UNIVERSIDAD NACIONAL DE INGENIERIA

CENTRO DE ESTUDIANTES DE LA FACULTAD DE INGENIERÍA ECONÓMICA Y CIENCIAS SOCIALES

SECCIÓN DE EXTENSIÓN UNIVERSITARIA Y PROYECCIÓN SOCIAL

Análisis Estadístico con "R"

Ing. Felipe de Mendiburu

Version 2.6.0 (2007-10-03)

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

http://tarwi.lamolina.edu.pe/~fmendiburu

http://cran.r-project.org/src/contrib/Descriptions/agricolae.html

Noviembre - 2007

CONTENIDO

Sesión 1

- Instalación del programa.
- La Consola, preparación del ambiente de trabajo
- Creación objetos
- Grabación del objeto en archivo permanente
- Lectura del archivo grabado
- Creación y edición de programas

Sesión 2

- Manejo de datos externos
- Excel y base datos a R.
- Formas directas para acceder a un archivo externo
- Importar datos de: dBase, Access. Minitab, SAS, SPSS

Sesión 3

- Creación de vectores y matrices
- Operaciones con vectores y matrices.
- Selección, ordenamiento de un conjunto de datos
- Gráficos de líneas, barras, pie, histogramas,
- Grafico de cajas: boxplot(). Tallos y hojas

Sesión 4

- Planeamiento de experimentos: Completamente randomizado, Bloques completos al azar, Cuadrado Latino, Greco latino, Bloques Incompletos balanceados y parcialmente balanceados, latice simple, diseños alfa, experimentos confundidos y experimentos fraccionados.
- Análisis de variancia de los diseños experimentales con uno o más factores.
- Comparación múltiple de tratamientos LSD, Bonferroni, Waller-Duncan y Tukey.
- Contrastes ortogonales.

Sesión 5

- Correlación
- Análisis de regresión lineal simple.
- Grafico y limites de confianza
- Regresión lineal múltiple.
- Análisis de residuales
- Selección de variables para el mejor modelo

Sesión 6

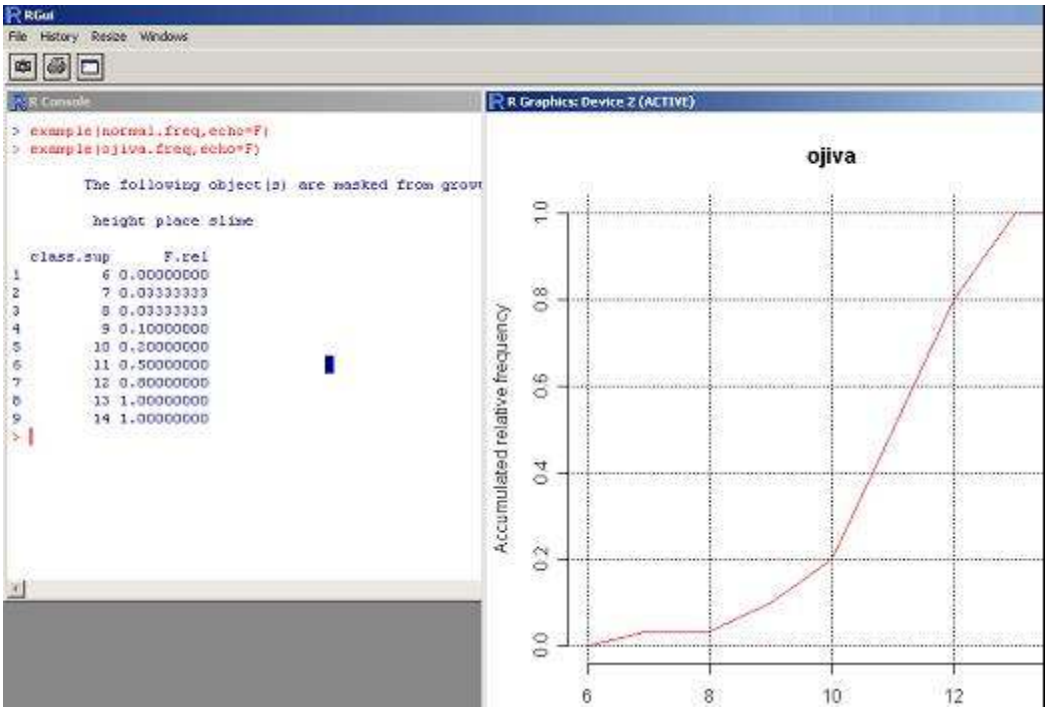
- Análisis multivarial. Componentes principales
- Clasificación no supervisada: cluster análisis, dendrograma y consensus (métodos jerárquicos).
- Clasificación supervisada. Análisis discriminante k vecinos mas cercanos.

Tabla de códigos ASCII utilizados en R	
Código	**Símbolo**
92	\
47	/
91	[
93	]
40	(
41	)
123	{
125	}
124	|
60	<
62	>
61	=
34	“
126	~
58	:
59	;
64	@
94	^
35	#
36	$
37	%
38	&
39	‘

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cite R or R packages in publications.

Sesión 1

- Instalación del programa.
- La Consola, preparación del ambiente de trabajo
- Creación objetos
- Grabación del objeto en archivo permanente
- Lectura del archivo grabado
- Creación y edición de programas



Archivo de trabajo: C:\UNI-R**
CD con toda la información del curso

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

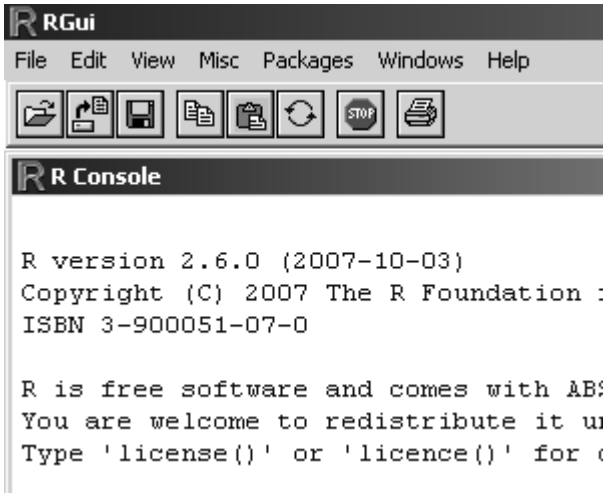
Una vez que lo tiene en le disco duro, ejecute este programa presionando el Icon:

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.

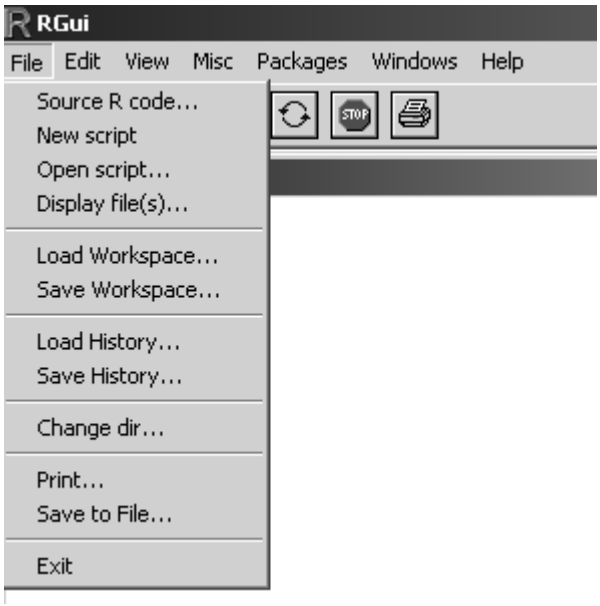
Al ejecutar, se ingresa a una ventana de trabajo llamada la consola, que es el lugar para ejecutar ordenes y a la vez recibir los resultados en texto, los cuales pueden ser enviados a un procesador de texto y los gráficos otra ventana que puede ser enviado a un archivo de gráficos o insertar en un documento.



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, gráficos y todo elemento que puede ser manipulado. 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: Notas de 10 exámenes:

```
notas <- c(12, 16, 09, 14, 08, 12, 11, 07, 15, 13)
```

Con esta orden se crea un vector de 10 elementos.

notas: es la variable que almacenara los 10 valores
`<-` : son dos símbolos juntos “`<`“ y “`-`“ , al juntar indica una flecha que se entiende como almacenar los datos de la derecha en la izquierda.

Este símbolo puede ser reemplazado por “=”

c() : es una función que permite crear un objeto de varios elementos separados por comas.

Cuando se ejecuta esta orden, R almacena los datos en el objeto notas y para observar su contenido se ejecuta la orden:

```
> print(notas)
```

o

```
> notas
```

El resultado será:

```
[1] 12 16  9 14  8 12 11  7 15 13
```

Que indica la lista 10 elementos.

El objeto notas también es creado mediante la siguiente orden:

```
> c(12, 16, 09, 14, 08, 12, 11, 07, 15, 13) -> notas
```

La flecha indica la asignación de los valores a la variable.

Para crear un objeto matricial a un vector fila o columna, se utiliza la función rbind() ó cbind():

```
> fila <- rbind(notas)
> columna <- cbind(notas)

> fila

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
notas   12   16    9   14    8   12   11    7   15    13

> columna

      notas
 [1,]    12
 [2,]    16
 [3,]     9
 [4,]    14
 [5,]     8
 [6,]    12
 [7,]    11
 [8,]     7
 [9,]    15
[10,]    13
```

Lista de objetos creados:

```
> objects()
[1] "columna" "fila"    "notas"
> ls()
[1] "columna" "fila"    "notas"
```

Remover objetos: rm

```
> rm(notas)
```

remueve le objeto notas

para remover todos los objetos

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

Limpiar la pantalla

Presione las teclas ctrl y “L” simultáneamente

Preparación del ambiente de trabajo

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:\UNI-R

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

O también:

```
> setwd("c:\\UNI-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 de elementos “A”,”B”,”C”
> x                     # Muestra el contenido de 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)
```

<- significa asignar

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
```

Problema: Se tiene 8 frutas que pesan 10 kilos y se debe repartir a 3 personas por igual cantidad. Mediante una sola operación determinar el resultado:

- Número de frutas que le corresponde a cada persona, teniendo en cuenta que no hay partición de la fruta.
- Y si hay partición de la fruta, cuanto le corresponde a cada uno?

Operaciones lógicas.

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.

```
> play <- c("f","b","f","v","b","v","b","f","v","b","v","b")

# extraer los que juegan basquet

> lista<-data.frame(Id=1:12,play)
> ob<-lista[,2]=="b"
> lista[ob,]

   Id play
2   2    b
5   5    b
7   7    b
10 10    b
12 12    b
```

```
# Contabilizar
> nrow(lista[ob,])
```

Examine las siguientes órdenes

```
> play=="b"
```

Observe que los resultados son TRUE y FALSE, dependiendo del valor de lista.

```
> play.b <- play[play=="b"]

> play.b
> length(play.b)
```

```
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"

Problema. Un objeto x contiene una secuencia de números { 1, 2, 3,…, 20}, hallar la suma, media, variancia y numero de elementos de "x" .

```
> x <- 1:20 # x<- seq(1, 20, 1)
> sum(x)
> mean(x)
> var(x)
> length(x)
```

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
 [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
```

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]
[1] 5 2 7 3 2 5 2
```

4. Cuantos elementos son:

```
> length( x[x%in%y] )   # cuenta los elementos.
[1] 7
```

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])
[1] 5 2 7 3
> length(unique(x[x%in%y]))
[1] 4
```

Nota: Cada resultado puede ser almacenado en un objeto

Ayudas a las órdenes estudiadas:

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

Ejercicios:

1- crear los siguientes objetos

```
xa = -1.9, -0.9, 1.1, 2.1,3.1
xb = 0, 2, 4,  6, .... , 20
xc = 20, 18, 16, 14, ... 0
xd = 30 números entre 5 y 10
```

2- Hallar el mínimo, máximo y el promedio de cada objeto

En cada operación, R almacena los valores con el máximo número de dígitos, hasta 17, pero muestra solo 8, si uno desea que muestre mayor número, debemos cambiar esta opción.

```
> options(digits = 16)
```

Observe el efecto en la siguiente operación.

```
> 1/3
```

Los objetos en R son de diferente forma como se pudo observar, estos pueden ser un conjunto de números o palabras, un vector, una matriz, una lista de resultados, una tabla de datos o alguna combinación entre ellos.

El usuario de R, debe conocer bien la descripción de los datos, para hacer correctamente las operaciones.

R no es bueno para preparar una salida en la forma que deseamos, requiere muchas instrucciones para obtener un formato, sin embargo, podemos generar los resultados y pasarlos a word, excel, power point u otro programa que nos sirva para elaborar nuestro informe. Veamos un ejemplo de presentar un resultado con R

Primero limpiemos la ventana de trabajo: Presione CTRL. + L y luego escribamos cada orden como sigue, respetando las mayúsculas y minúsculas en un editor, por ejemplo Tinn-R, luego grabe con el nombre ejemplo.R

```
# Inicio
x <- 1:20
cat("\n")
cat("Suma de x          : ", sum(x),"\n")
cat("promedio de x      : ", mean(x),"\n")
cat("variancia de x     : ", var(x),"\n")
cat("error estándar de x: ", sd(x)/sqrt(length(x)),"\n")
# final
```

Ahora en la consola escriba;

source(“ejemplo.R”)

"\n" indica salto de línea; puede usar "\t" para indicar fabulación.

Manejo de datos.

Los datos pueden ser creados en el ambiente R o en otros programas.

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

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])
```

Algunas funciones que permiten resúmenes

summary(), by()

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)
```

Ejercicios:

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])
estadisticas
```

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")
```

Lectura de un archivo

Lectura para datos separados por comas “,”

```
> datos1 <- read.table("camote1.txt", header = TRUE, sep = ",")
```

Lectura para datos separados por tabulaciones o espacios en blanco

```
> datos2 <- read.table("camote2.txt", header = TRUE)
```

Creación y edición de un programa personal (secuencia de instrucciones).
Para crear o un programa puede utilizar la opción de R del menú File o usar el programa Tinn-R.

El siguiente programa permite leer, ordenar e imprimir el archivo ordenado. Usar Tinn-R

```
# Ejemplo de programa
# Lectura de los datos “camote2.txt”
# Ordena los datos por Materia seca en forma descendente
#
datos<-read.table(“camote2.txt”,header=T)
datos.ordenados <-datos[order(datos[,3],decreasing=T),]
print(datos.ordenados)
# fin de mi programa
```

Puede ejecutar directo desde R:

- Seleccionando todas las instrucciones y copiando a la consola.
- Seleccionando y presionando simultáneamente Alt + R.
- Ejecutando directamente desde Tinn-R con los botones indicados en la parte superior.
- Grabando el programa en el disco Duro y luego ejecutar desde R con la orden source()

```
> source(“programa.R”)
```

Completado esta sesión, puede salir del sistema R e ingresar a la plataforma Windows.

Salida de programa R mediante la orden
> q()

Mediante menú: “File”, “exit”

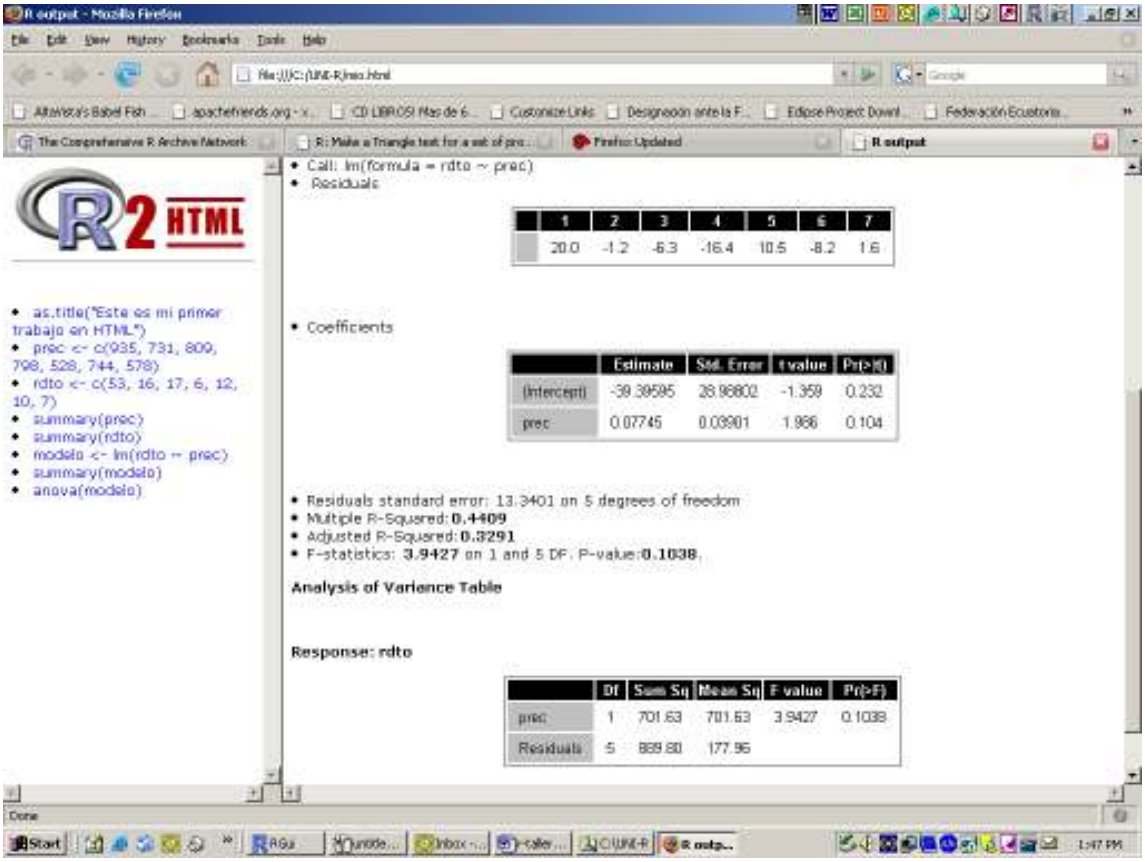
Ejercicios.
Crear el siguiente archivo de datos de nombre “germoplasma de papa”. Respuesta por planta

Variedad	tuberculos	Peso gramos
Revolución	4	70
Yungay	8	75
Liberteña	6	100
Muru	5	50
Yana	8	80
Andina	6	120

- Hallar las estadisticas: promedio y variancia de turberculo y peso por planta de las variedades
- Ordenar la tabla de datos por variedad.
- Ordenar la tabla por peso de mayor a menor.

Sesión 2

- Manejo de datos externos
- Excel y base datos a R.
- Formas directas para acceder a un archivo externo
- Importar datos de: dBase, Access. Minitab, SAS, SPSS
- Exportar a HTML



Pasos preliminares para esta sesión.

Revisar los archivos excel y asigne nombre a conjunto de celdas:

- ✓ En Labranza, asignar el nombre "datos" al área: B6:G21
- ✓ En papa2001, asignar el nombre "estadistica" al área F3:H7
- ✓ En papa2001, asignar el nombre "propagacion" al area A3:D27
- ✓ Bolivia.dbf (Es un archivo de FoxPro o dBASE III)
- ✓ camote.txt (Archivo texto, ver con NOTEPAD)
- ✓ maca.dat (Creado por SAS para uso de R)
- ✓ maca.sas7bdat (Es un DATA SET de SAS para exportar)
- ✓ papacip.mdb (Es un proyecto de ACCESS)
- ✓ suelo.txt (Datos de suelo para sus gráficos)
- ✓ camote raices.MTP (datos del Minitab)

Pasar datos de otros medios a R.

Uso del modulo RODBC para transferencia de datos de tablas y base de datos y foreign para otros sistemas.

1: Cargar las librerías RODBC y foreign

```
> library(RODBC)
> library(foreign)
```

2: Establecer la comunicación

```
> canal <- odbcDriverConnect("")
```

Ingrese a la opción: **Machine data Source**, luego el archivo. Para nuestro ejemplo: Labranza.xls

En estos momentos R esta disponible para acceder al archivo Labranza.XLS.

3: Realice la transferencia de los datos de EXCEL a R.

```
> base <- sqlFetch(canal,"datos")
```

Cerrar la conexión con la orden:

```
> odbcCloseAll()
```

4: ver la estructura con str()

```
> str(base)
```

Realice el mismo ejercicio para conectarse al archivo: PAPA2001.XLS, el área de lectura se llama: propagacion y asigne al objeto "propagacion"

Formas directas para acceder a un archivo externo

Caso Excel.

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

Importar los datos de papa2001.xls, el área "estadistica" para continuar realizando otros graficos.

Nota. Respetar mayúsculas y minúsculas en el nombre.

```
> canal <- odbcDriverConnect("")
> estimadores <- sqlFetch(canal,"estadistica")
> odbcCloseAll()
```

Para conocer las características del objeto "estimadores", ejecute:

```
> str(estimadores)
```

El objeto es un data.frame, para graficar en barras o en pie, necesitamos que sus elementos sean vectores.

```
> trat <- as.vector(estimadores$Tratamiento)
> prom <- as.vector(estimadores$Promedio)
```

Formas directas para acceder a un archivo externo

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")
> 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)
```

Nota. Cuando los datos son preparados sin encabezados, en R se puede leer utilizando el parámetro header=FALSE y R colocara encabezados como V1, V2,...etc. y con la función "edit" se puede editar los encabezados.

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)]
```

Ahora tiene una nueva tabla en su base de datos.

Cierre la comunicación:

```
> 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:\UNI-R\maca.dat';
libname sitio 'C:\UNI-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.
```

Entrada y salida manual

Uso del "copy" y "paste" de Microsoft.

Los datos de Excel pueden ser pasados a un manejador de textos (ej. NOTEPAD) y grabarlo con un nombre en el disco duro.

Realice el siguiente ejercicio:

1. Ingresar al archivo labranza.xls con Excel.
2. hacer un COPY de los datos, incluyendo los encabezados de columnas en un espacio de NOTEPAD
3. Abrir NOTEPAD y hacer PASTE.
4. Grabar el archivo como labranza.txt
5. Salir del NOTEPAD y EXCEL.
6. En R seguir el proceso:

```
> labranza <- read.table("Labranza.txt",header=TRUE)
```

Si un archivo de datos que esta en Excel tiene las características de un set de datos para análisis, entonces puede hacer almacenamiento de extensión CSV y poder leer de R directamente:

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

Nota. Cuando los datos son preparados sin encabezados, en R podemos leer utilizando el parámetro header=FALSE y R colocara los encabezados con V1, V2,...etc. y con la opción edit() puede editar los encabezados.

Salidas en HTML

```
> library(R2HTML)
```

Precipitación (mm) anual y rendimiento promedio de Oca en campo de agricultores

935	731	809	798	528	744	578
53	16	17	6	12	10	7

Hallar estadísticas y realizar el análisis de regresión (rdto en función de la prec).

Activar HTML y proceder a los cálculos.

```
> HTMLStart(outdir = "c:/UNI-R",filename="mio",echo=FALSE)
> as.title("Este es mi primer trabajo en HTML")

> prec <- c(935, 731, 809, 798, 528, 744,578)
> rdto  <- c(53, 16, 17, 6, 12, 10, 7)
> summary(prec)
> summary(rdto)
> modelo <- lm(rdto ~ prec)
> summary(modelo)
> anova(modelo)
```

Desactivar HTML

```
> HTMLStop()
```

Observar el resultado el archivo creado: **mio.html en el fólder C:\UNI-R**

Ejercicios: Trabajar con el archivo labranza.xls y realizar las siguientes tareas:

Papa de primera y de segunda

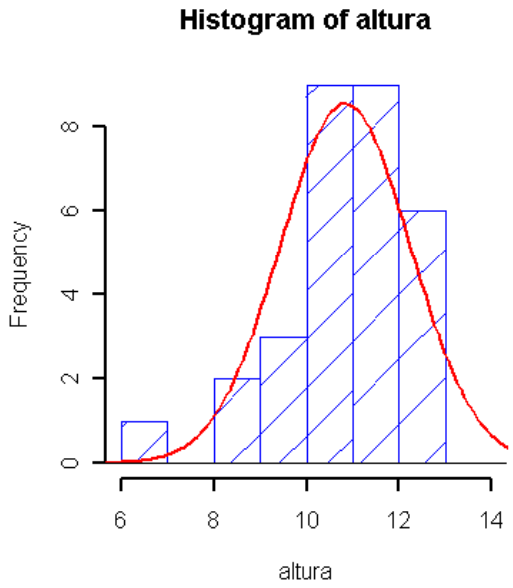
1. Estadísticas de centralización (Media, Mediana).
2. Estadísticas de Dispersión: Rango, Desviación estándar, Coeficiente de variación
3. Tabla de frecuencia.
4. Estadísticas de datos agrupados.

Solución:

```
> attach(labranza)
> mean(primera)
> median(primera)
> rango<- max(primera) - min(primera)
> sd(primera)
> var(primera)
> cv <- sd(primera)*100/mean(primera)
> h<- hist(primera)
> table.freq(h)
> stat.freq(h)
> detach(labranza)
```

Sesión 3

- Creación de vectores y matrices
- Operaciones con vectores y matrices.
- Selección, ordenamiento de un conjunto de datos
- Gráficos de líneas, barras, pie, histogramas,
- Grafico de cajas: boxplot(). Tallos y hojas
- Creación y uso de funciones personales



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
```

Verifique si la solución es correcta.

Multiplique A por "x" y compara con "b"

```
> A%*%x == b
```

También si A x – b = 0 entonces es correcto.

```
> A%*%x - b
```

La función solve puede ser utilizado para hallar la inversa de una matriz; esto significa que "x" puede ser obtenido de la operación: $x = A^{-1} b$

```
> y<- solve(A)%*%b
> y
```

Verifique si "x" es igual a "y"

Realice la operación x-y que debe ser cero completamente.

```
> x-y
```

Para este caso el cero absoluto no se produce porque el sistema tiene un máximo de 16 dígitos significativos.

Selección de subconjuntos

Seleccione el subconjunto definido por el tratamiento Will

```
> will<-subset(propagacion,Tratamiento=="Will")
> will
```

Defina los vectores genotipo y brotes

```
> genotipo<-as.vector(will[,1])
> brotes <-as.vector(will[,4])
```

Gráficos de barras y pie

```
> barplot(brotes,names=genotipo,cex.names=0.6,col="yellow")
```

Cambie lo limites del eje Y

```
> barplot(brotes,names=genotipo,cex.names=0.6,col=4,ylim=c(0,10))
```

Cambie los parámetros del grafico: tamaño de letra y fondo verde y márgenes .

```
> par(cex=0.7, bg="bisque1", mar=c(2,2,2,2))
> barplot(brotes,names=genotipo,ylim=c(0,10),col="burlywood")
```

Grafico en pie

```
> pie(brotes,genotipo)
```

Ejercicio: Realice el mismo proceso para el genotipo: DESIREE

```
> desiree <- subset(propagacion,genotipo=="Desiree")
> desiree

> brotes<-as.vector(desiree$brotes)
> tratamiento<-as.vector(desiree$Tratamiento)
> pie(brotes,tratamiento)
> pie(brotes,tratamiento, col=gray(seq(0.4,1,length=4)))

> barplot(brotes,name=tratamiento,horiz=TRUE)
> par(bg="white")
> barplot(brotes,name=tratamiento,horiz=TRUE,density=TRUE)
> barplot(brotes,name=tratamiento,density=c(10,20,30,40),space=0)
```

Crea el objeto "propaga" que representa a los brotes

```
> propaga<-as.vector(propagacion$brotes)
```

Histogramas.

```
> hist(propaga,nclass=4)
> hist(propaga,nclass=5,xlab="Propagación",col="yellow")

> hist(propaga,breaks = "Sturges",xlab="Propagación",col=8)
> hist(propaga,nclass=3,xlab="Propagación",col=4)
> hist(propaga,breaks = "Sturges",xlab="Propagación",col=10)
```

Barras superpuestas.

Tratamientos en los genotipos canchan y desiree

Extrae dos subconjuntos Canchan y Desiree del objeto "propagacion"

```
> canchan<-subset(propagacion,Genotipo=="Canchan-INIA")
> canchan
> desiree <- subset(propagacion,Genotipo=="Desiree")
> desiree
```

Crea nuevos objetos como vectores para juntarlos en una tabla.

```
> tratamiento<-as.vector(desiree$Tratamiento)
> desiree <-as.vector(desiree$brotes)
> canchan <-as.vector(canchan$brotes)
> datos<-data.frame(desiree,canchan,row.names=tratamiento)
> datos
```

Transforma el objeto datos en una matriz y se grafica.

```
> dd<-as.matrix(datos)

> barplot(dd,legend=rownames(dd))
> barplot(dd,legend=rownames(dd),col=c(4,5,6,7))
> barplot(dd,legend=rownames(dd),col=c(2,4,6,8), beside=T)
> barplot(dd,legend=rownames(dd),col=c(4,5,6,7),
beside=TRUE,density=c(10,20,30,40))
```

Grafico de cajas: boxplot()

Grafico de caja por cada genotipo.

```
> brotes <- propagacion$brotes
> genotipo<-propagacion$Genotipo
> tratamiento<- propagacion$Tratamiento
> boxplot(brotes~genotipo,col="yellowgreen")
```

También,

```
> plot(brotes~genotipo,col="yellowgreen")
> boxplot(brotes~tratamiento,col="whitesmoke")
> boxplot(brotes~tratamiento,col="blanchedalmond")
```

Se puede superponer gráficos de boxplot para fines comparativos.

```
>
boxplot(brotes~Tratamiento,data=propagacion,border="red",subset=Genotipo==”De
siree”,ylim=c(0,8))

>
boxplot(brotes~Tratamiento,data=propagacion,border="blue",subset=Genotipo==”C
anchan-INIA”,ylim=c(0,8),add = TRUE)

> legend(0.5,8, cex=0.8,c("Desiree", "Canchan-INIA"), fill = c("red",
"blue"))
```

Tallos y Hojas

Es una forma de visualizar la distribución de los datos, cuando estos no son muchos y puede realizarse manualmente. En el programa R se tiene la función stem para este proceso.

Considere el siguiente objeto:

```
> x<- c(58,4,73,68,82,60,1,69,60,36,15,6,63,56,86,62,68,48,3,89)
> stem(x)

   The decimal point is 1 digit(s) to the right of the |

  0 | 13465
  2 | 6
  4 | 868
  6 | 00238893
  8 | 269
```

Para descifrar este resultado, ordene los valores de “x”

```
> order(x)
 [1]  7 19  2 12 11 10 18 14  1  6  9 16 13  4 17  8  3  5 15 20
```

Estos valores indican la posición de los elementos dentro del objeto “x”

Para obtener los datos ordenados, se ubica como índice este resultado, así:

```
> x[order(x)]
 [1]  1  3  4  6 15 36 48 56 58 60 60 62 63 68 68 69 73 82 86 89

Se puede observar que en el:

grupo 0 (menos de 20)   : 01, 03, 04, 06, 15
grupo 2 (20 a < 40)     : 36
grupo 4 (40 a < 60)     : 48, 56, 58
grupo 6 (60 a < 80)     : 60, 60, 62, 63, 68, 68, 69, 73
grupo 8 (80 a mas )     : 82, 86, 89
```

Por lo tanto el grafico de tallos y hojas es:

```
  0 : 13465
  2 : 6
  4 : 868
  6 : 00238893
  8 : 269
```

Ejercicios.

Hallar las hojas y tallos del número de brotes

```
> stem(propagacion[,4])
```

Solo para el tratamiento "Will"

```
> stem(propagacion[propagacion[,2]=="Will",4])
```

O también:

Crear un subconjunto

```
> Will <- subset(propagacion,propagacion[,2]=="Will")
> stem(Will[,4])
```

Programación y creación de funciones

La programación es una herramienta fundamental para proporcionar nuevos procedimiento y funciones de acuerdo a sus necesidades, las funciones facilitan los procesos y pueden ser utilizadas por muchos programas en R.

Para crear sus funciones y procedimientos necesita un editor, para nuestro caso se usara Tinn-R.

Una función tiene las siguientes partes:

<u>Asignación y nombre de la función</u>

```
mifuncion <- function(a,b,c)
```

Mi función se llama "mifuncion" y tiene 3 parámetros "a, b y c"

Cuerpo de la función en encerrado entre llaves como en la programación C:

```
{
Instrucciones
 }
```

La primera llave es el inicio de la función, y la segunda llave es el final de la función.

Salida de datos de la función

return(x)

Esta función exporta el objeto “x” hacia el programa que este ejecutando la función

Ejemplo: una función para escribir un saludo.

Utilice Tinn-R y escriba:

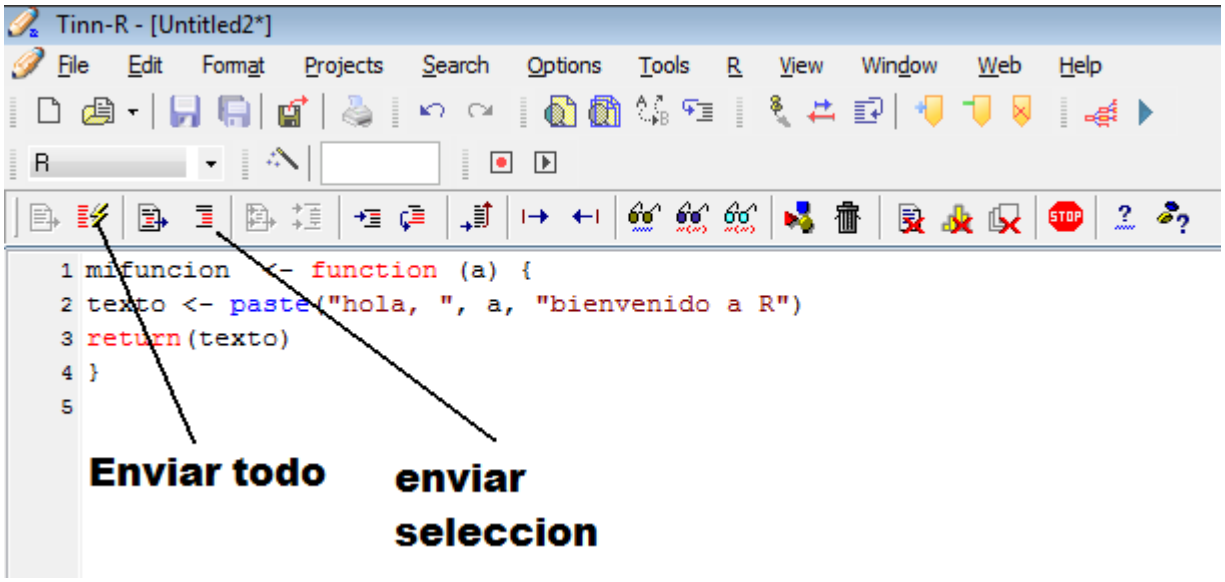
```
mifuncion  <- function (a) {
texto <- paste(“hola, ”, a, “bienvenido a R”)
return(texto)
}
```

Ejecute en R. Copiar y pegar en la consola la función

Ejecute la función mifuncion y coloque en el argumento su nombre entre comillas:

> mifuncion(“Maria”)

Desde Tinn-R puede enviar a R presionado un botón o seleccionando el procedimiento y luego ejecutando el botón correspondiente.



Ejercicio: crear una función para ordenar en forma ascendente o descendente los datos de una columna determinada.

Parámetros de la función:

File: Nombre del objeto de datos

Columna: numero de la columna
Criterio: Ascendente (FALSE) o descendente (TRUE)

Con estos parámetros se define la función:

Nombre: ordenar

Función: ordenar(file, columna, criterio)

Creación de la función en R (en NOTEPAD)

```
ordenar <- function(file, columna, criterio)
{
file[order(file[,columna], decreasing = criterio),]
}
```

Las llaves { } sirven para encerrar las instrucciones de la función.

Aplicación de la función ordenar.

Lectura de los datos.

```
> # Datos de morfología, seleccione la especie: SOLPHU; columna 1 es SPP
>  solphu<-subset(morfologia,morfologia[,1]=="SOLPHU")
```

En este objeto realice el ordenamiento por HABIT, corresponde a la columna 2.

```
> ordenar(solphu,2,F)
```

Guardar el resultado en el objeto salida.

```
> salida<-ordenar(solphu,2,F)
```

Por lo general las funciones tienen más de una instrucción.

Para el siguiente caso se requiere una función para hallar los valores estándar de una columna de datos, sin alterar los datos originales.

Un dato estándar (z) es igual al valor original menos el promedio, luego el resultado dividir por su desviación estándar:

```
z = (x -prom(x))/std(x)
```

Los parámetros para esta función son:

File: nombre del objeto (tabla de datos)
Columna: numero de la columna del objeto

La función será: estandar(archivo, columna)

```
# Estandariza valores de una columna de datos
estandar <- function(archivo, columna)
{
x<- archivo[,columna]
promedio<- mean(x,na.rm=T)
desv <- sqrt(var(x,na.rm=T))
archivo[,columna]<-(x-promedio)/desv
return(archivo)
}
```

En esta función se dispone de la instrucción “return” para exporar resultados.

En R se puede usar así:

```
> estandar(file,columna)
O
> nuevo<-estandar(file,columna)
```

Aplicación:

Utilizar el objeto “maca” que importado de SAS

```
> fix(maca)
```

Observe los datos y tome nota del nombre de las columnas.

Ejercicio: Realice los siguientes procesos:

a) estandarizar la columna peso
b) crear un nuevo objeto con datos estandarizados para peso y numero.

```
> estandar(maca,5)

> temporal<-estandar(maca,5)
> temporal<-estandar(temporal,6)
> fix(temporal)
```

Ejercicio.

Escriba una función para realizar las siguientes acciones:

1. Leer un archivo de datos numéricos con encabezados;
2. la función debe hacer un grafico de línea, donde la columna x debe ser ordenada de menor a mayor,
3. Finalmente hacer un grafico de “y” en función de “x”.

Como parámetros debe ser el nombre del archivo, los vectores “x”, “y” y el titulo del grafico.

La siguiente función escrita en Tinn-R realiza esta tarea:

```
miplot <- function(Fi, x, y, titulo="")
{
datos <- read.table(Fi,header=TRUE)
# ordena la primera columna de datos
datos<-datos[order(datos[,x]),]
nombres<-names(datos)
plot(datos[,x], datos[,y], frame=FALSE, xlab=nombres[x], ylab=nombres[y],
type="b", col="red")
title(titulo)
grid(col="blue",lty=2)
}
```

```
miplot        : es el nombre de la función
Fi,x,y,titulo : parámetros
{             : Llave de apertura, inicio de la función
}             : Llave de cierre, final de la función
```

El cuerpo de la función esta encerrado entre las llaves.
"miplot" realiza la siguiente operación:

1. Lee el archivo de datos con encabezados que se encuentra en el disco duro de nombre "Fi" y registra en el objeto "datos"
2. Ordena la columna indicada como "x" del objeto "datos"
3. Realiza un grafico de línea, donde la columna "x" representa el eje X y la segunda el Y
4. Escribe un titulo en el grafico de contenido "S"

Poner en practica la función miplot, utilice el archivo "suelo.txt" y obtenga gráficos de línea, relacionando los diferentes componentes del suelo.

```
> miplot("suelo.txt",1,2,"Relación de las componentes del suelo")
> miplot("suelo.txt",2,1,"Componentes del suelo ")
> miplot("suelo.txt",3,2,"Relación del suelo")
> miplot("suelo.txt",4,5,"Relación de las componentes del suelo \n en campos
dedicado al estudio de Ralstonia en CIP")
```

Elaboración de programas en R:

Utilice Tinn-R en el directorio correcto y crear una file en modo fuente R.

Este se almacenara en el disco duro, en la ruta establecida.

```
# mi primer programa
# determina el ranking estadístico
# mis datos están en el objeto x
rm(list=ls()) # elimina todos los objetos existentes.
x<- c( 14, 7, 8, 19, 23, 17, 7, 10)
rango <- rank(x)
y<- cbind(x,rango)
print(y)
# Termino mi programa
```

Guardar en el disco duro con el nombre “primero.R” y ejecutar desde R.

```
> source(“primero.R”)
```

Si su programa esta en otra dirección, escribir la ruta (path) apropiadamente, por ejemplo:

```
> source(“c:/UNI-R/primero.R”)
> source(“c:\\UNI-R\\primero.R”)
```

El programa puede ser editado por cualquier programa de texto en windows.

Importar datos de excel: archivo PAPA2001.xls mediante un programa

Escribir el siguiente programa en Tinn-R:

```
# Cargar la biblioteca
library(RODBC)
# establecer la comunicación
canal <- odbcConnectExcel("PAPA2001")
# cargar los datos de propagacion
propagacion <- sqlFetch(canal,"propagacion")
# cierra la comunicacion
odbcCloseAll()
# Fin de mi programa
```

Grabar el programa “`lee papa2001.R`”

Ejecutar en R:

```
> source(“lee papa2001.R”)
```

Ver los datos:

```
> propagacion
```

Ver su estructura:

```
> str(propagacion)
```

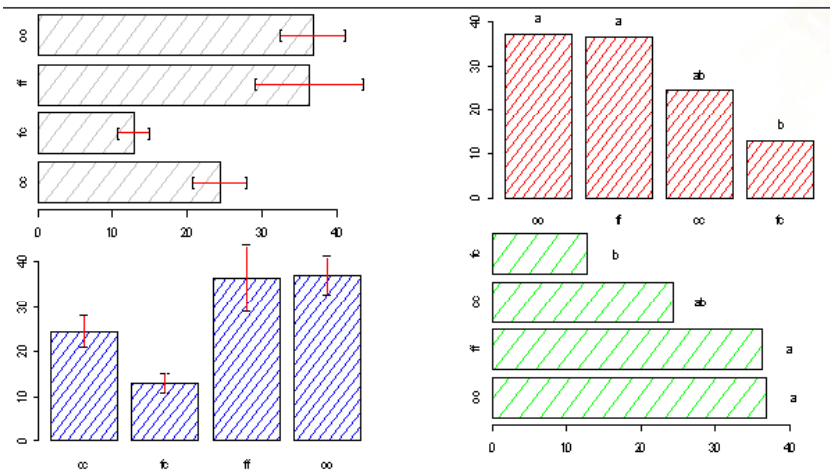
Ejercicios: Hallar funciones para calcular:

1. El área de un trapecio
2. El área de un triangulo conociendo el perímetro o la base y altura
3. Grados celsius a Fahrenheit
4. Galones americanos por acre a litros por hectárea.

Sesion 4

- Planeamiento de experimentos: Completamente randomizado, Bloques completos al azar, Cuadrado Latino, Greco latino, Bloques Incompletos balanceados y parcialmente balanceados, latice simple, diseños alfa, experimentos confundidos y experimentos fraccionados.
- Análisis de variancia de los diseños experimentales con uno o más factores.
- Comparación múltiple de tratamientos LSD, Bonferroni, Waller-Duncan y Tukey.
- Contrastes ortogonales.

bar.err & bar.group



Alpha design

(trt, k, r, number = 1, seed = 0, kinds = "Super-Duper")

```
> Trt <- letters[1:12]
> plan<-design.alpha(trt,k=3, r=2, number=101)
```

```
alpha design (0,1) - Serie  I
Parameters Alpha design
=======================
treatmeans : 12
Block size : 3
Blocks     : 4
Replication: 2

Efficiency factor

(E ) 0.6470588

<<< Book >>>
```

Field Book

```
> plan
    plots cols block trt replication
1     101    1     1   j           1
2     102    2     1   h           1
3     103    3     1   c           1
4     104    1     2   d           1
...
23    123    2     8   h           2
24    124    3     8   e           2
```

Diseño de experimentos.

Es planificación de la investigación por experimentación y comprende la selección del sitio adecuado, la aplicación de tratamientos y la elaboración de libros de campo para el registro de los datos experimentales según el diseño.

En R, se dispone de librerías apropiadas para este fin:

agricolae: diseños clásicos de experimentación
config.design: diseños confundidos
BHH2: Generación de la matriz diseño factorial completo y fraccionado
AlgDesign: Algoritmo de diseños experimentales
multcomp y mvtnorm: Para contrastes ortogonales.
experiment: randomizacion de tratamientos

Librería: agricolae.

Diseños clásicos, generación del libro de campo:

1) **Diseño completamente aleatorio CRD: 5 tratamientos con repeticiones diferentes**

```
> trt <- c("t1","t2","t3","t4","t5")
> rep <- c(4,5,3,3,4)
> campo1 <- design.crd(trt,rep)
> write.table(campo1,"hoja1.txt",sep="\t",row.names=FALSE)
> file.show("hoja1.txt")
```

2) **Bloques completos al azar: RCBD: 4 tratamientos en 5 bloques, semilla=45**

```
> trt<-c("A","B","C","D")
> rcbd <-design.rcbd(trt,5,number=101,45,"Super-Duper")
> trt <-as.character(rcbd[,3])
> dim(trt) <-c(4,5)
> print(t(trt))
```

3) **Cuadrado latino LSD: 4 tratamientos.**

```
> varieties<-c("perricholi","yungay","maria bonita","tomasa")
> lsd <-design.lsd(varieties,number=1001,seed=23)
> trt <-as.character(lsd[,4])
> dim(trt) <-c(4,4)
> print(t(trt))
```

4) **Greco Latino: 4x4**

```
> T1<-c("a","b","c","d")
> T2<-c("v","w","x","y")
> greco <-  design.graeco(T1,T2,number=101)
> trt <- paste(greco[,4],greco[,5])
> dim(trt) <-c(4,4)
> print(t(plots))
```

5) Diseño bloques incompletos balanceados: 4 tratamientos, tamaño del bloque 3, parcelas numeradas y semilla de randomizacion 41.

```
> trt<-c("A","B","C","D")
> k<-3
> bib <-design.bib(trt,k,number=101,seed =41,kinds ="Super-Duper")
> field <-as.character(bib[,3])
> t(matrix(field,c(3,4)))
```

6) Diseño alfa: 30 tratamientos, tamaño del bloque k=3 y repeticiones r=2.

```
> trt<-1:30
> t <- length(trt)
> # size block k
> k<-3
> # replications r
> r <- 2
> book<- design.alpha(trt,k,r)
```

7) Diseño latice simple: 100 tratamientos.

```
> lattice.simple(10)
```

Ejercicios:

1. DBCA para las siguientes variedades de papa nativa: “rosada jaspeada”, “ruky”, “rumu”,”rumy,”sacampaya” y “sani imilla” con 5 repeticiones c/u
2. DCA para 5 tratamientos (niveles de hongo para gorjojo) para igual repetición (5 c/u) y otra hoja de campo con diferente repetición c/u (4,3,5,4,6)
3. DCL para 5 tipos de trampa para mosca minadora en parcelas grandes.

Libreria: experiment.

Ejemplos:

1. Completamente al azar 2 tratamientos con igual repetición.

```
> library(experiment)
> # Parcelas numeradas del 101 al 116
> set1 <- data.frame(nro=101:116)
> set2 <- randomize(set1,group=c("treat","control"))
> plan<-data.frame(set1, trt=set2$treatment)
> plan
```

2. Bloques completos al azar 3 tratamientos en 4 bloques.

```
> # Parcelas numeradas del 101 al 112
> set1 <- data.frame(nro=101:112)
> set2 <- randomize(set1,group=c("A","B","C"), block=c(1:4))
> plan<-data.frame(set1,block=set2$block.id, trt=set2$treatment)
> plan
```

Análisis de datos experimentales

Asignar como factores a: trt, trt1, bloques, fila, columna.

Completamente al azar: `Y ~ trt`
Bloques completos al azar: `Y ~ bloques + trt`
Cuadrado latino: `Y ~ filas + columnas + trt`
Greco latino: `Y ~ filas + columnas + trt + trt1`

Análisis del diseño completo al azar

```
> datos <- read.table("virus.txt", header=TRUE)
> datos[,1] <- as.factor(datos[,1])
> #
> modelo <-aov(rdto ~ virus, datos)
> anova(modelo)
> gle <- df.residual(modelo)
> cme <- deviance(modelo)/gle
> attach(datos)
> compara <- LSD.test(rdto, virus, gle, cme, adjust="benferroni")
> detach(datos)
```

Análisis de diseño de bloques completos

```
> datos <- read.table("rendimiento.txt",header=TRUE)
> #
> datos[,1] <- as.factor(datos[,1])
> attach(datos)
> rdto<-comercial+no.comercial
> modelo <-aov(rdto ~ B + A, datos)
> anova(modelo)
> cv.model(modelo)
> gle <- df.residual(modelo)
> cme <- deviance(modelo)/gle
> attach(datos)
> compara <- LSD.test(rdto, virus, gle, cme, adjust="benferroni")
> detach(datos)
```

Análisis del diseño cuadrado latino

```
> datos <- read.table("melon.txt",header=TRUE)
> datos[,1] <- as.factor(datos[,1])
> datos[,2] <- as.factor(datos[,2])
> attach(datos)
> modelo<-lm(rdto ~ fila + columna + trat,conj4)
> cv.model(modelo)
> gle <- df.residual(modelo)
> cme <- deviance(modelo)/gle
> attach(datos)
> compara <- HSD.test(rdto, virus, gle, cme, adjust="benferroni")
```

Para los siguientes procesos, se sugiere hacer programas en R. Utilice la siguiente orden para crear y editar programas.

Procedimientos en Tinn-R: lectura, análisis de variancia y comparación de medias y grupos de tratamientos mediante contrastes.

```
camote <- read.table("virus.txt", header=TRUE)
camote[,1]<-as.factor(camote[,1])
modelo1 <-lm(rdto ~ virus, camote)
anova(modelo1)
cv.model(modelo1)
gle <- df.residual(modelo1)
cme <- deviance(modelo1)/gle
attach(camote)
comparacion <- LSD.test(rdto, virus, gle,cme)
```

Usar bonferroni.

```
> comparacion <- LSD.test(rdto, virus, gle,cme, p.adj="bonferroni")
```

Usar tukey

```
> comparacion <- HSD.test(rdto, virus, gle,cme)
```

Usar Waller y Duncan.

Requiere el valor del Fc. Del ANVA correspondiente al factor de comparación.

```
> Fc <- anova(modelo1)[1,4]
> comparacion <- waller.test(rdto, virus, gle,cme, Fc)
```

Graficar los resultados de la comparación, density = 4 es para hacer cajas rayadas, es un parámetro de gráficos.

```
> bar.group(comparacion, ylim=c(0,45),density=4, col="blue")
```

Realizar contrastes ortogonales.

Para realizar contrastes, es necesario construir la matriz de coeficientes del contraste. Si OO es el testigo, entonces este se compara con los demás (Test de Dunnett):

```
library(multcomp)
library(mvtnorm)
camote <- read.table("virus.txt", header=TRUE)
camote[,1]<-as.factor(camote[,1])
contraste <- rbind(c(1,0,0,-1),c(0,1,0,-1),c(0,0,1,-1))
filas<-c("CC vs OO","FC vs OO","FF vs OO")
columnas<-c("CC","FC","FF","OO")
dimnames(contraste)<-list(filas,columnas)
attach(camote)
modelo<-aov(rdto~virus, data=camote)
compara <-glht(modelo, linfct = mcp(virus = contraste))
summary(compara)

> contraste
```

```
         CC FC FF OO
CC vs OO  1  0  0 -1
FC vs OO  0  1  0 -1
FF vs OO  0  0  1 -1
```

Análisis del diseño de bloques incompleto balanceado. BIB

```
> # Example linear estimation and design of experiments. D.D. Joshi. 1987
> # Professor of Statistics, Institute of Social Sciences Agra, India
> # 6 varieties of wheat crop in a BIB whit 10 blocks of 3 plots each.
> y <-c(69,77,72,63,70,54,65,65,57,59,50,45,68,75,59,38,60,60,62,
      55,54,65,62,65,61,39,54,67,63,56)
> varieties<-gl(6,5)
> block <- c(1,2,3,4,5,1,2,6,7,8,1,3,6,9,10,2,4,7,9,10,3,5,7,8,9,4,5,6,8,10)
> model<-BIB.test(block, varieties, y)
```

Análisis del diseño de bloques incompleto parcialmente balanceado. Aplicado a latices cuadrado, rectangular y diseños alfa con la funcion `PBIB.test()`**.**

Un diseño alfa de 30 tratamientos con 2 repeticiones y bloques de tamaño 3.

```
> library(agricolae)
> library(corpcor)
> # alpha design
> trt<-1:30
> ntr<-length(trt)
> r<-2
> k<-3
> s<-10
> obs<-ntr*r
> b <- s*r
> book<-design.alpha(trt,k,r,seed=5)
```

Se simula datos al plan generado

```
> book[,3]<- gl(20,3)
> # dataset
> y<-c(5,2,7,6,4,9,7,6,7,9,6,2,1,1,3,2,4,6,7,9,8,7,6,4,3,2,2,1,1,2,
    1,1,2,4,5,6,7,8,6,5,4,3,1,1,2,5,4,2,7,6,6,5,6,4,5,7,6,5,5,4)
> book<-data.frame(book,y=y)
> rm(y,trt)
```

Realiza el análisis del diseño alfa.

```
> attach(book)
> model <- PBIB.test(block, trt, replication, y, k=3)
> detach(book)
```

Ejercicios:

Con el archivo rendimiento.txt.

1. Use el rendimiento comercial y no comercial y el total
2. Halle el cuadro del ANVA según el diseño de bloques.
3. Halle el rendimiento promedio total y el coeficiente de variación.
4. Realice las comparaciones múltiples.

R para diseños confundidos.

Requiere la biblioteca: **conf.design**

Valido solo para factorial $\mathbf{2^n, 3^n, 4^n}$, etc

Ejemplos:

1) Confundir la interacción AB en el factorial $\mathbf{2^2}$.

```
> D22 <- conf.design(c(1,1), p=2, treatment.names=LETTERS[1:2])
```

2) Confundir la interacción ABC en el factorial $\mathbf{2^3}$.\

```
D222 <- conf.design(c(1,1,1), p=2, treatment.names=LETTERS[1:3])
```

3) Confundir la interaccion AB en el factorial $\mathbf{2^3}$.

```
Dab <- conf.design(c(1,1,0), p=2, treatment.names=LETTERS[1:3])
```

R para diseños factoriales fraccionados.

R dispone de una librería: BHH2, definida por
“ Full or fractional factorial design matrix generation”

Es el generador de la matriz diseño de factoriales fraccionados.

La función a usar es: ffDesMatrix(k, gen = NULL)

K: representa el número de factores de 2 niveles
Gen: los generadores.

Ejemplos:

1) 2^{4-1} factorial ½ replicación con 4 factores; el generador es D=ABC

```
> ffDesMatrix(4,gen=list(c(4,1,2,3)))
```

2) 2^{4-2} factorial ¼ replicación con 4 factores; el generador es C=AB, D=AC significa de resolución III.

```
> ffDesMatrix(4,gen=list(c(3,1,2),c(4,1,3)))
```

Cuando es poco los factores y niveles no es recomendable, los diseños son muy pequeños, no se tiene buena estimación; en este ejemplo solo se tendria 4 datos para hacer el análisis.

3) 2^{5-2} factorial ¼ replicación con 5 factores; los generadores son D=AB, E=-AC significa de resolución III.

```
> ffDesMatrix(5,gen=list(c(4,1,2),c(-5,1,3)))
```

4) 2^{5-1} factorial ½ replicación con 5 factores; el generador es E=ABCD. Resolución V

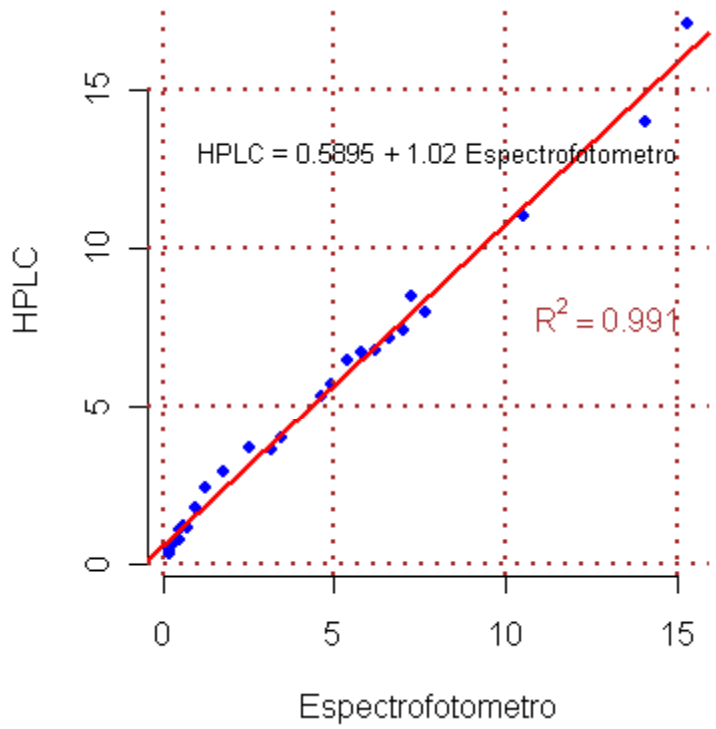
```
> ffDesMatrix(5,gen=list(c(5,1,2,3,4)))
```

Si se multiplica las columnas 1x2x3x4x5 el resultado siempre es +1, eso significa que esta confundido la interacción de mayor nivel ABCDE

Sesión 5

- Correlación
- Análisis de regresión lineal simple.
- Grafico y limites de confianza
- Regresión lineal múltiple.
- Análisis de residuales
- Selección de variables para el mejor modelo

-



Análisis de correlación lineal simple.

Mediante R, consiste en hallar las correlaciones y los correspondientes p-value mediante tres métodos: Pearson, Spearman y Kendall.

En R se tiene la función cor() y en agricolae la función correlation(), en ambos se encuentra las correlaciones, en este ultimo permite los pvalue de todas las relaciones.

correlation(x,y,method="kendall"), para otros cambiar por "spearman" o "pearson" que es por defecto.

Pearson: Cuando las variables son continuas y distribuidas normalmente.
Spearman y Kendall: Cuando los datos son categóricos o no-normales

En todo estudio de correlación es necesario graficar la asociación.

Para datos sensoriales sabor y textura de 25 variedades de papa nativa utilice el método de Spearman o Kendall.

Lectura de los datos

```
> library(agricolae)
> datos <- read.table("sensorial.txt",header=T)
> sensorial<- tapply.stat(datos[,2], datos[,3:4], mean)
> attach(sensorial)
> cor(textura,sabor,method="spearman")
```

Si la correlación es cero entre limo y altura

```
> data(growth)
> attach(growth)
> correlation(slime,height,method="kendall")
```

Correlación entre muchas variables

Si A es un data.frame en las cuales las columnas son las variables, entonces la correlación entre todas las variables se puede hallar directamente con cor(A) para cualquier método, el resultado será una matriz de correlaciones. Si requiere la validez estadística de cada correlación, utilizar la función correlation() de agricolae

Para este estudio, utilice el archivo "análisis sensorial.txt"
El archivo tiene 6 columnas, solo se debe utilizar las columnas 3, 4, 5 y 6
Solo para el tratamiento = 1

```
> evaluacion<-read.table("analisis sensorial.txt",header=T)
> trat.1<-subset(evaluacion[,c(-1,-2)],evaluacion$trat==1)
> trat.1
> cor(trat.1,method="kendall")
```

```
> correlation(trat.1,method="spearman")
```

Análisis de regresión lineal simple.

En R, se dispone de la funcion lm(), modelo lineal,

Para el ejemplo se utilizara el file Glycoalkaloids.txt

Los glicocoalcaloides son compuestos antinutricionales que se presentan en cantidades muy bajas en la papa destinada al consumo humano. Aunque estos compuestos existen en forma natural en el tubérculo, sus concentraciones podrían aumentar si las plantas son sometidas a diferentes situaciones de estrés, lo que se manifiesta en el sabor amargo de las papas. Los glicocoalcaloides en papa son principalmente de dos tipos: a-solanina y a-chaconina, se les conoce con estos nombres por ser alcaloides unidos a moléculas de azucares (glucosa, galactosa y sacarosa)

Los datos corresponden a muestras deshidratadas (liofilizadas) de papa se analizó el contenido de glicoalcaloides totales (TGA) por el método espectrofotométrico (Hellenas (1986) y Wang et al. (1976)) y por el método de HPLC.

El resultado del análisis de regresión obtuvo la siguiente ecuación de predicción: HPLC = 0.5895 + 1.02 E (E= TGA mg/100g peso fresco obtenido por el método espectrofotométrico), con un $r^2 = 0.99$ (P^3 0,95),

```
> glic <- read.table("Glycoalkaloids.txt",header=TRUE)
```

La variable HPLC es dependiente
La variable Espectrofotometro es independiente

Construir el modelo de regresion lineal

```
> modelo <- lm(HPLC ~ Espectrofotometro, data=glic)
```

Análisis del modelo

```
> summary(modelo)
> plot(modelo)
```

Información estadística almacenada en modelo

```
> names(modelo)
```

Extraer información del modelo: los residuales, los valores estimados por el modelo

```
> modelo$residuals
> modelo$fitted.values
```

Grafique los valores observados y superponga la línea de regresión

```
> HPLC <- glic$HPLC
> Espectrofotometro <- glic$Espectrofotometro

> plot(Espectrofotometro, HPLC,col="blue",pch=19,frame=F)
> abline(modelo, col="red",lwd=2)
> grid(col="brown",lwd=2)
```

```
> text(8,13,cex=0.8,"HPLC = 0.5895 + 1.02 Espectrofotometro")
> Rsq <- summary(modelo)$ "r.squared"
>
text(13,8,as.expression(substitute(R^2==r,list(r=round(Rsq,3)))),col="brown",
lwd=2)
```

Halle los intervalos de confianza

```
> pred <- predict(modelo,interval="confidence", level = 0.9999)
```

Junte los valores obervados con los predichos

```
> x <- data.frame(HPLC, Espectrofotometro, pred)
```

Ordene los datos por valores Espectrofotometro

```
> x <- x[order(Espectrofotometro),]

> HPLC <- x[,1]
> Espectrofotometro <- x[,2]
> lineas <- cbind(x[,3],x[,4],x[,5])
```

Grafique el modelo y los límites de confianza

```
> matplot(Espectrofotometro,lineas,type="l",lty=c(1,5,5),col=c(1,6,6),
xlim=c(0,18),ylim=c(0,18),ylab="HPLC",lwd=2,frame=F)
> points(Espectrofotometro,HPLC,col="blue",pch=19)
> grid(col="brown",lwd=2)
```

Observaciones al presente gráfico .

```
> par(new=TRUE)
> plot(HPLC,Espectrofotometro,cex=0.5,col=4,axes=FALSE, xlim=c(0,18),
ylim=c(0,18), ylab="")
> par(new=FALSE) # elimina la superposicion
```

Regresión Múltiple.

Una variable aleatoria en función de variables fijas independientes.

R, dispone de la función lm() linear model para estudiar modelos lineales, uno de los casos es la regresión múltiple.

Para la aplicación, se considero el comportamiento de la bacteria wilt en 13 suelos diferentes donse se cultiva de papa en Perú y esta afecto a la enfermedad marchites bacteriana. La evolución de la enfermedad se cuantifico en periodos de tiempo igual, 7 veces y se calculo el valor de AUDPC (avance del progreso de la enfermedad del cultivo).

AUDPC: Variable dependiente
Suelo: Variables independientes

El objetivo es estudiar que variables del suelo están mas relacionas con el progreso de la enfermedad y si es posible construir un modelo predictivo.

Correlación del AUDPC con las medidas de suelo.

```
> library(agricolae)
> suelo <- read.table("suelo con bacteria wilt.txt", header=TRUE)
> correlation( suelo[,1],suelo[,-1])
```

Se observa que hay muchas variables que están correlacionadas, entonces se puede hacer un análisis de regresión múltiple.

Las variables mas correlacionadas: pH CE CaCO3 MO CIC P K Arena Limo Arcilla Ca, B, Fe

```
> modelo <- lm( audpc ~ . , suelo)

> modelo <- lm( audpc ~ pH + CE + CaCO3 + P + K + Arena + Arcilla + Ca + B +
Fe, suelo)
```

```
Según el análisis, Múltiple R-Squared: 0.9325.

El siguiente proceso es una selección por pasos para buscar el mejor modelo.

Función step()
```

```
> estudio <- step(modelo, k=4)
```

```
k es un parámetro de selección por el criterio AIC, el valor aproximado es
log(n), donde n es el número de datos.

En el último paso se tiene:

Step:  AIC=-111.89
audpc ~ pH + P + Fe

       Df Sum of Sq      RSS      AIC
<none>                 0.001 -111.893
- P     1 0.0003621    0.001 -110.435
- Fe    1     0.001    0.001 -106.272
- pH    1     0.003    0.003  -95.443

Significa que la respuesta audpc depende del pH, P y Fe.

En el resumen:
```

```
> summary(estudio)
```

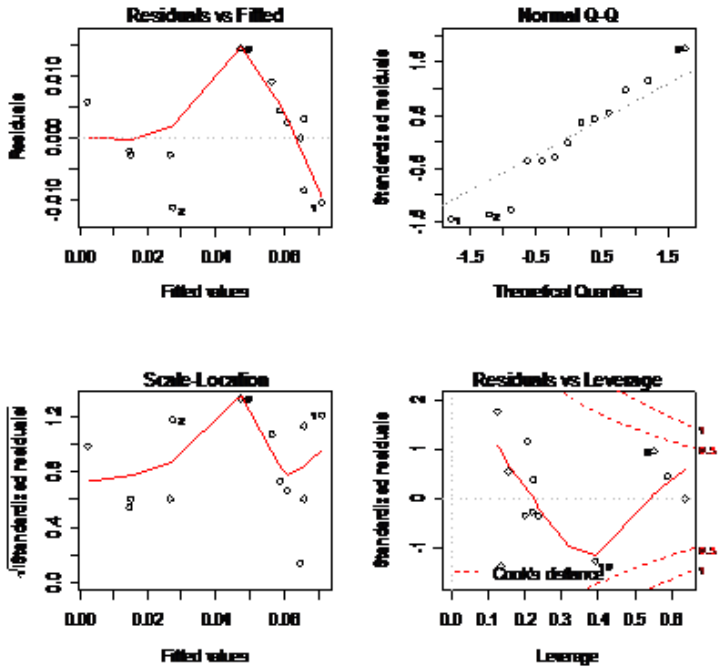
```
Múltiple R-Squared: 0.9074

Haciendo un análisis de residuales del modelo seleccionado:
```

```
> par(mfrow=c(2,2))
> plot(estudio)
```

Los 4 gráficos corresponden a:

1. Falta de ajuste
2. Normalidad de los errores
3. Magnitud de los errores
4. valores extremos.



Del grafico (1) se puede hacer un ajuste más al modelo con términos cuadráticos por ejemplo:

```
> nuevo <- lm( audpc ~ pH + P + Fe + I(pH^2) + I(P^2) + I(Fe^2) + I(pH*P) +
I(pH*Fe) + I(P*Fe)    , data = suelo)

> summary(nuevo) se obtiene un R cuadrado de:

Múltiple R-Squared: 0.9863

> estudio <- step( nuevo, k=4)
```

En el ultimo paso se tiene:

```
Step:  AIC=-123.97
audpc ~ pH + Fe + I(Fe^2) + I(pH * Fe) + I(P * Fe)

             Df Sum of Sq       RSS      AIC
<none>                    0.0001481 -123.970
- I(P * Fe)   1 0.0002118 0.0003599 -116.429
- Fe          1 0.0002738 0.0004220 -114.362
- I(Fe^2)     1 0.0003254 0.0004735 -112.863
- I(pH * Fe)  1     0.001     0.001 -106.834
- pH          1     0.001     0.001  -99.103

> summary(estudio)
```

Múltiple R-Squared: 0.9802

Modelo resultante:

```
> estudio

Call:
lm(formula = audpc ~ pH + Fe + I(Fe^2) + I(pH * Fe) + I(P * Fe),     data =
suelo)

Coefficients:
(Intercept)           pH           Fe      I(Fe^2)   I(pH * Fe)
  1.032e-01   -1.141e-02   -1.072e-04   -6.754e-08    3.486e-05
  I(P * Fe)
  8.240e-07
```

audpc = 0.1032 -0.01141 pH -1.072e-04 Fe -6.754e-08 Fe^2 + 3.486e-05 pH*Fe + 8.240e-07 P*Fe

con R^2 = 0.98

Según el modelo propuesto hacer la predicción del audpc para la siguiente información de un suelo caracterizado:

pH = 3.8
Fe = 295.4
P = 8.1

```
> predict(estudio,new=data.frame(pH=3.8, Fe=295.4, P=8.1))
```

audpc = 0.06335923

El valor observado fue: 0.0605

Ejercicio.

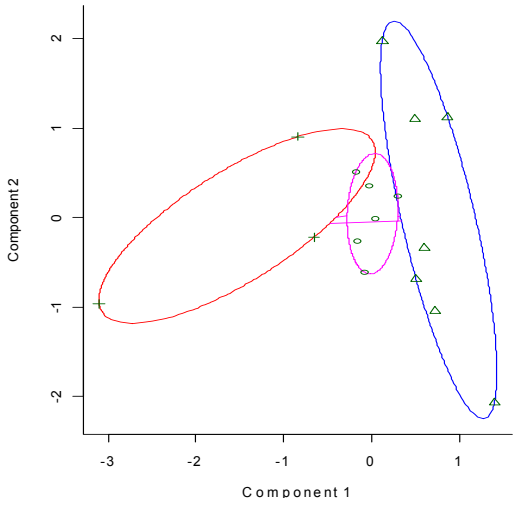
Construir el mejor modelo de regresión múltiple. Archivo: "oca.txt"

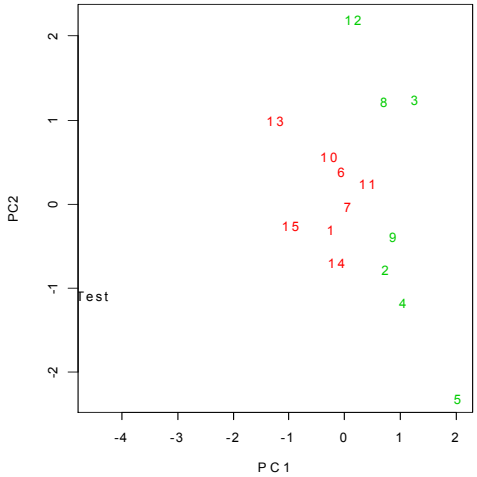
La variable dependiente: Grados brix (V6)

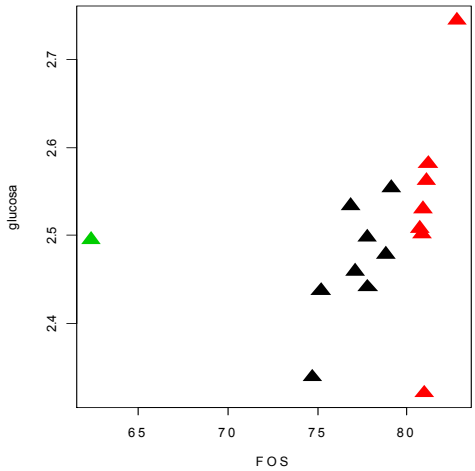
Materia seca	V1	g
Rend. almidón	V2	g
Almidón total	V3	g
Azúcar total	V4	g
Az. reductor	V5	g
Brix	V6	°B
Acidez	V7	%
Oxalato total	v8	mg
Acido ascorbico	v9	mg
Proteína	v10	g
Ceniza	v11	g
Fibra	v12	g
Grasa	v13	g

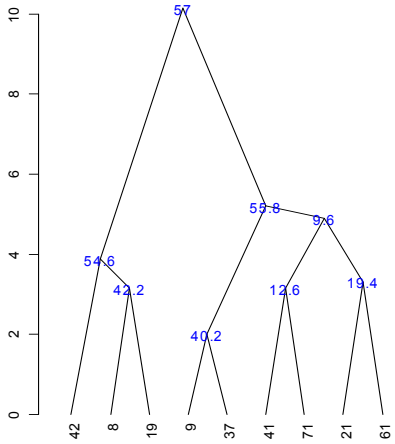
Sesión 6

- 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.txt”

FOS, glucosa, fructosa y sacarosa

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

```
> azucar <- read.table("azucares.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.

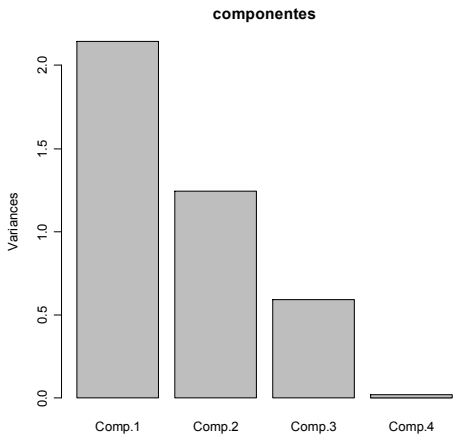
```
> componentes<-princomp(datos, cor=FALSE)
> summary(componentes)
```

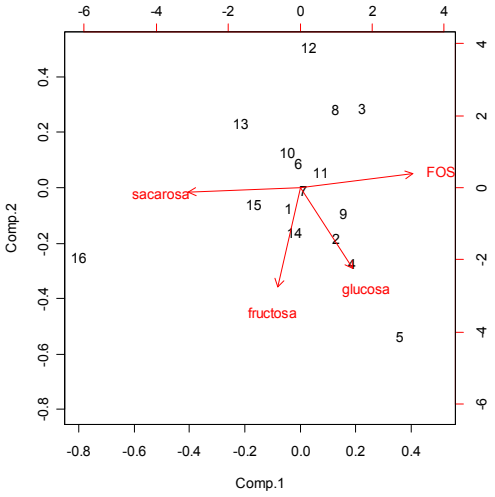
```
Importance of components:
                          Comp.1     Comp.2      Comp.3       Comp.4
Standard deviation     5.3956993 0.61247696 0.301047465 0.0835644307
Proportion of Variance 0.9840216 0.01267911 0.003063227 0.0002360218
Cumulative Proportion  0.9840216 0.99670075 0.999763978 1.0000000000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

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.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.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
```