Laboratorio 1 Bioinformática - Introducción al uso de R

Manuel Villalobos Cid (manuel.villalobos@usach.cl)

05 de septiembre de 2016

1 Introduccion

R es un lenguaje y entorno de programación para análisis estadístico y gráfico, creado en 1993 por el Departamento de Estadística de la Universidad de Auckland. Al ser un proyecto de software libre, licencia GNU-GPL, se ha convertido en uno de los lenguajes más utilizados en investigación científica.

Debido a la inclusión de paquetes específicos, el uso de R se ha extendido en campos especializados como las ciencias biomédicas y bioinformática. Entre sus ventajas destaca la integración de funciones provenientes de diversas aplicaciones en un solo entorno, y la incorporación de alternativas a soluciones comerciales.

La primera actividad de laboratorio tiene como objetivo proporcionar a los alumnos las herramientas básicas para la utilización de R.

2 Instalación de R y RStudio

La versión basal de R trabaja mediante la utilización de comandos ingresados en una consola. Con objetivo de incorporar una interfaz más amigable, se ha diseñado un IDE denominado RStudio. Éste establece un entorno de trabajo similar al utilizado por Matlab (MATrix LABoratory).

2.1 Instalación de R

Windows

- Vaya a http://ftp.heanet.ie/mirrors/cran.r-project.org y seleccione el vínculo “Download R for Windows”.
- Diríjase a “Download R for Windows” y haga click el vínculo “base”.
- Seleccione la opción “Download R 3.X.X for Windows”, descargue e instale siguiendo las instrucciones que aparecerán en pantalla.

Ubuntu

- Para instalar la última versión de R en Ubuntu utilice como entrada:
  - sudo add-apt-repository “deb http://cran.rstudio.com/bin/linux/ubuntu $(lsb_release -cs)/”
  - sudo apt-key adv –keyserver keyserver.ubuntu.com –recv-keys E084DAB9
  - sudo apt-get update
  - sudo apt-get install r-base r-base-dev

2.2 Instalación de RStudio

- Diríjase a https://www.rstudio.com/products/rstudio/download/.
- Seleccione la última versión disponible de “RStudio Desktop” según su sistema operativo, descargue y siga las instrucciones que aparecerán en pantalla.

3 Descripción de entorno de trabajo en RStudio

RStudio ofrece un entorno de trabajo como el mostrado en la Figura 1.

El menú con las principales opciones se encuentra en la parte superior (A). La inclusión de código, diseño de funciones, scripts y archivos de texto se efectúan en la la ventana superior izquierda (B). La consola de comandos corresponde a la ventana inferior izquierda (C). Ésta última es equivalente al entorno visual de R basal.

Las ventanas del lado derecho efectúan labores de apoyo. La superior (D) permite acceder las variables y al historial, mientras que la inferior efectúa exploración: directorio de trabajo, archivos, el listado de paquetes instalados y tareas de visualización (E).

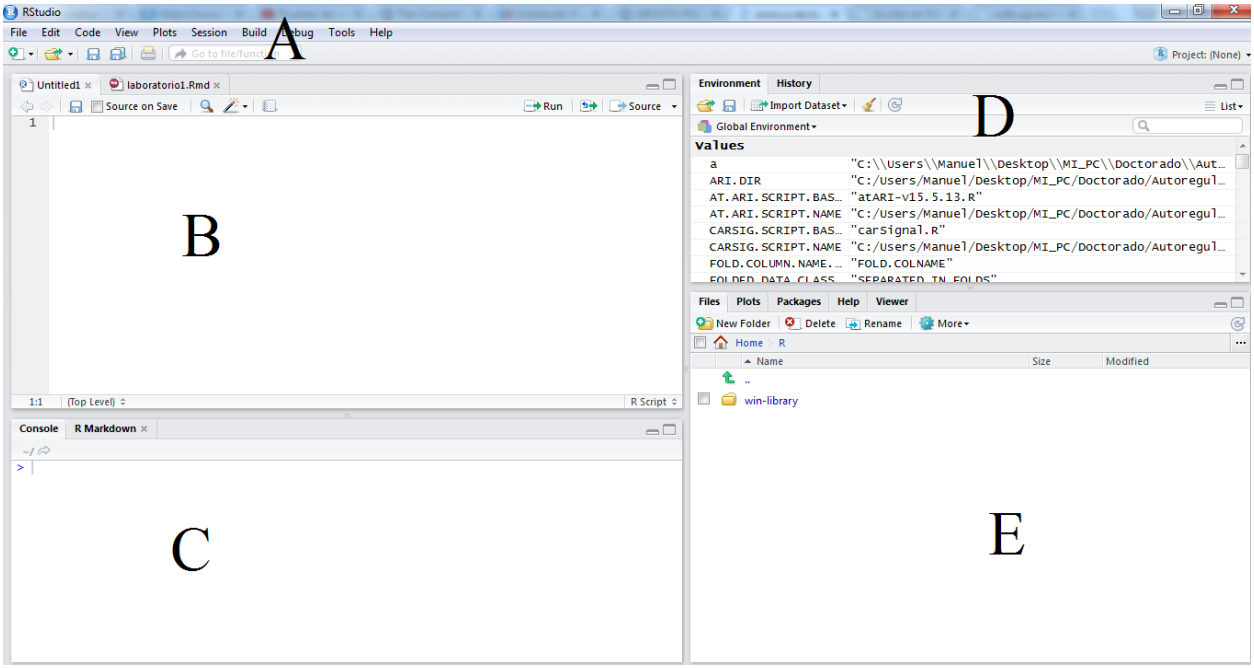


Figura 1. Entorno de trabajo en RStudio.

4 Actividad práctica de Laboratorio 1: introducción - nivel básico

Esta sección pretende ser una guía de aplicación para conocer el uso básico de R. Para ello se utilizarán ejemplos que pueden ser copiados directamente en la consola de comandos o en un script.

4.1 Definición de carpeta de trabajo

- Cree una carpeta en su escritorio denominada BIOLAB1.
- En RStudio, vaya al menú principal, seleccione FILE->NEW FILE->R-SCRIPT (CTRL+SHFT+N)
- En la ventana de su script (A) utilice la función **setwd** para seleccionar la carpeta de su proyecto y luego presione el botón RUN. También lo puede hacer seleccionado su carpeta en la ventana inferior derecha (E), ir a la opción "MORE" y escoger la opción "SET AS WORKING DIRECTORY".

```
setwd("C:/Users/Manuel/Desktop/MANUEL/UNIVERSIDAD/Bioinformática/Clases/Lab2015/BIOLAB1")
```

4.2 Variables y operaciones matemáticas básicas

Las siguientes secciones buscan que el alumno aprenda mediante el uso de ejemplos.

Variables

- En la ventana (A) defina dos variables y súmelas según el siguiente ejemplo. Ejecute su script usando el botón RE-RUN (CONTROL + SHIFT + ENTER).

```
#1.- Definición de variables
#============================
variable1<-1000
variable2=2000

#2.- Operación
#============================
salida=variable1+variable2
salida
```

```
## [1] 3000
```

Operaciones matemáticas

- Efectúe otro tipo de operaciones como división, multiplicación, logaritmo, raíz cuadrada, seno, u otra que estime conveniente.
- Escriba comentarios en su código utilizando el signo #. Ejemplo:

```
#1.- Definición de variables
#============================
variable1=1000
variable2=2000
variable3=9
variable4=sample(1:1000,1) #Generación de un número aleatorio entre 1 y 1000
variable5=180
variable6=TRUE

#2.- Operaciones
#============================

#Multiplicación
#----------------
s_mult<-variable1*variable2
s_mult
```

```
## [1] 2e+06
```

```
#División
#----------------
s_div<-variable2/variable1
s_div
```

```
## [1] 2
```

```
#Logaritmo base 10
#----------------
s_log10<-log10(variable1)
s_log10
```

```
## [1] 3
```

```
#Raiz cuadrada
#----------------
s_sqrt<-sqrt(variable3)
s_sqrt
```

```
## [1] 3
```

```
#Seno de 180 en radianes
#----------------
s_sinrad<-sin(variable5)
s_sinrad
```

```
## [1] -0.8011526
```

```
#Seno de 180 en grados
#----------------
s_singrad<-sin(variable5*pi/180)
s_singrad
```

```
## [1] 1.224606e-16
```

4.3 Vectores, matrices, listas y tablas

Vectores

- Los vecotres permiten agrupar elementos de un mismo tipo. Para crearlos, se debe utilizar la función combinar **c()**. Construya un vector con los primeros 5 números pares usando los siguientes comandos:

```
vector_par <- c(2, 4, 6, 8, 10)
vector_par
```

```
## [1]  2  4  6  8 10
```

- Puede acceder a los elemenos de cada vector utilizando un índice **[i]**. Reemplace el valor de alguno de los elemento del vector anterior. Ejemplo:

```
vector_par[5]<-12
vector_par
```

```
## [1]  2  4  6  8 12
```

Matrices

- Una matriz puede ser creada utilizando la funcion **matrix** y **c**. Los parámetros **ncol** y **nrow** permiten definir el número de filas y columnas. Cree una matrix de 5 x 2 utilizando el siguiente ejemplo:

```
datos<-matrix(c(6.2,3.4,5.1,7.2,5.3,50.3,33.2,45.3,60.3,47.8),nrow=5,ncol=2)
```

- Es posible asignar nombres a filas y columnas utilizando los comandos **colnames** y **rownames**. Agregue estos parámetros a la matriz anterior, suponiendo que ha registrado el peso y el tamaño de 5 bebés con 3 meses de edad. Por ejemplo:

```
colnames(datos)<-c("peso(Kg)","talla(cm)")
rownames(datos)<-c("SLOBODAN","ADOLF","KIM","FIDEL","AUGUSTO")
datos
```

```
##          peso(Kg) talla(cm)
## SLOBODAN      6.2      50.3
## ADOLF         3.4      33.2
## KIM           5.1      45.3
## FIDEL         7.2      60.3
## AUGUSTO       5.3      47.8
```

- Para efectuar operaciones matriciales dispone de varias sentencias, por ejemplo:

```
datos<-matrix(c(6.2,3.4,5.1,7.2,5.3,50.3,33.2,45.3,60.3),nrow=3,ncol=3)
print(datos)     #Permite visualizar la matriz
```

```
##      [,1] [,2] [,3]
## [1,]  6.2  7.2 33.2
## [2,]  3.4  5.3 45.3
## [3,]  5.1 50.3 60.3
```

```
t(datos)         #Genera la matriz transpuesta
```

```
##      [,1] [,2] [,3]
## [1,]  6.2  3.4  5.1
## [2,]  7.2  5.3 50.3
## [3,] 33.2 45.3 60.3
```

```
datos*datos      #Corresponde a la multiplicación elemento por elemento
```

```
##       [,1]    [,2]    [,3]
## [1,] 38.44   51.84 1102.24
## [2,] 11.56   28.09 2052.09
## [3,] 26.01 2530.09 3636.09
```

```
datos%*%(datos)  #Corresponde a multiplicación matricial
```

```
##        [,1]    [,2]    [,3]
## [1,] 232.24 1752.76 2533.96
## [2,] 270.13 2331.16 3084.56
## [3,] 510.17 3336.40 6084.00
```

```
det(datos)       #Calcula el determinante
```

```
## [1] -7178.06
```

```
solve(datos)     #Calcula la matriz inversa
```

```
##              [,1]        [,2]         [,3]
## [1,]  0.272914966 -0.17216351 -0.020924874
## [2,] -0.003623542 -0.02849516  0.023401866
## [3,] -0.020059738  0.03833069 -0.001167446
```

```
diag(datos)      #Calcula la diagonal de la matriz
```

```
## [1]  6.2  5.3 60.3
```

Listas

- Las listas permiten agrupar elementos de diversas caracterísitcas. Para utilizarlas se debe emplear la funcion **list**. Cree una lista usando el siguiente ejemplo.

```
lista <- list(nombre="diplodocups", genero="Saurópodos diplocínidos",
              antiguedad="150 Millones de años", largo_m=c(31.0, 36.5, 35.0, 20, 40))
```

- Para acceder a los atributo de una lista utilice el comando **attributes**.

```
attributes(lista)
```

```
## $names
## [1] "nombre"     "genero"     "antiguedad" "largo_m"
```

- Puede modificar los valores de una lista utilizando el símbolo **$**.

```
lista$nombre[1] = "Brachiosaurus";
lista$genero="Saurópodos  braquiosáuridos"
lista$largo_m[c(1,2,3,4,5)]  = 25.0;
lista
```

```
## $nombre
## [1] "Brachiosaurus"
##
## $genero
## [1] "Saurópodos  braquiosáuridos"
##
## $antiguedad
## [1] "150 Millones de años"
##
## $largo_m
## [1] 25 25 25 25 25
```

Tablas

- Es posible la creación de tablas combinando la **data.frame()** con **cbind** o **rbind**. A modo de ejemplo, la siguiente tabla muestra los cereales que han sido catalogados como alimentos transgénicos en nuestro país.

```
marcas<-c("BIOCENTURY","EL-GRANERO-INTEGRAL","GRANOVITA","PAGESA",
          "INTEGRAL-ESPIGAS","PASCUAL","SOJIVIT","HIPP","NUTREXPA",
          "NESTLÉ","KELLOGGS")

productos<-c("Bicentury","Todos","Todos","Diet_Rádisson","Todos",
             "Pascual/Essential/MásVital/ViveSoy","Todos","Todos","Cola-Cao",
             "Chocapic/Fitness/Fibre1/Estrellitas/Golden-Grahams/Crunch/Cheerios"
             ,"Todos")

transgenicos<-c("NO","NO","NO","NO","NO","NO","NO","NO","NO","SI","SI");

cereales = data.frame(cbind(marcas,productos,transgenicos))

head(cereales)
```

```
##               marcas                          productos transgenicos
## 1          BIOCENTURY                          Bicentury           NO
## 2 EL-GRANERO-INTEGRAL                              Todos           NO
## 3           GRANOVITA                              Todos           NO
## 4              PAGESA                      Diet_Rádisson           NO
## 5    INTEGRAL-ESPIGAS                              Todos           NO
## 6             PASCUAL Pascual/Essential/MásVital/ViveSoy           NO
```

- Este tipo de datos se puede almacenar en formato de texto con extensión ***.csv*** *o* **.xls**. Para ello se debe utilizar el comando **write.table**, donde el parámetro **row.names= FALSE** omite el uso de índices, **quote** evita que los números sean guardados en formato de texto, **dec** señala que el punto decimal será una coma y **sep** establece **";"** como caracter de separación. Guarde los datos usando la siguiente sentencia y revise el archivo en directorio correspondiente.

```
write.table(cereales,"cereales.csv", row.names = FALSE , quote = FALSE, dec = ",", sep = ";")
```

- Para leer archivos ***.csv** se utiliza la función **read.table**. Lea el archivo anterior.

```
nueva_tabla=read.table("cereales.csv", header = TRUE, sep = ";")
```

- También es posible guardar una tabla en formato ***.rda*** *o* **.rdata** utilizando la funcion **saveRDS**. Guarde según la siguiente sentencia y revise archivo en directorio correspondiente.

```
saveRDS(cereales, file = "cereales.Rdata")
```

- Para leer un archivo de estas características utilice la función **readRDS**. Lea el archivo.

```
rep_tabla<- readRDS("cereales.Rdata")
```

4.4 Estructuras de control: if, else, for, while.

Sentencia IF/ELSE

Se utiliza para comprobar una condición. Si ésta última es verdadera entonces se ejecuta determinada sentencia. En caso contrario, **else**, se realizará una sentencia diferente. Ejemplo:

```
mfARI=sample(1:9,1)
if (mfARI>=5) {
    print("El paciente tiene una autorregulación cerebral normal")
} else {
    print("El paciente tiene una autorregulación cerebral dañada")
}
```

Sentencia FOR

Implementa un bucle repitiendo un grupo de sentencias un número determinado de veces. Ejemplo:

```
for(i in 1:6) {print(i)}
```

Sentencia WHILE

Permite ejecutar repetidamente un bloque mientras una condición sea verdadera. Ejemplo:

```
i=1
while(i<=6){print(i);i<-i+1;}
```

4.5 Funciones, scripts y consola.

La capacidad de crear funciones propias es fundamental para estudios futuros y la optimización de código. Una función en R se define de la siguiente manera: nombre<-function{arg1,arg2,...}{expresion}

- Cree un nuevo script usando FILE->NEW FILE->R-SCRIPT (CONTROL+SHIFT+N). Desarrolle una función como la **definida** en el siguiente ejemplo. Posteriormente guarde el archivo con el nombre **sumar_num.R**.

```
#Definición de función
sumar_num<-function(x,y) {resultado<-x+y;}
```

A continuación se utilizará la consola de R/RStudio. Anter de cargar su script es recomendable eliminar las variables usadas anteriormente que no son de utilidad. El comando **ls()** permite chequear la tarea anterior.

```
rm(list=ls(all=TRUE))
ls()
```

```
## character(0)
```

- Si trabaja en RStudio puede cargar su script seleccionando en el menú FILE->OPEN FILE. Seleccione su archivo.R y en la ventana correspondiente presione RE-RUN (CRTL+SHFT+P). Si desea utilizar la consola de R base (también de RStudio) utilice la siguiente sentencia:

```
source("sumar_num.R")  #Revisar que el archivo se encuentre en la carpeta de trabajo
```

- La función **sumar_num** debe aparecer en la ventana de apoyo superior derecha (D). Puede revisar utilizando el comando **ls()**.

```
ls()
```

```
## [1] "sumar_num"
```

- Pruebe su función con distintas entradas siguiente el ejemplo planteado a continuación:

```
#Prueba de función
res_sum<-sumar_num(4,3)
res_sum
```

```
## [1] 7
```

```
res_sum2<-sumar_num(c(2,3),c(3,8))
res_sum2
```

```
## [1]  5 11
```

```
res_sum3<-sumar_num(matrix(c(1,1,1,1),nrow=2,ncol=2),matrix(c(2,2,2,2),nrow=2,ncol=2))
res_sum3
```

```
##      [,1] [,2]
## [1,]    3    3
## [2,]    3    3
```

4.6 Ayuda y descarga de paquetes

- Acceda a la ayuda de la función **log10** escribiendo la siguiente función en la consola o en un script (quite el símbolo de comentarios). Los resultados se verán en la ventana inferior derecha (D) de RStudio y/o en su explorador de referencia. Para limpiar la consola de trabajo (C) utilice CRTL+L.

```
#help("log10")
```

- Para instalar un paquete es necesario utilizar el comando **install.packages("paquete")**. Por ejemplo, si se desea instalar el paquete para gráficos avanzados denominado **ggplot2**, se debe utilizar el siguiente comando (quite el símbolo de comentarios).

```
#install.packages("ggplot2")
```

5 Actividad práctica de Laboratorio 1: estudio de tratamientos en pacientes con anorexia - nivel avanzado

Para efectuar este ejemplo se trabajará con datos reales disponibles en la librería MASS. El conjunto de datos a utilizar corresponde a **"anorexia"**. Se abordó un tema de estas caracterísiticas para no tratar una temática aún no cursada en la sección teórica de la asignatura. Fácilmente este ejemplo puede ser modificado con datos de expresión génica.

El conjunto compuesto por 72 filas y 3 columnas, presenta el peso de pacientes antes y después de efectuar dos tipos de tratamientos. La variable categórica es **Treat**, subdividida en: pacientes de control (CONT), tratamiento de conducta cognitiva (CBT) y tratamiento familia (FT). Las variables **Prewt** y **Postwt** cuantifican el peso en libras pre y post-tratamiento en forma respectiva.

Esta actividad consiste en evaluar si los tratamientos fueron efectivos comparando los pesos pre y post-tratamiento usando significancia estadística.

- Como primera actividad, se deben cargar y chequear los datos. Para ello se utilizarán las siguientes sentencias.

```
library(MASS)           #Carga librería con datos
data_axia<-anorexia     #Guarda el conjunto de datos de interés
head(data_axia)         #Presenta el encabezado y los primeros datos
```

```
##   Treat Prewt Postwt
## 1  Cont  80.7   80.2
## 2  Cont  89.4   80.1
## 3  Cont  91.8   86.4
## 4  Cont  74.0   86.3
## 5  Cont  78.1   76.1
## 6  Cont  88.3   78.1
```

- Posteriormente se separarán del conjunto de datos los pesos correspondientes a cada tratamiento. Para ello se utilizará la función **subset**.

```
data_axia_cont<-subset(data_axia, Treat == 'Cont')    #Datos pacientes control
data_axia_cbt<-subset(data_axia, Treat == 'CBT')      #Datos TTO. Conductual
data_axia_ft<-subset(data_axia, Treat == 'FT')        #Datos TTO. Familiar
```

- Seguidamente, sólo para tener una idea del comportamiento de los datos, se aplicará estadística descriptiva.

```
#install.packages("psych")
library(psych)
```

```
## Warning: package 'psych' was built under R version 3.2.5
```

```
#Estadística Control
describe(data_axia_cont,skew=FALSE)
```

```
##        vars  n  mean   sd  min  max range   se
## Treat*    1 26  2.00 0.00  2.0  2.0   0.0 0.00
## Prewt     2 26 81.56 5.71 70.5 91.8  21.3 1.12
## Postwt    3 26 81.11 4.74 73.0 89.6  16.6 0.93
```

```r
#Estadística CBT
describe(data_axia_cbt,skew=FALSE)
```

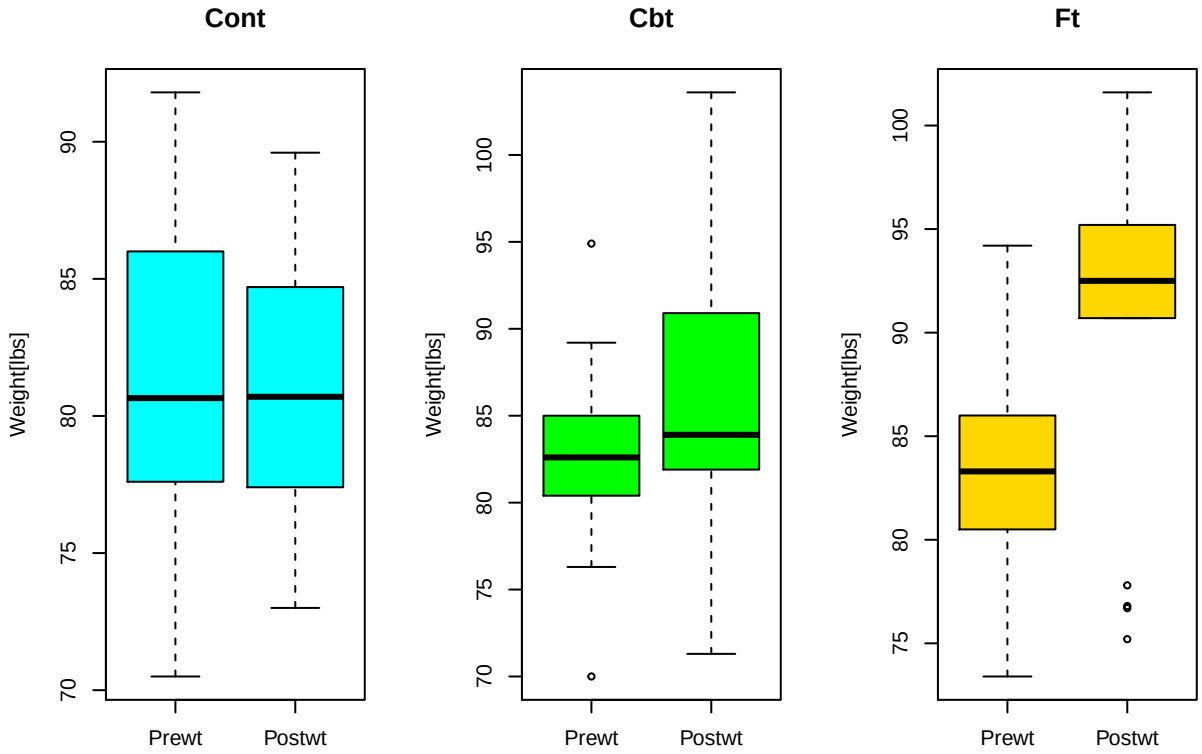
```
##        vars  n  mean   sd  min   max range   se
## Treat*    1 29  1.00 0.00  1.0   1.0   0.0 0.00
## Prewt     2 29 82.69 4.85 70.0  94.9  24.9 0.90
## Postwt    3 29 85.70 8.35 71.3 103.6  32.3 1.55
```

```r
#Estadística FT
describe(data_axia_ft,skew=FALSE)
```

```
##        vars  n  mean   sd  min   max range   se
## Treat*    1 17  3.00 0.00  3.0   3.0   0.0 0.00
## Prewt     2 17 83.23 5.02 73.4  94.2  20.8 1.22
## Postwt    3 17 90.49 8.48 75.2 101.6  26.4 2.06
```

- Antes de efectuar alguna conclusión se debe estudiar la varianza de los datos. En una primera instancia se utilizarán gráficos de caja:

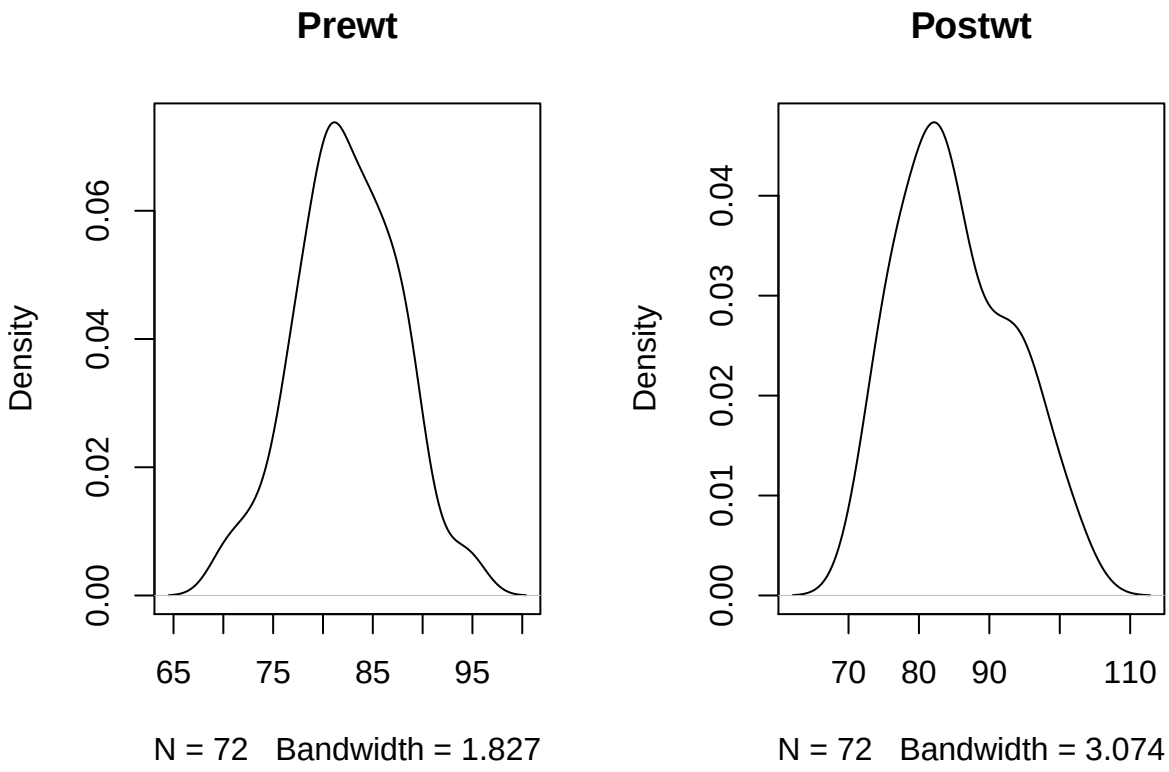
```r
par(mfrow=c(1,3))              #Define número de gráficos a imprimir en pantalla
boxplot(data_axia_cont[,2:3], main="Cont", ylab="Weight[lbs]", col="cyan")
boxplot(data_axia_cbt[,2:3], main="Cbt", ylab="Weight[lbs]", col="green")
boxplot(data_axia_ft[,2:3], main="Ft", ylab="Weight[lbs]", col="gold")
```



- A priori, parece que el tramiento familiar posee mejor resultado, sin embargo, es necesario evaluar cuantitativamente por medio de pruebas de hipótesis para comparación de grupos. La determinación de la técnica a usar, sea paramétrica o no paramétrica, dependerá de la distribución de los conjuntos de datos. Para esto se construirán gráficos de densidad en combinación a pruebas de normalidad usando la librería **nortest**.

La hipótesis nula asume distribución normal de los datos para todos los test.

```
par(mfrow=c(1,2))
plot(density(data_axia$Prewt), main="Prewt")
plot(density(data_axia$Postwt), main="Postwt")
```




```
#install.packages("nortest")
library("nortest")
p_Prewt<-0
p_Prewt[1]<-shapiro.test(data_axia$Prewt)$p.value #Shapiro -Wilk
p_Prewt[2]<-ad.test(data_axia$Prewt)$p.value     #Anderson-Darling
p_Prewt[3]<-cvm.test(data_axia$Prewt)$p.value     #Cramer-von Mises
p_Prewt[4]<-lillie.test(data_axia$Prewt)$p.value  #Lillie (Kolmo.-Smirnov)
p_Prewt[5]<-pearson.test(data_axia$Prewt)$p.value #Pearson chi-square test
p_Prewt[6]<-pearson.test(data_axia$Prewt)$p.value #Shapiro-Francia test
p_Prewt
```

```
## [1] 0.9483528 0.9352561 0.9231438 0.7978584 0.4685950 0.4685950
```

```
p_Postwt<-0;
p_Postwt[1]<-shapiro.test(data_axia$Postwt)$p.value       #Shapiro - Wilk
p_Postwt[2]<-ad.test(data_axia$Postwt)$p.value           #Anderson-Darling
p_Postwt[3]<-cvm.test(data_axia$Postwt)$p.value          #Cramer-von Mises
p_Postwt[4]<-lillie.test(data_axia$Postwt)$p.value       #Lillie(Kolm.-Smirnov)
p_Postwt[5]<-pearson.test(data_axia$Postwt)$p.value      #Pears.chi-square test
p_Postwt[6]<-pearson.test(data_axia$Postwt)$p.value      #Shapiro-Francia test
p_Postwt
```

```
## [1] 0.05780906 0.06273964 0.06034830 0.12772506 0.50093445 0.50093445
```

- Si se utiliza un umbral de confianza del 1%, ambas pruebas presentan un p-value mayor a esta cifra, lo que permite asumir una distribución normal para los datos de ambas variables y el uso de una técnica paramétrica de comparación.

En este caso se utilizará la prueba t-student y ANOVA ONE-way.

```
#Aplicación de t_student (Prueba de homogeneidad de varianza)
#Hipótesis nula indica que la distribución es igual
var_cont<-var.test(data_axia_cont$Prewt,data_axia_cont$Postwt); #p>0.01
var_cont
```

```
##
##  F test to compare two variances
##
## data:  data_axia_cont$Prewt and data_axia_cont$Postwt
## F = 1.4471, num df = 25, denom df = 25, p-value = 0.3617
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.6488219 3.2274009
## sample estimates:
## ratio of variances
##           1.447069
```

```
var_cbt<-var.test(data_axia_cbt$Prewt,data_axia_cbt$Postwt);    #p<0.01
var_cbt
```

```
##
##  F test to compare two variances
##
## data:  data_axia_cbt$Prewt and data_axia_cbt$Postwt
## F = 0.33659, num df = 28, denom df = 28, p-value = 0.005275
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.1580298 0.7169145
## sample estimates:
## ratio of variances
##          0.3365915
```

```r
var_ft<-var.test(data_axia_ft$Prewt,data_axia_ft$Postwt);       #p>0.01
var_ft
```

```
##
##  F test to compare two variances
##
## data:  data_axia_ft$Prewt and data_axia_ft$Postwt
## F = 0.35039, num df = 16, denom df = 16, p-value = 0.04327
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.1268894 0.9675449
## sample estimates:
## ratio of variances
##          0.3503872
```

```r
#Aplicación de t_student (Prueba de homogeneidad de varianza)
#(H0): No hay diferencia de pesos antes y despues de los tratamientos
#(H1): hay diferencia de pesos antes y despues de los tratamientos
#alfa=0,01
#Si p < 0.01 se rechaza la hipótesis nula
#Si p> 0.01 no se rechaza la hipótesis nula.

prueba_t_cont<-t.test(data_axia_cont$Prewt,data_axia_cont$Postwt,var.equal = TRUE)
prueba_t_cont
```

```
##
##  Two Sample t-test
##
## data:  data_axia_cont$Prewt and data_axia_cont$Postwt
## t = 0.30918, df = 50, p-value = 0.7585
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.473404  3.373404
## sample estimates:
## mean of x mean of y
##  81.55769  81.10769
```

```r
#como p-value = 0.7585 es mayor que 0,01 entonces No hay diferencia de pesos antes
#y despues en tratamiento control.

prueba_t_cbt<-t.test(data_axia_cbt$Prewt,data_axia_cbt$Postwt,var.equal = FALSE)
prueba_t_cbt
```

```
##
##  Welch Two Sample t-test
##
## data:  data_axia_cbt$Prewt and data_axia_cbt$Postwt
## t = -1.677, df = 44.931, p-value = 0.1005
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.6183919  0.6045988
## sample estimates:
```

```
## mean of x mean of y
##  82.68966  85.69655
```

```
#como p-value = 0.1005 es mayor que 0,01 entonces No hay diferencia de pesos antes
#y despues en tratamiento conducta cognitiva (CBT)

prueba_t_ft<-t.test(data_axia_ft$Prewt,data_axia_ft$Postwt,var.equal = TRUE)
prueba_t_ft
```

```
##
##  Two Sample t-test
##
## data:  data_axia_ft$Prewt and data_axia_ft$Postwt
## t = -3.0414, df = 32, p-value = 0.004673
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -12.130177  -2.399235
## sample estimates:
## mean of x mean of y
##  83.22941  90.49412
```

```
#como p-value = 0.004673 es menor que 0,01 entonces hay diferencia de pesos antes
#y despues en tratamiento familiar.
```

```
#Aplicación de ANOVA
#Hipótesis nula indica que la distribución es igual
p.cont <- aov(Prewt ~ Postwt, data=data_axia_cont, var.equal = TRUE); #p>0.01
```

```
## Warning in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...):
## extra argument 'var.equal' is disregarded.
```

```
summary(p.cont)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Postwt       1   21.2   21.22   0.642  0.431
## Residuals   24  793.0   33.04
```

```
p.cbt <- aov(Prewt ~ Postwt, data=data_axia_cbt, var.equal = FALSE);   #p<0.01
```

```
## Warning in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...):
## extra argument 'var.equal' is disregarded.
```

```
summary(p.cbt)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## Postwt       1  159.1  159.11   8.622 0.00671 **
## Residuals   27  498.3   18.46
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
p.ft <- aov(Prewt ~ Postwt, data=data_axia_ft, var.equal = TRUE);      #p>0.01
```

```
## Warning in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...):
## extra argument 'var.equal' is disregarded.
```

```r
summary(p.ft)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Postwt       1  116.6  116.64   6.117 0.0258 *
## Residuals   15  286.0   19.07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Los resultados de ANOVA indican que el tratamiento conductual (CBT) es el único que presenta cambios significativos en el peso de los pacientes, si se utiliza un intervalo de confianza menor al 1%. Por otro lado la prueba t indica que es el tratamiento familiar.

6 Actividad propuesta (opcional)

- Los gráficos de caja indican la presencia de valores atípicos (outliers) en CBT y TF. ¿Será que estos valores están alterando los resultados? ¿Al eliminarlos seguirá siendo efectivo sólo CBT?. Compruebe los resultados.