correct = TRUE, p = rep(1/length(x),length(x)))
```

Argumentos:

```
      x: un vector o  matriz.

      y: un vector; ignorado si 'x' es una matriz.

correct: un indicador lógico para la aplicación de la corrección por
         continuidad en el cálculo del estadísico del  test.

      p: un vector of probabilidades de la misma longitud que 'x'.
```

Veremos cómo utilizar esta función en las siguientes secciones.

8.4.1 Bondad de Ajuste

Para realizar un test Chi-cuadrado para evaluar un modelo probabilístico particular, el vector **x** debe contener la cantidad de observaciones en cada clase (si los datos son las observaciones crudas se debe utilizar la función **table()** para tabular las cantidades). El vector **p** debe contener las probabilidades asociadas con cada clase. El valor por defecto es suponer que las probabilidades de las clases son iguales.

Ejemplo: Se arroja un dado 300 veces y se observa:

Cara del dado	1	2	3	4	5	6
Frecuencia	43	49	56	45	66	41

Para testear si el dado es equilibrado podemos utilizar el test de bondad de ajuste con una probabilidad de 1/6 a cada celda:

```
> conteo <- c(43, 49, 56, 45, 66, 41)
> probs <- rep(1/6, 6)
> chisq.test(conteo, p = probs)

        Chi-squared test for given probabilities

data:  conteo
X-squared = 8.96, df = 5, p-value = 0.1107
>
```

La salida muestra el valor del estadístico del test, los grados de libertad y el p-valor.

8.4.2 Tablas de Contingencia

La forma más fácil de analizar una tabla de contingencia tabulada en R consiste en entrarla como una matriz (si tiene las cantidades crudas, es decir, a que categoría pertenece cada registro, se puede utilizar la función **table()**)

Ejemplo: Una muestra aleatoria de 1000 adultos fue clasificado de acuerdo al género y se era o no daltónico.

	Masculino	Femenino
Normal	442	514
Daltónico	38	6

```
>   visión <- matrix(c(442, 514, 38, 6), nrow=2, byrow=T)
>   visión
     [,1] [,2]
[1,]  442  514
[2,]   38    6
>
```

Asignamos nombres a las filas y columnas para que la matriz tenga un aspecto más parecido a una tabla:

```
> dimnames(visión) <-
list(c("normal","daltónico"),c("Masculino","Femenino"))
>


>  visión
          Masculino Femenino
normal          442      514
daltónico        38        6
```

Para testear si existe una relación entre el género y la incidencia de daltonismo utilizamos el estadístico Chi- cuadrado:

```
> chisq.test(visión, correct=F) # sin corrección

        Pearson's Chi-squared test

data:  visión
X-squared = 27.1387, df = 1, p-value = 1.894e-07

>
```

Al igual que en las funciones de ANOVA y modelo lineal, la función **`chisq.test()`** crea un objeto de salida del que puede extraerse la información. Por ejemplo:

```
salida <- chisq.test(visión, correct=F)
> attributes (salida)
$names
[1] "statistic" "parameter" "p.value"   "method"    "data.name"
"observed"
[7] "expected"  "residuals"

$class
[1] "htest"


> salida$expected            # estos son los valores esperados
```

```
           Masculino Femenino
normal        458.88   497.12
daltónico      21.12    22.88
```

8.5 Otros Tests

Hay muchos procedimientos estadísticos incluidos en R, la mayoría se utiliza en forma similar a la presentada en este capítulo. A continuación damos una lista y la descripción de los tests más comunes (utilice el help para más información):

- **`prop.test()`**
  Test para muestras grandes de una proporción ó para comparar dos o más proporciones que utiliza un estadístico chi-cuadrado.

- **`var.test()`**
  Test F para comparar las varianzas de dos muestras normales independientes.

- **`cor.test()`**
  Test que evalúa si la correlación entre dos vectores es significativa.

- **`wilcox.test()`**
  Tests no paramétricos para una y dos muestras. Los tests Suma de Rangos y de Rangos Signados de Wilcoxon.

- **`ks.test()`**
  Test para determinar si una muestra proviene de una distribucín especificada, o determinar si dos muestras provienen de la misma distribución.

Estas primeras 5 guías de R están basadas en
W. J. Owen 2006. **The R Guide**
http://cran.r-project.org/doc/contrib/Owen-TheRGuide.pdf