1er. COLOQUIO DE R

Algunos topicos en R

CONTENIDO

Explorando R

- Instalación del programa.
- La Consola, preparación del ambiente de trabajo
- Creación objetos
- Manejo de datos externos: Texto y Excel
- Creación de vectores y matrices
- Operaciones con vectores y matrices.
- Gráficos de líneas, barras, pie, histogramas,

Aplicación en analisis multivarial

- Análisis multivarial. Componentes principales
- Clasificación no supervisada: cluster análisis, dendrograma y consensus (métodos jerárquicos).
- Clasificación supervisada. Análisis discriminante k vecinos mas cercanos.

Tabla de codigos ASCII

Código	Símbolo
92	\
47	/
91	[
93	]
40	(
41	)
123	{
125	}
124	|
60	<
62	>
61	=

Código	Símbolo
34	"
126	~
58	:
59	;
64	@
94	^
35	#
36	$
37	%
38	&
39	'

Explorando R

Introducción

R es un lenguaje de programación funcional, de uso publico, disponible en Internet en la dirección: **http://www.r-project.org/**

El programa R tiene un modulo BASE, lo necesario para iniciar una sesión, el modulo BASE contiene las herramientas de programación, procedimientos estadísticos y gráficos frecuentemente utilizados. Adicional a este modulo, el proyecto R dispone de una cantidad muy grande de paquetes lo necesario para realizar un análisis estadístico. R también provee toda la documentación por manuales y libros de todos los paquetes propuestos.

El programa permite interactuar con el computador, pasos a paso, según los resultados que son mostrados en la consola, es decir que puede uno monitorear los procesos.

Una lista de ordenes pueden ser escritos en un archivo en ASCII con extensión .R y ser ejecutado mediante la orden: source("ejemplo.R"), si el nombre del programa es "ejemplo.R"

EL Programa R opera bajo diferentes plataformas: Windows, Linux, Unix y Mac. Las ordenes de ejecución de los procedimientos son las mismas en cada plataforma, la diferencia esta en la instalación y en algunas funciones de menú que presenta cada versión.

Descargar e instalar la base R en la plataforma Windows.

Ingresar a la dirección de Internet:

http://www.r-project.org/

En el menú Download, seleccione CRAN y una dirección cerca a su lugar de instalación, por ejemplo:

http://cran.at.r-project.org/ de Austria

Seleccione Windows (95 and later)

Luego dentro, seleccione BASE, que es la opción para el programa principal de R.

Finalmente encontrara una lista de información, en el cual esta el programa instalador; por lo general tiene un nombre específico:

R-2.6.0-win32.exe (es la versión que en el momento de editar este manual, se dispone en Internet).

Procedimiento a seguir para instalar el programa.

Localizar el programa en Internet:
Pagina web: http://www.r-project.org/

Buscar el programa R-2.6.0-win32:

- Seleccione DownLoad **CRAN**
- Seleccione un sitio, ejemplo: Australia**. http://cran.au.r-project.org/**
- La plataforma: **Windows**
- Seleccione: **base**
- Descargue el programa: R-2.6.0-win32.exe

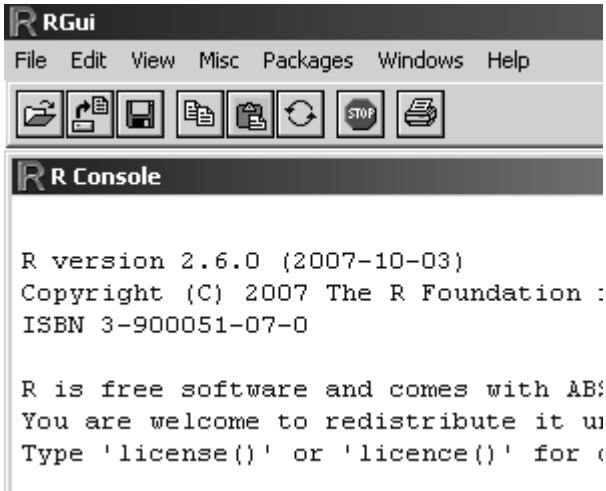
Una vez que lo tiene en le disco duro, ejecute este programa presionando el Icon:

El programa es la base del sistema R. Para tareas mas avanzadas requiere bajar las funciones de los contribuyentes (mas de 1200).

Entrada y salida del programa.

Cuando el programa esta bien instalado, se ejecuta el Icon de R que esta en el escritorio. El ambiente de trabajo se llama consola. Para salir del programa, utilice el menú o escriba la orden q() y presione Enter.

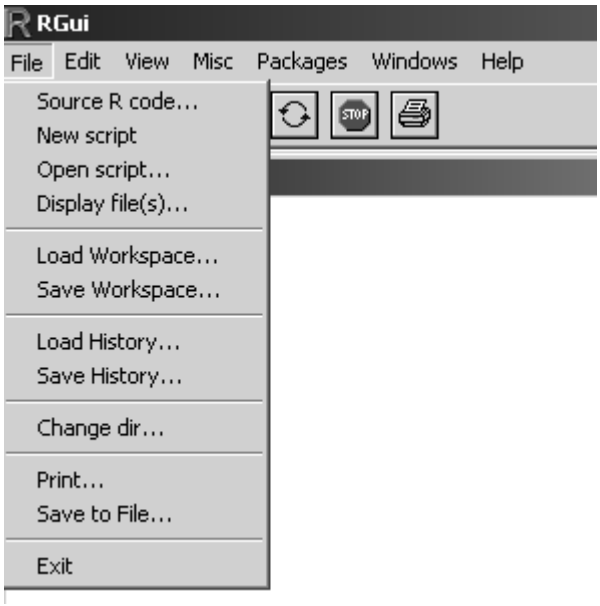
Al ejecutar, se ingresa a una ventana de trabajo llamada la consola, que es el lugar para ejecutar ordenes y a la vez recibir los resultados en texto, los cuales pueden ser enviados a un procesador de texto y los gráficos otra ventana que puede ser enviado a un archivo de gráficos o insertar en un documento.



Como se observa las opciones de menú son muy pocas:

File.- para editar o ejecutar un conjunto de ordenes, cambiar el directorio o salir del programa.
Edit.- para editar datos existentes creados por R
Misc.- para listar o remover objetos.
Packages.- es la opción mas importante, nos permite cargar un modulo de la biblioteca existente o cargar un modulo externo a nuestra base para trabajos futuros.
Windows.- manejar las ventanas activas.
Help.- Ayuda del programa.

Por ejemplo la opción File:



Las opciones New Script y Open Script son funciones para escribir y ejecutar procedimientos en R.

En general son pocas las opciones que ofrece el menú, esto significa que uno debe realizar todos los procesos en línea de comandos:

>

Este es el símbolo o señal para escribir una orden.

La Consola

Es el área de trabajo de R, lugar donde el usuario de R escribe las instrucciones necesarias para comunicarse con el sistema.

La señal de la línea de ordenes esta identificada con el símbolo “>”, si la orden es demasiado grande, presión enter y el sistema le indica con un ”+” para continuar escribiendo.

Es importante leer por lo menos una vez lo que aparece en la pantalla cuando ingresa al sistema “R”.
Es importante diferenciar entre ventana activa y no activa, la barra de titulo aparece iluminada si la ventana esta activa.

Escriba demo(graphics) y presione ENTER y continué presionando ENTER según lo solicite:

```
> demo(graphics)
```

Para visualizar mejor, ir a la opción del menú, seleccione windows y ejecute **Tile** para observar los gráficos paralelo a la ventana de la consola.

NOTA.- Usted observa dos ventanas; una es la consola y la otra es la ventana de gráficos. Estos resultados pueden ser exportados para cualquier sistema.

```
> help.start()
```

Es la ayuda en formato HTML de R. Los sistemas de ayuda son amplios en el sistema R. (Ayuda por manuales, Ayuda por Internet, Ayuda interactiva)

Como Crear Objetos

Los objetos en R esta formado por vectores, matrices, textos, tablas y todo elemento que puede ser manipulado por R. Un conjunto de objetos puede ser integrado en una lista como objeto de R.

El nombre de los objetos es muy sensible a mayúsculas y minúsculas, así como cualquier orden en R. Por ejemplo “print” existe pero “PRINT” no existe. Por lo general son minúsculas, en pocos casos se utiliza mayúsculas, para esto es necesario conocer la sintaxis apropiadamente.

Nuestro primer objeto: Rdto de papa en Kg. en 10 parcelas.

```
rdto <- c(12, 16, 9, 14, 8, 12, 11, 7, 15, 13)
rdto = c(12, 16, 9, 14, 8, 12, 11, 7, 15, 13)
c(12, 16, 9, 14, 8, 12, 11, 7, 15, 13) -> rdto
```

Creado el objeto, se puede hacer algunas estadisticas:

```
min(rdto)
max(rdto)
mean(rdto)
sd(rdto)
summary(rdto)
```

Lor datos pueden ser impresos y ordenados.

```
sort(rdto)
sort(rdto,decreasing = TRUE)
print(rdto)
```

Los datos pueden ser convertidos a matrices

```
fila <- rbind(rdto)
columna <- cbind(rdto)
fila
columna
```

Lista de objetos creados:

```
objects()
[1] "columna" "fila"    "rdto"
ls()
[1] "columna" "fila"    "rdto"
```

Remover objetos: rm

```
rm(fila)
rm(list=ls())
```

Limpiar la pantalla

Presione las teclas ctrl y "L" simultáneamente

Preparación del ambiente de trabajo: Crear el folder C:\COLOQUIO-R

Cada vez que ingrese a una nueva sesión de trabajo, localizar el lugar (fólder) para el manejo de su información.

Ubicación del área de trabajo

Get Working Directory: getwd()

```
getwd()
[1] "C:/Program Files/R/R-2.6.0"
```

Ir al forder c:\COLOQUIO-R

```
setwd("c:/COLOQUIO-R")
getwd()
[1] "c:/COLOQUIO-R"
```

O también:

```
setwd("c:\\COLOQUIO-R")
```

Algunas ordenes básicas:

```
list.files()       # Lista de archivos del área de trabajos
ls()               # Lista de Objetos: > ls()
x<-c("A","B","C") # crea el objeto x con elementos "A","B","C"
x                  # Muestra el contenido del objeto x
rm(x)              # Borra el objeto "x"
```

Revisión de los operadores aritméticos, lógicos, de relación y matriciales.

Ingrese help, html y en packages localice el modulo BASE.

Estudie los operadores, tome nota y realice operaciones creando objetos.

Operaciones Aritméticas:

a) Localice la simbología. > help("Arithmetic")

```
x <- 2*5/3; y <-2*2*2 ; v<- 2^3
z <- 13%/%4 ; w <- 13%%4 ; s<-c(1,2,3,4,5,6,7,8,9,10)
```

b) Revise el contenido de cada variable creada.

```
x ; y ; v ; z ; w ; s
```

c) Halle un nuevo objeto "p" de elementos {0,1} que represente a los valores pares e impares del objeto "s" ya creado.

```
p <- s%%2
p
```

Operadores logicos.

Revise la simbología y su uso > help("Logic")

```
x > y ;  z  < w ;  y==v
```

Problema: Se tiene una lista numerada de personas que practican diferentes deportes; b=basket, f=football y v=voley. Se quiere saber quienes y cuantos practican cada uno de los deportes.

```
Id <- c(1,2,3,4,5,6,7,8,9,10,11,12)
deporte <- c("f","b","f","v","b","v","b","f","v","b","v","b")

# formar una tabla y listar los que juegan basket

participan<-data.frame(Id=1:12,deporte)
selecto <- participan[,2]=="b"
participan[selecto,]
```

```
# Contabilizar
nrow(paricipan[selecto,])
```

```
Ejercicio: Contabilizar para los casos voley y fútbol
```

Secuencia de números.

Ejecute y describa cada resultado:

```
> 1:20
> seq(1,20,1)
> seq(from=1, to = 20, by=1)
> seq(0,1,length=20)
> seq(length=10, to = 20, from=5)
```

Halle las siguientes secuencias de números:

```
-  -3  -4  -5  -6  -7  -8  -9 -10 -11
-  -3 -1   1   3   5   7   9 11
-  3.0 3.2 3.4 3.6 3.8 4.0
-  20 18 16 14 12 10  8  6
```

Funciones estadística y matemática:

sum(x) : suma de los elementos del objeto "x"
mean(x) : media de los elementos del objeto "x"
length(x): longitud de los elementos del objeto "x"
var(x) : variancia muestral de los elementos del objeto "x"
sd(x) : desviacion estandar
min(x) : mínimo valor de "x"
max(x) : máximo valor de "x"
log(x) : logaritmo neperiano
log(x,2) : logaritmo base 2
exp(x) : exponente

Ejercicios desarrollados:

1. crear dos objetos y determinar cuantos y cuales son los valores comunes.

```
"x" contiene  { 5, 2, 7, 90, 24, 3, 6, 8,2, 5, 2}
"y" contiene  { 10, 7, 4, 7, 2, 10, 4, 3, 3, 1, 1, 3, 5}

x <- c(5, 2, 7, 90, 24, 3, 6, 8,2, 5, 2)
y <- c(10, 7, 4, 7, 2, 10, 4, 3, 3, 1, 1, 3, 5)
```

2. Identificar con verdadero o falso que elementos de "x" están en "y"

La orden x%in%y, indica cada elemento de "x" si esta en "y" con una respuesta TRUE o FALSE (verdadero o falso)

```
x%in%y
```

3. Identificar que elementos son:

`x[x%in%y]`, con esta instrucción se obtiene los elementos verdaderos, es decir los elementos de "x" que están en "y"

```
x[x%in%y]
```

4. Cuantos elementos son:

```
length( x[x%in%y] )   # cuenta los elementos.
```

5. Cuántos y cuáles son los valores únicos de ambos conjuntos.

unique(objeto) es la funcion para hallar los únicos valores que son distintos entre ellos.

```
unique(x[x%in%y])
length(unique(x[x%in%y]))
```

Cada resultado puede ser almacenado en un objeto

Ayudas a las órdenes estudiadas:

help(seq)
help(sum)
help(unique)
? mean
? var

Manejo de datos.

Los datos pueden ser creados en el ambiente R o en forma externa por Tinn-R, notepad, word, excel, etc u otros sistemas estadisticos como SAS, MiniTab, SPSS, etc. R puede importar y exportar de otros ambientes.

En el ambiente R, nosotros asignamos un nombre al objeto de datos, para luego registrar los datos según como deseamos.

Creando datos en R

Objeto: camote.
Campos: variedad, días acumulados y evaluación en materia seca.

```
camote <- edit(data.frame())
```

Información: Materia seca del follaje en camote cortes cada 90 días

variedad	dias	MS
ARB265	90	5
ARB265	180	13.1
ARB265	270	15.5
HELENA	90	2.3
HELENA	180	9.7
HELENA	270	12.8

Para modificar su contenido

```
> fix(camote)
```

Agregar los siguientes registros:

DLP3548	90	4.4
DLP3548	180	12.7
DLP3548	270	15.4

Observe los datos con las siguientes instrucciones:

```
camote
print(camote)
camote$variedad
camote$MS
camote[,1]
camote[2,3]
```

Algunas estadísticas de Materia seca

```
mean(camote$MS)
mean(camote[,3])
```

Formas directas para acceder a un archivo externo

Caso texto: Tinn-R, notepad, wordpad, word

azucar <- read.table("azucares en yacon.txt", header=T)

Caso dBase

Archivo: Bolivia.dbf

```
canal<-odbcConnect(dsn="dBase files")
bolivia <- sqlFetch(canal,"Bolivia")
odbcCloseAll()
```

Caso Excel

Archivo: Labranza.xls
Tabla de datos: datos

```
canal <- odbcConnectExcel("Labranza.xls")
labranza <- sqlFetch(canal,"datos")
odbcCloseAll()
```

Si en excel se dispone de una tabla de datos en una hoja, este se puede grabar en el disco duro como un archivo de tipo CSV, en R podemos leer con la opción:

```
labranza <- read.csv("Labranza.csv",header=TRUE)
```

Caso Access

Proyecto: papacip.mdb
Tabla de datos: Morfologia

```
canal <- odbcConnectAccess("papacip.mdb")
morfologia <- sqlFetch(canal,"Morfologia")
odbcCloseAll()
```

Listado y creación de nuevas tablas en Access.

Conectar R al sistema ACCESS, a la base de datos papacip.mdb

```
canal <- odbcConnectAccess("papacip.mdb")
```

Lista las tablas que existen

```
sqlTables(canal)[,c(4,3)]
```

La columna 4 es el tipo y la columna 3 es el nombre de la tabla.

Si en R se tiene algunas tablas (data.frame), estas pueden ser agregadas a la base de datos de access, activas en "canal".

Por ejemplo "bolivia" es una tabla existente.

```
sqlSave(canal,bolivia)
```

Liste nuevamente las tablas de la base de datos activa.

```
sqlTables(canal)[,c(4,3)]
```

Cierre todas las comunicaciones:

```
> odbcCloseAll()
```

Caso Minitab

Los datos almacenados por minitab deben ser en formato: Minitab Portable

Y su estructura debe ser numérica (no texto)

Para este ejercicio, considere el archivo "camote raices.MTP"

El procedimiento en R es:

```
> library(foreign)
> minitab<-read.mtp("camote raices.MTP")
```

Los datos leídos están en el objeto “minitab”, cuya estructura se puede visualizar con la orden:

```
> str(minitab)

> datos<-data.frame(minitab[1],minitab[2],minitab[3], minitab[4], minitab[5],
minitab[6], minitab[7], minitab[8])

> fix(datos)
```

Porque son 8 columnas

Caso SAS

Cuando uno dispone datos en SAS, estos deben ser almacenados en un formato de SAS, para esto se dispone del siguiente procedimiento en SAS para crear un archivo de datos portable.

```
/*
En SAS se prepara datos para R
*/
libname R sasv5xpt 'C:\COLOQUIO-R\maca.dat';
libname sitio 'C:\COLOQUIO-R';
DATA R.temp ;
set sitio.maca;
run;
quit;
```

Al correr este procedimiento en SAS, se genera automáticamente un archivo portable de nombre: "`maca.dat`", a partir del archivo de origen: `maca.sas7bdat`

Para leer en R, utilice el siguiente procedimiento:

```
> # Carga el paquete para la importación
> library(foreign)
> # lectura de los datos
> maca <- read.xport("maca.dat")
```

Caso SPSS

Cuando uno dispone datos creados por SPSS, estos pueden ser importados por R. El siguiente es un script escrito en Tinn-R:

```
# Lectura de un archivo SPSS
# archivo cip.sav
datos<-read.spss("Mouse survival.sav")
nuevo<-data.frame(datos)
# "nuevo" es la tabla que ahora contiene los datos del archivo SPSS
# para grabar el archivo en modo texto.
write.table(nuevo,"Mouse survival.txt",row.names=F,sep="\t")
# Este archivo puede ser leído desde Excel o de un editor de texto.
```

Algunas funciones que permiten resúmenes

Resumen estadístico de MS

```
resumen<-by(camote[,3], camote[,1], summary)
names(resumen)
resumen$"ARB265"
```

Promedio de MS por variedad

```
medias <- by(camote[,3], camote[,1], function(x) mean(x))
accesiones <-as.matrix(medias)
colnames(accesiones)<-c("promedio")
accesiones

library(agricolae)
attach(camote)
tapply.stat(variedad, MS, mean)
detach(camote)
```

Hallar la media, variancia, el mínimo y máximo valor por variedad. Usar Tinn-R

```
media <-tapply.stat(variedad, MS, mean)
variancia <-tapply.stat(variedad, MS, var)
minimo <-tapply.stat(variedad, MS, min)
maximo <-tapply.stat(variedad, MS, max)
estadisticas <- data.frame(variedad= media[,1],media=media[,2],
variancia=variancia[,2],minimo=minimo[,2],maximo=maximo[,2])
```

Grabación del objeto en archivo permanente en el disco duro

```
write.table(camote, file = "camote1.txt", sep = ",", row.name=FALSE)
write.table(camote, file = "camote2.txt", sep = "\t", row.name=FALSE)

Es equivalente a:

write.table(camote, "camote1.txt", sep = ",", row.name=F)
```

```
write.table(camote, "camote2.txt", sep = "\t", row.name=F)
```

La orden file.show permite visualizar los archivos grabados:

```
file.show("camote1.txt")
file.show("camote2.txt")
```

Creación de vectores y matrices

Borre todos los objetos

```
> rm(list=ls())
```

Crear un vector numérico de 10 elementos con ceros

```
> v<- rep(0,10)
```

Crear un vector de elementos { 2,4,3,6,5,8 }

```
> x <- c(2,4,3,6,5,8)
```

Crear una matriz de 4 filas y 5 columnas con ceros

```
> m <- rep(0,20)
> dim(m)<-c(4,5)
```

O también

```
> m<-matrix(m,4,5)
```

Crear la matriz A:

$$A = \begin{pmatrix} 3 & 1 & 5 & 4 & 2 \\ 1 & 6 & 7 & 5 & 3 \\ 5 & 3 & 8 & 4 & 1 \\ 2 & 1 & 4 & 9 & 2 \end{pmatrix}$$

Esta matriz tiene 4 filas y 5 columnas

Una forma puede ser:

```
> A <- c(3,1,5,2,1,6,3,1,5,7,8,4,4,5,4,9,2,3,1,2)
> dim(A)<-c(4,5)
```

También

```
> A<-matrix(A,4,5)
```

Juntando filas:

```
> A<-rbind(c(3,1,5,4,2),c(1,6,7,5,3),c(5,3,8,4,1),c(2,1,4,9,2))
```

Columnas

```
> A<-cbind(c(3,1,5,2),c(1,6,3,1),c(5,7,8,4),c(5,5,4,9),c(2,3,1,2))
```

Operaciones con vectores y matrices.

Considere el siguiente sistema de ecuaciones:

```
2a + 3b + 3c = 10
4a +  b + 2c = 11
 a + 6b +  c =  9
```

Para resolver el sistema de ecuaciones, escribimos el sistema en forma matricial: A x = b. Y mediante R, utilizando la función solve() podemos resolver.

A son los coeficientes del sistema, "x" el vector incógnita y "b" el vector del lado derecho.

```
> A <- rbind(c(2,3,3), c(4,1,2), c(1,6,1))
> b <- c(10, 11,9)
> x <- solve(A,b)
> x
```

Graficos

Funciones principales en r:

```
plot() generico
barplot (categorica) Barras en vertical y horizontal
pie()  (categorica) distribucion en torta
hist()  (continua)
pairs() (continua)
matplot() (continua)
boxplot() (continua por categorias)
```

```
ejemplo: Produccion de papa 2000-2006
```

Fuente: MINAG - DGIA

INDICADORES	2000	2001	2002	2003	2004	2005	2006
Producción (miles t)	3.274	2.680	3.297	3.141	2.918	3.290	3.224
Sup Cos (miles Has.)	284	234	271	275	235	264	259
Rendimiento (t/ha.)	11,5	11,5	12,2	11,4	12,4	12,5	12,4

```
year <- 2000:2006
produccion <- c(3.274, 2.680, 3.297, 3.141, 2.918, 3.290, 3.224)
cosecha <- c(284, 234, 271, 275, 235, 264, 259)
names(produccion)<- year
```

```
names(cosecha)<- year
barplot(produccion,col="yellow",ylim=c(0,3.5))
plot(year,produccion,type="b", col="blue", lwd=1.5, lty=4, ylim=c(0,4))
grid(col="brown")

indice<-barplot(produccion,col="yellow", ylim=c(0,4), axes=F,las=2)
axis(2,seq(0,4,0.5))
par(new=T)
barplot(cosecha,col=0, ylim=c(0,350),axes=F,border=0,las=2)
axis(4,seq(0,350,50))
points(indice,cosecha,pch=17,col="blue",cex=1)
lines(indice,cosecha,cex=1,col="blue")
legend ("top",c("produccion (miles ton.)", "cosecha(miles has.)"),
pch=c(22,17), col=c("black","blue"),lty=c(0,1), pt.bg=c("yellow","red"),
cex=1)
```

Produccion de papa 2006 por Dpto.

```
dpto <- c("Puno","Huanuco","Junin","La libertad", "Cusco", "Cajamarca",
"Ayacucho", "Lima", "Arequipa", "Apurimac", "Huancavelica", "otros")
prod<-c(14,13,10,9,9,9,6,5,5,5,4,11)
names(prod) <-dpto
pie(prod, main="Gráfico N°-: Producción de papa\npor Departamentos 2006" ,
cex.main=0.9 )
pie(prod,col=colors()[sample(1:657,12)])
title(main="Gráfico N°-: Producción de papa\npor Departamentos 2006" ,
cex.main=0.9 )

library(plotrix)
pie3D(prod,border=0,col=colors(),explode=0.1,labels=names(prod),labelcex=1)
title(main="Gráfico N°-: Producción de papa\npor Departamentos 2006" ,
cex.main=0.9 )
pie3D(prod,border=0,col=colors(),labels=names(prod),labelcex=1)
title(main="Gráfico N°-: Producción de papa\npor Departamentos 2006" ,
cex.main=0.9 )
```

Ejercicio 1

Leer el archivo "camote lamolina.xls" a R y seleccionar una nueva tabla correpondiente a los extremos de la medida Beta caroteno formado por los cuantiles 10 y 90.

```
setwd("C:/COLOQUIO-R")
library(RODBC)
canal <-odbcConnectExcel("camote lamolina.xls")
camote<- sqlFetch(canal, "camote")
odbcCloseAll()
str(camote)
minimo<-quantile(camote$BC,0.10,na.rm=T)
maximo<-quantile(camote$BC,0.90,na.rm=T)
A<-na.omit(camote[ camote$BC <= minimo, ])
B<-na.omit(camote[ camote$BC >= maximo, ])
selecto<-rbind(A,B)
```

Ejercicio 2

Construir un histograma, poligono de fecuencia, la tabla de frecuencia, La Ojiva y estadisticas de los datos agrupados de la variable Materia seca MS (7)

```
library(agricolae)
attach(camote)
H <- hist(MS, col="yellow", border="blue")
polygon.freq(H,col="brown")
table.freq(H)
ojiva.freq(H,type="b",col="blue")
stat.freq(H)
```

Ejercicio 3

Probar la normalidad de la respuesta Protein y graficar mediante cuantiles de la normalidad.

```
shapiro.test(Protein)
qqnorm(Protein)
qqline(Protein)
# probar si la materia seca (MS) sigue una distribución normal
```

Ejercicio 4

Hallar las correlaciones de beta caroteno con las otras medidas en la población y en las subpoblaciones de bajo y alto valor de caroteno.

```
# Poblacion
correlation(BC, camote[,c(-1,-6)])
correlation(A$BC, A[,c(-1,-6)])
correlation(B$BC, B[,c(-1,-6)])
```

Ejercicio 5

Seleccione una muestra aleatoria de 30 clones y halle las estadisticas de la materia seca y compare con las estadisticas de la población.

Medidas: Media, Mediana, Simetría, curtosis, desviación estándar, desviación media, rango y la correlacion con el Beta caroteno.

```
# Población camote de 206 clones.
# Muestra de 30 clones
detach(camote)
muestra <- sample(1:206, 30)
selecto <- camote[muestra,]
attach(selecto)
media <- mean(MS, na.rm=TRUE)
mediana <- median(MS, na.rm=TRUE)
skewness(MS)
kurtosis(MS)
sd(MS, na.rm=TRUE)
sqrt(sum(abs(MS - media),na.rm=TRUE)/30)
rango <- max(MS)-min(MS)
correlation(MS, BC)
detach(selecto)
```

Ejercicio 6

Con la población halle el Scatterplot para mostrar la asociación y el histograma del las medias BC, Protein, Fe, Zn, Ca que corresponden a las columnas: 6, 2, 3, 4 y 5.

```
# Población camote
source("panel.R")
camote1<- na.omit(camote)
pairs(camote1[,c(6,2:5)], panel=panel.smooth, cex = 0.8, pch = 19,
bg="light blue",  diag.panel=panel.hist, cex.labels = 1, font.labels=1)
pairs(camote1[,c(6,2:5)], lower.panel=panel.smooth, upper.panel=panel.cor)
```

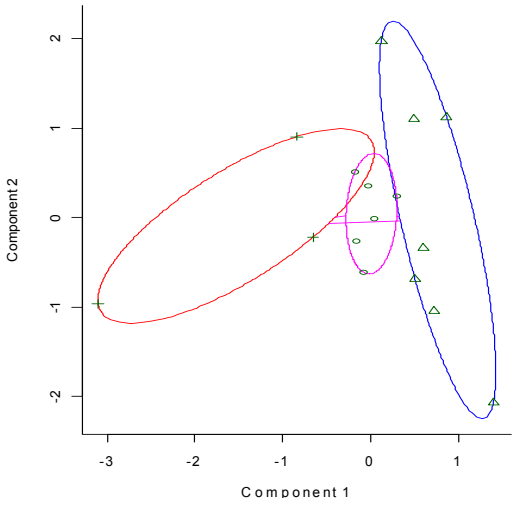
Ejercicio 7

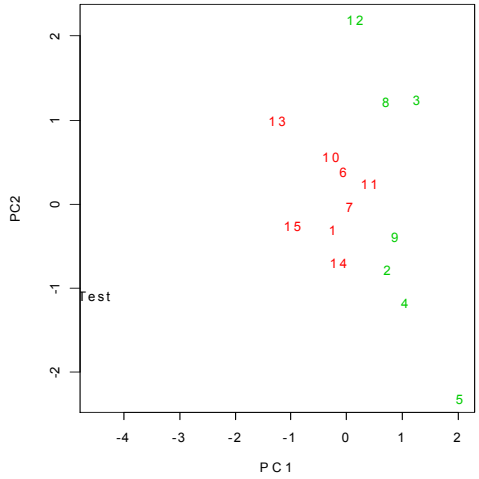
Con la población de camote1 asigne clases a la proteina y hierro según los percentiles, ejemplo: para proteina: p25 "Bajo", p75 "Alto", en otro caso "Medio"

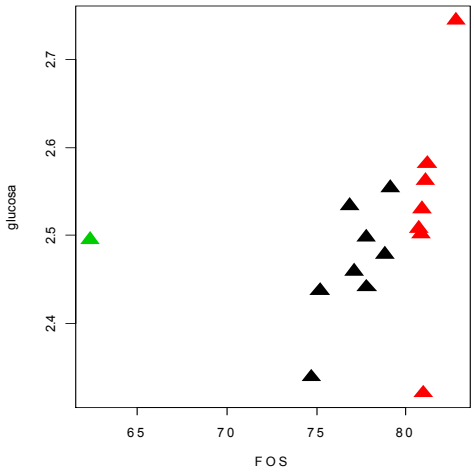
```
#proteina
camote2 <-data.frame(camote1, clase.Protein="medio", clase.Fe="moderado",
clase.BC="regular")
P25 <- quantile(camote2[,2],0.25)
P75 <- quantile(camote2[,2],0.75)
F25 <- quantile(camote2[,3],0.25)
F75 <- quantile(camote2[,3],0.75)
B25 <- quantile(camote2[,6],0.25)
B75 <- quantile(camote2[,6],0.75)

camote2[,8]<-as.character(camote2[,8])
camote2[,9]<-as.character(camote2[,9])
camote2[,10]<-as.character(camote2[,10])
camote2[camote2[,2] <= P25,8]<-"bajo"
camote2[camote2[,2] >= P75,8]<-"alto"
camote2[camote2[,3] <= F25,9]<-"poco"
camote2[camote2[,3] >= F75,9]<-"suficiente"
camote2[camote2[,6] <= B25,10]<-"malo"
camote2[camote2[,6] >= B75,10]<-"bueno"
matriz<- table(camote2[,10], camote2[,8])
matriz
matriz<-matriz[,c(2,3,1)]
matriz<-matriz[c(2,3,1),]

barplot(matriz, xlab="Proteina",col= c(3,5,7))
legend("topleft",c("malo", "regular", "bueno"), pch=22, pt.bg=c(3,5,7),
title="BC",cex=0.8)
title(main="Relacion del Betacaroteno y Proteina\nen clones de
camote",cex.main=0.8)

barplot(matriz, xlab="Proteina",col= c(3,5,7), beside=T)
legend("topleft",c("malo", "regular", "bueno"), pch=22, pt.bg=c(3,5,7),
title="BC", cex=0.8)
title(main="Relacion del Betacaroteno y Proteina\nen clones de
camote",cex.main=0.8)
```

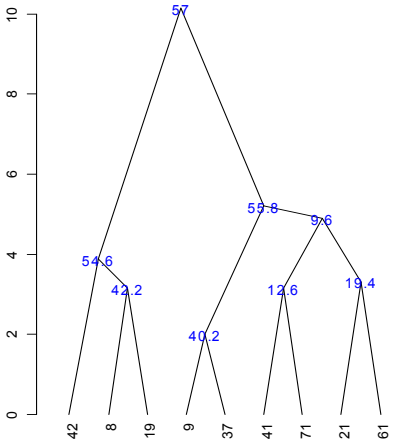
Analisis Multivarial

- Analisis multivarial. Componentes principales
- Clasificacion no supervisada: cluster analisis, dendrograma y consensus (metodos jerarquicos).
- Clasificacion supervisada. Analisis discriminante.y k vecinos mas cercanos.









Componentes principales

R dispone de funciones para hallar las componentes principales a partir de correlaciones o covariancias de las variables observadas.

Para este estudio considere el siguiente archivo: “azucares en yacon.txt”

FOS, glucosa, fructosa y sacarosa

Una variable adicional que es la identificación de cada registro (ID)

```
> azucar <- read.table("azucares en yacon.txt",header=TRUE)
> datos  <- azucar[,-1]
> correl <- cor(datos)
> valores <- eigen(correl)
> valores

$values
[1] 2.14206323 1.24405952 0.59276138 0.02111586

$vectors
           [,1]        [,2]       [,3]        [,4]
[1,]  0.6688031  0.10715436  0.1680706  0.71622111
[2,]  0.3107702 -0.62950240 -0.7115513 -0.02904034
[3,] -0.1326303 -0.76885749  0.6184480  0.09375192
[4,] -0.6622186 -0.03320897 -0.2880435  0.69093745
```

Para hallar los valores correspondientes a cada componente principal, es necesario primero estandarizar los datos, en este caso “datos”. Si usamos la función estándar descrita anteriormente (Pág. 34). Las 4 columnas se pueden estandarizar

```
> a1 <- estandar(datos,1)
> a2 <- estandar(a1,2)
> a3 <- estandar(a2,3)
> a4 <- estandar(a3,4)
```

Entonces se forma una matriz A.

```
> A <- as.matrix(a4)
```

```
Las ponderaciones de las componentes se encuentran en el objeto valores,
entonces obtenemos la matriz X:
```

```
> X <- valores$vectors
```

```
Las componentes principales se obtienen multiplicando la matriz A por X.
```

```
> CP1 <- A%*%X
```

Utilizando la matriz variancia-covariancia

```
> covar <- cov(datos)
> valores <- eigen(covar)
> valores

$values
[1] 31.054475402  0.400136559  0.096671548  0.007448548

$vectors
             [,1]        [,2]       [,3]        [,4]
[1,]  0.854321447  0.47752777 -0.2042005 -0.02010622
[2,]  0.005110203 -0.01540644 -0.1124739  0.99352209
[3,] -0.013639922 -0.37397769 -0.9207939 -0.10996962
[4,] -0.519540858  0.79490286 -0.3127147 -0.02040282
```

El aporte de cada componente se puede hallar según la proporción respecto al total.

En el caso de correlación:

```
> valores$value*100/sum(valores$value)
[1] 53.5515808 31.1014881 14.8190346  0.5278965
```

En covariancia:

```
> valores$value*100/sum(valores$value)
[1] 98.40216440  1.26791076  0.30632266  0.02360218
```

Y las componentes pueden ser calculadas como:

```
> X <- valores$vectors

Las componentes principales se obtienen multiplicando la matriz A por X.

> CP2 <- A%*%X
```

Otra función que permite hallar las componentes principales es: princomp(). Si usa correlaciones el parámetro cor=TRUE, si usa covariancias es cor=FALSE.

```
> componentes<-princomp(datos, cor=TRUE)
> summary(componentes)

Importance of components:
                          Comp.1    Comp.2    Comp.3      Comp.4
Standard deviation     1.4635789 1.1153742 0.7699100 0.145312978
Proportion of Variance 0.5355158 0.3110149 0.1481903 0.005278965
Cumulative Proportion  0.5355158 0.8465307 0.9947210 1.000000000
```

```
> names(componentes)
[1] "sdev"     "loadings" "center"   "scale"    "n.obs"    "scores"   "call"
```

Cada objeto proporciona información sobre las componentes principales.

La matriz de componentes esta en: componentes$scores que tiene la misma información obtenida en CP para correlaciones. Para obtener el mismo valor y signo, se debe particularizar cada matriz; por ejemplo si se normaliza ambas matrices CP1 y CP2 para la primera componente se tiene:

```
> CP1[,1]/CP1[1,1]
> componentes$scores[,1]/ componentes$scores[1,1]
```

Ambos vectores son iguales en signo y magnitud.

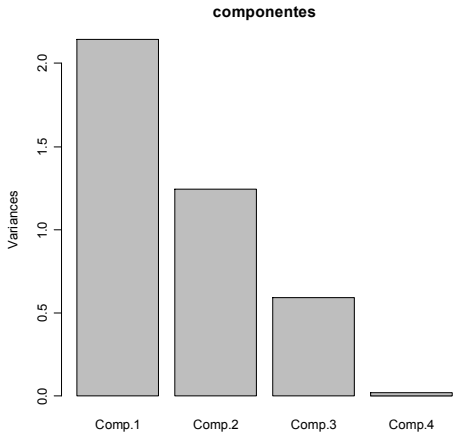
De igual forma para covariancias.

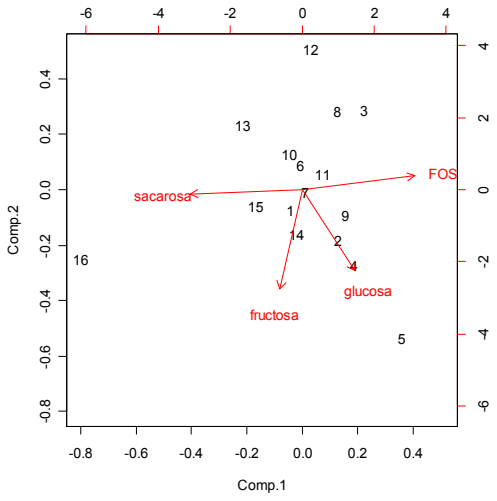
```
> componentes<-princomp(datos, cor=FALSE)
> summary(componentes)

Importance of components:
                          Comp.1     Comp.2      Comp.3       Comp.4
Standard deviation     5.3956993 0.61247696 0.301047465 0.0835644307
Proportion of Variance 0.9840216 0.01267911 0.003063227 0.0002360218
Cumulative Proportion  0.9840216 0.99670075 0.999763978 1.0000000000
```

Las proporciones acumulativas son las mismas encontradas con el procedimiento anterior.

Otra información es obtenida de:

```
> plot(componentes)
> biplot(componentes)
```





Clasificación no supervisada.
Conglomerados (cluster).

Permite formar grupos similares según las medidas que uno considere para el agrupamiento.

Hay muchas funciones para ello, las principales son:

dist() para el calculo de la distancia entre individuos
hclust() procedimiento para los agrupamientos.
cutree() para hacer los cortes del dendrograma.

y otras funciones de la librería cluster y agricolae como se indica

```
plclust()
rect.hclust()
daisy()
pam()
clusplot()
as.dendrogram()
cut()
consensus()
hcut()
```

Subir los paquetes

```
> library(cluster)
> library(agricolae)
```

Análisis cluster

Métodos para calcular distancia.

"euclidian".- la distancia entre los vectores x e y
"maximum".- la distancia Máxima entre dos componentes x,y
"manhattan": la distancia Absoluta entre los dos vectores
"canberra".- sum(|x_i - y_i | / |x_i + y_i |).
"minkowki".- la norma de p componentes
"binary".- distancia como una proporción de frecuencias.

"binary".- La distancia es determinada como una proporción entre el total de celdas de presencia del marcador (una sola vez) entre el total de celdas con presencia del marcador.

Ejemplo: Para hallar la distancia entre A y B, se tiene 5 celdas con presencia del marcador { (1,1), (1,1), (1,0), (1,1), (1,1) } y celdas con una sola vez { (1,0) }, entonces la distancia es1/5 = 0.2 . En otro caso, Distancia (B, D), total con presencia del maracador { (1,1), (1,0), (0,1), (1,0), (1,1) } y celdas con una sola vez { (1,0), (0,1), (1,0) }, entonces la distancia (B, D) es igual a 3/5 = 0.6

```
      Datos
  m1 m2 m3 m4 m5
A  1  1  1  1  1
B  1  1  0  1  1
C  0  0  1  0  0
D  1  0  1  0  1
E  0  0  0  0  0
F  1  1  1  1  1
```

Distancias:

```
     A    B    C    D    E
B 0.20
C 0.80 1.00
D 0.40 0.60 0.66
E 1.00 1.00 1.00 1.00
F 0.00 0.20 0.80 0.40 1.00
```

Métodos para hallar dendrogramas.

Cada una de las matrices de distancias puede ser utilizada para hallar el dendrograma. Se utiliza el criterio de la no-semejanza (DISSIMILARITIES)

"ward".- método de mínima variancia, racimos compactos y esféricos
"single".- relaciona las ramas del árbol
"complete".- racimos similares

Los otros métodos apuntan a alguna relación entre los racimos
"average"
"mcquitty"
"median"
"centroid"

Ejercicios.

Utilice las siguientes tablas de datos para los procedimientos de conglomerados y dendrogramas en R.

Datos.1 : Binarios
Datos.2 : medibles y categóricos, con valores muy diferenciados
Datos.3 : Tabla con datos faltantes.

```
datos.1
  m1 m2 m3 m4 m5
A  1  1  1  1  1
B  1  1  0  1  1
C  0  0  1  0  0
D  1  0  1  0  1
E  0  0  0  0  0
F  1  1  1  1  1
```

```
datos.2
    v1 v2 v3 v4
A 1200  2  6 20
B 1500  1  3 10
C 1400  1  3 20
D 1600  2  5 15
E 1000  1  2  5
```

```
datos.3
  x1 x2 x3 x4
+  4  2  6 15
+ NA  1  3 18
+  2  1 NA 16
.  6  2  5 15
. NA  1  2 NA
.  4  2  6 10
```

Preparación de las tablas:

```
> datos.1 <-
rbind(c(1,1,1,1,1),c(1,1,0,1,1),c(0,0,1,0,0),c(1,0,1,0,1),c(0,0,0,0,0),c(1,1,
1,1,1))
> dimnames(datos.1)<-
list(c("A","B","C","D","E","F"),c("m1","m2","m3","m4","m5"))
> datos.1
```

```
> datos.2 <-
rbind(c(1200,2,6,20),c(1500,1,3,10),c(1400,1,3,20),c(1600,2,5,15),c(1000,1,2,5
))
> dimnames(datos.2)<-list(c("A","B","C","D","E"),c("v1","v2","v3","v4"))
> datos.2

> datos.3 <-
rbind(c(4,2,6,15),c(NA,1,3,18),c(2,1,NA,16),c(6,2,5,15),c(NA,1,2,NA),c(4,2,6,1
0))
> dimnames(datos.3)<-list(c("+","+","+",".",".","."),c("x1","x2","x3","x4"))
> datos.3
```

Calcula las distancias por diferentes métodos

```
> d1<-dist(datos.1,method="euclidean")
> d2<-dist(datos.1,method="maximum")
> d3<-dist(datos.1,method="manhattan")
> d4<-dist(datos.1,method="canberra")
> d5<-dist(datos.1,method="binary")
> d6<-dist(datos.1,method="minkowski",p=3)
```

Halla la relación de conglomerados por diferentes métodos

```
> c1<-hclust(d1,method="ward")
> c2<-hclust(d1,method="single")
> c3<-hclust(d1,method="complete")
> c4<-hclust(d1,method="average")
> c5<-hclust(d1,method="mcquitty")
> c6<-hclust(d1,method="median")
> c7<-hclust(d1,method="centroid")
```

Presentar 2 gráficos por cada proceso

```
> par(mfrow=c(1,2))
> plot(c1,col="blue",main="ward",sub = "", xlab = "")
> plot(c2,col="brown",main="single",sub = "", xlab = "")

> plot(c3,col="red",main="complete",sub = "", xlab = "")
> plot(c4,col="chartreuse4",main="average",sub = "", xlab = "")

> plot(c5,col="magenta",main="mcquitty",sub = "", xlab = "")
> plot(c6,col="brown",main="median",sub = "", xlab = "")

> par(mfrow=c(1,1))
> plot(c7,col="black",main="centroid",sub = "", xlab = "")
```

Realizar todas las posibilidades con los métodos y los 3 tipos de datos y obtenga sus propias conclusiones respecto a las metodologías y los tipos de datos.

Procedimientos avanzados para dendogramas:

```
> dend1 <- as.dendrogram(c1)
> plot(dend1, nodePar=list(pch = 2:1,cex=.4*2:1, col = 2:3), horiz = TRUE)
> plot(dend1, nodePar=list(pch = c(1,NA),cex=0.8), type = "t", center=TRUE)
```

```
> plot(dend1, edgePar=list(col = 1:2, lty = 2:3), edge.root = TRUE)
```

Coeficiente de Divisivo (metodo Diana)

Una vez calculado la distancia de disimilaridad, se puede aplicar el método Diana para obtener el coeficiente de divisivo.

El método que proporciona un valor mas alto, cerca de 1.0, es el mejor método.

Aplicar este concepto a sus procedimientos realizados.

```
> individuos<-dist(datos.2,method="minkowski",p=3)
> dendograma<-diana(individuos)
> plot(dendograma,main="Metodo minkowski, p=3", col="blue")
```

Aplicación: Azucares en Oca.

```
> datos <- read.table("azucares en yacon.txt",header=TRUE)
```

Seleccione las variables para el estudio

```
> attach(datos)
> matriz<-datos[,-1]
> row.names(matriz)<-datos[,1]
```

Calculo de las distancias por el método average (distancia euclidiana)

```
> distancia <- dist(matriz)
```

Crear un objeto "hc" que contiene toda la información del cluster análisis.

```
> hc <- hclust(distancia)
```

Imprimir información del método de conglomerado

```
> hc
```

Mostrar la estructura de este objeto.

```
>str(hc)
```

Listar los elementos del objeto "hc"

```
> names(hc)
```

Mostrar la altura de cada individuo en el dendrograma.

```
> hc$height
```

Graficar el dendrograma

```
> par(cex=0.6)

> plclust(hc, hang = 0.2, unit = FALSE, level = FALSE, hmin = 0,
square = TRUE, labels = NULL, plot. = TRUE, axes = TRUE, frame.plot = FALSE,
ann = TRUE, main="Dendrograma de Clusters\ndata(Azucares)", sub = NULL, xlab
= NULL, ylab = "Altura")
```

Otro modelo de dendrograma

```
> plot(hc,cex=0.6, main="Dendrograma ", hang= -1)
```

Corte el dendrograma en 3 rectángulos (grupos):

```
> informacion<-rect.hclust(hc, k=3, border="green")
```

Lista los grupos

```
> informacion
```

Lista el grupo 2

```
> informacion[[2]]
```

Modelos especiales para cluster:

método Diana

```
> hc.diana <-diana(matriz,metric="manhattan")
```

Imprime la información obtenida con este método

```
> hc.diana
```

Graficar los resultados

```
> plot(hc.diana,ask=TRUE)
```

método Daisy

```
> azucar.diss <- daisy(matriz)
```

Formar 4 grupos utilizando la matriz de disimilaridades calculado "daisy" minimizando la suma de las disimilaridades:

```
> azucar.clus <- pam(azucar.diss, 4, diss = TRUE)$clustering
```

Para listar los grupos por este método, puede imprimir azucar.clus o usar cbind() para mostrar en columnas

```
> cbind(azucar.clus)
```

Graficar estos grupos con la función clusplot

```
> par(cex=0.6)
> clusplot(azucar.diss,  azucar.clus, lines=2,diss = TRUE,
  color=TRUE, col.p="black", labels=2)
```

Método Fanny

Utiliza un valor "k" que indica el numero de grupos a formar 0 < k < n/2; n es el numero de observaciones

Tres grupos, k=3 mediante las distancias euclidianas.

```
> hc.fanny<-fanny(matriz,  k=3, diss = inherits(matriz, "dist"))

> hc.fanny

> plot(hc.fanny, ask=TRUE,col=TRUE)
```

Método Agnes

```
> hc.agnes <- agnes(matriz, metric = "manhattan", stand = TRUE)
> hc.agnes
> plot(hc.agnes,ask=TRUE)
```

Mejore el coeficiente de aglomeración

```
hc.agnes2 <- agnes(daisy(matriz), diss = TRUE, method = "complete")

> plot(hc.agnes2,ask=TRUE)

> box(col="orange",lwd=2)
```

Observe que el coeficiente de agrupamiento mejora

Trabajando con dendrogramas y grupos

```
> distancia<-dist(matriz,method="maximum")
> hc <- hclust(distancia)
> ramas <-cutree(hc,h=1:5)
> ramas
```

Estudie este resultado:

A nivel h=1 se logra formar hasta 8 grupos diferentes
A nivel h=5 hasta 3 grupos.

Junte a los datos originales, la información de las ramas cortadas

```
A<-data.frame(matriz,ramas)
```

Con esta información liste las observaciones agrupadas a una altura de 4.

```
Lista <-by(A[,c(1,2,3,4)],A[,8],list)
lista
```

Información del dendrograma

```
dend <-as.dendrogram(hc)
grupos <- cut(dend, h=4)
```

Grafico por grupo: Grupo 2

```
plot(grupos$lower[[2]], nodePar=list(col=1),edgePar =
list(lty=1:2, col=2:1), edge.root=TRUE)
```

El mismo gráfico en forma horizontal

```
plot(grupos$lower[[2]], nodePar=list(pch = 2:1,cex=.4*2:1, col = 2:3), horiz
= TRUE)
```

Con otro tipo de línea para el gráfico y con márgenes definidos

```
par(mar=c(4,4,4,4))
plot(grupos$lower[[2]], edgePar=list(col = 1:2, lty = 2:3), edge.root = TRUE)
```

```
Clasificación no jerárquica.

R dispone de la función kmeans() con 4 métodos de agrupamiento, requiere
asignar un numero de grupos a priori.
```

```
azucar <- read.table("azucares en yacon.txt",header=TRUE)
clase1 <- kmeans(azucar[,-1], 3, algorithm = "Hartigan-Wong")
plot(azucar[,c(2,3)],cex=2,col=clase1$cluster,pch=17)
clase2 <- kmeans(azucar[,-1], 3, algorithm = "Lloyd")
plot(azucar[,c(2,3)],cex=2.5,col=clase2$cluster,pch=8)
clase3 <- kmeans(azucar[,-1], 3, algorithm = "Forgy")
plot(azucar[,c(2,3)],cex=0)
nombres<-as.character(azucar$ID)
text(azucar[,c(2,3)], nombres, col=clase4$cluster, cex=1)
clase4 <- kmeans(azucar[,-1], 3, algorithm = "MacQueen")
plot(azucar[,c(2,3)],cex=3.5,col=clase4$cluster,pch=1)
```

Análisis discriminante

Es otro método importante del análisis multivarial que consiste en construir modelos de clasificación conociendo a priori la pertenencia de individuos u objetos en grupos o poblaciones, el modelo discriminante puede ser útil para consistencias de los grupos, clasificar nuevos objetos en las poblaciones definidas y realizar inferencia estadística de los modelos construidos. Los métodos cambian de acuerdo a los tipos de variables y al grado de homogeneidad de variancia de las poblaciones a clasificar.

R dispone de muchas librerías. MASS es la librería base.

Las funciones disponibles son:
lda() para función discriminante lineal y
qda() para función discriminante cuadrático

Para la aplicación se utilizara el archivo "Maiz Selva de Peru".

Lectura:

```
maiz <- read.table("maiz selva.txt", header=TRUE)
```

Construcción del modelo discriminate lineal

```
library(MASS)
model <- lda(Raza ~ ., data=maiz)
```

Obtener la predicción con los mismos datos

```
clases <- predict(model, maiz)$class

# Comparar lo observado con lo esperado

nuevo <- data.frame(maiz,clases)
table(nuevo[,c(1,8)])
```

Tasa de error aparente

```
tea <- 1 - sum(clases == maiz$Raza)/nrow(maiz)
tea
```

Tasa de error de validación cruzada

```
model <- lda(Raza ~ ., data=maiz, CV=TRUE)
clases <- model$class
tevc <- 1 - sum(clases == maiz$Raza)/nrow(maiz)
tevc
```

El método de k vecinos mas próximos

Es otro método para clasificar mediante el algoritmo de k vecinos más próximos. La función en R es knn() del paquete class.

El parámetro más importante es el número de vecinos más cercanos que debe coger, que es "k"

Por ejemplo para clasificar los individuos en las razas de maíz para k vecinos = 3:

```
library(class)
maiz.knn <- knn(maiz[,-1],maiz[,-1], maiz$Raza,k=3)
```

El resultado es simplemente un vector que da en que grupo se incluye cada Individuo. Para calcular la tasa de error aparente, basta comparar con las verdaderas clases, igual que en el caso del análisis discriminante lineal.

```
Así, el error aparente.

tea <- 1 - sum(maiz.knn == maiz$Raza)/nrow(maiz)
tea

[1] 0.1506849
```

Validación cruzada en el método de k vecinos más próximos

La función en R es knn.cv()

```
maiz.knn.cv <- knn.cv(maiz[,-1], maiz$Raza,k=3)

Y el error aparente en validación cruzada.

1 - sum(maiz.knn.cv ==maiz$Raza)/nrow(maiz)

[1] 0.1643836
```

Ejercicio 1: Construir un nuevo set de datos para construir el modelo (entrenamiento) y otro set diferente para la prueba del modelo de clasificación. Hallar el error aparente de la muestra de prueba.

```
maiz <- read.table("maiz selva.txt", header=TRUE)
out <-sample(1:73, 10)
```

Muestra para el entrenamiento:

```
train <- maiz[-out,]
```

Muestra para la prueba:

```
test <- maiz[out,]
```

Análisis

```
model <- lda(Raza ~ ., data=train)
clases <- predict(model, test)$class
tea <- 1 - sum(clases == test$Raza)/nrow(test)
tea
```

Ejercicio 2: Formar un nuevo conjunto que este formado por 8 individuos aleatorios de la raza "CUBANOAMARILLO" y las otras razas. Con este conjunto realice el análisis discriminante con las predicciones y comparaciones con una muestra de prueba. Halle el error aparente y error de validación cruzada.

```
# Construir un nuevo set de datos
# los 8 individuos de raza Cubanoamarillo en el objeto cubanoA

cubanoA <- maiz[maiz[,1]=="CUBANOAMARILLO",]
selectoA <- cubanoA[sample(1:56,8), ]

# Los restantes
selectoB <- maiz[maiz[,1]!= "CUBANOAMARILLO",]

# Junto para el nuevo conjunto
selecto<-rbind(selectoA,selectoB)

# Construccion del modelo

model <- lda(Raza ~ ., data=selecto)

# verificacion del modelo

clases <- predict(model, selecto)$class
table(clases, selecto[,1])
```

Tasa de error aparente

```
tea <- 1 - sum(clases == selecto$Raza)/nrow(selecto)
tea
```

Tasa de error de validación cruzada

```
model <- lda(Raza ~ ., data=selecto, CV=TRUE)
clases <- model$class
tevc <- 1 - sum(clases == selecto$Raza)/nrow(selecto)
tevc
```

Ejercicio 3: Del nuevo conjunto formar dos conjuntos (10 individuos para la prueba y el resto para el entrenamiento).

```
ntotal <- nrow(selecto)
muestra<- sample(1:ntotal, 10)
test <- selecto[muestra,]
train <- selecto[-muestra,]

# modelo
model <- lda(Raza ~ ., data=train)

# Comparación entre lo predicativo y lo observado.
prod<-predict(model, test)$class
table(pred, test[,1])
```

¿Cuál es el error aparente de la muestra de prueba?

Ejercicio 4: Del nuevo conjunto clasificar y hallar el error aparente si se utiliza 4 vecinos más cercanos.

```
library(class)
selecto.knn <- knn(selecto[,-1],selecto[,-1], selecto$Raza,k=4)

# Error aparente

tea <- 1 - sum(selecto.knn == selecto$Raza)/nrow(selecto)
tea
```