



# UNIVERSIDAD NACIONAL AGRARIA LA MOLINA

FACULTAD DE ECONOMIA Y PLANIFICACIÓN  
DEPARTAMENTO DE ESTADISTICA E INFORMATICA



*Agricolae*

versión 1.1-6

## MANUAL PRÁCTICO PARA EL USO DE AGRICOLAE

Ing. Estadístico Felipe de Mendiburu (\*)  
Magister en Ingeniería de Sistemas

Diciembre - 2013

---

\* *Profesor Principal. Dpto. Estadística e Informática. Universidad Nacional Agraria La Molina.*  
\* *Researcher. International Potato Center*

## INDICE GENERAL

	Página
PRESENTACIÓN	1
1. INSTALACIÓN DE AGRICOLAE Y USO EN R	1
1.1 INSTALACIÓN	1
1.2 USO EN R	2
2. ESTADÍSTICA DESCRIPTIVA	3
2.1 HISTOGRAMA	3
2.2 MANEJANDO ESCALAS	4
2.3 TABLAS DE FRECUENCIA Y ESTADÍSTICAS	4
2.4 REPRODUCIENDO HISTOGRAMAS Y EL USO DE hist()	5
2.5 HISTOGRAMA A PARTIR DE DATOS AGRUPADOS	6
2.6 JUNTANDO CLASES	6
3. DISEÑO DE EXPERIMENTOS	6
3.1 COMPLETAMENTE AL AZAR	7
3.2 BLOQUES COMPLETOS AL AZAR	8
3.3 CUADRADO LATINO	8
3.4 GRECO LATINO	8
3.5 BLOQUES INCOMPLETOS BALANCEADOS	9
3.6 CÍCLICO	9
3.7 LATICE	10
3.8 ALFA LATICE	11
3.9 BLOQUES AUMENTADOS	12
3.10 PARCELAS DIVIDIDAS EN BLOQUES (SPLIT-PLOT )	13
3.11 BLOQUES DIVIDIDOS (STRIP-PLOT)	14
3.12 FACTORIALES	15

4. ANÁLISIS DE DISEÑOS EXPERIMENTALES	16
4.1 LSD	17
4.2 BONFERRONI	18
4.3 DUNCAN' NEWS	19
4.4 STUDENT-NEWMAN-KEULS (SNK)	20
4.5 TUKEY (HSD)	21
4.6 WALLER DUNCAN	21
4.7 SCHEFFE	23
4.8 COMPARACIÓN MÚLTIPLE DE UN FACTORIAL	24
4.9 GRÁFICOS DE LA COMPARACIÓN MÚLTIPLE	27
4.10 ANÁLISIS DE BLOQUES INCOMPLETOS BALANCEADOS	28
4.11 BLOQUES INCOMPLETOS PARCIALMENTE BALANCEADOS	30
4.12 BLOQUES AUMENTADOS	36
4.13 COMPARACIONES NO-PARAMÉTRICAS	38
4.14 KRUSKAL – WALLIS	39
4.15 FRIEDMAN	40
4.16 WAERDEN	41
4.17 DURBIN	42
5. ANÁLISIS DE ESTABILIDAD	43
5.1 ESTABILIDAD PARAMÉTRICA	43
5.2 ESTABILIDAD NO PARAMÉTRICA	45
5.3 AMMI	46
6. FUNCIONES ESPECIALES	48
6.1 CONSENSO DE DENDROGRAMA	48
6.2 MONTECARLO	50
6.3 RE-MUESTREO EN MODELO LINEAL	51
6.4 SIMULACIÓN EN MODELO LINEAL	51
6.5 ANÁLISIS PATH	53
6.6 LINEA POR PROBADOR	53
6.7 UNIFORMIDAD DEL SUELO	56
6.8 LÍMITES DE CONFIANZA EN ÍNDICES DE BIODIVERSIDAD	57
6.9 CORRELACIÓN	57
6.10 OTRAS FUNCIONES	58
REFERENCIAS BIBLIOGRÁFICAS	65

## PRESENTACIÓN

R es un sistema de programación funcional exclusivo para el manejo de datos en estadística y ciencias afines como las matemáticas, en ambientes como Windows, Linux y Mac y “agricolae” es un paquete de funciones para R aplicadas a la investigación agrícola.

El paquete “agricolae” ofrece una amplia funcionalidad en el diseño de experimentos, especialmente para experimentos en la agricultura y mejoramientos de plantas, los cuales pueden también ser usados para otros propósitos. Contiene los diseños latice, alfa, cíclico, bloques incompletos balanceados, bloques completos aleatorios, latino, greco latino, diseños aumentados en bloques, parcelas divididas, bloques divididos; también tiene varios procedimientos de análisis de datos experimentales, tales como las comparaciones de tratamientos de Waller-Duncan, Bonferroni, Duncan, Student-Newman-Keuls, Scheffe o los clásicos como LSD y Tukey, y comparaciones no-paramétricas, como Kruskal-Wallis, Friedman, Durbin y Waerden, análisis de estabilidad y otros procedimientos aplicados en genética, así como también procedimientos en biodiversidad y estadística descriptiva.

Para más detalles sobre el uso de “agricolae” se encuentran el manual de referencia y el sistema de ayudas del sistema en html, los cuales pueden ser invocados desde el menú de R y en la web: <http://tarwi.lamolina.edu.pe/~fmendiburu/>

## 1 INSTALACIÓN DE AGRICOLAE Y USO EN R

### 1.1 INSTALACIÓN

El programa principal de R debe ser instalado en la plataforma de su computadora (Windows, Linux o Mac). Si aún no está instalado, éste debe ser descargado del proyecto R ([www.r-project.org](http://www.r-project.org)) del CRAN de un repositorio (R Development Core Team, 2013). Como es un programa libre, no se requiere ninguna identificación. Los paquetes pueden ser incorporadas mediante un proceso de instalación, directamente desde la plataforma de R.

“Agricolae” es un paquete para R; como tal, su instalación es igual que cualquier otro paquete de R.

Para Windows se requiere el programa R versión 3.0.0 o superior.

Si el programa R está instalado en Windows o en otra plataforma, la instalación de “agricolae” puede hacerse directamente desde la consola de R en conexión con Internet, así:

```
install.packages("agricolae")
```

Se selecciona un repositorio y el sistema se instala automáticamente.

Si no se tiene conexión a Internet, es necesario copiar de la página del proyecto R el archivo agricolae\_x.x-x. zip para Windows.

El archivo agricolae\_ x.x-x.zip (De Mendiburu, 2013) puede ser descargado del repositorio de R en la siguientes direcciones: [www.r-project.org](http://www.r-project.org) o de la dirección <http://cran.at.r-project.org/web/packages/agricolae/index.html>

El archivo puede ser incorporado directamente a R instalando desde la consola ejecutando la siguiente instrucción si el archivo está en la dirección E:  
`install.packages("E:/agricolae_x.x-x.zip")`

También puede ser instalado desde el menú de R:

Packages, Install package(s) from local zip files ... . Seleccionar el archivo zip no requiere desempaquetar.

“agricolae” para su completa funcionalidad requiere otros paquetes.

MASS: para la inversa generalizada utilizada en la función `PBIB.test()`  
nlme: para los metodos REML and LM en `PBIB.test`  
klaR: para la función `triplot()` utilizada en la función `AMMI()`  
akima: para el uso de la función `interp()` utilizada en `grid3p()` para interpolación.  
Cluster: para el uso de la función `consensus()`

## 1.2 USO EN R

Como “agricolae” es un paquete de funciones, éstas son operativas cuando se las invoca directamente desde la consola de R y son integradas a todas las funciones de Base de R.

Las siguientes órdenes son frecuentes:

Cargar el paquete a la memoria: `library(agricolae)`

Descargar: `detach(package:agricolae)`

Una vez cargado el paquete, éste puede ser usado así:

Listar la base de datos: `data(package="agricolae")`

Cargar los datos de camote: `data(sweetpotato)`

Ver su estructura: `str(sweetpotato)`

Editar su contenido: `fix(sweetpotato)`

Para continuar con la línea de comandos, siempre debe cerrar las ventanas abiertas con alguna orden de R.

Para ayudas: `help(sweetpotato); ? sweetpotato`

Para la búsqueda de alguna función: `apropos("design")`

```
[1] "design.ab"          "design.alpha"      "design.bib"        "design.crd"
[5] "design.cyclic"      "design.dau"        "design.graeco"      "design.lattice"
[9] "design.lsd"         "design.rcbd"       "design.split"       "design.strip"
```

Para uso de símbolos que no figuran en el teclado en español, como por ejemplo: ~, [, ], &, ^, |, <, >, {, }, \, % u otros, utilice la tabla 6.10.

## 2 ESTADÍSTICA DESCRIPTIVA

El paquete “agricolae” proporciona algunas funciones complementarias al programa R, específicamente para el manejo del histograma.

### 2.1 HISTOGRAMA

El histograma es construido con la función `graph.freq()` y está asociado a otras funciones: `polygon.freq`, `table.freq`, `stat.freq`, `intervals.freq`, `sturges.freq`, `join.freq`, `ojiva.freq` y `normal.freq`.

Ejemplo 1.1 Datos generados en R. (peso de estudiantes). Figura 2.1

```
c( 68, 53, 69.5, 55, 71, 63, 76.5, 65.5, 69, 75, 76, 57, 70.5, 71.5,
56, 81.5, 69, 59, 67.5, 61, 68, 59.5, 56.5, 73, 61, 72.5, 71.5,
59.5, 74.5, 63) -> peso
```

cargar el paquete “agricolae” :

```
library(agricolae)
par(mfrow=c(2,2))
h1<- graph.freq(peso, col="yellow", frequency =1, axes= FALSE,
main="frecuencia absoluta\nh1")
axis(1,h1$breaks); axis(2, las=1)
h2<- graph.freq(peso, frequency =2 , axes= FALSE, main="polígono de
frecuencia\nh2")
polygon.freq(h2, col="blue", lwd=2, frequency =2)
TIC<- h2$breaks[2]- h2$breaks[1]
axis(1,c(h2$mids[1]-TIC, h2$mids, h2$mids[6]+TIC ),cex=0.6)
axis(2, cex=0.6,las=1)
h3<- graph.freq(peso, col="brown", frequency =3, axes= FALSE,
main="densidad\nh3")
axis(1,h3$breaks); axis(2,las=1)
h4<- graph.freq(peso, col="blue", frequency =3, axes= FALSE,
main="densidad normal\nh4", density=4)
normal.freq(h4, col="red", lty=4,lwd=2, frequency=3)
axis(1,h3$breaks); axis(2,las=1)
```

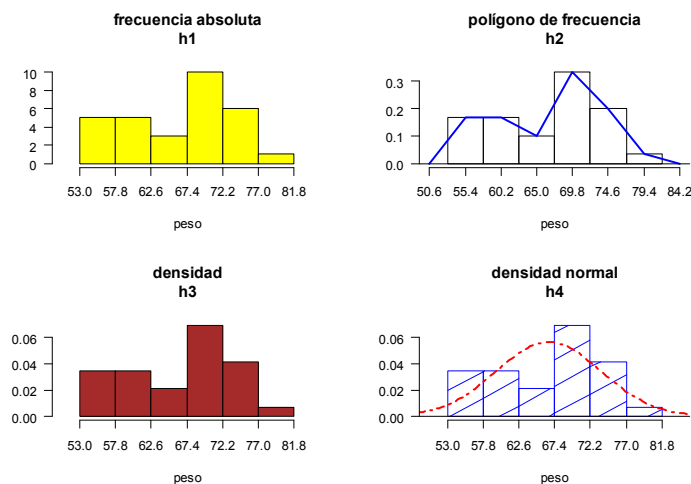


Figura 2.1 Histogramas, polígono y densidad

## 2.2 MANEJANDO ESCALAS

Se refiere a los cambios de escala en los ejes. Figura 2.2

```
h5<- graph.freq(peso, axes=FALSE, frequency =1, main="frecuencia
absoluta\nh5")
axis(1,h5$breaks,las=2); axis(2,las=1)
h6<- graph.freq(peso, axes=FALSE, nclass=5, main="frecuencia con 5
clases\nh6")
axis(1,h6$breaks,las=2); axis(2,las=1)
normal.freq(h6,col="red")
h7<- graph.freq(peso, density=6, col="blue", frequency =3,
axes=FALSE, main="densidad\nh7")
lines(density(peso),col="brown",lwd=2)
axis(1,h7$breaks,las=2); axis(2,las=1)
h8<- graph.freq(peso, border=0, frequency =3, axes=FALSE,
main="polígono y densidad\nh8")
polygon.freq(h8,col="blue", frequency =3)
lines(density(peso),col="brown",lwd=2)
axis(1,h8$breaks,las=2); axis(2,las=1)
```

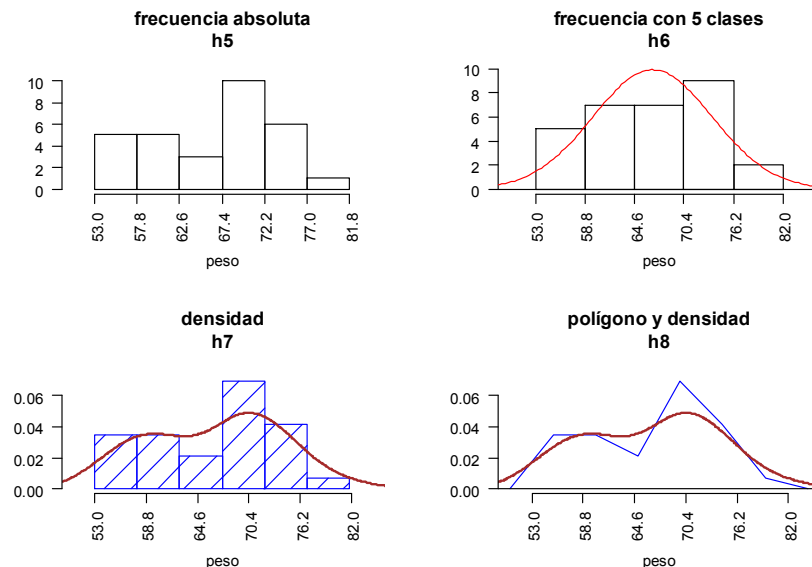


Figura 2.2. Cambio de escala de los ejes de coordenadas

## 2.3 TABLAS DE FRECUENCIA Y ESTADÍSTICAS

Redondeado a dos decimales:							<b>stat.freq(h6)</b>						
<b>round(table.freq(h6), 2)</b>							\$variance						
							[1] 50.42133						
							\$mean						
							[1] 66.72667						
							\$median						
							[1] 67.08571						
							\$mode						
							[- -] mode						
							[1,] 70.4 76.2 71.68889						
Lower	Upper	Main	freq	relative	CF	RCF							
53.0	58.8	55.9	5	0.17	5	0.17							
58.8	64.6	61.7	7	0.23	12	0.40							
64.6	70.4	67.5	7	0.23	19	0.63							
70.4	76.2	73.3	9	0.30	28	0.93							
76.2	82.0	79.1	2	0.07	30	1.00							

## 2.4 REPRODUCIENDO HISTOGRAMAS Y EL USO DE HIST()

La clase de graph.freq() es graph.freq. Figura 2.3

Reproduciendo el histograma h6 (5 clases)

```
h9<-ojiva.freq(h5,axes=FALSE,type="b", main="ojiva de h5\nh9",
col="red")
axis(1,round(h9[,1],1),las=2); axis(2, las=1)
h10<-plot(h6, axes=FALSE, main="frecuencia con 5 clases\nh10")
axis(1,h6$breaks,las=2); axis(2,seq(0,10))
normal.freq(h6,col="red")
summary(h6)
```

Lower	Upper	Main	freq	relative	CF	RCF
53.0	58.8	55.9	5	0.16666667	5	0.1666667
58.8	64.6	61.7	7	0.23333333	12	0.4000000
64.6	70.4	67.5	7	0.23333333	19	0.6333333
70.4	76.2	73.3	9	0.30000000	28	0.9333333
76.2	82.0	79.1	2	0.06666667	30	1.0000000

El tipo de clase de la función hist() es "histogram" y de graph.freq() es "graph.freq", sin embargo es posible establecer compatibilidad entre ambas funciones.

```
hh <- hist(peso,nclass=5, plot=FALSE) # Reporta 7 clases
# hist(peso,nclass=4) # Reporta 4 clases
```

Para mostrar las frecuencias relativas se puede usar graph.freq() usando el objeto hh creado por hist(), sin modificar las clases.

```
h11<-graph.freq(hh, frequency=2, col=colors()[367]
,main="relativa\nh11" ,axes=F)
axis(1,h11$breaks,las=2)
axis(2,las=1)
```

Ver los resúmenes: > summary(hh)  
> summary(h11)

Las funciones de "agricolae" para el manejo de histogramas funcionan correctamente sobre los objetos creados por la función hist() de R.

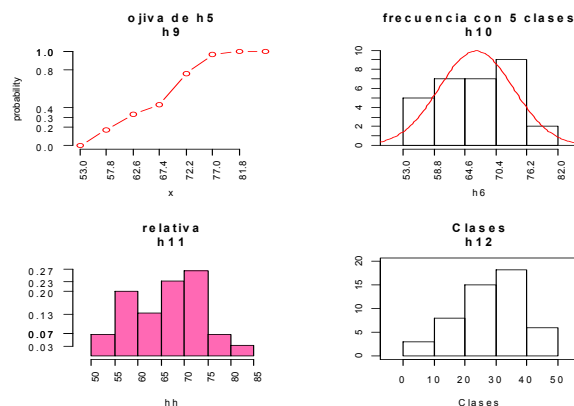


Figura 2.3 Nuevas escalas para los histogramas



## 2.5 HISTOGRAMA A PARTIR DE DATOS AGRUPADOS

Si se tiene datos agrupados, con la función `graph.freq()` se puede graficar y obtener los resúmenes del histograma; así por ejemplo en la siguiente tabla:

0-10	10-20	20-30	30-40	40-50
3	8	15	18	6

Organizando en R se tiene:

```
clases <- c(0, 10, 20, 30, 40, 50)
frec <- c(3, 8, 15, 18, 6)
h12 <- graph.freq(clases, counts=frec, xlab="Clases",
main="Clases\nh12")
summary(h12)
```

Lower	Upper	Main	frec	relative	CF	RCF
0	10	5	3	0.06	3	0.06
10	20	15	8	0.16	11	0.22
20	30	25	15	0.30	26	0.52
30	40	35	18	0.36	44	0.88
40	50	45	6	0.12	50	1.00

Todas las funciones de “agricolae” pueden aplicarse, incluyendo `plot()`.

## 2.6 JUNTANDO CLASES

Con la información de los pesos de los estudiantes, los intervalos originales pueden ser cambiados, juntando por ejemplo:

<code>intervals.freq(h5\$breaks)</code>	<code>nuevas&lt;- join.freq(h5\$breaks,1:2)</code>
lower upper	
[1,] 53.0 57.8	<code>intervals.freq(nuevas)</code>
[2,] 57.8 62.6	lower upper
[3,] 62.6 67.4	[1,] 53.0 62.6
[4,] 67.4 72.2	[2,] 62.6 67.4
[5,] 72.2 77.0	[3,] 67.4 72.2
[6,] 77.0 81.8	[4,] 72.2 77.0
	[5,] 77.0 81.8
	<code>h13 &lt;- graph.freq(peso, breaks=nuevas,</code>
	<code>main="Clases\nh13", axes=FALSE)</code>
	<code>axis(1,h13\$breaks);</code>
	<code>axis(2,seq(0,10,2))</code>

## 3 DISEÑO DE EXPERIMENTOS

El paquete “agricolae” presenta funciones especiales para la creación del libro de campo para diseños experimentales. La generación aleatoria hace que este paquete sea bastante utilizado en la investigación agrícola.

Para esta generación se requiere ciertos parámetros como por ejemplo el nombre de cada uno de los tratamientos, el número de repeticiones y otros según el diseño (Cochran, 1992; Kuehl, 2000; Montgomery, 2002; LeClerg, 1962). Otros parámetros de generación aleatoria son adicionales, como la semilla para reproducir la misma generación aleatoria o el método de generación (Ver manual de referencia de agrícola <http://cran.at.r-project.org/web/packages/agricolae/agricolae.pdf>)

### Parametros importantes en la generación de diseño:

serie: Es una constante que se utiliza para poner etiqueta numérica a los bloques, por ejemplo serie = 2, las etiquetas serán: 101, 102, para la primera fila o bloque, 201, 202, para las siguientes, en el caso del diseño completo al azar, la numeración es secuencial.

design: Algunas funciones de agricolae solicitan el diseño a ser aplicado, específicamente para design.ab (factorial) o design.split(parcelas divididas) y sus valores posibles son: "rcbd", "crd" y "lsd".

seed: Es la semilla para la generación aleatoria, y su valor es cualquier valor real positivo, si el valor es cero, se tiene una generación no reproducible, para otro valor si es reproducible el diseño.

Kinds: es el método para la generación aleatoria, el recomendado es "Super-Duper".

first: Para algunos diseños se requiere mantener ordenado (sistemático) la primera repetición, especialmente en el diseño de bloques; si desea cambiar a aleatorio, colocar first = TRUE.

### 3.1 COMPLETAMENTE AL AZAR

Sólo se requiere los nombres de tratamientos y el número de sus repeticiones y sus parámetros son:

```
> args(design.crd)
function (trt, r, serie = 2, seed = 0, kinds = "Super-Duper")
```

```
trt <- c("A", "B", "C")
repeticion <- c(4, 3, 4)
plan1 <- design.crd(trt, r=repeticion, seed=777, serie=0)
```

	plots	r	trt
1	1	1	B
2	2	1	A
3	3	2	A
4	4	1	C
5	5	2	C
6	6	3	A
7	7	2	B
8	8	3	C
9	9	3	B
10	10	4	A
11	11	4	C

Para Excel.

```
write.csv(plan1, "plan1.csv", row.names=FALSE)
```

### 3.2 BLOQUES COMPLETAMENTE AL AZAR

Se requiere los nombres de tratamientos y el número de bloques y sus parámetros son:

```
> args(design.rcbd)
function (trt, r, serie = 2, seed = 0, kinds = "Super-Duper",
         first = FALSE)

trt <- c("A", "B", "C")
repeticion <- 4
plan2 <- design.rcbd(trt,r=repeticion, seed=543, serie=2)
t(matrix(plan2[,3],c(3,4)))
      [,1] [,2] [,3]
[1,] "A"  "B"  "C"
[2,] "B"  "A"  "C"
[3,] "B"  "C"  "A"
[4,] "C"  "B"  "A"
```

El plan puede ser enviado a excel como libro de campo.

```
write.csv(plan2,"plan2.csv",row.names=FALSE)
```

### 3.3 CUADRADO LATINO

Se requiere los nombres de tratamientos.

```
> args(design.lsd)
function (trt, serie = 2, seed = 0, kinds = "Super-Duper", first =
FALSE)
NULL

trt <- c("A", "B", "C", "D")
plan3 <- design.lsd(trt, seed=543, serie=2)
t(matrix(plan3[,4],c(4,4)))
      [,1] [,2] [,3] [,4]
[1,] "A"  "B"  "C"  "D"
[2,] "B"  "C"  "D"  "A"
[3,] "D"  "A"  "B"  "C"
[4,] "C"  "D"  "A"  "B"
```

### 3.4 GRECO LATINO

Se requiere los nombres de tratamientos de cada factor de estudio y sus parámetros son:

```
> args(design.graeco)
function (trt1, trt2, serie = 2, seed = 0, kinds = "Super-Duper")

trt1 <- c("A", "B", "C", "D")
trt2 <- 1:4
plan4 <- design.graeco(trt1,trt2, seed=543, serie=2)
t(matrix(paste(plan4[,4],plan4[,5]),c(4,4)))
      [,1] [,2] [,3] [,4]
[1,] "A 1" "D 4" "B 3" "C 2"
[2,] "D 3" "A 2" "C 1" "B 4"
[3,] "B 2" "C 3" "A 4" "D 1"
[4,] "C 4" "B 1" "D 2" "A 3"
```

### 3.5 BLOQUES INCOMPLETOS BALANCEADOS

Se requiere los nombres de tratamientos y el tamaño del bloque.

```
> args(design.bib)
function (trt, k, serie = 2, seed = 0, kinds = "Super-Duper")

trt <- c("A", "B", "C", "D", "E" )
k <- 4
plan5 <- design.bib(trt,k, seed=543, serie=2)
```

```
Parameters BIB
=====
Lambda      : 3
treatmeans  : 5
Block size  : 4
Blocks      : 5
Replication : 4

Efficiency factor 0.9375
```

<<< Book >>>

Según la información producida, son 5 bloques de tamaño 4, siendo la matriz formada:

```
plan5$parameters
      lambda treatmeans blockSize blocks r Efficiency
values      3          5         4     5 4      0.9375

book<-plan5$book
t(matrix(book[,3],c(4,5)))
      [,1] [,2] [,3] [,4]
[1,] "C"  "B"  "E"  "A"
[2,] "C"  "D"  "A"  "B"
[3,] "B"  "A"  "E"  "D"
[4,] "D"  "C"  "E"  "B"
[5,] "A"  "D"  "E"  "C"
```

Se puede observar que los tratamientos tienen 4 repeticiones con lambda=3, lo cual significa que un par de tratamientos está junto en 3 oportunidades. Por ejemplo, B y E se encuentran en el bloque I, III y V.

### 3.6 CÍCLICO

Se requiere los nombres de tratamientos, el tamaño del bloque y el número de repeticiones. Se usa para 6 a 30 tratamientos. Las repeticiones son un múltiplo del tamaño del bloque; si son 6 tratamientos y el tamaño es 3, entonces las repeticiones pueden ser 6, 9, 12, etc. Sus parámetros:

```
> args(design.cyclic)
function (trt, k, r, serie = 2, rowcol = FALSE, seed = 0, kinds =
"Super-Duper")

trt <- c("A", "B", "C", "D", "E", "F" )
plan6 <- design.cyclic(trt,k=3, r=6, seed=543, serie=2)
```

```
cyclic design
Generator block basic:
1 2 4
1 3 2
Parameters
=====
treatments : 6
Block size : 3
Replication: 6
```

```
> plan6$design[[1]]
      [,1] [,2] [,3]
[1,] "A"  "E"  "D"
[2,] "D"  "F"  "C"
[3,] "A"  "D"  "B"
[4,] "A"  "C"  "F"
[5,] "C"  "B"  "E"
[6,] "B"  "E"  "F"
```

```
> plan6$design[[2]]
      [,1] [,2] [,3]
[1,] "B"  "D"  "C"
[2,] "C"  "A"  "B"
[3,] "F"  "A"  "B"
[4,] "C"  "D"  "E"
[5,] "E"  "A"  "F"
[6,] "F"  "E"  "D"
```

Se han generado 12 bloques de 4 tratamientos para cada uno.

### 3.7 LATTICE

Se requiere un número de tratamientos de un cuadrado perfecto, por ejemplo 9, 16, 25, 36, 49, etc.

Puede generar un lattice simple (2 rep.) o lattice triple (3 rep.)

Generando un diseño lattice triple para 9 tratamientos 3x3.

```
> args(design.lattice)
function (trt, r = 3, serie = 2, seed = 0, kinds = "Super-Duper")
> trt<-letters[1:9]
> plan7 <-design.lattice(trt, r = 3, serie = 2, seed = 0, kinds =
"Super-Duper")
```

```
Lattice design,   triple    3 x 3
```

```
Efficiency design  0.7272727
```

```
> plan
$parameters
      treatments blockSize blocks r Efficiency
values          9         3     3 3  0.7272727

$square1
      [,1] [,2] [,3]
[1,] "b"  "i"  "g"
[2,] "h"  "f"  "e"
```

```

[3,] "d" "a" "c"

$square2
      [,1] [,2] [,3]
[1,] "d"  "h"  "b"
[2,] "a"  "f"  "i"
[3,] "c"  "e"  "g"

$square3
      [,1] [,2] [,3]
[1,] "a"  "e"  "b"
[2,] "d"  "f"  "g"
[3,] "c"  "h"  "i"

$plan
      plots r block trt
1      101 1      1  b
2      102 1      1  i
3      103 1      1  g
...
9      109 1      3  c
10     201 2      4  d
11     202 2      4  h
12     203 2      4  b
...
19     301 3      7  a
20     302 3      7  e
21     303 3      7  b
22     304 3      8  d
23     305 3      8  f
24     306 3      8  g
25     307 3      9  c
26     308 3      9  h
27     309 3      9  i

```

### 3.8 ALFA LATICE

Son los diseños generados por los arreglos alpha (Patterson & Williams, 1976). Son similares a los diseños latice, pero los cuadros son rectangulares de s bloques por k tratamientos por bloque. El número de tratamientos debe ser igual a s\*k y el total de unidades experimentales r\*s\*k. Sus argumentos

```

> args(design.alpha)
function (trt, k, r, serie = 2, seed = 0, kinds = "Super-Duper")

trt <- letters[1:15]
plan8 <- design.alpha(trt,k=3,r=2,seed=543)
alpha design (0,1) - Serie I

Parameters Alpha design
=====
treatmeans : 15
Block size : 3
Blocks      : 5
Replication: 2

```

```
Efficiency factor
(E ) 0.6363636
```

```
<<< Book >>>
```

#### **plan8\$parameters**

```
      treatments blockSize blocks r Efficiency
values          15           3     5 2  0.6363636
```

#### **plan8\$design**

```
$rep1
```

```
      [,1] [,2] [,3]
[1,] "l"  "m"  "e"
[2,] "g"  "c"  "i"
[3,] "o"  "k"  "d"
[4,] "h"  "f"  "j"
[5,] "a"  "n"  "b"
```

```
$rep2
```

```
      [,1] [,2] [,3]
[1,] "o"  "a"  "m"
[2,] "l"  "k"  "g"
[3,] "d"  "n"  "h"
[4,] "j"  "b"  "c"
[5,] "f"  "i"  "e"
```

#### **# codificacion de las parcelas**

```
> A1<-plan8$book[plan8$book$replication==1,1]
> dim(A1)<-c(3,5)
> A1<-t(A1)
> A1
```

```
      [,1] [,2] [,3]
[1,]  101  102  103
[2,]  104  105  106
[3,]  107  108  109
[4,]  110  111  112
[5,]  113  114  115
```

```
> A2<-plan8$book[plan8$book$replication==2,1]
> dim(A2)<-c(3,5)
> A2<-t(A2)
> A2
```

```
      [,1] [,2] [,3]
[1,]  201  202  203
[2,]  204  205  206
[3,]  207  208  209
[4,]  210  211  212
[5,]  213  214  215
```

### **3.9 BLOQUES AUMENTADOS**

Se tiene estos diseños para dos tipos de tratamientos: los de control (comunes) y los aumentados. Los comunes son aplicados en bloques completos al azar y los aumentados, aleatoriamente, aplicandose cada uno en algún bloque una sola vez. Se entiende que los comunes son de mayor interés; el error estándar de la

diferencia es mucho menor que entre dos aumentados que están en diferentes bloques. La función `design.dau()` logra este propósito. Sus parámetros:

```
> args(design.dau)
function (trt1, trt2, r, serie = 2, seed = 0, kinds = "Super-Duper",
        name = "trt")

rm(list=ls())
trt1 <- c("A", "B", "C", "D")
trt2 <- c("t", "u", "v", "w", "x", "y", "z")
plan9 <- design.dau(trt1, trt2, r=5, seed=543, serie=2)
attach(plan9)
by(trt, block, as.character)

> by(trt, block, as.character)
block: 1
[1] "D" "C" "A" "u" "B" "t"
-----
block: 2
[1] "D" "z" "C" "A" "v" "B"
-----
block: 3
[1] "C" "w" "B" "A" "D"
-----
block: 4
[1] "A" "C" "D" "B" "y"
-----
block: 5
[1] "C" "B" "A" "D" "x"

detach(plan9)
print(plan9)
  plots block trt
1    101     1  D
2    102     1  C
3    103     1  A
4    104     1  u
5    105     1  B
6    106     1  t
7    201     2  D
...
27   505     5  x
```

Para el diseño completamente al azar aumentado, utilizar la función `design.crd()`.

### 3.10 PARCELAS DIVIDIDAS EN BLOQUES (SPLIT-PLOT)

Estos diseños tienen dos factores, un factor aplicado en parcelas definido como A en un diseño de bloques completos al azar y un segundo factor en las subparcelas de cada parcela aplicado aleatoriamente. La función `design.split()` permite hallar el plan experimental para este diseño. Sus parámetros:

```
> args(design.split)
function (trt1, trt2, r = NULL, design = c("rcbd", "crd", "lsd"),
        serie = 2, seed = 0, kinds = "Super-Duper", first = FALSE)

trt1<-c("A", "B", "C", "D")
```



```

trt2<-c("a","b","c")
plan10 <-design.split(trt1,trt2,r=3,serie=2,seed=543)
print(plan10)
  plots splots block trt1 trt2
1    101      1     1    A    c
2    101      2     1    A    a
3    101      3     1    A    b
4    102      1     1    B    b
5    102      2     1    B    c
6    102      3     1    B    a
7    103      1     1    C    b
8    103      2     1    C    c
9    103      3     1    C    a
10   104      1     1    D    a
11   104      2     1    D    b
12   104      3     1    D    c
13   201      1     2    A    b
14   201      2     2    A    a
...
36   304      3     3    D    b

p<-plan10$trt1[seq(1,36,3)]
q<-NULL
for(i in 1:12) q<-c(q,paste(plan10$trt2[3*(i-1)+1],plan10$trt2[3*(i-1)+2],plan10$trt2[3*(i-1)+3]))

```

En las parcelas

```

print(t(matrix(p,c(4,3))))
      [,1] [,2] [,3] [,4]
[1,] "A"  "B"  "C"  "D"
[2,] "A"  "C"  "B"  "D"
[3,] "A"  "C"  "B"  "D"

```

En las subparcelas (división de cada parcela)

```

print(t(matrix(q,c(4,3))))
      [,1] [,2] [,3] [,4]
[1,] "c a b" "b c a" "b c a" "a b c"
[2,] "b a c" "a b c" "a c b" "b c a"
[3,] "a b c" "a c b" "a c b" "c a b"

```

### 3.11 BLOQUES DIVIDIDOS (STRIP-PLOT)

Este diseño se utiliza cuando se tiene dos tipos de tratamientos (factores) y se aplica separadamente en parcelas grandes, llamadas franjas, en vertical y horizontal del bloque, obteniéndose los bloques divididos. Cada bloque constituye una repetición. Sus parámetros:

```

> args(design.strip)
function (trt1, trt2, r, serie = 2, seed = 0, kinds = "Super-Duper")

trt1<-c("A","B","C","D")
trt2<-c("a","b","c")
plan11 <-design.strip(trt1,trt2,r=3,serie=2,seed=543)
print(plan11)

```

```

      plots block trt1 trt2
1      101      1    A    a
2      102      1    A    b
3      103      1    A    c
...
13     201      2    D    a
14     202      2    D    b
15     203      2    D    c
16     204      2    A    a
17     205      2    A    b
...
36     312      3    A    a

t3<-paste(plan11$trt1,plan11$trt2)
B1<-t(matrix(t3[1:12],c(4,3)))
B2<-t(matrix(t3[13:24],c(3,4)))
B3<-t(matrix(t3[25:36],c(3,4)))

> print(B1)
      [,1] [,2] [,3] [,4]
[1,] "A a" "A b" "A c" "D a"
[2,] "D b" "D c" "B a" "B b"
[3,] "B c" "C a" "C b" "C c"
> print(B2)
      [,1] [,2] [,3]
[1,] "D a" "D b" "D c"
[2,] "A a" "A b" "A c"
[3,] "B a" "B b" "B c"
[4,] "C a" "C b" "C c"

> print(B3)
      [,1] [,2] [,3]
[1,] "B b" "B c" "B a"
[2,] "D b" "D c" "D a"
[3,] "C b" "C c" "C a"
[4,] "A b" "A c" "A a"

```

### 3.12 FACTORIAL

El arreglo factorial completo de n factores aplicados a un diseño experimental (CRD, RCBD y LSD) es frecuente y este procedimiento en agricolae permite aplicar el factorial a uno de estos tres diseños.

La función es design.ab() con parámetros:

```

> str(design.ab)
function (trt, r = NULL, serie = 2, design = c("rcbd", "crd", "lsd"),
  seed = 0, kinds = "Super-Duper", first = FALSE)

```

Para generar el factorial, es necesario crear un vector de niveles de cada factor, el procedimiento genera automáticamente hasta 25 factores y con “r” repeticiones.

```
trt<- c(4,2,3) # tres factores a 4, 2 y 3 niveles respectivamente.
```

Para los diseños rcdb y crd, es necesario dar valor a “r” como numero de repeticiones, este puede ser un vector si es desigual repetición o constante para igual repetición (recomendado).

```
> library(agricolae)
> trt<-c(3,2) # factorial 3x2
> rcdb <-design.ab(trt, r=3, serie=2)
> head(rcdb,10) # print of the field book
```

	plots	block	A	B
1	101	1	1	1
2	102	1	1	2
3	103	1	2	1
4	104	1	2	2
5	105	1	3	1
6	106	1	3	2
7	201	2	1	2
8	202	2	3	2
9	203	2	1	1
10	204	2	2	2

```
> # factorial 2 x 2 x 2 with 5 replications in completely randomized design.
> trt<-c(2,2,2)
> crd<-design.ab(trt, r=5, serie=2,design="crd")
> print(crd)
```

	plots	r	A	B	C
1	101	1	2	1	2
2	102	1	2	2	2
3	103	2	2	2	2
4	104	1	1	2	2
5	105	1	1	2	1
...					

#### 4 ANÁLISIS DE DISEÑOS EXPERIMENTALES

Para estos fines, “agricolae” tiene las siguientes funciones: BIB.test(), PBIB.test(), LSD.test(), HSD.test(), duncan.test(), SNK.test(), scheffe.test() (Steel, 1996) y durbin.test(), kruskal(), friedman() y waerden.test (Conover, 1999).

Para todo análisis estadístico, los datos deben estar organizados en columna. Para la demostración se utilizará la base de datos de “agricolae”.

Los datos de “sweetpotato” corresponden a un experimento completamente aleatorio en campo con parcelas de 50 plantas de camote y sometidas al efecto de virus y un control sin virus (ver manual de referencia del paquete).

```
data(sweetpotato)
modelo<-aov(yield~virus, data=sweetpotato)
cv.model(modelo)
[1] 17.16660
attach(sweetpotato)
mean(yield)
[1] 27.625
```

Parámetros del modelo: Grados de libertad y su variancia del error:

```
df<-df.residual(modelo)
MSerror<-deviance(modelo)/df
```

#### 4.1 LSD

Comprende la comparación múltiple mediante el método de la mínima diferencia significativa (Least Significant Difference), (Steel, 1997).

```
# out <- LSD.test(yield,virus,df,MSerror)
out<-LSD.test(modelo, "virus",console=TRUE)
Study:
```

LSD t Test for yield

Mean Square Error: 22.48917

	yield	std.err	r	LCL	UCL	Min.	Max.
cc	24.40000	2.084067	3	19.594134	29.20587	21.7	28.5
fc	12.86667	1.246774	3	9.991602	15.74173	10.6	14.9
ff	36.33333	4.233727	3	26.570341	46.09633	28.0	41.8
oo	36.90000	2.482606	3	31.175100	42.62490	32.1	40.4

alpha: 0.05 ; Df Error: 8  
Critical Value of t: 2.306004

Least Significant Difference 8.928965  
Means with the same letter are not significantly different.

Groups, Treatments and means

a	oo	36.9
a	ff	36.33
b	cc	24.4
c	fc	12.87

```
names(out)
```

```
1) "statistics" "parameters" "means" "comparison" "groups"
```

```
out$statistics
```

Mean	CV	MSerror	LSD
27.625	17.1666	22.48917	8.928965

En la función LSD.test() se realizó la comparación múltiple. Para obtener las probabilidades de las comparaciones se debe indicar que no se requiere grupos, así:

```
out <- LSD.test(yield, virus,df, MSerror, group=F,console=TRUE)
```

LSD t Test for yield

Mean Square Error: 22.48917

virus, means and individual ( 95 %) CI							
	yield	std.err	r	LCL	UCL	Min.	Max.
cc	24.40000	2.084067	3	19.594134	29.20587	21.7	28.5

```
fc 12.86667 1.246774 3 9.991602 15.74173 10.6 14.9
ff 36.33333 4.233727 3 26.570341 46.09633 28.0 41.8
oo 36.90000 2.482606 3 31.175100 42.62490 32.1 40.4
```

```
alpha: 0.05 ; Df Error: 8
Critical Value of t: 2.306004
```

Comparison between treatments means

	Difference	pvalue	sig.	LCL	UCL
cc - fc	11.5333333	0.0176377595	*	2.604368	20.462299
cc - ff	-11.9333333	0.0150730851	*	-20.862299	-3.004368
cc - oo	-12.5000000	0.0120884239	*	-21.428965	-3.571035
fc - ff	-23.4666667	0.0003023690	***	-32.395632	-14.537701
fc - oo	-24.0333333	0.0002574929	***	-32.962299	-15.104368
ff - oo	-0.5666667	0.8872673216		-9.495632	8.362299

La diferencia corresponde al primer tratamiento menos el segundo; así:

$$\text{"fc"} - \text{"oo"} = -24.0333333,$$

donde se compara ambos "fc" vs. "cc", el p.valor es 0.0002574929, lo cual indica que son diferentes significativamente. Así se procede con los otros resultados.

El código de la significación "sig" se interpreta como:

```
***": p.valor < 0.001
**  ": 0.001 < p.valor < 0.01
*   ": 0.01 < p.valor < 0.05
.   ": 0.05 < p.valor < 0.10
```

## 4.2 BONFERRONI

Mediante la función `LSD.test()` se puede hacer ajustes a las probabilidades encontradas como por ejemplo el ajuste por Bonferroni.

```
LSD.test(modelo, "virus", group=F, p.adj= "bon", console=TRUE)
```

```
alpha: 0.05 ; Df Error: 8
Critical Value of t: 3.478879
```

Comparison between treatments means

	Difference	pvalue	sig.	LCL	UCL
cc - fc	11.5333333	0.105827		-1.937064	25.0037305
cc - ff	-11.9333333	0.090439	.	-25.403730	1.5370638
cc - oo	-12.5000000	0.072531	.	-25.970397	0.9703971
fc - ff	-23.4666667	0.001814	**	-36.937064	-9.9962695
fc - oo	-24.0333333	0.001545	**	-37.503730	-10.5629362
ff - oo	-0.5666667	1.000000		-14.037064	12.9037305

Otras pruebas de comparación pueden ser aplicadas, tales como "duncan", "Student-Newman-Keuls", "tukey", "waller-duncan".

Para "duncan" la función `duncan.test()`; para "Student-Newman-Keuls", la función `SNK.test()`; para "tukey", la función `HSD.test()`; para "scheffe" la función `scheffe.test()`; y para "waller-duncan", la función `waller.test()`. Los parámetros son los

mismos. En el caso de Waller, se requiere además el valor de F-calculado de tratamientos del ANOVA. Si se utiliza el modelo como parámetro, ya no es necesario.

### 4.3 DUNCAN' NEW'S

Corresponde a la nueva prueba de rango multiple de Duncan (Duncan's New Multiple-Range Test), (Steel, 1997).

```
out<-duncan.test(modelo, "virus", console=TRUE)
```

Study:

Duncan's new multiple range test  
for yield

Mean Square Error: 22.48917

virus, means

	yield	std.err	r	Min.	Max.
cc	24.40000	2.084067	3	21.7	28.5
fc	12.86667	1.246774	3	10.6	14.9
ff	36.33333	4.233727	3	28.0	41.8
oo	36.90000	2.482606	3	32.1	40.4

alpha: 0.05 ; Df Error: 8

Critical Range

	2	3	4
	8.928965	9.304825	9.514910

Means with the same letter are not significantly different.

Groups, Treatments and means

a	oo	36.9
a	ff	36.33
b	cc	24.4
c	fc	12.87

out\$Duncan

	Table	CriticalRange
2	3.261182	8.928965
3	3.398460	9.304825
4	3.475191	9.514910

```
duncan.test(modelo, "virus", group=FALSE, console=TRUE)
```

alpha: 0.05 ; Df Error: 8

Critical Range

	2	3	4
	8.928965	9.304825	9.514910

Comparison between treatments means

	Difference	pvalue	sig.	LCL	UCL
--	------------	--------	------	-----	-----

```

cc - fc 11.5333333 0.017638 * 2.604368 20.462299
cc - ff -11.9333333 0.015073 * -20.862299 -3.004368
cc - oo -12.5000000 0.014544 * -21.804825 -3.195175
fc - ff -23.4666667 0.000388 *** -32.771492 -14.161842
fc - oo -24.0333333 0.000387 *** -33.548244 -14.518423
ff - oo -0.5666667 0.887267 -9.495632 8.362299

```

#### 4.4 STUDENT-NEWMAN-KEULS (SNK)

Student, Newman y Keuls contribuyeron a mejorar la prueba de Newman-Keuls de 1939, el cual se conocia como el metodo Keuls, (Steel, 1997).

```
out<-SNK.test(modelo, "virus", alpha=0.05,console=TRUE)
```

```
Student Newman Keuls Test
```

```
for yield
```

```
Mean Square Error: 22.48917
```

```
virus, means
```

```

      yield std.err r Min. Max.
cc 24.40000 2.084067 3 21.7 28.5
fc 12.86667 1.246774 3 10.6 14.9
ff 36.33333 4.233727 3 28.0 41.8
oo 36.90000 2.482606 3 32.1 40.4

```

```
alpha: 0.05 ; Df Error: 8
```

```
Critical Range
```

```

      2      3      4
8.928965 11.064170 12.399670

```

```
Means with the same letter are not significantly different.
```

```
Groups, Treatments and means
```

```

a      oo      36.9
a      ff      36.33
b      cc      24.4
c      fc      12.87

```

```
names(out)
```

```

[1] "statistics" "parameters" "SNK"          "means"        "comparison"
[6] "groups"

```

```
out$SNK
```

```

Table CriticalRange
2 3.261182      8.928965
3 4.041036     11.064170
4 4.528810     12.399670

```

```
out<-SNK.test(modelo, "virus", group=FALSE,console=TRUE)
```

```
alpha: 0.05 ; Df Error: 8
```

```
Critical Range
```

```

      2      3      4
8.928965 11.064170 12.399670

```

Comparison between treatments means

	Difference	pvalue	sig.	LCL	UCL
cc-fc	11.53333333	0.017638	*	2.604368	20.462299
cc-ff	-11.93333333	0.015073	*	-20.862299	-3.004368
cc-oo	-12.50000000	0.029089	*	-23.564170	-1.435830
fc-ff	-23.46666667	0.000777	***	-34.530836	-12.402497
fc-oo	-24.03333333	0.001162	**	-36.433003	-11.633664
ff-oo	-0.56666667	0.887267		-9.495632	8.362299

**out\$comparison**

	Difference	pvalue	sig	LCL	UCL
cc-fc	11.53333333	0.017638	*	2.604368	20.462299
...					
ff-oo	-0.56666667	0.887267		-9.495632	8.362299

#### 4.5 TUKEY (HSD)

Llamado de rango estudiantizado construido por Tukey en 1953, conocido como la prueba de la diferencia significativa honesta HSD (Honestly significant difference), (Steel, 1997).

```
out1 <- HSD.test(modelo, "virus", console=TRUE)
```

HSD Test for yield

Mean Square Error: 22.48917

alpha: 0.05 ; Df Error: 8

Critical Value of Studentized Range: 4.52881

Honestly Significant Difference: 12.39967

Means with the same letter are not significantly different.

Groups, Treatments and means

a	oo	36.9
ab	ff	36.33
bc	cc	24.4
c	fc	12.87

#### 4.6 WALLER-DUNCAN

En 1975, Duncan continuo con los procedimientos de comparacion multiple, introduciendo el criterio de minimizar ambos errores experimentales; para esto, utilizo la teoria bayesiana, obteniendo una nueva prueba llamada Waller-Duncan, (Steel, 1997).

```
# análisis de variancia:
```

```
anova(modelo)
```

Analysis of Variance Table

Response: yield



```

          Df Sum Sq Mean Sq F value    Pr(>F)
virus      3 1170.21   390.07   17.345 0.0007334 ***
Residuals  8  179.91    22.49
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

El valor de F calculado es 17.345. Para el análisis comparativo de tratamientos se requiere que esté activo sweetpotato; así:

```
attach(sweetpotato)
```

entonces:

```
out2 <- waller.test(yield,virus,DFerror=8,MSerror=22.49,Fc= 17.345,  
group=F)
```

En otro caso con sólo invocar el objeto `modelo`:

```
out2 <- waller.test(modelo, "virus", group=F, console=TRUE)
```

Waller-Duncan K-ratio t Test for yield

This test minimizes the Bayes risk under additive loss and certain other assumptions.

```

          . . . . .
K ratio              100.00000
Error Degrees of Freedom    8.00000
Error Mean Square          22.48917
F value                   17.34500
Critical Value of Waller    2.23600

```

virus, means

```

      yield std.error Min. Max.
cc 24.40000 2.084067 3 21.7 28.5
fc 12.86667 1.246774 3 10.6 14.9
ff 36.33333 4.233727 3 28.0 41.8
oo 36.90000 2.482606 3 32.1 40.4

```

Minimum Significant Difference 8.657906  
Comparison between treatments means

```

          Difference significant
cc - fc  11.5333333      TRUE
cc - ff -11.9333333      TRUE
cc - oo -12.5000000      TRUE
fc - ff -23.4666667      TRUE
fc - oo -24.0333333      TRUE
ff - oo  -0.5666667     FALSE

```

Se indica que el efecto del virus “ff” no es significativamente del control “oo”.

El objeto encontrado “out” y tiene información para hacer otros procedimientos.

```

> out1
$statistics
      Mean      CV  MSerror      HSD
27.625 17.1666 22.48917 12.39967

$parameters
      Df ntr StudentizedRange
      8   4           4.52881

$means
...
> out2
$statistics
      Mean      CV  MSerror  F.Value CriticalDifference
27.625 17.1666 22.48917 17.34478           8.657906

$parameters
      Df ntr   K Waller
      8   4 100   2.236

$means
...

```

#### 4.7 SCHEFFE

Este método fué construido en 1959 por Scheffe, es muy general para todos los posibles contraste y sus intervalos de confianza. Los intervalos de confianza para las medias son muy amplias, resultando una prueba muy conservadora para la comparacion entre medias de tratamientos (Steel, 1997).

```

# análisis de variancia:
modelo<-aov(yield~virus, data=sweetpotato)
out3 <- scheffe.test(modelo,"virus", group=TRUE,
main="Yield of sweetpotato\nDealt with different virus")

```

```

Study: Yield of sweetpotato
Dealt with different virus

```

```
Scheffe Test for yield
```

```
Mean Square Error   : 22.48917
```

```
virus, means
```

```

      yield  std.err r Min. Max.
cc 24.40000 2.084067 3 21.7 28.5
fc 12.86667 1.246774 3 10.6 14.9
ff 36.33333 4.233727 3 28.0 41.8
oo 36.90000 2.482606 3 32.1 40.4

```

```

alpha: 0.05 ; Df Error: 8
Critical Value of F: 4.066181

```

```
Minimum Significant Difference: 13.52368
```

```
Means with the same letter are not significantly different.
```

```
Groups, Treatments and means
```

a	oo	36.9
a	ff	36.33
ab	cc	24.4
b	fc	12.87

El valor mínimo de significación es muy alto.

Si se requiere las probabilidades aproximadas de comparación, se puede utilizar la opción Group=FALSE.

```
out3 <- scheffe.test(modelo,"virus", group=FALSE)
```

Scheffe Test for yield

Mean Square Error : 22.48917

virus, means

	yield	std.err	r	Min.	Max.
cc	24.40000	2.084067	3	21.7	28.5
fc	12.86667	1.246774	3	10.6	14.9
ff	36.33333	4.233727	3	28.0	41.8
oo	36.90000	2.482606	3	32.1	40.4

alpha: 0.05 ; Df Error: 8  
Critical Value of F: 4.066181

Comparison between treatments means

	Difference	pvalue	sig.	LCL	UCL
cc - fc	11.5333333	0.097816	.	-1.000348	24.0670149
cc - ff	-11.9333333	0.085487	.	-24.467015	0.6003483
cc - oo	-12.5000000	0.070607	.	-25.033682	0.0336816
fc - ff	-23.4666667	0.002331	**	-36.000348	-10.9329851
fc - oo	-24.0333333	0.001998	**	-36.567015	-11.4996517
ff - oo	-0.5666667	0.999099		-13.100348	11.9670149

#### 4.8 COMPARACION MULTIPLE DE UN FACTORIAL

En un factorial se tiene efectos combinados de los tratamientos. Las pruebas comparativas: LSD, HSD, Waller-Duncan, Duncan, Scheffe, SNK pueden ser aplicadas.

```
# modelo <-aov(y ~ A*B*C,data)
# out<-LSD.test(modelo,c("A","B","C"))
```

La comparación será de la combinación de A:B:C.

Datos de un diseño bloques con un factorial clones x nitrógeno. La variable respuesta rendimiento:

```

Rendimiento <-
scan(text=
"6 7 9 13 16 20 8 8 9
 7 8 8 12 17 18 10 9 12
 9 9 9 14 18 21 11 12 11
 8 10 10 15 16 22 9 9 9"
)
bloque<-gl(4,9)
clon<-rep(gl(3,3,labels = c("c1","c2","c3")),4)
nitrogeno<-rep(gl(3,1,labels=c("n1","n2","n3")),12)
A<-data.frame(bloque,clon,nitrogeno,rendimiento)
head(A)

```

```

      bloque clon nitrogeno rendimiento
1         1    c1         n1           6
2         1    c1         n2           7
3         1    c1         n3           9
4         1    c2         n1          13
5         1    c2         n2          16
6         1    c2         n3          20

```

```

modelo<-aov(rendimiento ~ bloque + clon*nitrogeno, data=A)
anova(modelo)

```

Analysis of Variance Table

Response: rendimiento

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
bloque	3	20.75	6.917	5.8246	0.0038746	**
clon	2	497.72	248.861	209.5673	6.370e-16	***
nitrogeno	2	54.06	27.028	22.7602	2.865e-06	***
clon:nitrogeno	4	43.28	10.819	9.1111	0.0001265	***
Residuals	24	28.50	1.187			

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

```

out<-LSD.test(modelo,c("clon","nitrogeno"),main="Rendimiento ~
bloques + clon + nitrogeno + clon:nitrogeno")

```

Study: Rendimiento ~ bloques + clon + nitrógeno + clon:nitrogeno

LSD t Test for rendimiento

Mean Square Error: 1.1875

clon:nitrogeno, means and individual ( 95 %) CI

	rendimiento	std.err	r	LCL	UCL	Min.	Max.
c1:n1	7.50	0.6454972	4	6.167759	8.832241	6	9
c1:n2	8.50	0.6454972	4	7.167759	9.832241	7	10
c1:n3	9.00	0.4082483	4	8.157417	9.842583	8	10
c2:n1	13.50	0.6454972	4	12.167759	14.832241	12	15
c2:n2	16.75	0.4787136	4	15.761984	17.738016	16	18
c2:n3	20.25	0.8539126	4	18.487611	22.012389	18	22
c3:n1	9.50	0.6454972	4	8.167759	10.832241	8	11
c3:n2	9.50	0.8660254	4	7.712611	11.287389	8	12
c3:n3	10.25	0.7500000	4	8.702076	11.797924	9	12

alpha: 0.05 ; Df Error: 24  
Critical Value of t: 2.063899

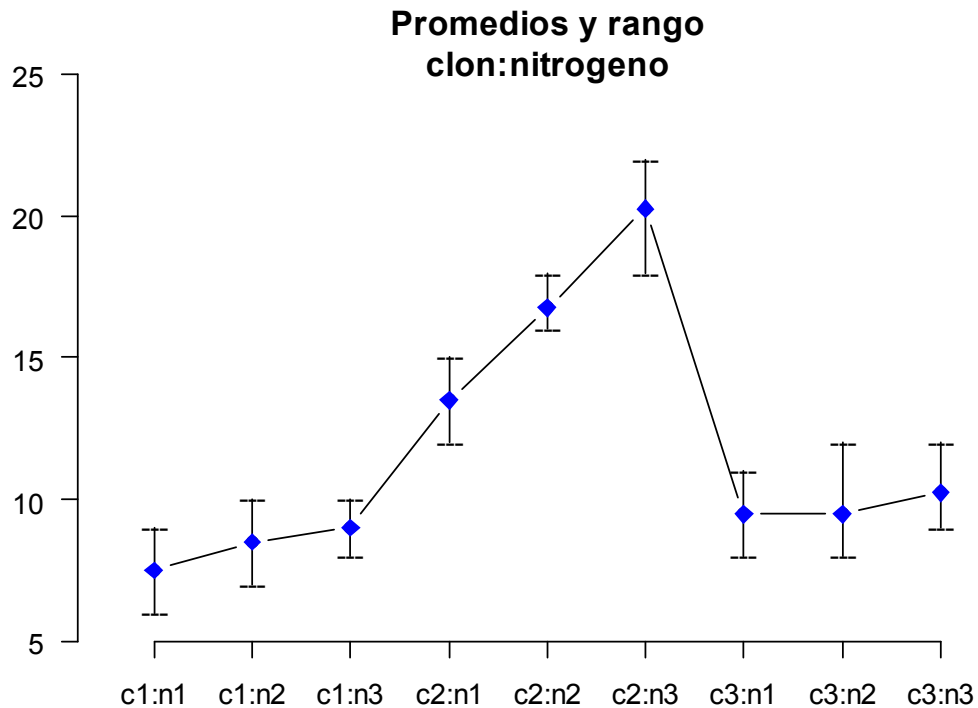
Least Significant Difference 1.590341  
Means with the same letter are not significantly different.

Groups, Treatments and means

a	c2:n3	20.25
b	c2:n2	16.75
c	c2:n1	13.5
d	c3:n3	10.25
de	c3:n1	9.5
de	c3:n2	9.5
def	c1:n3	9
ef	c1:n2	8.5
f	c1:n1	7.5

```
par(mar=c(3,3,2,0))
grafico<-bar.err(out$means,variation="range",ylim=c(5,25),
bar=FALSE,col=0,las=1)
points(grafico$index,grafico$means,pch=18,cex=1.5,col="blue")
axis(1,grafico$index,labels=FALSE)
title(main="Promedios y rango\nclon:nitrogeno")
```

En la siguiente figura 4.1 se muestra el resultado.



**Figura 4.1 Combinado clon:nitrogeno**

Es importante el orden como se presenta en el anova (solo para la prueba de waller-duncan en otras pruebas no es necesario)

## 4.9 GRÁFICOS DE LA COMPARACIÓN MÚLTIPLE

Los resultados de una comparación pueden ser vistos graficamente con las funciones `bar.group()` y `bar.err()`.

El objeto encontrado de una comparación es la entrada para estas funciones, Figura 4.2.

Los objetos `out1` (Tukey) y `out2` (Waller-Duncan) son utilizados en el siguiente ejercicio:

`out1`, para las funciones `bar.group()` y `bar.err()`

`out2`, para la función `bar.err()`

```
par(mfrow=c(2,2))
c1<-colors()[480]; c2=colors()[65]; c3=colors()[15];
c4=colors()[140]
G1<-bar.group(out1$groups, ylim=c(0,45),
main="Tukey\nG1",col=c1,las=1)
G2<-bar.group(out1$groups, horiz=TRUE, xlim=c(0,45),
main="Tukey\nG2",col=c2,las=1)
G3<-bar.err(out2$means, variation="range",ylim=c(0,45), col=c3,
main="Rango\nG3",las=1)
G4<-bar.err(out2$means, variation="SE", horiz=TRUE, xlim=c(0,45),
col=c4, main="Error estándar \nG4",las=1)
```

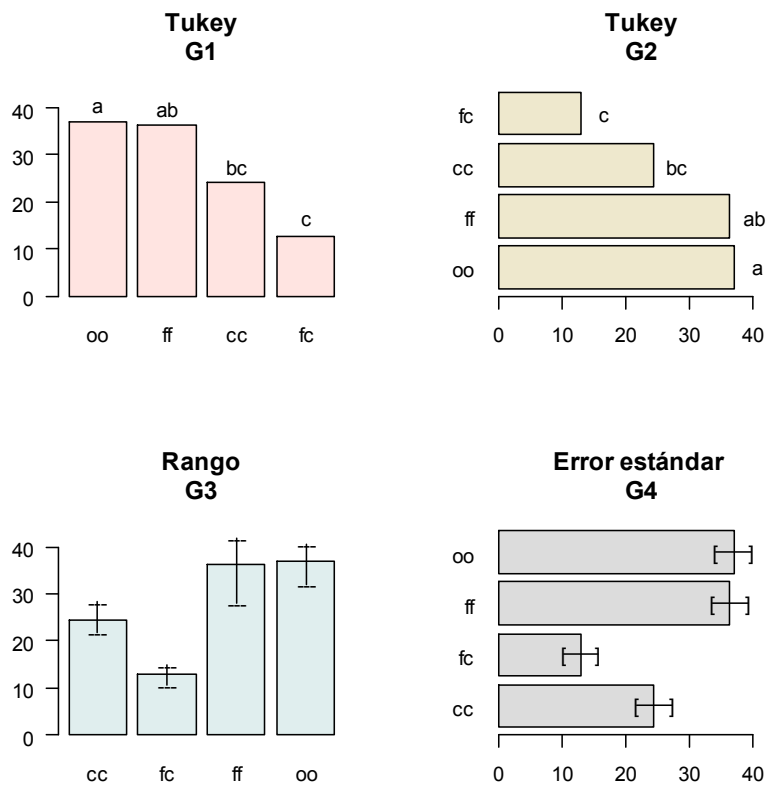


Figura 4.2 Comparación entre tratamientos

#### 4.10 ANÁLISIS DE BLOQUES INCOMPLETOS BALANCEADOS

Éstos pueden provenir de diseños balanceados o parcialmente balanceados. La función BIB.test() es para diseños balanceados, y PBIB.test(), para diseños parcialmente balanceados. Para el ejemplo, se utilizarán los datos de “agricolae”.

```
#Example linear estimation and design of experiments. (Joshi, 1987)
# Profesor de Estadística, Institute of Social Sciences Agra, India
# 6 variedades de trigo en 10 bloques de 3 parcelas cada una.
bloque<-gl(10,3)
variedad<-c(1,2,3,1,2,4,1,3,5,1,4,6,1,5,6,2,3,6,2,4,5,2,5,6,3,4,5,3,
4,6)
y<-c(69,54,50,77,65,38,72,45,54,63,60,39,70,65,54,65,68,67,57,60,62,
59,65,63,75,62,61,59,55,56)
analisis<-BIB.test(block=bloque, trt=variedad, y)
```

```
ANALYSIS BIB:  y
Class level information
```

```
Block:  1 2 3 4 5 6 7 8 9 10
Trt   :  1 2 3 4 5 6
```

```
Number of observations:  30
```

```
Analysis of Variance Table
```

```
Response: y
          Df Sum Sq Mean Sq F value Pr(>F)
block.unadj  9  466.97  51.885   0.9019 0.54712
trt.adj       5 1156.44 231.289   4.0206 0.01629 *
Residuals    15  862.89  57.526
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coefficient of variation: 12.6 %
y Means: 60.3
```

```
variedad,  statistics
```

	y	mean.adj	SE r	std.err	Min.	Max.
1	70.2	75.13333	3.728552	5 2.267157	63	77
2	60.0	58.71667	3.728552	5 2.190890	54	65
3	59.4	58.55000	3.728552	5 5.537147	45	75
4	55.0	54.96667	3.728552	5 4.404543	38	62
5	61.4	60.05000	3.728552	5 2.014944	54	65
6	55.8	54.38333	3.728552	5 4.810405	39	67

```
LSD test
Std.diff   : 5.363111
Alpha      : 0.05
LSD        : 11.4312
Parameters BIB
Lambda     : 2
treatmeans : 6
Block size : 3
Blocks     : 10
Replication: 5
```

Efficiency factor 0.8

<<< Book >>>

Means with the same letter are not significantly different.

Comparison of treatments

Groups, Treatments and means

a	1	75.13
b	5	60.05
b	2	58.72
b	3	58.55
b	4	54.97
b	6	54.38

**analisis**

\$parameters

lambda	treatmeans	blockSize	blocks	r
2	6	3	10	5

\$statistics

Mean Efficiency	CV
60.3	0.8 12.57808

\$comparison

NULL

\$means

	y	mean.adj	SE	r	std.err	Min.	Max.
1	70.2	75.13333	3.728552	5	2.267157	63	77
2	60.0	58.71667	3.728552	5	2.190890	54	65
3	59.4	58.55000	3.728552	5	5.537147	45	75
4	55.0	54.96667	3.728552	5	4.404543	38	62
5	61.4	60.05000	3.728552	5	2.014944	54	65
6	55.8	54.38333	3.728552	5	4.810405	39	67

\$groups

	trt	mean.adj	M
1	1	75.13333	a
2	5	60.05000	b
3	2	58.71667	b
4	3	58.55000	b
5	4	54.96667	b
6	6	54.38333	b

Se puede utilizar LSD, Duncan, SNK, Tukey y Waller-Duncan. También se puede obtener las probabilidades de la comparación. Sólo se debe indicar group=FALSE, así:

```
out<-BIB.test(block=bloque, trt=variedad, y, group=F, test= "tukey")
```

**out\$comparison**

	Difference	pvalue	sig.
1 - 2	16.416667	0.070509	.



```

1 - 3 16.5833333 0.066649 .
1 - 4 20.1666667 0.019092 *
1 - 5 15.0833333 0.109602
1 - 6 20.7500000 0.015510 *
2 - 3 0.1666667 1.000000
2 - 4 3.7500000 0.979184
2 - 5 -1.3333333 0.999840
2 - 6 4.3333333 0.961588
3 - 4 3.5833333 0.982927
3 - 5 -1.5000000 0.999715
3 - 6 4.1666667 0.967375
4 - 5 -5.0833333 0.927273
4 - 6 0.5833333 0.999997
5 - 6 5.6666667 0.890815

```

El objeto "modelo" encontrado puede ser utilizado para las funciones de `bar.group()` y `bar.err()` para los gráficos de barras, en la misma forma como se realizó anteriormente.

#### 4.11 BLOQUES INCOMPLETOS PARCIALMENTE BALANCEADO

La función `PBIB.test()` (Joshi, 1987). puede ser utilizada para los diseños latice y alfa.

Considere el caso siguiente: Construir el diseño alfa con 30 tratamientos, 2 repeticiones y un tamaño del bloque igual a 3.

```

library(agricolae)
library(MASS)
require(nlme)
# alpha design
genotipo<-paste("geno",1:30,sep="")
r<-2
k<-3
plan<-design.alpha(genotipo,k,r,seed=5)

alpha design (0,1) - Serie I

Parameters Alpha design
=====
treatments : 30
Block size : 3
Blocks      : 10
Replication: 2

Efficiency factor
(E ) 0.6170213

<<< Book >>>

plan$parameters
      treatments blockSize blocks r Efficiency
values          30         3    10 2   0.6170213

```

El plan generado es `plan$book`.

Suponga que la observación correspondiente a cada unidad experimental es:

```
rdto<-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)
```

Se construye la tabla de datos para el análisis. En teoría, se supone que uno aplica un diseño y realiza el experimento; posteriormente se observa las variables de estudio a partir de cada unidad experimental.

```
tabla<-data.frame(plan$book,rdto)
rm(rdto,genotipo)
```

El análisis:

```
attach(tabla)
modelo <- PBIB.test(block, genotipo, replication, rdto, k=3,
method="ML",console=TRUE)
detach(tabla)
```

ANALYSIS PBIB: rdto

Class level information

block : 20

genotipo : 30

Number of observations: 60

Estimation Method: Maximum likelihood

Parameter Estimates

	Variance
block:replication	3.355931e+00
replication	5.997912e-09
Residual	5.923666e-01

	Fit Statistics
-2 Res Log Likelihood	-98.32886
AIC	262.65771
BIC	331.77108

Analysis of Variance Table

Response: rdto

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
genotipo	29	32.182	1.10974	1.8734	0.1359
Residuals	11	6.516	0.59237		

coefficient of variation: 17 %

rdto Means: 4.533333

Parameters PBIB

	.
genotipo	30
block size	3
block/replication	10
replication	2

Efficiency factor 0.6170213

Comparison test lsd

Means with the same letter are not significantly different.

Groups, Treatments and means

a	20	7.759	
ab	24	6.294	
ab	13	6.171	
abc	1	6.14	
abcd	8	6.079	
abcd	23	5.97	
abcde	22	5.792	
abcde	16	5.67	
bcdef	27	5.08	
bcdefg	4	5.042	
bcdefg	10	4.927	
bcdefgh	14	4.885	
bcdefghi	3	4.698	
bcdefghij	6	4.631	
bcdefghij	7	4.442	
bcdefghij	19	4.386	
bcdefghij	21	4.324	
bcdefghij	5	4.194	
bcdefghij	17	4.181	
bcdefghij	28	3.965	
bcdefghij	25	3.812	
cdefghij	15	3.646	
cefghij	2	3.634	
defghij	11	3.53	
efghij	12	3.427	
fghij	26	3.418	
fhi	30	2.652	
gij	18	2.533	
ij	9	2.515	
j	29	2.201	

Comparison between treatments means and its name

<<< to see the objects: means, comparison and groups. >>>

**> names(modelo)**

```
[1] "method"      "parameters"  "statistics"  "comparison"  "means"
[6] "groups"      "vartau"
```

**modelo\$parameters**

```
treatments blockSize blocks r
          30           3     10 2
```

**modelo\$statistics**

```
Efficiency      Mean      CV
0.6170213 4.533333 16.97765
```

Las medias ajustadas pueden ser extraídas del modelo.

```
head(modelo$means)
```

	rdto	trt	mean.adj	SE	r	std.err	Min.	Max.
geno1	7.5	1	6.140390	0.8813469	2	1.5	6	9
geno10	4.5	2	3.633591	0.8813469	2	0.5	4	5
geno11	5.5	3	4.698153	0.8754092	2	0.5	5	6
geno12	4.0	4	5.042196	0.8813469	2	3.0	1	7
geno13	4.0	5	4.194444	0.8813469	2	2.0	2	6
geno14	3.5	6	4.631101	0.8754092	2	2.5	1	6

Los datos sobre las medias ajustadas y su error estándar pueden ser graficados (figura 4.2), dado que el objeto creado es muy similar a los objetos generados por las comparaciones múltiples.

```
par(mfrow=c(2,2),cex=0.6)
C1<-bar.err(modelo$means[1:7,], ylim=c(0,9), main="C1",
variation="range",col=colors()[15])
C2<-bar.err(modelo$means[8:15,], ylim=c(0,9), main="C2",
variation="range", col=colors()[20])
C3<-bar.err(modelo$means[16:22,], ylim=c(0,9), main="C3",
variation="range", col=colors()[23])
C4<-bar.err(modelo$means[23:30,], ylim=c(0,9), main="C4",
variation="range", col=colors()[47])
```

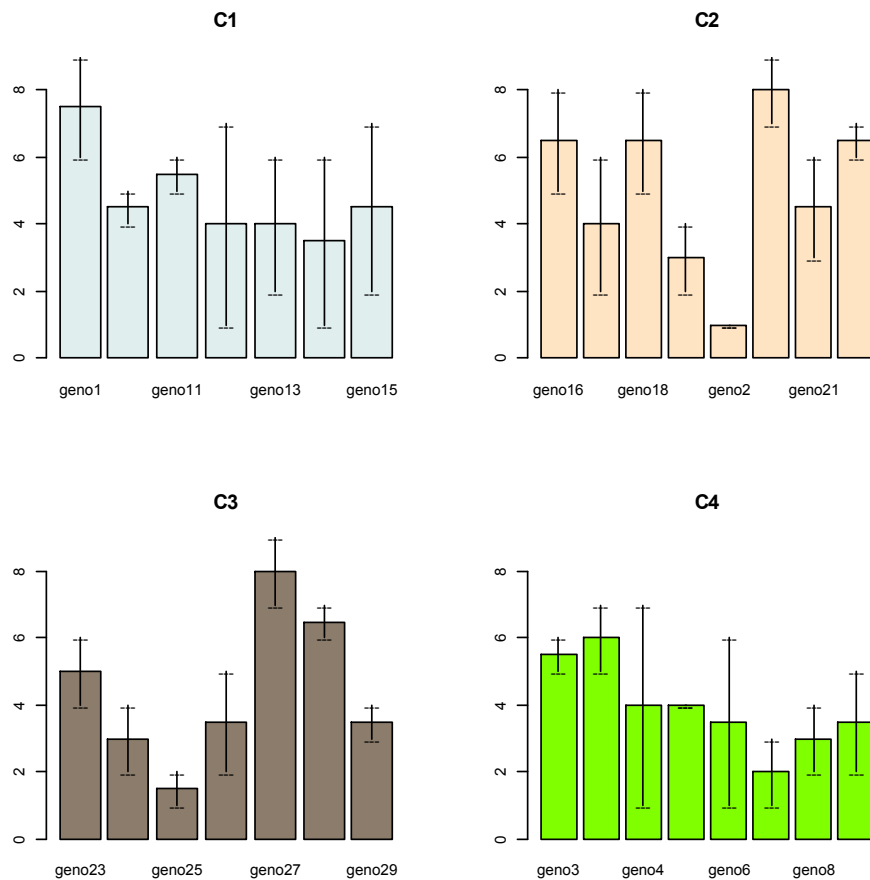


Figura 4.3. Rango de variacion en cada tratamiento.

Crear los datos en un archivo de texto: `latice3x3.txt` y leer con R:

sqr	block	trt	yield	sqr	block	trt	yield	sqr	block	trt	yield
1	1	1	48.76	1	1	4	14.46	1	1	3	19.68
1	2	8	10.83	1	2	6	30.69	1	2	7	31.00
1	3	5	12.54	1	3	9	42.01	1	3	2	23.00
2	4	5	11.07	2	4	8	22.00	2	4	1	41.00
2	5	2	22.00	2	5	7	42.80	2	5	3	12.90
2	6	9	47.43	2	6	6	28.28	2	6	4	49.95
3	7	2	27.67	3	7	1	50.00	3	7	6	25.00
3	8	7	30.00	3	8	5	24.00	3	8	4	45.57
3	9	3	13.78	3	9	8	24.00	3	9	9	30.00
4	10	6	37.00	4	10	3	15.42	4	10	5	20.00
4	11	4	42.37	4	11	2	30.00	4	11	8	18.00
4	12	9	39.00	4	12	7	23.80	4	12	1	43.81

Response:	yield						
	Df	Sum Sq	Mean Sq	F value	Pr(>F)		
trt	8	3749.4	468.68	8.2315	0.0001987	***	
Residuals	16	911.0	56.94				

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coefficient of variation: 25.9 %
yield Means: 29.16167
```

```
Parameters PBIB
```

```
      .
trt    9
block size 3
block/sqr 3
sqr     4
```

```
Efficiency factor 0.75
```

```
Comparison test lsd
```

```
Means with the same letter are not significantly different.
```

```
Groups, Treatments and means
```

```
a      1      45.89
ab     9      39.61
ab     4      38.09
bc     7      31.9
bc     6      30.24
cd     2      25.67
d      8      18.71
d      5      16.9
d      3      15.45
```

```
Comparison between treatments means and its name
```

```
<<< to see the objects: means, comparison and groups. >>>
```

```
modelo2$parameters
```

```
treatments blockSize blocks r
          9          3          3 4
```

```
modelo2$means
```

```
      yield trt mean.adj      SE r  std.err  Min.  Max.
1 45.8925   1  45.8925 3.772839 4 2.108860 41.00 50.00
2 25.6675   2  25.6675 3.772839 4 1.900585 22.00 30.00
3 15.4450   3  15.4450 3.772839 4 1.505133 12.90 19.68
4 38.0875   4  38.0875 3.772839 4 8.027584 14.46 49.95
5 16.9025   5  16.9025 3.772839 4 3.068910 11.07 24.00
6 30.2425   6  30.2425 3.772839 4 2.536390 25.00 37.00
7 31.9000   7  31.9000 3.772839 4 3.966947 23.80 42.80
8 18.7075   8  18.7075 3.772839 4 2.906984 10.83 24.00
9 39.6100   9  39.6100 3.772839 4 3.647335 30.00 47.43
```

```
head(modelo2$comparison)
```

```
      Difference  stderr  pvalue
1 - 2      20.2250 5.335599 0.001604
1 - 3      30.4475 5.335599 0.000032
1 - 4       7.8050 5.335599 0.162884
```

```
1 - 5      28.9900 5.335599 0.000056
1 - 6      15.6500 5.335599 0.009746
1 - 7      13.9925 5.335599 0.018476
```

## 4.12 BLOQUES AUMENTADOS

La función `DAU.test()` puede ser utilizada para el análisis del diseño de bloques aumentados.

Los datos deben estar organizados en una tabla, conteniendo los bloques, tratamientos y la respuesta.

```
block<-c(rep("I",7),rep("II",6),rep("III",7))
trt<-c("A","B","C","D","g","k","l","A","B","C","D","e","i","A","B",
"C","D","f","h","j")
yield<-c(83,77,78,78,70,75,74,79,81,81,91,79,78,92,79,87,81,89,96,
82)
data.frame(block, trt, yield)
```

	block	trt	yield
1	I	A	83
2	I	B	77
3	I	C	78
4	I	D	78
5	I	g	70
6	I	k	75
7	I	l	74
8	II	A	79
9	II	B	81
10	II	C	81
11	II	D	91
12	II	e	79
13	II	i	78
14	III	A	92
15	III	B	79
16	III	C	87
17	III	D	81
18	III	f	89
19	III	h	96
20	III	j	82

En cada bloque están los tratamientos:

```
by(trt,block,as.character)
block: I
[1] "A" "B" "C" "D" "g" "k" "l"
-----
block: II
[1] "A" "B" "C" "D" "e" "i"
-----
block: III
[1] "A" "B" "C" "D" "f" "h" "j"
```

con sus respectivas respuestas:

```

by(yield,block,as.character)
block: I
[1] "83" "77" "78" "78" "70" "75" "74"
-----
block: II
[1] "79" "81" "81" "91" "79" "78"
-----
block: III
[1] "92" "79" "87" "81" "89" "96" "82"

model<- DAU.test(block,trt,yield,method="lsd",console=TRUE)

ANALYSIS DAU:  yield
Class level information

Block:  I II III
Trt   :  A B C D e f g h i j k l

Number of observations:  20

ANOVA, Treatment Adjusted
Analysis of Variance Table

Response: yield
              Df Sum Sq Mean Sq F value Pr(>F)
block.unadj    2  360.07  180.036
trt.adj       11  285.10   25.918   0.9609 0.5499
Control        3   52.92   17.639   0.6540 0.6092
Control + control.VS.aug.  8 232.18   29.022   1.0760 0.4779
Residuals      6  161.83   26.972

ANOVA, Block Adjusted
Analysis of Variance Table

Response: yield
              Df Sum Sq Mean Sq F value Pr(>F)
trt.unadj     11  575.67   52.333
block.adj      2   69.50   34.750   1.2884 0.3424
Control        3   52.92   17.639   0.6540 0.6092
Augmented      7  505.88   72.268   2.6793 0.1253
Control vs augmented  1  16.88   16.875   0.6256 0.4591
Residuals      6  161.83   26.972

coefficient of variation: 6.4 %
yield Means: 81.5

Critical Differences (Between)          Std Error Diff.

Two Control Treatments                  4.240458
Two Augmented Treatments (Same Block)    7.344688
Two Augmented Treatments(Different Blocks) 8.211611
A Augmented Treatment and A Control Treatment 6.360687

Means with the same letter are not significantly different.

Groups, Treatments and means
a          h          93.5
ab         f          86.5

```



ab	A	84.67
ab	D	83.33
ab	C	82
ab	j	79.5
ab	B	79
ab	e	78.25
ab	k	78.25
ab	i	77.25
ab	l	77.25
b	g	73.25

Comparison between treatments means

<<< to see the objects: pvalue and means >>>

**model\$means**

	yield	std.err	r	Min.	Max.	mean.adj	se.adj	block
A	84.66667	11.532563	3	79	92	84.66667	2.998456	
B	79.00000	3.464102	3	77	81	79.00000	2.998456	
C	82.00000	7.937254	3	78	87	82.00000	2.998456	
D	83.33333	11.789826	3	78	91	83.33333	2.998456	
e	79.00000	NA	1	79	79	78.25000	5.193479	II
f	89.00000	NA	1	89	89	86.50000	5.193479	III
g	70.00000	NA	1	70	70	73.25000	5.193479	I
h	96.00000	NA	1	96	96	93.50000	5.193479	III
i	78.00000	NA	1	78	78	77.25000	5.193479	II
j	82.00000	NA	1	82	82	79.50000	5.193479	III
k	75.00000	NA	1	75	75	78.25000	5.193479	I
l	74.00000	NA	1	74	74	77.25000	5.193479	I

**model<- DAU.test(block,trt,yield,method="lsd", group=FALSE)**

**model\$comparison**

	Difference	pvalue	sig.
A - B	5.666667	0.229886	
A - C	2.666667	0.552612	
A - D	1.333333	0.763840	
A - e	6.416667	0.352008	
...			
B - h	-14.500000	0.062832	.
B - i	1.750000	0.792448	
...			
f - l	9.250000	0.303000	
g - h	-20.250000	0.048720	*
...			
k - l	1.000000	0.896154	

#### 4.13 COMPARACIONES NO-PARAMÉTRICAS

Las funciones para comparaciones múltiples no-paramétricas incluidas en "agricolae" son: `kruskal()`, `waerden.test()`, `friedman()` y `durbin.test()` (Conover, 1999).

La función `kruskal()` se utiliza para muestras de N (N>2) poblaciones o datos provenientes de un experimento completamente aleatorio (poblaciones = tratamientos).

La función `waerden.test()`, similar a `kruskal-wallis`, utiliza score normal en vez de rangos como `kruskal`.

La función `friedman()` se utiliza para evaluaciones organolépticas de diferentes productos por jueces (cada juez evalúa todos los productos) o para el análisis de tratamientos del diseño de bloques completos al azar, donde la respuesta no puede ser tratada mediante al análisis de variancia.

La función `durbin.test()` para el análisis del diseño de bloques incompletos balanceados es muy utilizada para pruebas de degustación, donde los jueces evalúan sólo una parte de los tratamientos.

Datos del libro de Montgomery (Montgomery, 2002)  
Incluidos en el paquete “`agricolae`”

```
library(agricolae)
data(corn)
attach(corn)
str(corn)

'data.frame':  34 obs. of  3 variables:
 $ method      : int  1 1 1 1 1 1 1 1 1 2 ...
 $ observation: int  83 91 94 89 89 96 91 92 90 91 ...
 $ rx          : num  11 23 28.5 17 17 31.5 23 26 19.5 23 ...
```

Para los ejemplos se utilizará los datos del paquete “`agricolae`”.

#### 4.14 KRUSKAL-WALLIS

```
compara<-kruskal(observation,method,group=TRUE, main="corn")
detach(corn)
```

```
Study: corn
Kruskal-Wallis test's
Ties or no Ties
```

```
Value: 25.62884
degrees of freedom: 3
Pvalue chisq  : 1.140573e-05
```

```
method, means of the ranks
```

	observation	r
1	21.83333	9
2	15.30000	10
3	29.57143	7
4	4.81250	8

```
t-Student: 2.042272
Alpha      : 0.05
LSD        : 4.9175
```

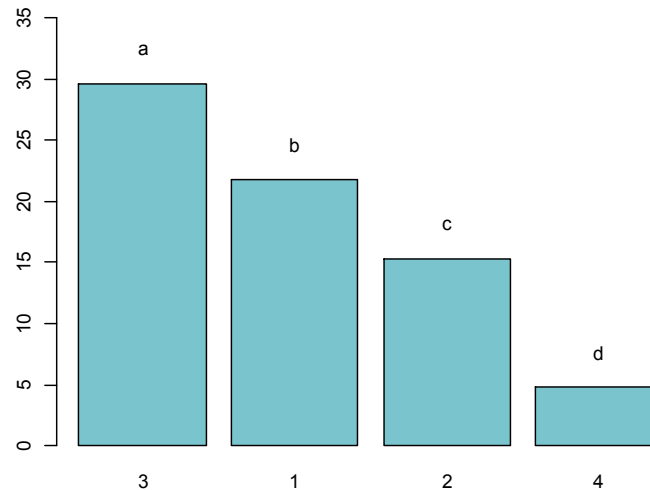
```
Harmonic Mean of Cell Sizes  8.351284
Means with the same letter are not significantly different
```

```
Groups, Treatments and mean of the ranks
```

a	3	29.57
b	1	21.83
c	2	15.3
d	4	4.812

El objeto “compara” tiene la misma estructura de las comparaciones (figura 4.3).

```
par(cex=0.8,mar=c(3,3,1,0))
bar.group(compara$groups,ylim=c(0,35),col=colors()[45])
```



**Figura 4.4. Comparación según Kruskal-Wallis.**

#### 4.15 FRIEDMAN

```
friedman()

library(agricolae)
rm(list=ls())
data(grass)
attach(grass)
compara<-friedman(judge,trt, evaluation,alpha=0.05, group=FALSE,
main="Datos del libro de Conover")
detach(grass)
```

Study: Datos del libro de Conover

trt, Sum of the ranks

	evaluation	r
t1	38.0	12
t2	23.5	12
t3	24.5	12
t4	34.0	12

```
Friedman's Test
=====
Adjusted for ties
Value: 8.097345
Pvalue chisq : 0.04404214
```

F value : 3.192198  
Pvalue F: 0.03621547

Alpha : 0.05  
t-Student : 2.034515

Comparison between treatments  
Sum of the ranks

	Difference	pvalue	sig.	LCL	UCL
t1 - t2	14.5	0.014896	*	3.02	25.98
t1 - t3	13.5	0.022602	*	2.02	24.98
t1 - t4	4.0	0.483434		-7.48	15.48
t2 - t3	-1.0	0.860438		-12.48	10.48
t2 - t4	-10.5	0.071736	.	-21.98	0.98
t3 - t4	-9.5	0.101742		-20.98	1.98

#### 4.16 WAERDEN

waerden.test(), con datos de camote en la base "agricolae".

```
data(sweetpotato)
attach(sweetpotato)
compara<-
waerden.test(yield,virus,alpha=0.01,group=TRUE,console=TRUE)
```

Study:  
Van der Waerden (Normal Scores) test's

Value : 8.409979  
Pvalue: 0.03825667  
Degrees of freedom: 3

virus, means of the normal score

	yield	std.err	r
cc	-0.2328353	0.1748697	3
fc	-1.0601764	0.2002213	3
ff	0.6885684	0.4396858	3
oo	0.6044433	0.2160981	3

t-Student: 3.355387  
Alpha : 0.01  
LSD : 1.322487

Means with the same letter are not significantly different

Groups,	Treatments	means of the normal score
a	ff	0.6886
a	oo	0.6044
ab	cc	-0.2328
b	fc	-1.06

Las probabilidades de comparación se obtienen con el parámetro group=FALSE.

```
compara<-waerden.test(yield,virus,group=FALSE,console=TRUE)
detach(sweetpotato)
```

Study:  
Van der Waerden (Normal Scores) test's

Value : 8.409979  
Pvalue: 0.03825667  
Degrees of freedom: 3

virus, means of the normal score

	yield	std.err	r
cc	-0.2328353	0.1748697	3
fc	-1.0601764	0.2002213	3
ff	0.6885684	0.4396858	3
oo	0.6044433	0.2160981	3

Comparison between treatments means  
mean of the normal score

	Difference	pvalue	sig.	LCL	UCL
cc - fc	0.8273411	0.069032	.	-0.08154345	1.73622564
cc - ff	-0.9214037	0.047582	*	-1.83028827	-0.01251917
cc - oo	-0.8372786	0.066376	.	-1.74616316	0.07160593
fc - ff	-1.7487448	0.002176	**	-2.65762936	-0.83986026
fc - oo	-1.6646197	0.002902	**	-2.57350426	-0.75573516
ff - oo	0.0841251	0.836322		-0.82475944	0.99300965

#### 4.17 DURBIN

durbin(); ejemplo: Myles Hollander (pág. 311) Fuente: W. Moore and C.I. Bliss. (1942)

```
dias <-gl(7,3)
quimico<-c("A","B","D","A","C","E","C","D","G","A","F","G",
"B","C","F","B","E","G","D","E","F")
toxico<-c(0.465,0.343,0.396,0.602,0.873,0.634,0.875,0.325,0.330,
0.423,0.987,0.426,0.652,1.142,0.989,0.536,0.409,0.309,
0.609,0.417,0.931)
```

```
compara<-durbin.test(dias,quimico,toxico,group=F,
main="Logaritmo de la dosis tóxica",console=TRUE)
```

Study: Logaritmo de la dosis tóxica  
quimico, Sum of ranks

	sum
A	5
B	5
C	9
D	5
E	5
F	8
G	5

```
Durbin Test
=====
Value      : 7.714286
Df 1       : 6
```

```

P-value      : 0.2597916
Alpha        : 0.05
Df 2         : 8
t-Student    : 2.306004

```

```

Least Significant Difference
between the sum of ranks:  5.00689

```

```

Parameters BIB
Lambda       : 1
treatmeans   : 7
Block size   : 3
Blocks       : 7
Replication:  3

```

```

Comparison between treatments sum of the ranks

```

	Difference	pvalue	sig.
A - B	0	1.000000	
A - C	-4	0.102688	
A - D	0	1.000000	
A - E	0	1.000000	
A - F	-3	0.204420	
A - G	0	1.000000	
....			
E - G	0	1.000000	
F - G	3	0.204420	

## 5 ANÁLISIS DE ESTABILIDAD

En “agricolae” se tiene dos métodos para el estudio de estabilidad y el modelo AMMI. Éstos son: un modelo paramétrico para una selección simultánea en rendimiento y estabilidad “SHUKLA'S STABILITY VARIANCE AND KANG'S”, y un método no paramétrico de Haynes, basado en el rango de los datos.

### 5.1 ESTABILIDAD PARAMÉTRICA

Uso del modelo paramétrico función `stability.par()`

Preparar una tabla de datos donde las filas sean los genotipos y las columnas, los ambientes. Los datos deben corresponder a promedios de rendimiento u otra variable medida. Determinar la variancia del error común para todos los ambientes y el número de repeticiones que fue evaluado para cada genotipo. Si las repeticiones son diferentes, hallar un promedio armónico que representará al conjunto. Finalmente, asigne un nombre a cada fila que representará al genotipo. Para el ejemplo se considera 5 ambientes:

```

v1 <- c(10.2, 8.8, 8.8, 9.3, 9.6, 7.2, 8.4, 9.6, 7.9, 10, 9.3, 8.0, 10.1, 9.4, 10.8, 6.3, 7.4)
v2 <- c(7, 7.8, 7.0, 6.9, 7, 8.3, 7.4, 6.5, 6.8, 7.9, 7.3, 6.8, 8.1, 7.1, 7.1, 6.4, 4.1)
v3 <- c(5.3, 4.4, 5.3, 4.4, 5.5, 4.6, 6.2, 6.0, 6.5, 5.3, 5.7, 4.4, 4.2, 5.6, 5.8, 3.9, 3.8)
v4 <- c(7.8, 5.9, 7.3, 5.9, 7.8, 6.3, 7.9, 7.5, 7.6, 5.4, 5.6, 7.8, 6.5, 8.1, 7.5, 5.0, 5.4)
v5 <- c(9, 9.2, 8.8, 10.6, 8.3, 9.3, 9.6, 8.8, 7.9, 9.1, 7.7, 9.5, 9.4, 9.4, 10.3, 8.8, 8.7)

```

Para 17 genotipos, se identifica por letras.

```
estudio <- data.frame(v1, v2, v3, v4, v5)
rownames(estudio) <- LETTERS[1:17]
```

Se asume una variancia del error de 2 y 4 repeticiones.

```
estabilidad <- stability.par(estudio, rep=4, MSError=2)
```

INTERACTIVE PROGRAM FOR CALCULATING SHUKLA'S STABILITY VARIANCE AND KANG'S

YIELD - STABILITY (YSi) STATISTICS

Environmental index - covariate

Analysis of Variance

	d.f.	Sum of Squares	Mean Squares	F	p.value
TOTAL	84	1035.6075			
GENOTYPES	16	120.0875	7.5055	2.65	0.003
ENVIRONMENTS	4	734.2475	183.5619	91.78	<0.001
INTERACTION	64	181.2725	2.8324	1.42	0.033
HETEROGENEITY	16	52.7128	3.2945	1.23	0.281
RESIDUAL	48	128.5597	2.6783	1.34	0.0815
POOLED ERROR	240		2		

Genotype. Stability statistics

	Mean	Sigma-square	. s-square	. Ecovalence
A	7.86	1.671833	ns	2.209084
B	7.22	1.822233	ns	1.977299
C	7.44	0.233967	ns	0.134103
D	7.42	4.079567	ns	1.443859
E	7.64	2.037967	ns	2.369090
F	7.14	5.161967	*	6.763106
G	7.90	1.759300	ns	1.058092
H	7.68	1.757167	ns	2.028880
I	7.34	5.495300	*	0.423680
J	7.54	4.129967	ns	5.125514
K	7.12	3.848900	ns	4.360772
L	7.30	2.675300	ns	3.610982
M	7.66	3.473167	ns	2.198229
N	7.92	0.806233	ns	1.097156
O	8.30	1.951300	ns	1.459578
P	6.08	3.647833	ns	4.919102
Q	5.88	3.598500	ns	4.353030

Signif. codes: 0 '\*\*\*' 0.01 '\*\*' 0.05 'ns' 1

Simultaneous selection for yield and stability (++)

	Yield	Rank	Adj.rank	Adjusted	Stab.var	Stab.rating	YSi	...
A	7.86	14	1	15	1.671833	0	15	+
B	7.22	5	-1	4	1.822233	0	4	
C	7.44	9	1	10	0.233967	0	10	+
D	7.42	8	1	9	4.079567	-2	7	
E	7.64	11	1	12	2.037967	0	12	+
F	7.14	4	-1	3	5.161967	-4	-1	
G	7.90	15	1	16	1.759300	0	16	+

```

H  7.68  13      1      14 1.757167      0 14  +
I  7.34   7     -1      6 5.495300     -4  2
J  7.54  10      1     11 4.129967     -2  9  +
K  7.12   3     -1      2 3.848900      0  2
L  7.30   6     -1      5 2.675300      0  5
M  7.66  12      1     13 3.473167      0 13  +
N  7.92  16      1     17 0.806233      0 17  +
O  8.30  17      2     19 1.951300      0 19  +
P  6.08   2     -2      0 3.647833      0  0
Q  5.88   1     -3     -2 3.598500      0 -2

Yield Mean: 7.378824
YS      Mean: 8.352941
LSD (0.05): 0.7384513
- - - - -
+   selected genotype
++  Reference: Kang, M. S. 1993. Simultaneous selection for yield
and stability: Consequences for growers. Agron. J. 85:754-757.

```

Los genotipos seleccionados son: A, C, E, G, H, J, M, N y O. Estos genotipos tienen un rendimiento más alto y una variación más baja. Según el ANOVA, la interacción es significativa.

Si se tiene un índice ambiental, por ejemplo, se puede adicionar como una covariable. Para este caso se incluye la altitud de las localidades.

```

altitud<-c(1200, 1300, 800, 1600, 2400)
estabilidad <- stability.par(estudio,rep=4,MSerror=2,cova=TRUE,
name.cov="Altitud", file.cov= altitud)

```

## 5.2 ESTABILIDAD NO-PARAMÉTRICA

Para estabilidad no-paramétrica, la función en “agricolae” es `stability.nonpar()`. Esta función requiere que en la primera columna se incluya los nombres de los genotipos y en las otras columnas, la respuesta por ambientes.

```

datos <- data.frame(nombre=row.names(estudio), estudio)
modelo<-stability.nonpar(datos, "YIELD", ranking=TRUE)

```

```

Non-parametric Method for Stability Analysis
-----
Estimation and test of non-parametric measures
Variable: YIELD

```

```

Ranking...
      v1  v2 v3  v4  v5
A 16.0  8.0  9 14.0  8.0
B  7.5 14.0  5  5.5 10.0
C  7.5  8.0  9  9.0  6.0
D  9.5  6.0  5  5.5 17.0
E 12.5  8.0 11 14.0  3.0
F  2.0 17.0  7  7.0 11.0
G  6.0 13.0 16 16.0 15.0
H 12.5  3.0 15 10.5  6.0
I  4.0  4.5 17 12.0  2.0
J 14.0 15.0  9  2.5  9.0
K  9.5 12.0 13  4.0  1.0
L  5.0  4.5  5 14.0 14.0

```



```

M 15.0 16.0 3 8.0 12.5
N 11.0 10.5 12 17.0 12.5
O 17.0 10.5 14 10.5 16.0
P 1.0 2.0 2 1.0 6.0
Q 3.0 1.0 1 2.5 4.0

```

Statistics...

	Mean	Rank	s1	Z1	s2	Z2
A	7.86	14	5.4	0.02	21.5	0.04
B	7.22	5	6.2	0.12	25.7	0.02
C	7.44	9	3.0	2.73	7.5	1.83
D	7.42	8	7.4	1.20	36.5	1.05
E	7.64	11	5.6	0.00	21.8	0.03
F	7.14	4	7.8	1.81	39.2	1.55
G	7.90	15	5.2	0.08	18.7	0.19
H	7.68	13	6.2	0.12	25.3	0.01
I	7.34	7	8.8	3.87	51.5	5.08
J	7.54	10	7.2	0.94	34.3	0.71
K	7.12	3	7.8	1.81	43.0	2.43
L	7.30	6	7.4	1.20	34.7	0.77
M	7.66	12	7.6	1.49	38.2	1.36
N	7.92	16	4.2	0.82	14.8	0.57
O	8.30	17	7.0	0.71	31.7	0.40
P	6.08	2	6.6	0.35	27.7	0.09
Q	5.88	1	7.0	0.71	32.3	0.46

Sum of Z1: 17.97158

Sum of Z2: 16.59462

Test...

The Z-statistics are measures of stability. The test for the significance of the sum of Z1 or Z2 is compared to a Chi-Square value of chi.sum. Individually, Z1 or Z2 are compared to a Chi-square value of chi.ind.

	MEAN	es1	es2	vs1	vs2	chi.ind	chi.sum
1	7.378824	5.647059	24	2.566667	148.8	8.843605	27.58711

---  
expectation and variance: es1, es2, vs1, vs2

### 5.3 AMMI

El modelo AMMI utiliza el biplot construido mediante las componentes principales generadas por la interacción ambiente y genotipo. Si existe dicha interacción, el porcentaje de las dos componentes principales explicaría más del 50% de la variación total; entonces el biplot es una buena alternativa para estudiar la relación ambiente y genotipo.

Los datos para AMMI deben provenir de experimentos similares conducidos en ambientes distintos. Se requiere homogeneidad de variancia del error experimental producido en los diferentes ambientes. El análisis se realiza combinando los experimentos.

Los datos pueden estar organizados por columna, así: ambiente, genotipo, repetición y variable.

Los datos también pueden ser los promedios de los genotipos en cada ambiente, pero es necesario considerar un promedio armónico para las repeticiones y una variancia común del error. Los datos deben estar organizados en columnas: ambiente, genotipo y variable.

Al ejecutar AMMI, éste genera los gráficos de BIPLLOT, ver figura 5.1.

Para la aplicación, se considera los datos utilizados en el ejemplo de estabilidad paramétrica (estudio):

```
rdto <- c(estudio[,1], estudio[,2], estudio[,3], estudio[,4],
estudio[,5])
ambiente <- gl(5,17)
genotipo <- rep(rownames(estudio),5)
```

```
modelo<-AMMI(ENV=ambiente, GEN=genotipo, REP=4, Y=rdto, MSE=2,
ylim=c(-2,2), xlim=c(-2,2), number=FALSE)
```

```
ANALYSIS AMMI: rdto
Class level information
```

```
ENV: 1 2 3 4 5
GEN: A B C D E F G H I J K L M N O P Q
REP: 4
```

```
Number of means: 85
```

```
Dependent Variable: rdto
```

```
Analysis of variance
```

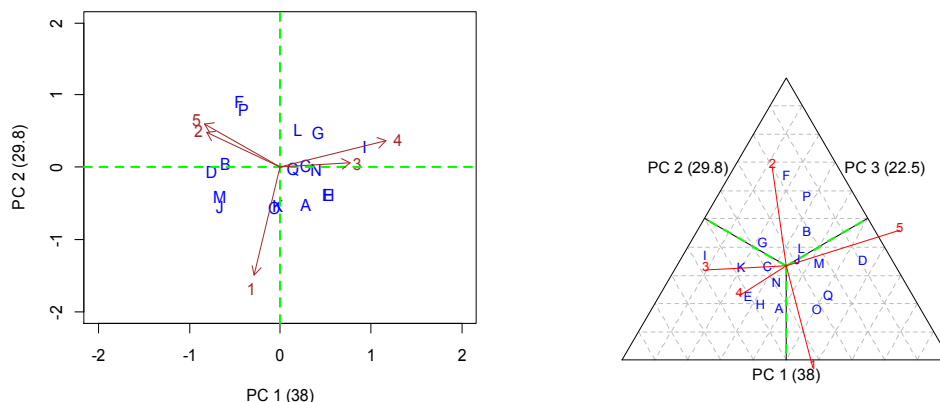
	Df	Sum Sq	Mean Sq	F value	Pr(>F)
ENV	4	734.2475	183.561882		
REP(ENV)	15				
GEN	16	120.0875	7.505471	3.752735	3.406054e-06
ENV:GEN	64	181.2725	2.832382	1.416191	3.279630e-02
Residuals	240	480.0000	2.000000		

```
Coeff var      Mean rdto
19.16584      7.378824
```

```
Analysis
```

	percent	acum	Df	Sum.Sq	Mean.Sq	F.value	Pr.F
PC1	38.0	38.0	19	68.96258	3.629609	1.81	0.0225
PC2	29.8	67.8	17	54.02864	3.178155	1.59	0.0675
PC3	22.5	90.3	15	40.84756	2.723170	1.36	0.1680
PC4	9.6	99.9	13	17.43370	1.341054	0.67	0.7915
PC5	0.0	99.9	11	0.00000	0.000000	0.00	1.0000

```
require(klaR)
par(mar=c(4,4,0,0))
model<-AMMI(ENV=environment, GEN=genotype, REP=4, Y=rdto, MSE=2,
graph="triplot",number=F)
```



**Figura 5.1. Biplot y Triplot**

En este caso, la interacción es significativa. Las dos primeras componentes explican el 67.8%; entonces el biplot puede proporcionar información sobre la relación de genotipos y ambientes. Con el triplot, se explicaría el 90.3%.

# para una evaluación de las componentes principales, ejecute:

```
CP<-princomp(model$genXenv)
summary(CP)
CP$loadings
```

## 6 FUNCIONES ESPECIALES

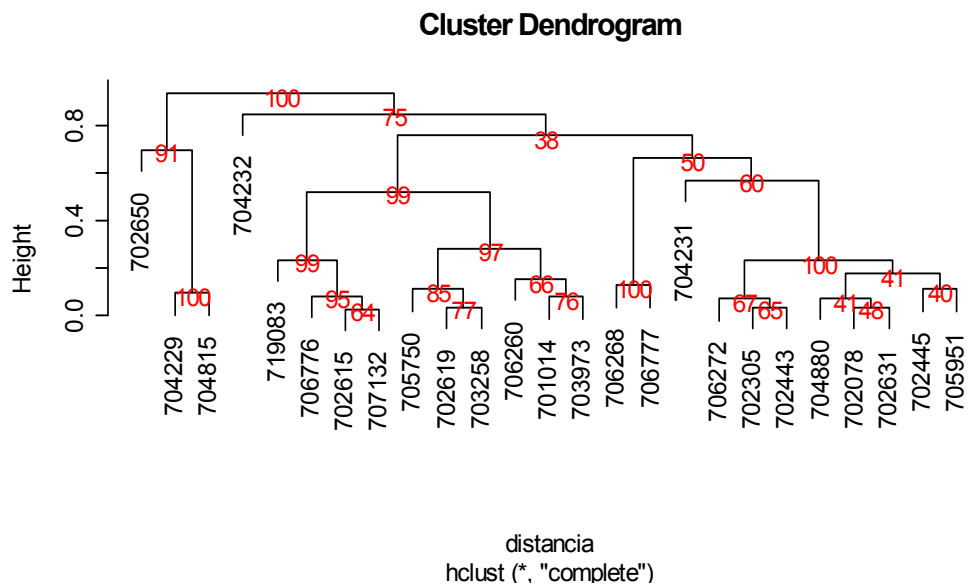
### 6.1 CONSENSO DE DENDROGRAMA

El consenso es el grado o semejanza de los vértices de un árbol respecto a sus ramas del dendrograma construido. La función a aplicar es consensus().

Los datos corresponden a una tabla, con el nombre de los individuos en las filas y el nombre de las variables en las columnas. Para la demostración, se utilizará los datos "pamCIP" de "agricolae", correspondientes a marcadores moleculares de 43 entradas de un banco de germoplasma (filas) y 107 marcadores (columnas).

El programa identifica duplicados en las filas y puede operar en ambos casos. El resultado es un dendrograma, en el que se incluye el porcentaje de consenso, figura 6.1.

```
data(pamCIP)
rownames(pamCIP)<-substr(rownames(pamCIP),1,6)
par(cex=0.8)
output<-consensus(pamCIP,distance="binary", method="complete",
nboot=500)
```



**Figura 6.1. Dendrograma, producción por consenso()**

Duplicates: 18

New data : 25 Records

Consensus hclust

Method distance: binary

Method cluster : complete

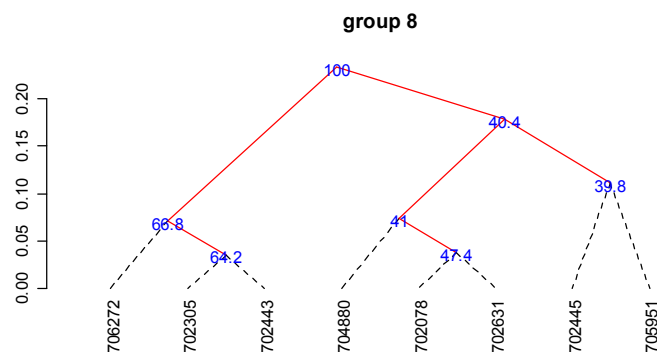
rows and cols : 25 107

n-bootstrap : 500

Run time : 20.469 secs

Cuando el dendrograma es complejo, es conveniente extraer parte de éste con la función `hcut()`, figura 6.2.

```
hcut(output,h=0.4,group=8,type="t",edgePar = list(lty=1:2,
col=2:1),main="group 8",
,col.text="blue",cex.text=1)
```



**Figura 6.2. Dendrograma, producción por `hcut()`**

El objeto "output" obtenido contiene información del proceso:

```
names(output)
```

```
[1] "table.dend" "dendrogram" "duplicates"
```

Esto significa que se puede conocer los duplicados, volver a construir el diagrama del árbol y tener las relaciones.

```
output$ table.dend
```

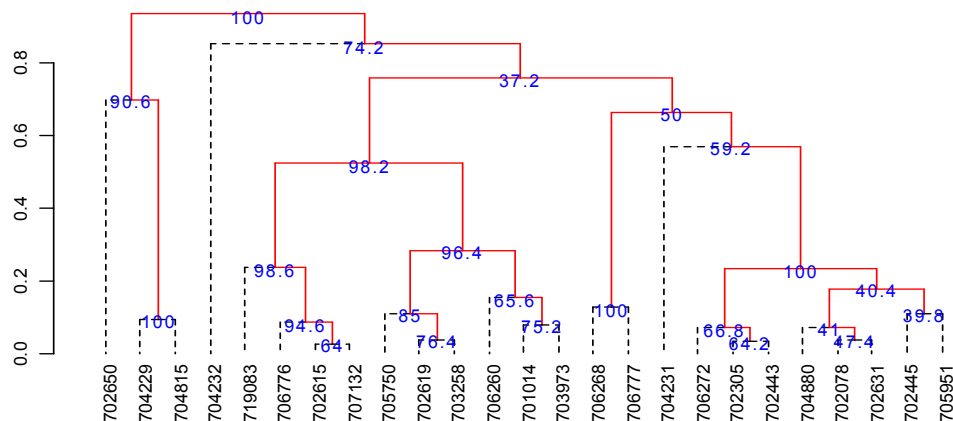
	X1	X2	xaxis	height	percentage	groups
1	-6	-24	7.500000	0.02857143	64.0	6-24
2	-3	-4	19.500000	0.03571429	64.2	3-4
...						
24	21	23	5.099609	0.93617021	100.0	1-2-3-4-5-6-7-8-9-10-11-
						12-13-14-15-16-17-18-19-20-21-22-23-24-25

Reproducir el dendrograma:

```
dend<-output$dendrogram
data<-output$table.dend
plot(dend)
text(data[,3],data[,4],data[,5])
```

Construir un dendrograma clásico, figura 6.3

```
dend<-as.dendrogram(output$dendrogram)
plot(dend,type="r",edgePar = list(lty=1:2, col=2:1))
text(data[,3],data[,4],data[,5],col="blue",cex=1)
```



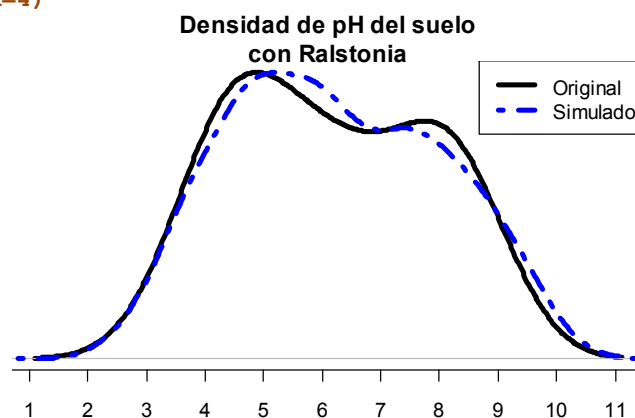
**Figura 6.3. Dendrograma clásico**

## 6.2 MONTECARLO

Es un método para generar números aleatorios de una distribución desconocida; utiliza un conjunto de datos y mediante el comportamiento acumulativo de su frecuencia relativa genera los posibles valores aleatorios que siguen la distribución de los datos. Estos nuevos números son utilizados por algún proceso de simulación.

La densidad de probabilidad de los datos originales y simulados pueden ser comparados, figura 6.4.

```
data(soil)
set.seed(9473)
simulado <- montecarlo(soil$pH,1000)
par(mar=c(3,0,2,1))
plot(density(soil$pH),axes=F,main="Densidad de pH del suelo\ncon
Ralstonia",xlab="",lwd=4)
lines(density(simulado), col="blue", lty=4,lwd=4)
h<-graph.freq(simulado,plot=F)
axis(1,0:12)
legend("topright",c("Original","Simulado"),lty=c(1,4),col=c("black",
"blue"), lwd=4)
```



**Figura 6.4. Distribución de los datos simulados y el original**

Se han generado 1000 datos y la tabla de frecuencia es:

```
round(table.freq(h),2)
```

Inf	Sup	MC	fi	fri	Fi	Fri
1.60	2.45	2.03	4	0.00	4	0.00
. . .						
2.45	3.31	2.88	42	0.04	46	0.05
10.14	10.99	10.57	8	0.01	1000	1.00

Véanse algunas estadísticas:

```
summary(soil$pH)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
3.800	4.700	6.100	6.154	7.600	8.400

```
summary(simulado)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.443	4.698	6.022	6.209	7.762	10.950

### 6.3 RE-MUESTREO EN MODELO LINEAL

Utiliza el método de permutación para el cálculo de las probabilidades de las fuentes de variación del ANOVA según el modelo

lineal de regresión o el diseño utilizado. El principio es que la respuesta Y no depende de las medias planteadas en el modelo propuesto; por lo tanto, se puede permutar los valores Y y construir muchos estimados del modelo. En base al comportamiento de las variables aleatorias de los elementos en estudio, se calcula la probabilidad para medir la significación.

Los datos deben ser preparados en forma similar para un análisis de variancia. La función a utilizar es: `resampling.model()`

```
data(potato)
potato[,1]<-as.factor(potato[,1])
potato[,2]<-as.factor(potato[,2])
model<-"cutting~variety + date + variety:date"
analysis<-resampling.model(1000,potato,model)
```

```
Resampling of the experiments
- - - - -
Proposed model: cutting~variety + date + variety:date
---
Resampling of the analysis of variance for the proposed model
Determination of the P-Value by Resampling
Samples: 1000
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	Resampling
variety	1	25.086806	25.086806	7.2580377	0.01952218	0.025
date	2	13.891758	6.945879	2.0095604	0.17670768	0.200
variety:date	2	4.853025	2.426513	0.7020312	0.51483592	0.530
Residuals	12	41.477005	3.456417			

```
---
```

La función `resampling.model()` puede ser utilizada cuando los errores tienen una distribución distinta a la normal.

## 6.4 SIMULACIÓN EN MODELO LINEAL

Bajo el supuesto de normalidad, la función genera errores seudo experimentales bajo el modelo propuesto y determina la proporción de resultados válidos según el análisis de variancia encontrado.

La función es: `simulation.model()`. Los datos son preparados en una tabla, en forma idéntica a la de un análisis de variancia.

Para el ejemplo planteado en el procedimiento anterior:

```
model <- simulation.model(1000, potato, model)

Simulation of experiments
Under the normality assumption
- - - - -
Proposed model: cutting~variety + date + variety:date
Analysis of Variance Table
```

```
Response: cutting
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
variety	1	25.087	25.087	7.2580	0.01952 *
date	2	13.892	6.946	2.0096	0.17671

```

variety:date  2  4.853    2.427  0.7020 0.51484
Residuals    12 41.477    3.456
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
---
Validation of the analysis of variance for the proposed model
Simulations: 1000

```

	Df	F value	% Acceptance	% Rejection	Criterion
variety	1	7.2580377	51.5	48.5	acceptable
date	2	2.0095604	61.1	38.9	acceptable
variety:date	2	0.7020312	67.5	32.5	acceptable

```

---

```

La validación es referida al porcentaje de resultados de decisión iguales al resultado de decisión del ANOVA. Así, 67.5% de los resultados simulados sobre la interacción variety\*date dio el mismo resultado de aceptación o rechazo obtenido en el ANOVA.

## 6.5 ANALISIS PATH

Corresponde al método “path análisis”; los datos corresponden a matrices de correlación de las independientes con la dependiente (XY) y entre las independientes (XX).

Es necesario asignar nombres a las filas y columnas para identificar los efectos directos e indirectos.

```

corr.x<- matrix(c(1,0.5,0.5,1),c(2,2))
corr.y<- rbind(0.6,0.7)
names<-c("X1","X2")
dimnames(corr.x)<-list(names,names)
dimnames(corr.y)<-list(names,"Y")
resultado <- path.analysis(corr.x,corr.y)

Direct(Diagonal) and indirect effect path coefficients
=====
              X1          X2
X1 0.3333333 0.2666667
X2 0.1666667 0.5333333

Residual Effect^2 =  0.4266667

names(resultado)
[1] "Coeff"      "Residual"

```

## 6.6 LINEA POR PROBADOR

Corresponde a un análisis de cruza de un diseño genético. Los datos deben estar organizados en una tabla. Sólo se necesita 4 columnas: repetición, hembras, machos y respuesta. En el caso que corresponda a progenitores, el campo de hembras o machos sólo será llenado con el que corresponda. Ver los datos heterosis.

Ejemplo con los datos de heterosis, localidad 2.



	Replication	Female	Male	v2
109	1	LT-8	TS-15	2.65
110	1	LT-8	TPS-13	2.26
...				
131	1	Achirana	TPS-13	3.55
132	1	Achirana	TPS-67	3.05
133	1	LT-8	<NA>	2.93
134	1	TPS-2	<NA>	2.91
...				
140	1	Achirana	<NA>	3.35
...				
215	3	<NA>	TPS-67	2.91

donde <NA> es vacío.

Si es una progenie, ésta proviene de un "Female" y un "Male".  
Si es un progenitor, sólo será "Female" o "Male".

En este ejemplo se tiene datos de la localidad 2, correspondientes a:

24 progenies (cruzas)  
8 hembras  
3 machos  
3 repeticiones

Son 35 tratamientos (24, 8, 3) aplicados a 3 bloques.

```
rm(list=ls())
data(heterosis)
site2<-subset(heterosis,heterosis[,1]==2)
site2<-subset(site2[,c(2,5,6,8)],site2[,4]!="Control")
attach(site2)
output1<-lineXtester(Replication, Female, Male, v2)
detach(site2)
```

ANALYSIS LINE x TESTER: v2

ANOVA with parents and crosses

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Replications	2	0.519190476	0.259595238	9.801	0.0002
Treatments	34	16.101605714	0.473576639	17.879	0.0000
Parents	10	7.731490909	0.773149091	29.189	0.0000
Parents vs. Crosses	1	0.005082861	0.005082861	0.192	0.6626
Crosses	23	8.365031944	0.363697041	13.731	0.0000
Error	68	1.801142857	0.026487395		
Total	104	18.421939048			

ANOVA for line X tester analysis

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Lines	7	4.9755431	0.71079187	3.632	0.0191
Testers	2	0.6493861	0.32469306	1.659	0.2256
Lines X Testers	14	2.7401028	0.19572163	7.389	0.0000
Error	68	1.8011429	0.02648739		

# ANOVA for line X tester analysis including parents

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Replications	2	0.519190476	0.259595238	9.801	0.0002
Treatments	34	16.101605714	0.473576639	17.879	0.0000
Parents	10	7.731490909	0.773149091	29.189	0.0000
Parents vs. Crosses	1	0.005082861	0.005082861	0.192	0.6626
Crosses	23	8.365031944	0.363697041	13.731	0.0000
Lines	7	4.975543056	0.710791865	3.632	0.0191
Testers	2	0.649386111	0.324693056	1.659	0.2256
Lines X Testers	14	2.740102778	0.195721627	7.389	0.0000
Error	68	1.801142857	0.026487395		
Total	104	18.421939048			

## GCA Effects:

=====

### Lines Effects:

	LT-8	MF-I	MF-II	Serrana	TPS-2	TPS-25
Achirana						
TPS-7						
	0.022	-0.338	0.199	-0.449	0.058	-0.047
	0.141					0.414

### Testers Effects:

TPS-13	TPS-67	TS-15
0.087	0.046	-0.132

## SCA Effects:

=====

	TPS-13	TPS-67	TS-15
Achirana	0.061	0.059	-0.120
LT-8	-0.435	0.519	-0.083
MF-I	-0.122	-0.065	0.187
MF-II	-0.194	0.047	0.148
Serrana	0.032	-0.113	0.081
TPS-2	0.197	-0.072	-0.124
TPS-25	0.126	-0.200	0.074
TPS-7	0.336	-0.173	-0.162

## Standard Errors for Combining Ability Effects:

=====

S.E. (gca for line)	: 0.05424983
S.E. (gca for tester)	: 0.0332211
S.E. (sca effect)	: 0.09396346
S.E. (gi - gj)line	: 0.07672084
S.E. (gi - gj)tester	: 0.04698173
S.E. (sij - skl)tester	: 0.1328844

## Genetic Components:

=====

Cov H.S. (line)	: 0.05723003
Cov H.S. (tester)	: 0.00537381
Cov H.S. (average)	: 0.003867302
Cov F.S. (average)	: 0.1279716
F = 0, Additive genetic variance	: 0.01546921
F = 1, Additive genetic variance	: 0.007734604
F = 0, Variance due to Dominance	: 0.1128228
F = 1, Variance due to Dominance	: 0.05641141

Proportional contribution of lines, testers  
and their interactions to total variance

=====

Contributions of lines : 59.48026

Contributions of testers: 7.763104

Contributions of lxt : 32.75663

## 6.7 UNIFORMIDAD DEL SUELO

El índice de Smith es un indicador de la uniformidad, utilizado para determinar el tamaño de la parcela para fines de investigación. Los datos corresponden a una matriz o tabla que contenga la respuesta por unidad básica, un número de n filas por m columnas, y un total de n\*m unidades básicas.

Para la prueba se utilizará el archivo de arroz. El gráfico es un resultado con el ajuste de un modelo para el tamaño de la parcela y el coeficiente de variación, figura 6.5.

```
data(rice)
table<-index.smith(rice,
  main="Relación entre el CV y el tamaño de la parcela" ,col="red",
  type="l",xlab="Tamaño")
uniformidad <- data.frame(table$uniformity)
uniformidad
```

	Size	Width	Length	plots	Vx	CV
1	1	1	1	648	9044.539	13.0
2	2	1	2	324	7816.068	12.1
3	2	2	1	324	7831.232	12.1
4	3	1	3	216	7347.975	11.7
5	3	3	1	216	7355.216	11.7
...						
40	162	9	18	4	4009.765	8.6

El tamaño es el producto del ancho por el largo de la parcela, y la forma rectangular el tamaño del ancho y el largo.

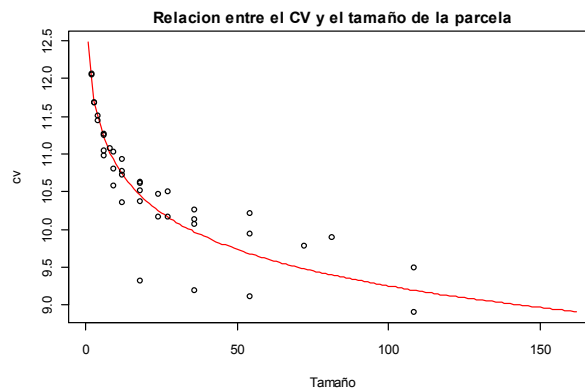


Figura 6.5. Curva de ajuste para el tamaño óptimo de parcela

## 6.8 LÍMITES DE CONFIANZA EN ÍNDICES DE BIODIVERSIDAD

Los índices de biodiversidad son muy utilizados para medir la presencia de seres vivos en un área ecológica. Muchos programas indican su valor. La función de "agricolae" es mostrar, además, los intervalos de confianza, los cuales pueden ser utilizados para una comparación estadística. El procedimiento es por bootstrap. Los datos se organizan en una tabla; en una columna, las especies; y en otra, la cantidad de individuos. Los índices que se pueden calcular con la función `index.bio()` de "agricolae" son: "Margalef", "Simpson.Dom", "Simpson.Div", "Berger.Parker", "McIntosh", y "Shannon".

Para el ejemplo se utilizará los datos obtenidos en la localidad de Paracsho, distrito de Huasahuasi, provincia de Tarma del departamento de Junín.

La evaluación se realizó el 17 de noviembre de 2005 en las parcelas, sin aplicación de insecticidas. Se contabilizó los especímenes siguientes:

```
data(paracsho)
especies <- paracsho[79:87,4:6]
especies
```

	Order	Family	Number.of.specimens
79	DIPTERA	TIPULIDAE	3
80	LEPIDOPTERA	NOCTUIDAE	1
81	NOCTUIDAE	PYRALIDAE	3
82	HEMIPTERA	ANTHOCORIDAE	1
83	DIPTERA	TACHINIDAE	16
84	DIPTERA	ANTHOCORIDAE	3
85	DIPTERA	SCATOPHAGIDAE	5
86	DIPTERA	SYRPHIDAE	1
87	DIPTERA	MUSCIDAE	

El índice de Shannon es:

```
output <-
index.bio(especies[,3],method="Shannon",level=95,nboot=200)
```

Method: Shannon

The index: 3.52304

95 percent confidence interval:  
3.090695 ; 4.264727

## 6.9 CORRELACIÓN

La función `correlation()` de "agricolae" realiza las correlaciones mediante los métodos Pearson, Spearman y Kendall para vectores y/o matrices. Si son dos vectores, realiza la prueba para una o dos colas; si es matricial, determina las probabilidades para una diferencia, sea ésta mayor o menor.

Para su aplicación, considere los datos del suelo: `data(soil)`

```
data(soil)
correlation(soil[,2:4],method="pearson")
```

## Correlation Analysis

Method : pearson  
Alternative: two.sided

```
$correlation
      pH    EC CaCO3
pH    1.00 0.55  0.73
EC    0.55 1.00  0.32
CaCO3 0.73 0.32  1.00
```

```
$pvalue
      pH          EC          CaCO3
pH    1.0000000000 0.0525330 0.004797027
EC    0.052532997 1.0000000 0.294159813
CaCO3 0.004797027 0.2941598 1.000000000
```

```
$n.obs
[1] 13
```

```
attach(soil)
correlation(pH,soil[,3:4],method="pearson")
Correlation Analysis
```

Method : pearson  
Alternative: two.sided

```
$correlation
      EC CaCO3
pH 0.55  0.73
```

```
$pvalue
      EC CaCO3
pH 0.0525 0.0048
```

```
$n.obs
[1] 13
```

```
correlation(pH,CaCO3,method="pearson")
```

Pearson's product-moment correlation

```
data: pH and CaCO3
t = 3.520169 , df = 11 , p-value = 0.004797027
alternative hypothesis: true rho is not equal to 0
sample estimates:
cor
0.7278362
```

## 6.10 OTRAS FUNCIONES

Funciones de conveniencia que facilitan el manejo de datos:

`tapply.stat()` Cálculo de estadísticas y operaciones matemáticas en columnas de una tabla en función de factores agrupados.

## Tabla de factores y variables

Aplicación con datos de “agricolae”:

```
data(RioChillon)
attach(RioChillon$babies)
tapply(yield,farmer,function(x) max(x)-min(x))
detach(RioChillon$babies)
```

	farmer	yield
1	AugustoZambrano	7.5
2	Caballero	13.4
3	ChocasAlto	14.1
4	FelixAndia	19.4
5	Huarangal-1	9.8
6	Huarangal-2	9.1
7	Huarangal-3	9.4
8	Huatocay	19.4
9	IgnacioPolinario	13.1

Corresponde al rango de variación en el rendimiento de los agricultores.

La función “tapply” puede ser utilizada en forma directa o con función.

Si A es una tabla con columnas 1,2 y 3 como categorías y 5,6 y 7 como variables, los siguientes procedimientos son válidos:

```
tapply.stat(A[,5:7], A[,1:3],mean)
tapply.stat(A[,5:7], A[,1:3],function(x) mean(x,na.rm=TRUE))
tapply.stat(A[,c(7,6)], A[,1:2],function(x) sd(x)*100/mean(x))
```

## Coeficiente de variación de un experimento

Si “modelo” es el objeto resultado de un análisis de variancia de la función aov() o lm() de R, entonces la función cv.model() calcula el coeficiente de variación.

```
data(sweetpotato)
modelo <- aov(yield ~ virus, data=sweetpotato)
cv.model(modelo)
[1] 17.16660
```

## Asimetría y curtosis

Los resultados de asimetría y curtosis obtenidos por “agricolae” son iguales a los obtenidos por SAS, MiniTab, SPSS, InfoStat y Excel.

Si x representa a un conjunto de datos:

```
> x<-c(3,4,5,2,3,4,5,6,4,NA,7),
```

la asimetría se calcula con:

```
> skewness(x)
[1] 0.3595431
```

y curtosis con:

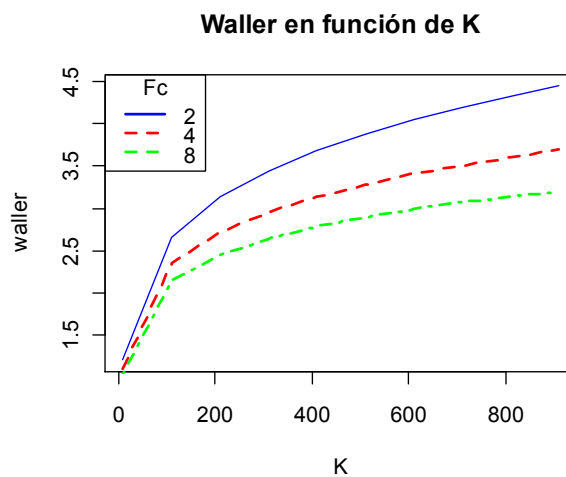
```
> kurtosis(x)
[1] -0.1517996
```

### Valor tabular de Waller-Duncan

La función Waller determina el valor tabular de Waller-Duncan. Para el cálculo, es necesario el valor F, calculado del análisis de variancia del factor de estudio, con sus grados de libertad y el estimado de la variancia del error experimental. El valor K parámetro de la función es la razón entre los dos tipos de errores (I y II); para el uso, se asigna un valor que guarde relación con el nivel alfa. Cuando el nivel alfa es 0.10, se asigna 50 a K; para 0.05, K=100; y para 0.01, K=500. K puede tomar cualquier valor.

La figura 6.6 grafica la función para valores diferentes de K con grados de libertad de 5 para el numerador y 15 para el denominador, y valores de F calculados, iguales a 2, 4 y 8.

```
q<-5
f<-15
K<-seq(10,1000,100)
n<-length(K)
y<-rep(0,3*n)
dim(y)<-c(n,3)
for(i in 1:n) y[i,1]<-waller(K[i],q,f,Fc=2)
for(i in 1:n) y[i,2]<-waller(K[i],q,f,Fc=4)
for(i in 1:n) y[i,3]<-waller(K[i],q,f,Fc=8)
plot(K,y[,1],type="l",col="blue",ylab="waller")
lines(K,y[,2],type="l",col="red",lty=2,lwd=2)
lines(K,y[,3],type="l",col="green",lty=4,lwd=2)
legend("topleft",c("2","4","8"),col=c("blue","red","green"),lty=c(1,
8,20),lwd=2,title="Fc")
title(main="Waller en función de K")
```



**Figura 6.6. Función de Waller a diferente valor de los parámetros K y Fc**

## AUDPC

Área bajo la curva del progreso de la enfermedad, figura 6.7. La función AUDPC calcula el absoluto y relativo del progreso de la enfermedad. Se requiere medir la enfermedad en porcentaje durante varias fechas, de preferencia equidistantes.

```
dias<-c(7,14,21,28,35,42)
evaluacion<-data.frame(E1=10,E2=40,E3=50,E4=70,E5=80,E6=90)
plot(dias,evaluacion,type="h",ylim=c(0,100),axes=F,col="red",xlab="Días",ylab="Evaluación")
lines(dias,evaluacion,col="red")
axis(1,dias)
axis(2,seq(0,100,20),las=2)
abline(v=7,h=100,lty=4,lwd=2,col="blue")
abline(v=42,h=0,lty=4,lwd=2,col="blue")
audpc(evaluacion,dias)
audpc(evaluacion,dias,"relative")
text(15,80,"Audpc Absoluta = 2030")
text(15,70,"Audpc Relativa = 0.58")
```

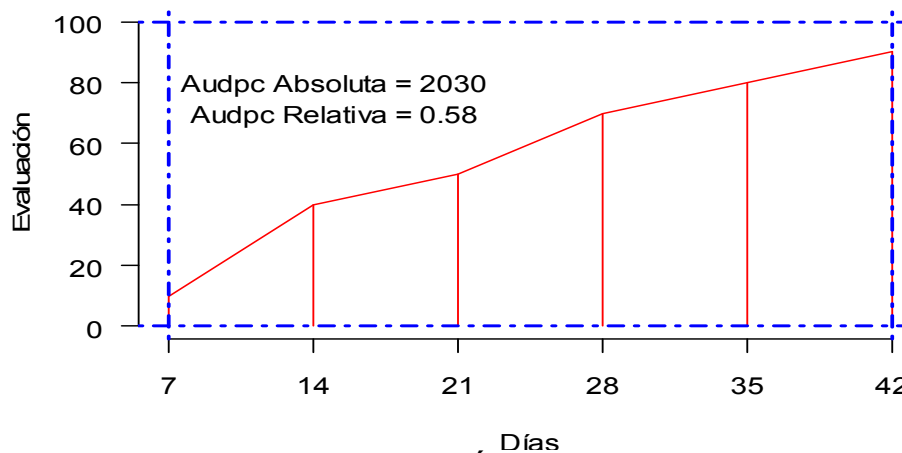


Figura 6.7. AUDPC: Área bajo la curva

## NO ADITIVIDAD

La prueba de no aditividad de Tukey en un modelo se utiliza cuando se tiene dudas sobre la veracidad de aditividad del mismo. Esta prueba verifica tal supuesto y se espera aceptar la hipótesis de nulidad del efecto no aditivo del modelo.

La prueba requiere todos los datos experimentales utilizados en la estimación del modelo aditivo lineal.

La función `nonadditivity()` de "agricolae" realiza esta prueba. Para su demostración, se utilizará los datos experimentales "potato" del paquete "agricolae". En este caso, el modelo corresponde al diseño de bloques completos al azar, donde los tratamientos son las variedades.

```
data(potato)
potato[,1]<-as.factor(potato[,1])
model<-lm(cutting ~ date + variety,potato)
df<-df.residual(model)
```



```
MSerror<-deviance(model)/df
attach(potato)
analysis<-nonadditivity(cutting, date, variety, df, MSerror)
detach(potato)
```

```
Tukey's test of non-additivity
cutting
```

```
P : 15.37166
Q : 77.4444
```

```
Analysis of Variance Table
```

```
Response: residual
              Df Sum Sq Mean Sq F value Pr(>F)
Non-additivity 1   3.051    3.051    0.922 0.3532
Residuals     14  46.330    3.309
```

Según los resultados, el modelo es aditivo porque el p.valor 0.35 es mayor de 0.05.

## LATEBLIGHT

LATEBLIGHT es un modelo matemático que simula el efecto de tiempo, el crecimiento de acogida y la resistencia, y el uso de fungicidas en el desarrollo asexual y el crecimiento de *Phytophthora infestans* en el follaje de la papa.

Versión TIZÓN TARDÍO LB2004 fue creado en octubre de 2004 (Andrade-Piedra et al., 2005a, byc), con sede en la C-versión escrita por BE Ticknor ("APUESTA 21.191 modificación de cbm8d29.c '), informó Doster et al. (1990) y descrito en detalle por Fry et al. (1991) (Esta versión se conoce como LB1990 por Andrade-Piedra et al. [2005a]). La primera versión de LATEBLIGHT fue desarrollado por Bruhn y Fry (1981) y se describe en detalle por Bruhn et al. (1980).

```
library(agricolae)
f <- system.file("external/weather.csv", package="agricolae")
weather <- read.csv(f,header=FALSE)
f <- system.file("external/severity.csv", package="agricolae")
severity <- read.csv(f)
weather[,1]<-as.Date(weather[,1],format = "%m/%d/%Y")
# Parameters dates
dates<-c("2000-03-25","2000-04-09","2000-04-12","2000-04-16","2000-
04-22")
dates<-as.Date(dates)
EmergDate <- as.Date('2000/01/19')
EndEpidDate <- as.Date("2000-04-22")
dates<-as.Date(dates)
NoReadingsH<- 1
RHthreshold <- 90
WS<-weatherSeverity(weather,severity,dates,EmergDate,EndEpidDate,
NoReadingsH,RHthreshold)
# Parameters Lateblight
InocDate<-"2000-03-18"
LGR <- 0.00410
IniSpor <- 0
SR <- 292000000
IE <- 1.0
LP <- 2.82
```

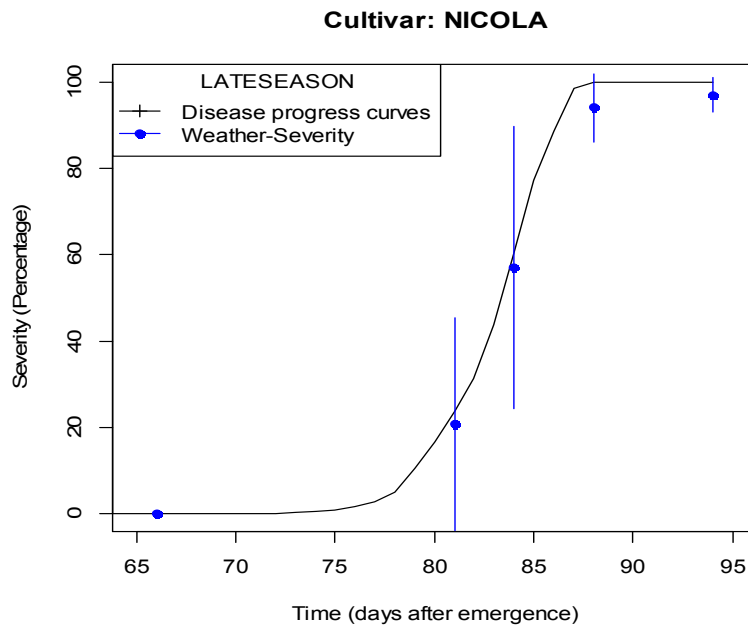
```

InMicCol <- 9
Cultivar <- 'NICOLA'
ApplSys <- "NOFUNGICIDE"
main<-"Cultivar: NICOLA"
#-----
model<-lateblight(WS, Cultivar,ApplSys, InocDate, LGR,IniSpor,SR,IE,
LP,MatTime='LATESEASON',InMicCol,main=main,type="l",xlim=c(65,95),lwd=1.5,xlab="Time (days after emergence)", ylab="Severity (Percentage)")

> head(model$Gfile)

```

	dates	nday	MeanSeverity	StDevSeverity	MinObs	MaxObs
Eval1	2000-03-25	66	0.1	0.000000	0.100000	0.10000
Eval2	2000-04-09	81	20.8	24.722459	-3.922459	45.52246
Eval3	2000-04-12	84	57.0	32.710854	24.289146	89.71085
Eval4	2000-04-16	88	94.0	7.968689	86.031311	101.96869
Eval5	2000-04-22	94	97.0	4.000000	93.000000	101.00000



**Figure 6.8. lateblight: LATESEASON**

```

# reproduce graph

x<- model$Ofile$nday
y<- model$Ofile$SimSeverity
w<- model$Gfile$nday
z<- model$Gfile$MeanSeverity
Min<-model$Gfile$MinObs
Max<-model$Gfile$MaxObs
plot(x,y,type="l",xlim=c(65,95),lwd=1.5,xlab="Time (days after emergence)",
ylab="Severity (Percentage)")
points(w,z,col="red",cex=1,pch=19)
npoints <- length(w)
for ( i in 1:npoints){

```

```
segments(w[i],Min[i],w[i],Max[i],lwd=1.5,col="red")
}
legend("topleft",c("Disease progress curves","Weather-Severity"),
title="Description",lty=1,pch=c(3,19),col=c("black","red"))
```

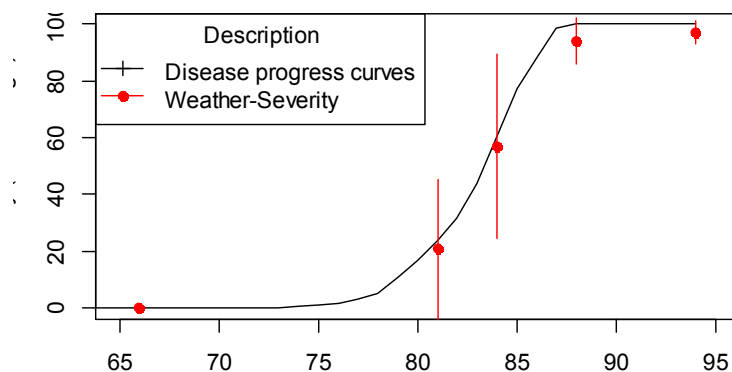


Figure 6.9. lateblight: LATESEASON

Tabla 6.10. Referencia de códigos ASCII para el uso de símbolos

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

## REFERENCIAS BIBLIOGRÁFICAS

1. Cochran and Cox. (1992). Experimental Design. Second edition. Wiley Classics Library Edition published. John Wiley & Sons, INC.
2. Conover, W.J. (1999). Practical Nonparametrics Statistics. John Wiley & Sons, INC, New York.
3. De Mendiburu, Felipe (2009). Una herramienta de análisis estadístico para la investigación agrícola. Tesis. Universidad Nacional de Ingeniería (UNI – LIMA - PERU).
4. Joshi, D.D. (1987). Linear Estimation and Design of Experiments. WILEY EASTERN LIMITED, New Delhi, India.
5. Kang, M. S. (1993). Simultaneous Selection for Yield and Stability: Consequences for Growers. Agron. J. 85:754-757.
6. Kuehl, Robert (2000). Design of Experiments. 2nd ed., Duxbury.
7. LeClerc, Erwin (1962). Field Plot Technique, Burgess Publishing Company.
8. Montgomery (2002). Diseño y Análisis de Experimentos (2ª Ed) WILEY.
9. Patterson, H.D. and Williams, E.R. Biometrika (1976). A New Class of Resolvable Incomplete Block Designs. Printed in Great Britain.
10. R Core Team (2012). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL <http://www.R-project.org>.
11. Steel & Torry & Dickey (1997). Principles and Procedures of Statistics. A Biometrical Approach. Third Edition.