USO DE LA LIBRERÍA TCL/TK EN EL DESARROLLO DE UNA INTERFAZ GRÁFICA DE USUARIO (GUI) EN R

Ospina G. Johann A.*

*Escuela de Estadística, Facultad de Ingeniería, Universidad del Valle

e-mail: johann.ospina@correounivalle.edu.co

Abstract: The statistical software R has been designed as an object-oriented programing language, by offering through the package that incorporated Tcl / Tk and Java, multiple tools for building objects using a graphical interface. The commands of Tcl / Tk are embedded in R. This package creates simple functions called Widgets, windows, menus, dialog boxes and interactive charts that can help teaching statistics and the solution of specific problems, demos and examples. It is even used as a platform for the creation of new statistical packages such as Rcmdr. This paper presents the basic structure in the creation of a Graphical User Interface through implementation in the illustration of the behavior of the Binomial distribution, F distribution, simulation coverage level of confidence intervals and the effect of the Box-Cox transformation.

Keywords: Graphical User Interface, Tcl / Tk library, Widgets.

Resumen: El programa estadístico R ha sido pensado como un lenguaje orientado a objetos, ofreciendo mediante el paquete que incorpora la interface Tcl/Tk y Java, múltiples herramientas que permiten construir mediante una interfaz gráfica objetos. Los comandos de Tcl/Tk se encuentran embebidos en R. Este paquete crea simples funciones llamadas Widgets, ventanas, menús, cuadros de dialogo, gráficos interactivos que pueden ayudar a la enseñanza de conceptos estadísticos y a la solución de problemas específicos e incluso usados como plataforma para la creación de nuevos paquetes estadísticos como el Rcmdr. Este artículo presenta la estructura básica para la creación de una Interfaz Gráfica de Usuario (GUI), a través de su implementación en la ilustración del comportamiento de la distribución Binomial, la distribución F, la simulación del nivel de cobertura en intervalos de confianza y el efecto de la transformación de Box-Cox.

Palabras claves: Interfaz Gráfica de Usuario, Librería Tcl/Tk, Widgets.

1. INTRODUCCIÓN

Desde el pensamiento estadístico se ha visto un progreso continuo de múltiples técnicas para el análisis y desarrollo de soluciones a múltiples problemas, muchas de estas técnicas se desarrollaron teóricamente varias décadas atrás, pero solo hasta la llegada de los computadores, en especial el computador personal, ha permitido a investigadores y académicos acceder de manera práctica a estos métodos, sin embargo, todos estos avances tecnológicos se han visto igualados por una necesidad creciente de capacidad analítica que obliga a una nueva valoración del análisis de datos.

La llegada de sistemas para la visualización de gráficos interactivos y dinámicos fue un significativo hito en la computación estadística [Bowman et al., 2007]. Sin embargo, sólo un número relativamente pequeño de herramientas, tales como girar gráficos tridimensionales, han encontrado su camino en algunos paquetes estándar. La experimentación con nuevos métodos gráficos dinámicos ha limitado a usuarios que tienen la capacidad de programar en lenguajes como Java, C, etc. Por ejemplo, hay muchos ejemplos de Applets de Java que utilizan gráficos dinámicos. Sin embargo, la creación de este tipo de aplicaciones ha limitado a la comunidad estadística que necesariamente tiene que aprender a programar en otros lenguajes [Dalgaard, 2001b].

El desarrollo del programa R Core Team (2013), como un entorno estándar y universal de computación estadística, junto con sus herramientas de interacción con otros lenguajes, ha cambiado esta imagen. Siendo un desarrollo importante, el paquete tcltk, creado por [Dalgaard, 2001b], que proporciona un enlace de R con herramientas de control de una Interfaz Gráfica de Usuario por sus siglas en inglés (GUI) del sistema Tcl/Tk. Esto se ha vuelto más popular como un medio de control de funciones de R. Por ejemplo, [Fox, 2005] describe el sistema de R Commander que permite a los usuarios controlar las funciones a través de un amplio menú y sistema de diálogo, basado en el paquete tcltk.

Considerando lo anterior, este artículo pretende mostrar el uso de herramientas sofisticadas, para ilustrar conceptos dentro de la enseñanza estadística tales como: los comportamientos de las distribuciones, el concepto del intervalo de confianza y en el análisis de regresión. En la primera sección, se exhibe el concepto del lenguaje de programación Orientado a Objetos y como éste se combina mediante el paquete Tcl/Tk para crear una GUI. Posteriormente se explica su instalación y se introduce al entorno del paquete. Por último se expone como ejemplos, aplicaciones de la distribución en el comportamiento de la distribución Binomial, distribución F, la simulación del nivel de cobertura en intervalos de confianza y el efecto de la transformación Box-Cox sobre el modelo de regresión lineal.

* Estudiante Msc. en Estadística, Universidad del Valle. Estadístico, Universidad del Valle. Profesor Escuela de Estadística, Universidad del Valle. Profesor Departamento de Matemáticas, Universidad Autónoma de Occidente.

2. PROGRAMACIÓN ORIENTADA A OBJETOS

R como lenguaje Orientado a Objetos esconde bajo esta denominación la potencia del entorno. El término entorno lo caracteriza como un sistema completamente diseñado y coherente, antes que como una agregación incremental de herramientas específicas e inflexibles, como ocurre frecuentemente con otros programas de análisis de datos [Team et al., 2012].

A pesar que muchos usuarios no se identifican con los lenguajes de programación, cabe resaltar que R no es un lenguaje compilado como C++ ó Pascal, es un lenguaje interpretado como Java, esto significa que los comandos escritos con el teclado son ejecutados directamente, sin la necesidad de construir ejecutables. El concepto Orientado a Objetos indica que las variables, datos, funciones, resultados, etc., son almacenados en la memoria activa del computador, como un objeto de nombre específico, estos objetos pueden ser manipulados mediante operadores bien sea aritméticos, lógicos, comparativos ó funciones que a su vez son objetos y los productos de estas transformaciones o comparaciones son objetos también [Paradis, 2002].

Igualmente, R permite concebir objetos propios con atributos, clases, operadores y funciones a partir de objetos nativos, de tal manera que estos permitan contextualizar de una manera más precisa la realidad. Suponga, por ejemplo, que se está interesado en determinar de qué manera un conjunto de variables, posiblemente con relación de dependencia entre ellas, afectan la respuesta de otra variable; hasta aquí, simplemente con un Objeto denominado Modelo Lineal General se podría abstraer el problema, pero si a partir de este objeto creamos otro que además tenga el atributo de analizar mediante Componentes Principales las variables dependientes y devolver los scores, para así "correr" el modelo, se tendría un Objeto más preciso y consistente con la realidad.

3. TCL/TK: TOOL COMMAND LENGUAGE / TOOL KIT

Tcl (Tool Command Language) es un lenguaje de programación de comandos muy popular para el desarrollo de pequeñas aplicaciones en entornos UNIX. Permite programar de forma rápida y sencilla aplicaciones. Sin embargo, la velocidad de ejecución de éstas no será muy elevada ya que nos encontramos

ante un lenguaje interpretado y no ante un lenguaje compilado. Una característica muy importante de este lenguaje es la facilidad con la que se pueden añadir nuevos comandos a los ya existentes en el Tcl estándar [Bowman et al., 2007]. Estos nuevos comandos pueden implementarse utilizando el lenguaje de programación C e integrarse de manera sencilla en Tcl. Así, se han escrito bastantes extensiones para la realización de ciertas tareas comunes como, por ejemplo, OTcl (Tcl orientado a objetos). Tk es simplemente una de estas extensiones aplicada a la creación de interfaces gráficas de usuario [Dalgaard, 2001b]. Cabe anotar que Tcl posee todas las estructuras necesarias para la programación procedimental: Variables, Estructuras de control, Caracteres, Vectores, Arrays y Listas.

Una de las múltiples funcionalidades de Tcl/Tk son los denominados Widgets [Lawrence and Verzani, 2012], elementos que tienen el efecto de reducir el número de líneas de código, esta es una gran ventaja respecto a otras interfaces ya que agiliza el desarrollo de las aplicaciones. Es necesario hacer énfasis en que esta interfaz por sí sola tiene el poder de crear aplicaciones computacionales en cualquier área y es independiente al entorno de R [Dalgaard, 2001a].

Los modelos computacionales, modelos estadísticos en este caso, envuelven variedad de componentes que se relacionan entre sí y deben ser organizados de manera tal que se relacionen entre ellos de manera eficiente, sobre todo cuando se trabaja con otras aplicaciones. Lo anteriormente descrito, permite al usuario acceder a la creación de todos y cada uno de los elementos que componen una ventana, menús, cuadros de edición, listas, opciones, etc. [Ousterhout and Jones, 2009].

4. VENTAJAS DE LA CREACIÓN DE UNA INTERFAZ GRÁFICA DE USUARIO (GUI)

Los programadores siempre habían estado interesados en la creación de una GUI que permitiera una conexión con el usuario común de R, esto con el objetivo de crear aplicaciones personalizadas y portables para que usuarios con poca experiencia en el lenguaje pero que requieran modelar, personalizar, simular o probar modelos estadísticos a partir de paquetes del entorno R tuvieran la oportunidad de hacerlo.

Una perspectiva esquemática de como una GUI permite conjugar los elementos de un modelo computacional se muestra en la Figura 1, se observa como todos los componentes se entrelazan en la (GUI), existiendo una retroalimentación entre R, el código, los archivos, la documentación y las aplicaciones. En el caso de una encuesta, por ejemplo, tendría una GUI, que contendría los formularios. La base de datos se trataría mediante paquetes de R para MySQL, y con interfaces HTLM. El análisis se programaría en R y se convertiría en una serie de botones, opciones, o menús

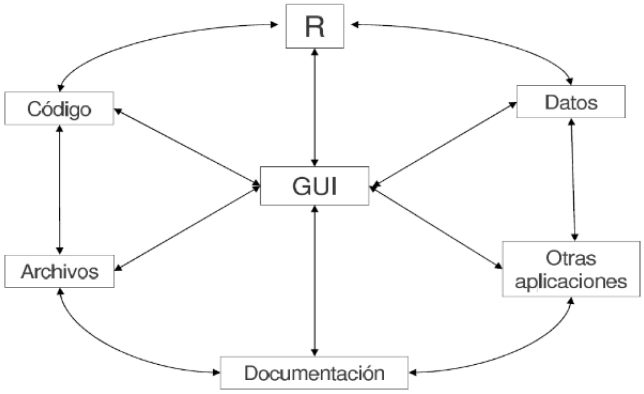

Figura 1. Estructura teórica de una Interfaz Gráfica de Usuario (GUI).

que permitiesen obtener los resultados deseados. Esto ayudaría a la reducción del tiempo de procesamiento de la información y la disminución de errores de digitación, lo cual permite almacenar información más precisa y consistente, proporcionando mayor confiabilidad en los resultados obtenidos.

En general una GUI permite:

- Desarrollar aplicaciones específicas a usuarios sin experiencia en R.
- Integrar en una sola aplicación, todos los componentes de un objeto estadístico.
- Tener mejor organizado todos los componentes necesarios para el desarrollo de una metodología de análisis de datos.
- Reducir los tiempos de procesamiento y obtención de resultados.

5. INTRODUCCIÓN AL PAQUETE TCLTK

El paquete tcltk de R es una interfaz que vincula y lenguaje vinculado a los elementos Tcl/Tk GUI. Este paquete provee acceso a la plataforma independiente de lenguaje Tcl, y a los elementos GUI de Tk mediante los denominados TkWidgets, ya que el lenguaje Tcl/Tk se encuentra embebido en R. Además permite configurar e inicializar el intérprete Tcl a través de funciones específicas de R. Así mediante un método de comunicación se definen pequeñas funciones que toman cadenas de R y pasan a Tcl para su ejecución. [Dalgaard, 2001b].

5.1 *Generación de Widgets*

Una importante diferencia entre los dos lenguajes es que Tcl usa sustitución de variables. Esto puede ayudar a crear una estructura de trabajo simple para

el desarrollo de aplicaciones más complejas. Dentro de las múltiples funciones que ofrece R para la creación de Widgets, existe una para cada elemento de una ventana, por ejemplo:

- `tkwidget(parent, type, ...)`
- `tkbutton(parent, ...)`
- `tkcanvas(parent, ...)`
- `tkcheckbutton(parent, ...)`
- `tkentry(parent, ...)`
- `tkframe(parent, ...)`
- `tklabel(parent, ...)`
- `tklistbox(parent, ...)`
- `tkmenu(parent, ...)`

El argumento parent hace referencia al contenedor donde va a estar el objeto Widget. El paquete tcltk permite crear botones, botones de opción, entradas, capas, listas, menús, etc. En este documento solo se presenta algunas aplicaciones básicas de estos elementos.

Existe una forma simple de vincular los comandos de Tcl y las funciones de R mediante el enlace de los correspondientes nombres en R y las opciones de Tcl. Los valores actuales, en su mayoría, son pasados como cadenas de texto después de la conversión con as.character, incluyendo caracteres especiales, dichos caracteres necesitan un tratamiento especial y deben ser convertidos al valor de su ID, es importante anotar que los argumentos NULL son convertidos a una cadena vacía, solo en algunos casos donde es necesario la opción explícita se puede pasar con name=NULL. Los vectores son convertidos en una cadena plana, es decir, que convierte cada vector en una cadena separada por espacios.

Casi todas las funciones en el paquete tcltk son creadas como objetos de la clase tkcmd. La excepción principal son los comandos que crean Widgets, ya que estos generan y retornan un objeto de clase tkwin.

5.2 *Variables de control*

Varios TkWidgedts pueden ser controlados por variables Tcl. Por ejemplo, un botón de chequeo puede ser configurado para prender o apagar mediante un click en la ventana con la configuración de una variable dicotómica (1 ó 0). A pesar que la correspondencia en R no es tan explícita ya que se debe utilizar el operador $ para hacer esta asignación a la variable Tcl. Este procedimiento es general para cualquier variable control [Fox, 2005].

5.3 *Eliminación de objetos*

Como R trabaja con la memoria activa, es decir, en la RAM, limitando en cierta parte algunos procesos. Cuando un objeto no es usado por largo tiempo, éste puede ser removido por el colector de basura. No obstante, cuando por ejemplo un objeto creado en el pasado es llamado a un Widget, R no podrá automáticamente tener conocimiento sobre qué función está aún en uso. Esto es posible corregirlo mediante un "Alias" para cada invocación del objeto almacenado en el objeto ventana al cual pertenece bajo un único nombre. Cuando una ventana es destruida con `tkdestroy`, ésta entrada en el entorno podría quedar almacenada en alguna parte. Este proceso asegura que la memoria no será copada rápidamente.

6. INSTALACIÓN DEL PAQUETE TCLTK

Para instalar el paquete, se descargar del Comprehensive R Archive Network (CRAN), si se posee una conexión de internet, en caso contrario, se puede instalar desde repositorios locales con archivos .zip y cargarlo mediante `library(tcltk)`. Este paquete requiere del intérprete de Tcl/Tk para poder lograr la interface, se recomienda el sistema Active Tcl 8.4.14.0, el cual es libre. Éste es un paquete de archivos binarios básico, pero bastante estable, con licencia tanto para uso comercial como no comercial, lo cual implica que no existe garantía ni soporte técnico, además, está habilitado para todas las plataformas (Windows, LINUX y OS/MAC) [Lawrence and Verzani, 2012].

7. APLICACIÓN

En esta sección se presentan cuatro aplicaciones para ilustrar el uso de los comandos. Para un mejor entendimiento se sugiere al lector ejecutar los códigos en la consola de R o RStudio.

7.1 *Aplicación 1: Distribución Binomial*

El siguiente ejemplo ilustra cómo va cambiando la forma de una variable que se distribuye Binomial, a medida que se va cambiando cada uno de sus parámetros. En la Figura 2 se observa los componentes generados por esta aplicación.

```
## Aplicación 1
## Instalación de páquetes
install.packages("tcltk")
install.packages("rpanel")
install.packages("TeachingDemos")

## Cargar páquetes
library(tcltk)
library(rpanel)
library(TeachingDemos)
```

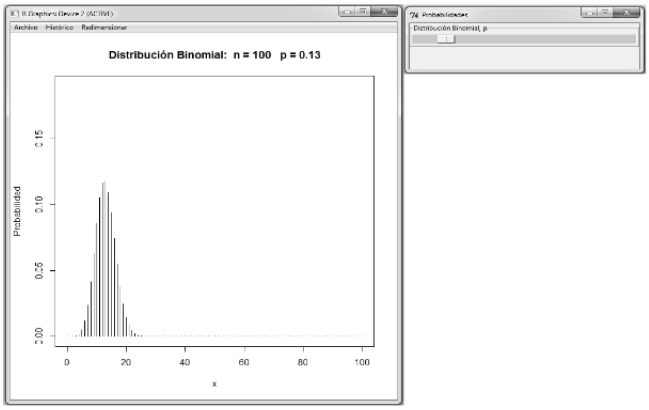

Figura 2. Interfaz Gráfica de Usuario (GUI) generada de la Aplicación 1.

```
#Función que calcula la distribución de probabilidad
#Binomial para una combinación de valores del parámetro
#"n" y "p".

plot.binomial <- function(panel) {
  with(panel, {
       probs <- dbinom(0:n, n, probabilidad)
       plot(c(0,n), c(0,1), type = "n", xlab = "x",
       ylab = "Probabilidad",ylim=c(0,.19),col=2)
       segments(0:n, rep(0, n+1), 0:n, probs)
       ptext <- as.character(round(probabilidad, 3))
       title(paste("Distribución Binomial:  n =", n, "
       p =", ptext))})
       panel}

#Crea el contenedor donde va a ir el widget que permite
#ir cambiando el parámetro "p" de la distribución Binomial.

binomial.panel <- rp.control("Probabilidades", n = 100,
probabilidad = 0.8)

#Waiget: Barra que permite variar ""p".
rp.slider(binomial.panel, probabilidad, 0, 1,action =
plot.binomial,title="Distribución Binomial, p:")
```

7.2 *Aplicación 2: Intervalos de Confianza*

La siguiente aplicación muestra la simulación de 100 intervalos de confianza generados con una media de 2 y una desviación estándar de 4 con un nivel de confianza del 95 %. En la Figura 3 se observa los resultados que se generan a partir de esta GUI.

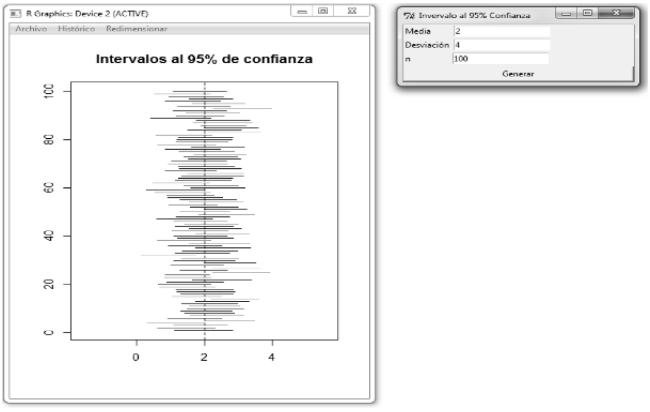

Figura 3. Interfaz Gráfica de Usuario (GUI) generada de la Aplicación 2.

```
## Aplicación 2

#Función que calcula los intervalos de confianza con un
#nivel de confianza del 95%.

ic.plot<-function(panel){

with(panel, {
n<-as.numeric(tm)
m<-as.numeric(m)
d<-as.numeric(d)
X<-matrix(rnorm(n*100, m, d), ncol=n)
Xm<-apply(X,1, mean)
Xd<-apply(X,1, sd)
li<-Xm-qt(.975, n-1)*(Xd/sqrt(n))
ls<-Xm+qt(.975, n-1)*(Xd/sqrt(n))

condicion<-((m>li) & (m<ls))
color<-rep("blue", 100)
color[!condicion]<-"green"

#Gráfica los intervalos
plot(m+c(-5,5)*d/sqrt(30),c(1,100), type="n", xlab=" "
, ylab=" ")
segments(li, 1:100, ls, 1:100, col=color)
abline(v=m, lty=2)
title("Intervalos al 95% de confianza")
})
panel
}

#Widget: Cuadro de dialogo donde se entre el número de
#intervalos a genera, la media y la desviación estándar.

ic.panel<-rp.control("Invervalo al 95% Confianza",
m=2,d=4,tm=100)
rp.textentry(ic.panel, m, title="Media ", action=ic.plot)
rp.textentry(ic.panel, d, title="Desviación",
action=ic.plot)
rp.textentry(ic.panel, tm, title="n ", action=ic.plot)
rp.button(ic.panel, title="Generar", action=ic.plot)
```

7.3 *Aplicación 3: Distribución F*

El siguiente ejemplo permite generar la densidad de una distribución F e ilustrar un valor observado específico. En la Figura 4 se observa los resultados que se generan a partir de esta GUI.

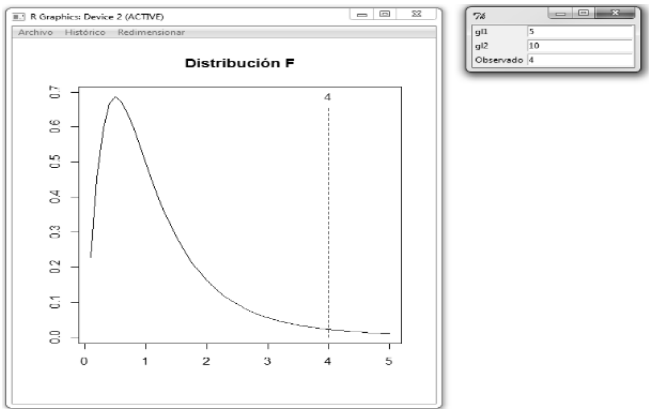

Figura 4. Interfaz Gráfica de Usuario (GUI) generada de la Aplicación 3.

```
## Aplicación 3

#Función que crea un gráfico de la distribución F.

if (interactive()) {
  plotf <- function(panel) {
    with(panel, {
pars   <- as.numeric(pars)
```

```
xgrid <- seq(0.1, max(c(pars[3], 5), na.rm
= TRUE), length = 50)
dgrid <- df(xgrid, pars[1], pars[2])
plot(xgrid, dgrid, type = "l", col = 1,
lwd = 1,xlab="", ylab="", main="Distribución F")
if (!is.na(pars[3])) {
  lines(rep(pars[3], 2), c(0, 0.95 * max(dgrid)),
  lty = 2, col = "red")
  text(pars[3], max(dgrid), as.character(pars[3]),
  col = "red")
  }
})
    panel
    }

#Widget: Cuadro de dialogo que llama la función plotf y
#tiene parámetros de entrada los grados de libertad.
   panel <- rp.control(pars = c(5, 10, NA))
   rp.textentry(panel, pars, plotf, labels = c("gl1",
   "gl2", "Observado"),
         initval = c(10, 5, 3))
   rp.do(panel, plotf)
   }
```

7.4 *Aplicación 4: Regresión Box-Cox*

La siguiente aplicación consiste en crear una ventana de los cuatro gráficos del análisis de residuales una regresión lineal simple, posteriormente se crea una barra de deslizamiento para ir cambiando el parámetro λ de la transformación de Box-Cox y observar cómo va cambiando el ajuste de los residuales de forma interactiva. En la Figura 5 se observa los resultados que se generan a partir de esta GUI.

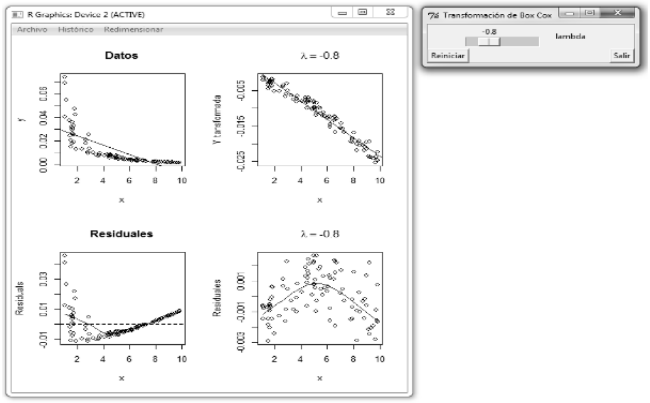

Figura 5. Interfaz Gráfica de Usuario (GUI) generada de la Aplicación 4.

```
## Aplicación 4

#Función que permite generar valores de una variable
#"x" y "y", luego se estable una ecuación funcional
#de y(x) = 3+2*x+e, donde "e" es error aleatorio que
#se distribuye N(0,1). Luego la función permite ir
#transformando "y" en terminos de potencias de 1
#hasta un parámetro lambda.

ejemplo.boxcoxfunction <-(lambda =
  sample(c(-1, -0.5, 0, 1/3, 1/2, 1, 2), 1))
{
x <- runif(100, 1, 10)
y <- 3 + 2 * x + rnorm(100)
if (min(y) <= 0)
y <- y - min(y) + 0.05
if (lambda == 0) {
y <- exp(y)
}
else {
y <- y^(1/lambda)
}

#Permite invocar una barra deslizadora para ir
#variando los valores de lambda. Luego se ajusta
#un modelo de regresión lineal simple, luego se
#calcula la transformación de Box-Cox con la función
#bct para los diferentes valores de lambda.

if (!exists("slider.env"))
slider.env <<- new.env()
library(TeachingDemos)
library(tcltk)
lam <- 1
assign("lam", tclVar(lam), env = slider.env)
bc.refresh <- function(...) {
lam <- as.numeric(evalq(tclvalue(lam), env =
  slider.env))
old.par <- par(mfcol = c(2, 2))
on.exit(par(old.par))
tmp1 <- lm(y ~ x)
tmp2 <- lm(bct(y, lam) ~ x)

#Gráfico de la relación "x" y "y"

plot(x, y, main = "Datos")
abline(tmp1)
scatter.smooth(x, resid(tmp1), main = "Residuales",
ylab = "Residuals")
abline(h = 0, lty = 2)

#Gráfico de la relación entre "x" y la variable
#"y" transformada.

plot(x, bct(y, lam), main = bquote(lambda == .(lam)),
ylab = "Y tansformada")
abline(tmp2)

#Gráfico de la suavización entre x y los valores de "y"
#transformados.

scatter.smooth(x, resid(tmp2), main = bquote(lambda ==
.(lam)), ylab = "Residuales")
}

#Widget: En esta secuencia de código se llama la barra
#deslizadora generada anteriormente y se incorporan
#elementos al cuadro de dialogo como el título, el tamaño
#de la ventana y se integran los botones de reiniciar
#o salir el proceso.

m<- tktoplevel()
tkwm.title(m, "Transformación de Box Cox")
tkwm.geometry(m, "+0+0")
tkpack(fr <- tkframe(m), side = "top")
tkpack(tklabel(fr, text = "lambda", width = "10"),
  side = "right")
tkpack(sc <- tkscale(fr, command = bc.refresh, from = -2,
to = 3, orient = "horiz", resolution = 0.1, showvalue = T),
side = "left")
assign("sc", sc, env = slider.env)
evalq(tkconfigure(sc, variable = lam), env = slider.env)

#Botón para reiniciar el proceso.
tkpack(tkbutton(m, text = "Reiniciar", command = bc.refresh),
side = "left")

#Botón para salir del proceso.
tkpack(tkbutton(m, text = "Salir", command = function()
  tkdestroy(m)),
side = "right")
}
```

8. CONCLUSIONES

El desarrollo de una GUI a partir del programa R ha proporcionado una amplia gama de ilustraciones donde la interacción gráfica puede utilizarse para potencializar el desarrollo de nuevas herramientas estadísticas.

El objetivo del uso del paquete tcltk es generar una interacción lo más accesible posible para los usuarios expertos y no expertos de R, proporcionando una forma simple y versátil para la ilustración de conceptos estadísticos y el desarrollo de aplicaciones especificas de acuerdo a las necesidades de cada usuario.

REFERENCIAS

[Bowman et al., 2007] Bowman, A., Crawford, E., Alexander, G., and Bowman, R. (2007). rpanel: Simple interactive controls for r functions using the tcltk package. *Journal of Statistical Software*, 17(9):1–18.

[Dalgaard, 2001a] Dalgaard, P. (2001a). A primer on the r-tcl/tk package. *R News*, 1(3):27–31.

[Dalgaard, 2001b] Dalgaard, P. (2001b). The r-tcl/tk interface. In *Proceedings of DSC*, page 2.

[Fox, 2005] Fox, J. (2005). Getting started with the r commander: A basic-statistics graphical user interface to r. *Journal of Statistical Software*, 14(9):1–42.

[Lawrence and Verzani, 2012] Lawrence, M. and Verzani, J. (2012). *Programming graphical user interfaces in R*. CRC Press.

[Ousterhout and Jones, 2009] Ousterhout, J. K. and Jones, K. (2009). *Tcl and the Tk toolkit*. Pearson Education.

[Paradis, 2002] Paradis, E. (2002). R for beginners.

[Team et al., 2012] Team, R. C. et al. (2012). R: A language and environment for statistical computing.