Tutorial de agricolae(Versión 1.2-8)

Felipe de Mendiburu*

2017-09-14

Índice

Prefacio **4**

1. Introducción **4**
1.1. Instalación . . . 4
1.2. Uso en **R** . . . 5
1.3. Base de datos de **agricolae** . . . 5

2. Estadística descriptiva **6**
2.1. Histograma . . . 6
2.2. Estadísticas y tabla de frecuencia . . . 8
2.3. La reproducción del histograma y compatibilidad con la función hist() . . . 10
2.4. Histograma basado en datos agrupados . . . 11
2.5. Juntando clases . . . 12

3. Diseño de experimentos **13**
3.1. Diseño completamente aleatorizado . . . 14
3.2. Diseño de bloques completos al azar . . . 15
3.3. Diseño cuadrado latino . . . 15
3.4. Diseño Greco-latino . . . 16
3.5. Youden design . . . 16
3.6. Diseño de bloque incompleto balanceado . . . 18
3.7. Diseño Ciclico . . . 19
3.8. Diseño Latice . . . 21
3.9. Diseño Alfa . . . 23
3.10. Diseño de bloques Aumentados . . . 24
3.11. Diseño de parcelas divididas . . . 26
3.12. Diseño bloques divididos o franjas . . . 28
3.13. Factorial . . . 30

*Profesor Principal del Departamento Academico de Estadística e Informática de la Facultad de Economía y Planificación. Universidad Nacional Agraria La Molina-PERU

4. Comparación múltiple **32**
4.1. La diferencia mínima significativa (DLS) . . . 33
4.2. holm, hommel, hochberg, bonferroni, BH, BY, fdr . . . 34
4.3. La Nueva prueba múltiple de Duncan . . . 35
4.4. Student-Newman-Keuls . . . 36
4.5. Ryan, Einot, Gabriel y Welsch . . . 37
4.6. Procedimiento de Tukey (HSD) . . . 38
4.7. Waller-Duncan prueba Bayesiana . . . 39
4.8. La prueba de Scheffe . . . 41
4.9. Comparación múltiple de tratamientos en factorial . . . 42
4.10. Análisis de los bloques incompletos balanceados . . . 44
4.11. Bloques incompletos parcialmente balanceados . . . 48
4.12. Bloques Aumentados . . . 56

5. Comparaciones no paramétricas **60**
5.1. Kruskal-Wallis . . . 61
5.2. Friedman . . . 62
5.3. Waerden . . . 63
5.4. Mediana . . . 65
5.5. Durbin . . . 66

6. Gráficos de comparación multiple **69**
6.1. bar.group . . . 69
6.2. bar.err . . . 69
6.3. plot.group . . . 69
6.4. diffograph . . . 70

7. Análisis de Estabilidad **71**
7.1. Estabilidad paramétrico . . . 72
7.2. Estabilidad no paramétrico . . . 74
7.3. AMMI . . . 75
7.4. indice de estabilidad de AMMI y rendimiento . . . 76

8. Funciones especiales **78**
8.1. Consenso de dendrograma . . . 79
8.2. Montecarlo . . . 81
8.3. Re-muestreo en el modelo lineal . . . 82
8.4. Simulación en el modelo lineal . . . 83
8.5. Análisis Path . . . 84
8.6. Linea X Probador . . . 84
8.7. La uniformidad del suelo . . . 87
8.8. Límites de confianza de los índices de biodiversidad . . . 88
8.9. Correlación . . . 89
8.10. Otras funciones . . . 90

Código para caracteres ASCII **97**

Bibliografía **97**

Índice de figuras

1. Frecuencia absoluta y relativa con polígono. 7
2. Densidad de frecuencia. 8
3. Frecuencia absoluta y porcentaje con 5 clases 9
4. Polígono Comparación con densidad 10
5. Ojiva y 5 clases 11
6. histograma con clases definida 12
7. Clases nuevas para el histograma 13
8. combinado clone:nitrogen 45
9. Tratamientos agrupados 51
10. Rango en cada tratamiento 54
11. Grupos de Tratamiento 56
12. Comparación según Kruskal-Wallis 62
13. Agrupación de tratamientos y su variación, Método mediana 67
14. Comparación entre los tratamientos 70
15. Agrupamiento de tratamientos y su variación, Método de Duncan 71
16. Mean-Mean de la comparacion entre tratamientos 71
17. Biplot y Triplot 76
18. Influencia del genotipo 77
19. Dendrograma, la producción por consenso 79
20. Dendrograma, producción por hcut() 80
21. dendrograma Clásico 80
22. dendrograma Clasico 81
23. Distribución de la simulación y los datos originales 82
24. Histograma de los datos originales y con datos simulados 83
25. Curva de ajuste para el tamaño óptimo de parcela 87
26. Función de Waller para diferentes valores de los parámetros K y Fc 92
27. AUDPC: Área bajo la curva 94
28. Area under the curve (AUDPC) and Area under the Stairs (AUDPS) 95
29. lateblight: LATESEASON 97

Prefacio

El siguiente documento fue desarrollado para facilitar el uso del paquete agricolae en R, se entiende que el usuario conoce la metodología estadística para el diseño y análisis de experimentos y mediante el uso de las funciones programadas en agricolae facilita la generación de la libro de campo según el diseño experimental y su análisis. En la primera parte se describe el uso de la función *graph.freq* que es complementaria a la función *hist* para hallar la tabla de frecuencias, histograma y gráficos como el polígono de frecuencias y ojiva; la segunda parte corresponde al desarrollo del diseño experimental y la numeración de las unidades como se utiliza en experimentos agrícolas y una tercera parte corresponde a las pruebas comparativas entre tratamientos, estabilidad de cultivares, estudio de homogeneidad del suelo, diseños geneticos, simulación del tizón tardío en el cultivo de papa y otros.

1. Introducción

El paquete agricolae ofrece una amplia funcionalidad en el diseño de experimentos, especialmente en experimento agricola para la mejora de las plantas, las cuales también pueden ser utilizadas para otros fines. Contiene las siguientes opciones: latice, alfa, diseño de bloques incompletos balanceados, cíclicos, bloques completos al azar, cuadrado latino, greco latino, diseño de bloques aumentados, parcelas divididas, bloques divididos. También cuenta con varios procedimientos de análisis de datos experimentales, tales como las comparaciones de tratamientos de Waller-Duncan, Bonferroni, Duncan, Student-Newman-Keuls, Ryan-Einot-Gabriel-Welsch (REGW) Scheffe, o la diferencia minima de significación (DLS) clásico y de Tukey; asi también las comparaciones no paramétricas como: Kruskal-Wallis, Friedman, Durbin, Waerden y la prueba de la Mediana; análisis de estabilidad, y otros procedimientos aplicados en la genética, así como los procedimientos de la biodiversidad y la estadística descriptiva, De Mendiburu (2009)
Para más detalles sobre el uso de agricolae , consulte el manual de referencia.

1.1. Instalación

El programa principal de **R** debe ser instalado en la plataforma de su computador *(Windows, Linux o MAC)*. Si no esta instalado, Usted puede bajar desde el proyecto R *(www.r-project.org)* de un repositorio del CRAN, R Core Team (2017)

`>install.packages(.ªgricolae")` Una vez instalado el paquete `agricolae`, este debe ser accesible en una sesión de **R**

```
> library(agricolae)
```

Para facilidades de ayuda o detalles de una orden en particular (tal como la funcion `waller.test`) Usted puede escribir:

```
> help(package="agricolae")
> help(waller.test)
```

Para una completa funcionalidad de **agricolae** requiere de otros paquetes.

MASS: para la inversa generalizada utilizada en la función *PBIB.test*
nlme: para los metodos REML y LM en *PBIB.test*
klaR: para la función *triplot* usada en la función *AMMI*
akima: para la función *interpp* usada en *grid3p* para interpolación
Cluster: para la función *consensus*
spdep: para la relación entre los genotipos en biplot de la función *AMMI*
algDesign: para el diseño de bloques incompletos*design.bib*

1.2. Uso en R

Desde que **agricolae** es un paquete de funciones, éstos son operativos cuando son llamados directamente desde la consola de **R** y se integran a todas las funciones de base de **R** .

Las siguientes órdenes son frecuentes:

```
> detach(package:agricolae) # retirar el paquete agricole de la memoria
> library(agricolae) # cargar el paquete agricole a la memoria
> designs<-apropos("design")
> print(designs[substr(designs,1,6)=="design"], row.names=FALSE)

 [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"
[13] "design.youden"
```

Para el uso de símbolos que no aparecen en el teclado en español, tales como:

```
~, [, ], &, ^, |. <, >, {, }, \%
```

u otros, usar la tabla de códigos ASCII.

Con el fin de continuar con la línea de ordenes, no se olvide de cerrar las ventanas abiertas con cualquier orden R.

Para obtener ayudas:

```
help(graph.freq)
? (graph.freq)
str(normal.freq)
example(join.freq)
```

1.3. Base de datos de agricolae

```
> A<-as.data.frame(data(package="agricolae")$results[,3:4])
> A[,2]<-paste(substr(A[,2],1,35),"..",sep=".")
> head(A)
```

```
            Item                                    Title
1            CIC    Data for late blight of potatoes...
2        Chz2006         Data amendment Carhuaz 2006...
3  ComasOxapampa         Data AUDPC Comas - Oxapampa...
4             DC Data for the analysis of carolina g...
5 Glycoalkaloids                  Data Glycoalkaloids...
6        Hco2006         Data amendment Huanuco 2006...
```

2. Estadística descriptiva

El paquete *agricolae* proporciona algunas funciones complementarias para el programa R, específicamente para la gestión del histograma.

2.1. Histograma

El histograma se construye con la función graph.freq() y se asocia a otras funciones: polygon.freq, table.freq, stat.freq, intervals.freq, sturges.freq, join.freq, ogive.freq y normal.frec, véase las Figuras 1, 2, 3, 4 para mas detalles.

```
> apropos("freq")
```

```
 [1] "frequency"          "graph.freq"         "intervals.freq"
 [4] "join.freq"          "normal.freq"        "ogive.freq"
 [7] "plot.graph.freq"    "polygon.freq"       "stat.freq"
[10] "sturges.freq"       "summary.graph.freq" "table.freq"
```

Cargar el paquete *agricolae*:

```
> #data(package="agricolae") # Lista la base de datos
> data(sweetpotato) # Carga datos de camote:
> head(sweetpotato) # Lista datos:
```

```
  virus yield
1    cc  28.5
2    cc  21.7
3    cc  23.0
4    fc  14.9
5    fc  10.6
6    fc  13.1
```

```
> str(sweetpotato) # Ver su estructura:
```

```
'data.frame':        12 obs. of  2 variables:
 $ virus: Factor w/ 4 levels "cc","fc","ff",..: 1 1 1 2 2 2 3 3 3 4 ...
 $ yield: num  28.5 21.7 23 14.9 10.6 13.1 41.8 39.2 28 38.2 ...
```

```
> # Editar su contenido: fix(sweetpotato)
```

```
> par(mfrow=c(1,2),mar=c(4,4,0,1),cex=0.6)
> h1<- graph.freq(peso,col=colors()[84],frequency=1,las=2,density=20,ylim=c(0,12),ylab
> x<-h1$breaks
> h2<- plot(h1, frequency =2, axes= FALSE,ylim=c(0,0.4),xlab="peso",ylab="Relativa (%)
> polygon.freq(h2, col=colors()[84], lwd=2, frequency =2)
> axis(1,x,cex=0.6,las=2)
> y<-seq(0,0.4,0.1)
> axis(2, y,y*100,cex=0.6,las=1)
```

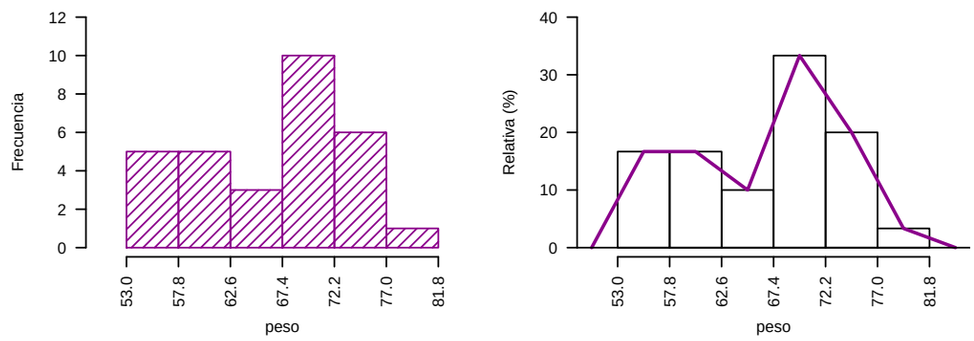

Figura 1: Frecuencia absoluta y relativa con polígono.

Ejemplo. Los datos generados en R. (peso de los estudiantes).

```
> peso<-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)
> print(summary(peso))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  53.00   59.88   68.00   66.45   71.50   81.50


> print(h1)

$breaks
[1] 53.0 57.8 62.6 67.4 72.2 77.0 81.8

$counts
[1]  5  5  3 10  6  1

$mids
[1] 55.4 60.2 65.0 69.8 74.6 79.4

$relative
[1] 0.1667 0.1667 0.1000 0.3333 0.2000 0.0333

$density
[1] 0.03472917 0.03472917 0.02083333 0.06943750 0.04166667 0.00693750
```

```
> par(mfrow=c(1,2),mar=c(4,3,1,1),cex=0.6)
> h3<- graph.freq (peso, col="brown", frequency =3,las=2)
> h4<- graph.freq(peso, col="blue", frequency =3)
> normal.freq(h4, col="red", lty=4,lwd=2, frequency=3,las=2)
```

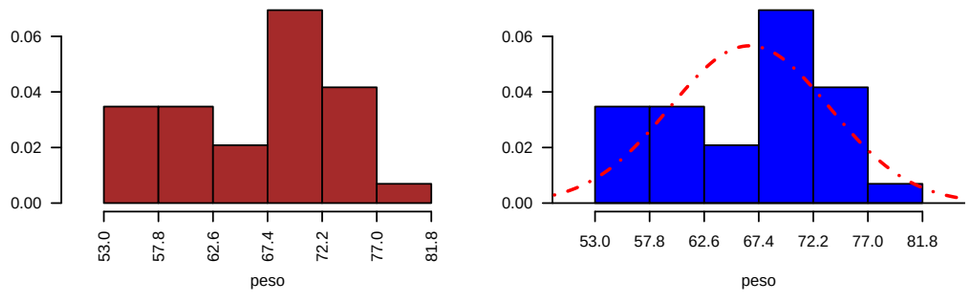

Figura 2: Densidad de frecuencia.

```
attr(,"class")
[1] "graph.freq" "histogram"
```

```
> print(summary(h1))
```

```
  Lower Upper Main Frequency Percentage CF   CPF
1  53.0  57.8 55.4         5       16.7  5  16.7
2  57.8  62.6 60.2         5       16.7 10  33.3
3  62.6  67.4 65.0         3       10.0 13  43.3
4  67.4  72.2 69.8        10       33.3 23  76.7
5  72.2  77.0 74.6         6       20.0 29  96.7
6  77.0  81.8 79.4         1        3.3 30 100.0
```

2.2. Estadísticas y tabla de frecuencia

Usar table.freq() equivalente con summary() y stat.freq().

Limites de clase: **Lower and Upper**
Marca de clase: **Main**
Frequencia: freq
frecuencia relativa porcentual: **percentage**
Frecuencia acumulativa: **CF**
Frecuencia acumulativa relativa porcentual: **CPF**

por ejemplo:

```
 Lower  Upper   Main freq relative CF  RCF
-1.910 -1.334 -1.622    2     0.04  2 0.04
-1.334 -0.758 -1.046    7     0.14  9 0.18
-0.758 -0.182 -0.470   12     0.24 21 0.42
-0.182  0.394  0.106   10     0.20 31 0.62
```

```
> par(mfrow=c(1,2),mar=c(4,4,1,1),cex=0.6)
> h7<- graph.freq(peso, nclass=5,frequency =1,xlab="h7")
> h8<- graph.freq(peso, nclass=5, frequency=2,axes=FALSE,xlab="h8")
> title(ylab="%")
> normal.freq(h8,col="red",frequency=2)
> axis(1); axis(2,seq(0,1,0.1),100*seq(0,1,0.1),las=2)
```

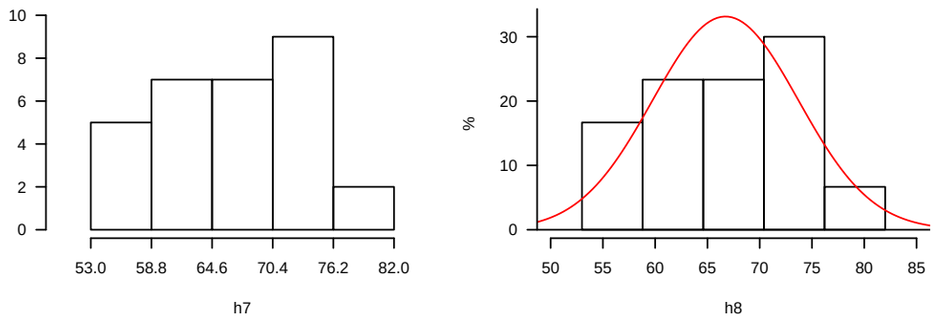

Figura 3: Frecuencia absoluta y porcentaje con 5 clases

```
 0.394  0.970  0.682   10     0.20 41 0.82
 0.970  1.546  1.258    5     0.10 46 0.92
 1.546  2.122  1.834    4     0.08 50 1.00

$variance
  0.8395964
$mean
  0.106
$median
  0.0484
$mode : mode and interval mode
     [-     -]       mode
  -0.758 -0.182 -0.3465714


> round(table.freq(h7), 2)

  Lower Upper Main Frequency Percentage CF   CPF
1  53.0  58.8 55.9         5       16.7  5  16.7
2  58.8  64.6 61.7         7       23.3 12  40.0
3  64.6  70.4 67.5         7       23.3 19  63.3
4  70.4  76.2 73.3         9       30.0 28  93.3
5  76.2  82.0 79.1         2        6.7 30 100.0
```

Estadísticas

Redondeado a dos decimales:

```
> stat.freq(h7)
```

```
> par(mfrow=c(1,2),mar=c(4,3,1,1),cex=0.6)
> wd<-density(peso)
> h9<- graph.freq(peso, density=6, col="blue", frequency =3,xlab="h9")
> lines(wd,col="brown",lwd=2)
> h10<- graph.freq(peso, border=0, frequency =3,xlab="h10")
> polygon.freq(h10,col="blue", frequency =3)
> lines(wd,col="brown",lwd=2)
```

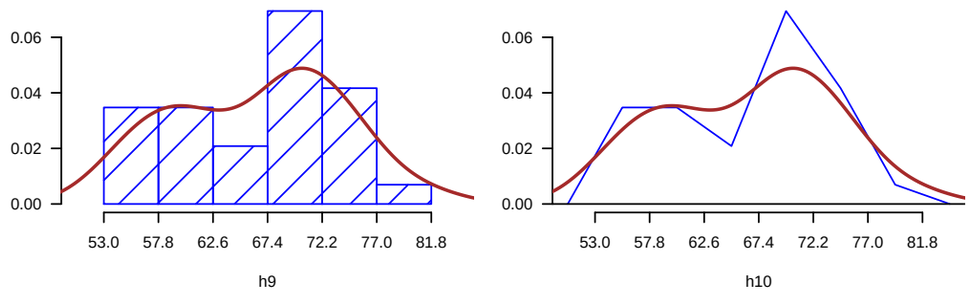

Figura 4: Polígono Comparación con densidad

```
$variance
[1] 50.42133

$mean
[1] 66.72667

$median
[1] 67.08571

$mode
       [-   -]     mode
[1,] 70.4 76.2 71.68889
```

2.3. La reproducción del histograma y compatibilidad con la función hist()

El objeto de la función graph.freq() define la clase graph.freq, Ver Figuras 5, 6, 7. La reproducción del histograma h8 (5 clases)

```
> round(summary(h8),2)
```

	Lower	Upper	Main	Frequency	Percentage	CF	CPF
1	53.0	58.8	55.9	5	16.7	5	16.7
2	58.8	64.6	61.7	7	23.3	12	40.0
3	64.6	70.4	67.5	7	23.3	19	63.3
4	70.4	76.2	73.3	9	30.0	28	93.3
5	76.2	82.0	79.1	2	6.7	30	100.0

```
> par(mfrow=c(1,2),mar=c(4,3,1,1),cex=0.7)
> h11<-ogive.freq(h7, type="b",  col="red ",xlab="h11")
> h12<-plot(h8, xlab="PESO (h12)",ylim=c(0,15))
> normal.freq(h12,col="red")
```

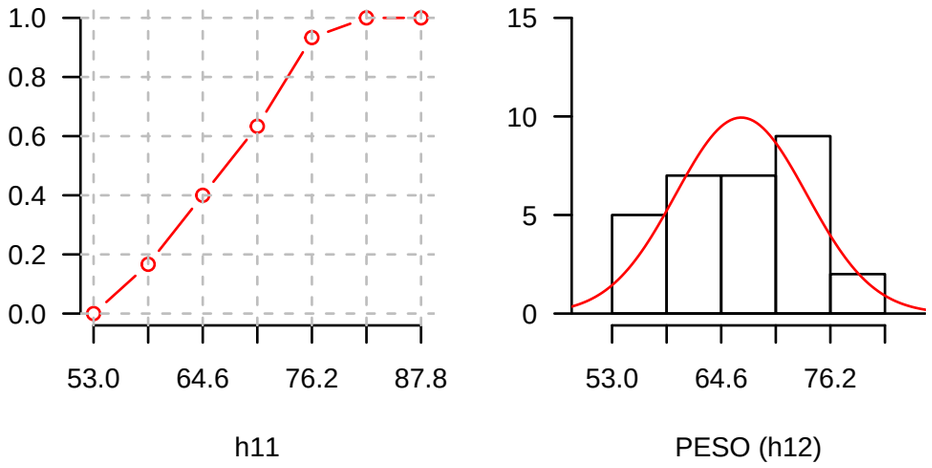

Figura 5: Ojiva y 5 clases

Las clases de las funciones hist() y graph.freq() son 'histogram'y 'graph.freq', respectivamente. Sin embargo, es posible establecer la compatibilidad entre ambas funciones.

Con el fin de mostrar las frecuencias relativas, puede utilizar graph.freq() con el objeto creado por hist() sin modificar las clases.

Las funciones de **agricolae** para la gestión de histogramas funcionan correctamente con los objetos creados por la función hist() de R.

then:
h8<-graph.freq(...)
h12<-plot(h8)
normal.freq(h12)
h8<-hist(...)
table.freq(h8)
h12<-graph.freq(h8)
summary(h12)

2.4. Histograma basado en datos agrupados

Si hay datos agrupados,se puede gráficar y obtener los resúmenes del histograma con la función graph.freq(), asi por ejemplo, en la siguiente tabla:

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

```
> par(mar=c(4,3,2,2),cex=0.6)
> classes <- c(0, 10, 20, 30, 40, 50)
> freq <- c(3, 8, 15, 18, 6)
> h13 <- graph.freq(classes,counts=freq, xlab="Classes (h13)")
```

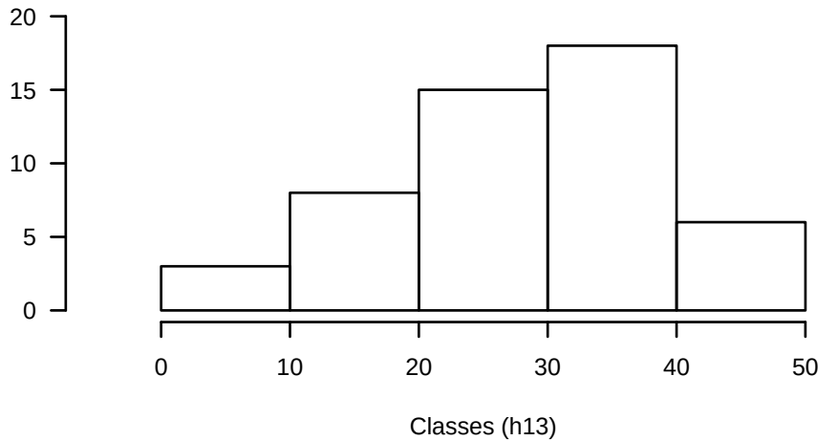

Figura 6: histograma con clases definida

En R tenemos:

```
> print(summary(h13))
```

	Lower	Upper	Main	Frequency	Percentage	CF	CPF
1	0	10	5	3	6	3	6
2	10	20	15	8	16	11	22
3	20	30	25	15	30	26	52
4	30	40	35	18	36	44	88
5	40	50	45	6	12	50	100

2.5. Juntando clases

Conociendo el intervalo original del objeto peso de estudiantes, estos pueden ser juntados, por ejemplo:

```
> intervals.freq(h13)
```

```
     lower upper
[1,]     0    10
[2,]    10    20
[3,]    20    30
[4,]    30    40
[5,]    40    50
```

```
> h14<- join.freq(h13,1:2)
> intervals.freq(h14)
```

```
> par(mfrow=c(1,2),mar=c(4,3,2,2),cex=0.6)
> plot(h13, xlab="Original Class (h13)", main="target")
> plot(h14, xlab="Changes in class (h14)", main="join.freq")
```

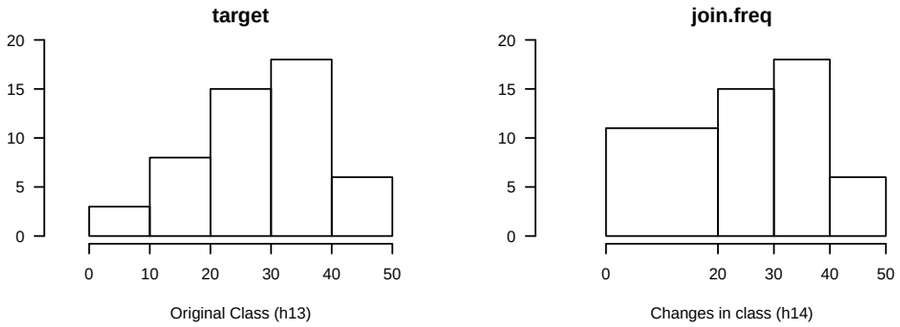

Figura 7: Clases nuevas para el histograma

```
     lower upper
[1,]     0    20
[2,]    20    30
[3,]    30    40
[4,]    40    50
```

3. Diseño de experimentos

agricolae presenta funciones especiales para la creación del libro de campo para diseños experimentales. Debido a la generación aleatoria, este paquete se utiliza bastante en la investigación agrícola.

Para esta generación, se requieren ciertos parámetros, como por ejemplo el nombre de cada tratamiento, el número de repeticiones, y otros, de acuerdo a la referencia de diseño Cochran and Cox (1957); kueh (2000); LeClerg (1992); Montgomery (2002). Hay otros parámetros de generación aleatoria, como la semilla para reproducir la misma generación al azar o el método de generación (Consulte el manual de referencia de agricolae.

http://cran.at.r-project.org/web/packages/agricolae/agricolae.pdf

Los parámetros importantes en la generación de diseño:

Series: Una constante que se utiliza para establecer los bloques de etiquetas numéricos, por ejemplo, número = 2, las etiquetas será: 101, 102, para la primera fila o bloque, 201, 202, para la siguiente, en el caso de diseño completamente al azar, la numeración es secuencial.

design: Algunas de las características del diseño solicitado agricolae aplicarse específicamente a design.ab (factorial) o design.split (parcelas divididas) y sus valores posibles son: RCBD, crd y lsd.

seed: La semilla para la generación aleatoria y su valor es cualquier valor real, si el valor es cero, no tiene generación reproducible, en este caso copia del valor de los parámetros outdesign$parameters.

Kinds: el método de generación aleatoria, de forma predeterminada $Super - Duper$.

first: Para algunos diseños no se requiere al azar de la primera repetición, especialmente en el diseño de bloques, si usted quiere cambiar al azar, cambiar a TRUE.

randomization: TRUE o FALSE. Si es FALSE, la randomización no se realiza.

Output design:

parameters: la entrada a la generación de diseño, incluyen la semilla de generación aleatoria, si la semilla, seed = 0, el programa genera un valor y es posible reproducir el diseño.

book: Libro de campo

statistics: las estadísticas de información del diseño para el índice de eficiencia de ejemplo, el número de tratamientos.

sketch: distribución de los tratamientos en el campo.

La enumeración de las parcelas

zigzag es una función que le permite colocar la numeración de las parcelas en la dirección de la serpentina: El zigzag es la salida generada por un diseño bloques, cuadrado latino, greco, parcelas divididas, tiras parcela, en bloques de bloques incompletos factorial, equilibrado, celosía cíclico, alfa y bloques aumentados.

fieldbook: La salida zigzag contiene el libro de campo.

3.1. Diseño completamente aleatorizado

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

```
> str(design.crd)

function (trt, r, serie = 2, seed = 0, kinds = "Super-Duper", randomization = TRUE)

> trt <- c("A", "B", "C")
> repeticion <- c(4, 3, 4)
> outdesign <- design.crd(trt,r=repeticion,seed=777,serie=0)
> book1 <- outdesign$book
> print(book1)

   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
```

Excel:write.csv(plan1,**"plan1.csv"**,row.names=FALSE)

3.2. Diseño de bloques completos al azar

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

```
> str(design.rcbd)

function (trt, r, serie = 2, seed = 0, kinds = "Super-Duper", first = TRUE,
    continue = FALSE, randomization = TRUE)

> trt <- c("A", "B", "C","D","E")
> repeticion <- 4
> outdesign <- design.rcbd(trt,r=repeticion, seed=-513, serie=2)
> # book2 <- outdesign$book
> book2<- zigzag(outdesign) # zigzag numeracion
> print(outdesign$sketch)

     [,1] [,2] [,3] [,4] [,5]
[1,] "D"  "B"  "C"  "E"  "A"
[2,] "E"  "A"  "D"  "B"  "C"
[3,] "E"  "D"  "B"  "A"  "C"
[4,] "A"  "E"  "C"  "B"  "D"

> print(matrix(book2[,1],byrow = TRUE, ncol = 5))

     [,1] [,2] [,3] [,4] [,5]
[1,]  101  102  103  104  105
[2,]  205  204  203  202  201
[3,]  301  302  303  304  305
[4,]  405  404  403  402  401
```

3.3. Diseño cuadrado latino

Requieren los nombres de los tratamientos y sus parámetros son:

```
> str(design.lsd)

function (trt, serie = 2, seed = 0, kinds = "Super-Duper", first = TRUE,
    randomization = TRUE)

> trt <- c("A", "B", "C", "D")
> outdesign <- design.lsd(trt, seed=543, serie=2)
> print(outdesign$sketch)

     [,1] [,2] [,3] [,4]
[1,] "C"  "A"  "B"  "D"
[2,] "D"  "B"  "C"  "A"
[3,] "B"  "D"  "A"  "C"
[4,] "A"  "C"  "D"  "B"
```

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> print(t(matrix(book[,1],c(4,4))),digit=0)

     [,1] [,2] [,3] [,4]
[1,]  101  102  103  104
[2,]  204  203  202  201
[3,]  301  302  303  304
[4,]  404  403  402  401
```

3.4. Diseño Greco-latino

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

```
> str(design.graeco)

function (trt1, trt2, serie = 2, seed = 0, kinds = "Super-Duper", randomization = TRUE)

> trt1 <- c("A", "B", "C", "D")
> trt2 <- 1:4
> outdesign <- design.graeco(trt1,trt2, seed=543, serie=2)
> print(outdesign$sketch)

     [,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"
```

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> print(t(matrix(book[,1],c(4,4))),digit=0)

     [,1] [,2] [,3] [,4]
[1,]  101  102  103  104
[2,]  204  203  202  201
[3,]  301  302  303  304
[4,]  404  403  402  401
```

3.5. Youden design

Reuiere los nombres de tratamientos de cada factor de estudio, corresponde a un cuadrado latino rectangular y sus argumentos son:

```
> str(design.youden)
```

```
function (trt, r, serie = 2, seed = 0, kinds = "Super-Duper", first = TRUE,
    randomization = TRUE)
```

```
> varieties<-c("perricholi","yungay","maria bonita","tomasa")
> r<-3
> outdesign <-design.youden(varieties,r,serie=2,seed=23)
> print(outdesign$sketch)
```

```
     [,1]           [,2]           [,3]
[1,] "maria bonita" "perricholi"   "tomasa"
[2,] "yungay"       "tomasa"       "maria bonita"
[3,] "tomasa"       "yungay"       "perricholi"
[4,] "perricholi"   "maria bonita" "yungay"
```

```
> book <- outdesign$book
> print(book) # field book.
```

```
   plots row col    varieties
1    101   1   1 maria bonita
2    102   1   2   perricholi
3    103   1   3       tomasa
4    201   2   1       yungay
5    202   2   2       tomasa
6    203   2   3 maria bonita
7    301   3   1       tomasa
8    302   3   2       yungay
9    303   3   3   perricholi
10   401   4   1   perricholi
11   402   4   2 maria bonita
12   403   4   3       yungay
```

```
> print(matrix(as.numeric(book[,1]),byrow = TRUE, ncol = r))
```

```
     [,1] [,2] [,3]
[1,]  101  102  103
[2,]  201  202  203
[3,]  301  302  303
[4,]  401  402  403
```

Serpentine enumeration:

```
> book <- zigzag(outdesign)
> print(matrix(as.numeric(book[,1]),byrow = TRUE, ncol = r))
```

```
     [,1] [,2] [,3]
[1,]  101  102  103
[2,]  203  202  201
[3,]  301  302  303
[4,]  403  402  401
```

3.6. Diseño de bloque incompleto balanceado

Se requieren los nombres de los tratamientos y el tamaño del bloque y sus parámetros son:

```
> str(design.bib)

function (trt, k, r = NULL, serie = 2, seed = 0, kinds = "Super-Duper",
    maxRep = 20, randomization = TRUE)

> trt <- c("A", "B", "C", "D", "E" )
> k <- 4
> outdesign <- 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 >>>

> book5 <- outdesign$book
> outdesign$statistics

       lambda treatmeans blockSize blocks r Efficiency
values      3          5         4      5 4     0.9375

> outdesign$parameters

$design
[1] "bib"

$trt
[1] "A" "B" "C" "D" "E"

$k
[1] 4

$serie
[1] 2

$seed
[1] 543

$kinds
[1] "Super-Duper"
```

De acuerdo con la información que se produce, que son cinco bloques de tamaño 4, siendo la matriz:

```
> t(matrix(book5[,3],c(4,5)))
```

```
     [,1] [,2] [,3] [,4]
[1,] "D"  "B"  "E"  "C"
[2,] "B"  "A"  "C"  "D"
[3,] "D"  "B"  "E"  "A"
[4,] "E"  "A"  "C"  "D"
[5,] "B"  "C"  "E"  "A"
```

Ser observa que los tratamientos tienen cuatro repeticiones. el parametro lambda tiene tres repeticiones, lo que significa que un par de tratamientos están juntos en tres ocasiones. Por ejemplo, B y E se encuentran en los bloques I, III y V.

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> t(matrix(book[,1],c(4,5)))
```

```
     [,1] [,2] [,3] [,4]
[1,]  101  102  103  104
[2,]  204  203  202  201
[3,]  301  302  303  304
[4,]  404  403  402  401
[5,]  501  502  503  504
```

3.7. Diseño Ciclico

Se requieren los nombres de los tratamientos, el tamaño del bloque y el número de repeticiones. Este diseño se utiliza para 6 a 30 tratamientos. Las repeticiones son un múltiplo del tamaño del bloque; si son seis tratamientos y el tamaño es 3, entonces las repeticiones pueden ser 6, 9, 12, etc y sus parámetros son:

```
> str(design.cyclic)
```

```
function (trt, k, r, serie = 2, rowcol = FALSE, seed = 0, kinds = "Super-Duper",
    randomization = TRUE)
```

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

```
cyclic design
Generator block basic:
1 2 4
1 3 2
```

```
Parameters
==================
treatmeans : 6
Block size : 3
Replication: 6
```

```
> book6 <- outdesign$book
> outdesign$sketch[[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"
```

```
> outdesign$sketch[[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"
```

12 bloques de 4 tratamientos cada uno se han generado. **Serpentine enumeration:**

```
> book <- zigzag(outdesign)
> array(book$plots,c(3,6,2))->X
> t(X[,,1])
```

```
     [,1] [,2] [,3]
[1,]  101  102  103
[2,]  106  105  104
[3,]  107  108  109
[4,]  112  111  110
[5,]  113  114  115
[6,]  118  117  116
```

```
> t(X[,,2])
```

```
     [,1] [,2] [,3]
[1,]  201  202  203
[2,]  206  205  204
[3,]  207  208  209
[4,]  212  211  210
[5,]  213  214  215
[6,]  218  217  216
```

3.8. Diseño Latice

Ellos requieren una serie de tratamientos de un cuadrado perfecto; por ejemplo 9, 16, 25, 36, 49, etc y sus parámetros son:

```
> str(design.lattice)

function (trt, r = 3, serie = 2, seed = 0, kinds = "Super-Duper", randomization = TRUE)
```

Pueden generar un latice simple (2 rep.) O un latice triple (3 rep.) la generación de un diseño latice triple para 9 tratamientos 3*3 se obtiene con el siguiente procedimiento

```
> trt<-letters[1:9]
> outdesign <-design.lattice(trt, r = 3, serie = 2, seed = 33,
+     kinds =  "Super-Duper")

Lattice design,  triple   3 x 3

Efficiency factor
(E ) 0.7272727

<<< Book >>>

> book7 <- outdesign$book
> outdesign$parameters

$design
[1] "lattice"

$type
[1] "triple"

$trt
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i"

$r
[1] 3

$serie
[1] 2

$seed
[1] 33

$kinds
[1] "Super-Duper"

> outdesign$sketch
```

```
$rep1
     [,1] [,2] [,3]
[1,] "i"  "d"  "a"
[2,] "b"  "c"  "e"
[3,] "h"  "f"  "g"

$rep2
     [,1] [,2] [,3]
[1,] "c"  "f"  "d"
[2,] "b"  "h"  "i"
[3,] "e"  "g"  "a"

$rep3
     [,1] [,2] [,3]
[1,] "e"  "h"  "d"
[2,] "b"  "f"  "a"
[3,] "c"  "g"  "i"
```

```
> head(book7)
```

```
  plots r block trt
1   101 1     1   i
2   102 1     1   d
3   103 1     1   a
4   104 1     2   b
5   105 1     2   c
6   106 1     2   e
```

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> array(book$plots,c(3,3,3)) -> X
> t(X[,,1])
```

```
     [,1] [,2] [,3]
[1,]  101  102  103
[2,]  106  105  104
[3,]  107  108  109
```

```
> t(X[,,2])
```

```
     [,1] [,2] [,3]
[1,]  201  202  203
[2,]  206  205  204
[3,]  207  208  209
```

```
> t(X[,,3])
```

```
     [,1] [,2] [,3]
[1,]  301  302  303
[2,]  306  305  304
[3,]  307  308  309
```

3.9. Diseño Alfa

Estos diseños son generados por los criterios alfa referencia, Patterson and Williams (1976). Son similares a los diseños de latice, pero los grupos son rectangulares, con s bloques por k tratamientos por bloque. El número de tratamientos debe ser igual a s*k y el total de unidades experimentales es R*s*K y sus parámetros son:

```
> str(design.alpha)

function (trt, k, r, serie = 2, seed = 0, kinds = "Super-Duper", randomization = TRUE)

> trt <- letters[1:15]
> outdesign <- 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 >>>

> book8 <- outdesign$book
> outdesign$statistics

       treatments blocks Efficiency
values         15      5  0.6363636

> outdesign$sketch

$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"
```

```
> # codification of the plots
> A<-array(book8[,1], c(3,5,2))
> t(A[,,1])

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

> t(A[,,2])

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

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> A<-array(book[,1], c(3,5,2))
> t(A[,,1])

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

> t(A[,,2])

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

3.10. Diseño de bloques Aumentados

Estos son los diseños para dos tipos de tratamientos: los tratamientos de control (comunes) y el aumento de tratamientos. Los tratamientos comunes son aplicadas en bloques completos al azar, y el aumento de los tratamientos, al azar. Cada tratamiento se debe aplicar en cualquier bloque de una sola vez. Se entiende que los tratamientos comunes son de un mayor interés; el error estándar de la diferencia es mucho menor que cuando entre dos aumento de los de diferentes bloques. El design.dau función () alcanza este propósito y sus parámetros son:

```
> str(design.dau)

function (trt1, trt2, r, serie = 2, seed = 0, kinds = "Super-Duper", name = "trt",
    randomization = TRUE)

> rm(list=ls())
> trt1 <- c("A", "B", "C", "D")
> trt2 <- c("t","u","v","w","x","y","z")
> outdesign <- design.dau(trt1, trt2, r=5, seed=543, serie=2)
> book9 <- outdesign$book
> with(book9,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"
```

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> with(book,by(plots, block, as.character))

block: 1
[1] "101" "102" "103" "104" "105" "106"
------------------------------------------------------------
block: 2
[1] "206" "205" "204" "203" "202" "201"
------------------------------------------------------------
block: 3
[1] "301" "302" "303" "304" "305"
------------------------------------------------------------
block: 4
[1] "405" "404" "403" "402" "401"
------------------------------------------------------------
block: 5
[1] "501" "502" "503" "504" "505"

> print(book)
```

```
   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    206     2   D
8    205     2   z
9    204     2   C
10   203     2   A
11   202     2   v
12   201     2   B
13   301     3   C
14   302     3   w
15   303     3   B
16   304     3   A
17   305     3   D
18   405     4   A
19   404     4   C
20   403     4   D
21   402     4   B
22   401     4   y
23   501     5   C
24   502     5   B
25   503     5   A
26   504     5   D
27   505     5   x
```

Para los diseños completamente aleatorizados aumentados, utilice la función design.crd () .

3.11. Diseño de parcelas divididas

Estos diseños tienen dos factores, uno es aplicado en las parcelas y se define como una en un diseño de bloques completos al azar; y un segundo factor, que se aplica en las subparcelas de cada parcela aplica al azar. la función design.split() permite encontrar el plan experimental para este diseño y sus parámetros son:

```
> str(design.split)

function (trt1, trt2, r = NULL, design = c("rcbd", "crd", "lsd"), serie = 2,
    seed = 0, kinds = "Super-Duper", first = TRUE, randomization = TRUE)
```

Aplicación

```
> trt1<-c("A","B","C","D")
> trt2<-c("a","b","c")
> outdesign <-design.split(trt1,trt2,r=3,serie=2,seed=543)
```

```
> book10 <- outdesign$book
> print(book10)
```

```
   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    D    b
5    102      2     1    D    c
6    102      3     1    D    a
7    103      1     1    B    b
8    103      2     1    B    c
9    103      3     1    B    a
10   104      1     1    C    a
11   104      2     1    C    b
12   104      3     1    C    c
13   105      1     2    A    b
14   105      2     2    A    a
15   105      3     2    A    c
16   106      1     2    C    a
17   106      2     2    C    b
18   106      3     2    C    c
19   107      1     2    B    a
20   107      2     2    B    c
21   107      3     2    B    b
22   108      1     2    D    b
23   108      2     2    D    c
24   108      3     2    D    a
25   109      1     3    A    a
26   109      2     3    A    b
27   109      3     3    A    c
28   110      1     3    C    a
29   110      2     3    C    c
30   110      3     3    C    b
31   111      1     3    B    a
32   111      2     3    B    c
33   111      3     3    B    b
34   112      1     3    D    c
35   112      2     3    D    a
36   112      3     3    D    b
```

```
> p<-book10$trt1[seq(1,36,3)]
> q<-NULL
> for(i in 1:12)
+ q <- c(q,paste(book10$trt2[3*(i-1)+1],book10$trt2[3*(i-1)+2], book10$trt2[3*(i-1)+3]
```

In plots:

```
> print(t(matrix(p,c(4,3))))
```

```
     [,1] [,2] [,3] [,4]
[1,] "A"  "D"  "B"  "C"
[2,] "A"  "C"  "B"  "D"
[3,] "A"  "C"  "B"  "D"
```

Ind sub plots (split plot)

```
> 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"
```

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> head(book,5)
```

```
  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    D    b
5   102      2     1    D    c
```

3.12. Diseño bloques divididos o franjas

Estos diseños se utilizan cuando hay dos tipos de tratamientos (factores) y se aplican por separado en parcelas grandes, llamado bandas, en una dirección vertical y horizontal del bloque, la obtención de los bloques divididos. Cada bloque constituye una repetición y sus parámetros son:

```
> str(design.strip)
```

```
function (trt1, trt2, r, serie = 2, seed = 0, kinds = "Super-Duper", randomization = TRU
```

Aplication

```
> trt1<-c("A","B","C","D")
> trt2<-c("a","b","c")
> outdesign <-design.strip(trt1,trt2,r=3,serie=2,seed=543)
> book11 <- outdesign$book
> head(book11)
```

```
  plots block trt1 trt2
1   101     1    A    a
2   102     1    A    b
```

```
3   103     1    A    c
4   104     1    D    a
5   105     1    D    b
6   106     1    D    c
```

```
> t3<-paste(book11$trt1, book11$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"
```

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> head(book)
```

```
  plots block trt1 trt2
1   101     1    A    a
2   102     1    A    b
3   103     1    A    c
4   106     1    D    a
5   105     1    D    b
6   104     1    D    c
```

```
> array(book$plots,c(3,4,3))->X
> t(X[,,1])
```

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

```
> t(X[,,2])
```

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

```
> t(X[,,3])
```

```
     [,1] [,2] [,3]
[1,]  301  302  303
[2,]  306  305  304
[3,]  307  308  309
[4,]  312  311  310
```

3.13. Factorial

El factorial completo de n factores aplicados a un diseño experimental (CRD, DBCA y LSD) es común y esta pocedimiento en agricolae aplica el factorial a uno de estos tres diseños y sus parámetros son:

```
> str(design.ab)
```

```
function (trt, r = NULL, serie = 2, design = c("rcbd", "crd", "lsd"), seed = 0,
    kinds = "Super-Duper", first = TRUE, randomization = TRUE)
```

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

```
> trt <- c (4,2,3) # three factors with  4,2 and 3 levels.
```

En los diseños crd y DBCA, es necesario el valor r como el número de repeticiones, esto puede ser un vector si es desigual el número de repeticiones, igual o constante es recomendado.

```
> trt<-c(3,2) # factorial 3x2
> outdesign <-design.ab(trt, r=3, serie=2)
> book12 <- outdesign$book
> head(book12) # print of the field book
```

```
  plots block A B
1   101     1 3 1
2   102     1 2 2
3   103     1 1 1
4   104     1 1 2
5   105     1 3 2
6   106     1 2 1
```

Enumeración en serpentina:

```
> book <- zigzag(outdesign)
> head(book)

  plots block A B
1   101     1 3 1
2   102     1 2 2
3   103     1 1 1
4   104     1 1 2
5   105     1 3 2
6   106     1 2 1
```

Factorial 2x2x2 con 5 repeticiones en el diseño completamente al azar.

```
> trt<-c(2,2,2)
> crd<-design.ab(trt, r=5, serie=2,design="crd")
> names(crd)

[1] "parameters" "book"

> crd$parameters

$design
[1] "factorial"

$trt
[1] "1 1 1" "1 1 2" "1 2 1" "1 2 2" "2 1 1" "2 1 2" "2 2 1" "2 2 2"

$r
[1] 5 5 5 5 5 5 5 5

$serie
[1] 2

$seed
[1] 970386955

$kinds
[1] "Super-Duper"
```

```
[[7]]
[1] TRUE

$applied
[1] "crd"
```

```
> head(crd$book,25)
```

```
   plots r A B C
1    101 1 2 2 1
2    102 1 1 1 2
3    103 1 2 1 2
4    104 1 2 1 1
5    105 1 2 2 2
6    106 2 2 1 2
7    107 3 2 1 2
8    108 1 1 2 1
9    109 1 1 2 2
10   110 1 1 1 1
11   111 2 1 1 2
12   112 2 1 2 2
13   113 2 1 2 1
14   114 2 2 2 2
15   115 3 1 1 2
16   116 2 2 1 1
17   117 3 1 2 2
18   118 2 1 1 1
19   119 4 1 1 2
20   120 5 1 1 2
21   121 2 2 2 1
22   122 3 2 2 1
23   123 3 1 1 1
24   124 3 1 2 1
25   125 4 2 2 1
```

4. Comparación múltiple

Para el análisis, las siguientes funciones de agricolae se utilizan: *L*SD.test, HSD.test, duncan.test, scheffe.test, waller.test, SNK.test, REGW.test, Steel and Torry and Dickey (1997); Hsu (1996) y *d*urbin.test, kruskal, friedman, waerden.test and Median.test, Conover (1999). Para cada análisis estadístico, los datos se deben organizar en columnas. Para la demostración, se utilizará la base de datos agricolae.

Los datos de camote corresponden a un experimento completamente al azar en el campo con parcelas de 50 plantas de camote, sometidos al efecto del virus y con un control sin virus (Consulte el manual de referencia del paquete).

```
> data(sweetpotato)
> model<-aov(yield~virus, data=sweetpotato)
> cv.model(model)

[1] 17.1666

> with(sweetpotato,mean(yield))

[1] 27.625
```

Parámetros del modelo: Grados de libertad y la varianza del error:

```
> df<-df.residual(model)
> MSerror<-deviance(model)/df
```

4.1. La diferencia mínima significativa (DLS)

Se incluye la comparación múltiple a través del método de la diferencia mínima significativa, referencia, Steel and Torry and Dickey (1997).

```
> # comparison <- LSD.test(yield,virus,df,MSerror)
> LSD.test(model, "virus",console=TRUE)

Study: model ~ "virus"

LSD t Test for yield

Mean Square Error:  22.48917

virus,  means and individual ( 95 %) CI

      yield      std r       LCL      UCL  Min  Max
cc 24.40000 3.609709 3 18.086268 30.71373 21.7 28.5
fc 12.86667 2.159475 3  6.552935 19.18040 10.6 14.9
ff 36.33333 7.333030 3 30.019601 42.64707 28.0 41.8
oo 36.90000 4.300000 3 30.586268 43.21373 32.1 40.4

Alpha: 0.05 ; DF Error: 8
Critical Value of t: 2.306004

least Significant Difference: 8.928965

Treatments with the same letter are not significantly different.

      yield groups
oo 36.90000      a
ff 36.33333      a
cc 24.40000      b
fc 12.86667      c
```

En la función LSD.test (), la comparación múltiple se llevó a cabo. Con el fin de obtener las probabilidades de las comparaciones, se debe indicar que no se requieren grupos; por lo tanto:

```
> # comparison <- LSD.test(yield, virus,df, MSerror, group=FALSE)
> outLSD <-LSD.test(model, "virus", group=FALSE,console=TRUE)

Study: model ~ "virus"

LSD t Test for yield

Mean Square Error:  22.48917

virus,  means and individual ( 95 %) CI

      yield      std r       LCL      UCL  Min  Max
cc 24.40000 3.609709 3 18.086268 30.71373 21.7 28.5
fc 12.86667 2.159475 3  6.552935 19.18040 10.6 14.9
ff 36.33333 7.333030 3 30.019601 42.64707 28.0 41.8
oo 36.90000 4.300000 3 30.586268 43.21373 32.1 40.4

Alpha: 0.05 ; DF Error: 8
Critical Value of t: 2.306004

Comparison between treatments means

         difference pvalue signif.        LCL        UCL
cc - fc  11.5333333 0.0176       *   2.604368  20.462299
cc - ff -11.9333333 0.0151       * -20.862299  -3.004368
cc - oo -12.5000000 0.0121       * -21.428965  -3.571035
fc - ff -23.4666667 0.0003     *** -32.395632 -14.537701
fc - oo -24.0333333 0.0003     *** -32.962299 -15.104368
ff - oo  -0.5666667 0.8873          -9.495632   8.362299
```

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

4.2. holm, hommel, hochberg, bonferroni, BH, BY, fdr

Con la función LSD.test() se puede realizar ajustes en las probabilidades encontradas, como por ejemplo el ajuste de Bonferroni, Holm y otras opciones ver Ajustar P-valores para comparaciones múltiples, función p.adjust (stats), R Core Team (2017).

```
> LSD.test(model, "virus", group=FALSE, p.adj= "bon",console=TRUE)

Study: model ~ "virus"

LSD t Test for yield
```

```
P value adjustment method: bonferroni

Mean Square Error:  22.48917

virus,  means and individual ( 95 %) CI

      yield      std r       LCL      UCL  Min  Max
cc 24.40000 3.609709 3 18.086268 30.71373 21.7 28.5
fc 12.86667 2.159475 3  6.552935 19.18040 10.6 14.9
ff 36.33333 7.333030 3 30.019601 42.64707 28.0 41.8
oo 36.90000 4.300000 3 30.586268 43.21373 32.1 40.4

Alpha: 0.05 ; DF Error: 8
Critical Value of t: 3.478879

Comparison between treatments means

         difference pvalue signif.        LCL         UCL
cc - fc  11.5333333 0.1058          -1.937064  25.0037305
cc - ff -11.9333333 0.0904       . -25.403730   1.5370638
cc - oo -12.5000000 0.0725       . -25.970397   0.9703971
fc - ff -23.4666667 0.0018      ** -36.937064  -9.9962695
fc - oo -24.0333333 0.0015      ** -37.503730 -10.5629362
ff - oo  -0.5666667 1.0000         -14.037064  12.9037305
```

Other comparison tests can be applied, such as ”duncan”, ”Student-Newman-Keuls”, ”tukey”, and ”waller-duncan.”

Para Duncan, utilice la función duncan.test(); para ”Student-Newman-Keuls”, SNK.test(); para ”Tukey”, HSD.test(); para ”Scheffe”, scheffe.test(); para REGW la prueba REGW.test y para Waller-Duncan”la función waller.test(). Los parámetros son los mismos. Waller requiere ademas el valor de F calculado de los tratamientos del ANOVA. Si se utiliza la salida del modelo aov o lm como un parámetro, ya no es necesario los parametros F, grados de libertad y la variancia del error.

4.3. La Nueva prueba múltiple de Duncan

Corresponde a la prueba de Duncan presentado en la referencia, Steel and Torry and Dickey (1997).

```
> duncan.test(model, "virus",console=TRUE)

Study: model ~ "virus"

Duncan's new multiple range test
for yield

Mean Square Error:  22.48917
```

```
virus,  means

      yield      std r  Min  Max
cc 24.40000 3.609709 3 21.7 28.5
fc 12.86667 2.159475 3 10.6 14.9
ff 36.33333 7.333030 3 28.0 41.8
oo 36.90000 4.300000 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.

      yield groups
oo 36.90000      a
ff 36.33333      a
cc 24.40000      b
fc 12.86667      c
```

4.4. Student-Newman-Keuls

Student, Newman y Keuls ayudado a mejorar la prueba de Newman-Keuls de 1939, que fue conocido como el método de Keuls, véase la referencia, Steel and Torry and Dickey (1997).

```
> SNK.test(model, "virus", alpha=0.05,console=TRUE)

Study: model ~ "virus"

Student Newman Keuls Test
for yield

Mean Square Error:  22.48917

virus,  means

      yield      std r  Min  Max
cc 24.40000 3.609709 3 21.7 28.5
fc 12.86667 2.159475 3 10.6 14.9
ff 36.33333 7.333030 3 28.0 41.8
oo 36.90000 4.300000 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.

      yield groups
oo 36.90000      a
ff 36.33333      a
cc 24.40000      b
fc 12.86667      c

> # Comparison treatments
> SNK.test(model, "virus", group=FALSE,console=TRUE)

Study: model ~ "virus"

Student Newman Keuls Test
for yield

Mean Square Error:  22.48917

virus,  means

      yield      std r  Min  Max
cc 24.40000 3.609709 3 21.7 28.5
fc 12.86667 2.159475 3 10.6 14.9
ff 36.33333 7.333030 3 28.0 41.8
oo 36.90000 4.300000 3 32.1 40.4

Comparison between treatments means

         difference pvalue signif.        LCL        UCL
cc - fc  11.5333333 0.0176       *   2.604368  20.462299
cc - ff -11.9333333 0.0151       * -20.862299  -3.004368
cc - oo -12.5000000 0.0291       * -23.564170  -1.435830
fc - ff -23.4666667 0.0008     *** -34.530836 -12.402497
fc - oo -24.0333333 0.0012      ** -36.433003 -11.633664
ff - oo  -0.5666667 0.8873          -9.495632   8.362299
```

4.5. Ryan, Einot, Gabriel y Welsch

Util cuando las comparaciones multiples producen desiguales limites de confianza y son pruebas de rango multiple, Hsu (1996).

```
> # REGW.test(model, "virus", alpha=0.05,console=TRUE)
> REGW.test(model, "virus", group=FALSE,console=TRUE)

Study: model ~ "virus"
```

```
Ryan, Einot and Gabriel and Welsch multiple range test
for yield

Mean Square Error:  22.48917

virus,  means

      yield      std r  Min  Max
cc 24.40000 3.609709 3 21.7 28.5
fc 12.86667 2.159475 3 10.6 14.9
ff 36.33333 7.333030 3 28.0 41.8
oo 36.90000 4.300000 3 32.1 40.4

Comparison between treatments means

         difference pvalue signif.         LCL         UCL
cc - fc  11.5333333 0.0350       *   0.9112173  22.1554494
cc - ff -11.9333333 0.0360       * -22.9975029  -0.8691637
cc - oo -12.5000000 0.0482       * -24.8996698  -0.1003302
fc - ff -23.4666667 0.0006     *** -34.0887827 -12.8445506
fc - oo -24.0333333 0.0007     *** -35.0975029 -12.9691637
ff - oo  -0.5666667 0.9873         -11.1887827  10.0554494
```

4.6. Procedimiento de Tukey (HSD)

Esta prueba de rango studentized, creado por Tukey en 1953, es conocido como HSD de Tukey (Diferencias honestamente significativa) se referencia en Steel and Torry and Dickey (1997).

```
> outHSD <-HSD.test(model, "virus",console=TRUE)

Study: model ~ "virus"

HSD Test for yield

Mean Square Error:  22.48917

virus,  means

      yield      std r  Min  Max
cc 24.40000 3.609709 3 21.7 28.5
fc 12.86667 2.159475 3 10.6 14.9
ff 36.33333 7.333030 3 28.0 41.8
oo 36.90000 4.300000 3 32.1 40.4

Alpha: 0.05 ; DF Error: 8
Critical Value of Studentized Range: 4.52881

Minimun Significant Difference: 12.39967
```

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

      yield groups
oo 36.90000      a
ff 36.33333     ab
cc 24.40000     bc
fc 12.86667      c
```

```
> print(outHSD)

$statistics
   MSerror Df    Mean      CV      MSD
  22.48917  8 27.625 17.1666 12.39967

$parameters
   test name.t ntr StudentizedRange alpha
  Tukey  virus   4          4.52881  0.05

$means
      yield      std r  Min  Max   Q25  Q50   Q75
cc 24.40000 3.609709 3 21.7 28.5 22.35 23.0 25.75
fc 12.86667 2.159475 3 10.6 14.9 11.85 13.1 14.00
ff 36.33333 7.333030 3 28.0 41.8 33.60 39.2 40.50
oo 36.90000 4.300000 3 32.1 40.4 35.15 38.2 39.30

$comparison
NULL

$groups
      yield groups
oo 36.90000      a
ff 36.33333     ab
cc 24.40000     bc
fc 12.86667      c

attr(,"class")
[1] "group"
```

4.7. Waller-Duncan prueba Bayesiana

En 1975, Duncan siguió los procedimientos de comparación múltiple, introduciendo el criterio de minimizar los dos tipos de error experimental por procedimientps Bayesianos, resultando la prueba de Waller-Duncan, **?**, Steel and Torry and Dickey (1997).

```
> anova(model)

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
```

```
> with(sweetpotato,
+ waller.test(yield,virus,df,MSerror,Fc= 17.345, group=FALSE,console=TRUE))
```

```
Study: yield ~ virus

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 r  Min  Max
cc 24.40000 3.609709 3 21.7 28.5
fc 12.86667 2.159475 3 10.6 14.9
ff 36.33333 7.333030 3 28.0 41.8
oo 36.90000 4.300000 3 32.1 40.4

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
```

En otro caso, con sólo invocar el objeto de modelo:

```
> outWaller <- waller.test(model, "virus", group=FALSE,console=FALSE)
```

El objeto outWaller encontrado tiene información para hacer otros procedimientos.

```
> names(outWaller)
```

```
[1] "statistics" "parameters" "means"      "comparison" "groups"
```

```
> print(outWaller$comparison)

         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 virus efecto ff no es significativo para el control oo.

```
> outWaller$statistics

    Mean Df      CV  MSerror  F.Value Waller CriticalDifference
  27.625  8 17.1666 22.48917 17.34478  2.236           8.657906
```

4.8. La prueba de Scheffe

Este método, creado por Scheffe en 1959, es muy general para todos los posibles contrastes y sus intervalos de confianza. Los intervalos de confianza para los promedios son muy amplias, lo que resulta en una prueba muy conservador para la comparación entre las medias de tratamiento véase la referencia, Steel and Torry and Dickey (1997).

```
> # analysis of variance:
> model<-aov(yield~virus, data=sweetpotato)
> scheffe.test(model,"virus", group=TRUE,console=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 r  Min  Max
cc 24.40000 3.609709 3 21.7 28.5
fc 12.86667 2.159475 3 10.6 14.9
ff 36.33333 7.333030 3 28.0 41.8
oo 36.90000 4.300000 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.

      yield groups
oo 36.90000      a
ff 36.33333      a
cc 24.40000     ab
fc 12.86667      b
```

El valor mínimo significativo es muy alta. Si necesita las probabilidades aproximadas de comparación, se puede utilizar la opcion group=FALSE.

```
> outScheffe <- scheffe.test(model,"virus", group=FALSE, console=TRUE)

Study: model ~ "virus"

Scheffe Test for yield

Mean Square Error  : 22.48917

virus,  means

      yield      std r  Min  Max
cc 24.40000 3.609709 3 21.7 28.5
fc 12.86667 2.159475 3 10.6 14.9
ff 36.33333 7.333030 3 28.0 41.8
oo 36.90000 4.300000 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 0.0978   .  -1.000348  24.0670149
cc - ff -11.9333333 0.0855   . -24.467015   0.6003483
cc - oo -12.5000000 0.0706   . -25.033682   0.0336816
fc - ff -23.4666667 0.0023  ** -36.000348 -10.9329851
fc - oo -24.0333333 0.0020  ** -36.567015 -11.4996517
ff - oo  -0.5666667 0.9991     -13.100348  11.9670149
```

4.9. Comparación múltiple de tratamientos en factorial

En factoriales se puede aplicar las pruebas comparativas de: LSD, HSD, Waller-Duncan, Duncan, Scheffe, SNK .

```
> # model <-aov (y ~ A * B * C, data)
> # compare <-LSD.test (model, c ("A", "B", "C"),console=TRUE)
```

La comparación es la combinación de A:B:C.
Diseño de datos DBCA con un factorial clon*nitrógeno. La variable de respuesta el rendimiento (yield).

```
> yield <-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 "
+  )
> block <-gl (4, 9)
> clone <-rep (gl (3, 3, labels = c ("c1", "c2", "c3")), 4)
> nitrogen <-rep (gl (3, 1, labels = c ("n1", "n2", "n3")), 12)
> A <-data.frame (block, clone, nitrogen, yield)
> head(A)

  block clone nitrogen yield
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

> modelAov <-aov (yield ~ block + clone * nitrogen, data = A)
> anova (modelAov)

Analysis of Variance Table

Response: yield
               Df Sum Sq Mean Sq  F value    Pr(>F)
block           3  20.75   6.917   5.8246 0.0038746 **
clone           2 497.72 248.861 209.5673 6.370e-16 ***
nitrogen        2  54.06  27.028  22.7602 2.865e-06 ***
clone:nitrogen  4  43.28  10.819   9.1111 0.0001265 ***
Residuals      24  28.50   1.188
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> out<-LSD.test(modelAov,c("clone","nitrogen"),
+               main="Yield ~ block+nitrogen+clone+clone:nitrogen",console=TRUE)

Study: Yield ~ block+nitrogen+clone+clone:nitrogen

LSD t Test for yield

Mean Square Error:  1.1875
```

```
clone:nitrogen,  means and individual ( 95 %) CI

      yield       std r       LCL       UCL Min Max
c1:n1  7.50 1.2909944 4  6.375459  8.624541   6   9
c1:n2  8.50 1.2909944 4  7.375459  9.624541   7  10
c1:n3  9.00 0.8164966 4  7.875459 10.124541   8  10
c2:n1 13.50 1.2909944 4 12.375459 14.624541  12  15
c2:n2 16.75 0.9574271 4 15.625459 17.874541  16  18
c2:n3 20.25 1.7078251 4 19.125459 21.374541  18  22
c3:n1  9.50 1.2909944 4  8.375459 10.624541   8  11
c3:n2  9.50 1.7320508 4  8.375459 10.624541   8  12
c3:n3 10.25 1.5000000 4  9.125459 11.374541   9  12

Alpha: 0.05 ; DF Error: 24
Critical Value of t: 2.063899

least Significant Difference: 1.590341

Treatments with the same letter are not significantly different.

      yield groups
c2:n3 20.25      a
c2:n2 16.75      b
c2:n1 13.50      c
c3:n3 10.25      d
c3:n1  9.50     de
c3:n2  9.50     de
c1:n3  9.00    def
c1:n2  8.50     ef
c1:n1  7.50      f
```

4.10. Análisis de los bloques incompletos balanceados

Este análisis puede provenir de diseños equilibrados o parcialmente equilibradas. La función BIB.test() está en diseños balanceados y PBIB.test(), por diseños parcialmente balanceadas. En el siguiente ejemplo, los datos de los *agricolae* serán utilizados, Joshi (1987).

```
> # Ejemplo -linear estimation and design of experiments-. (Joshi)
> # Profesor de Estadistica, Institute of Social Sciences Agra, India
> # 6 variedades de trigo en 10 bloques de 3 parcelas cada una.
> block<-gl(10,3)
> variety<-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)
> BIB.test(block, variety, y,console=TRUE)

ANALYSIS BIB:  y
Class level information
```

```
> par(mar=c(3,3,2,1))
> pic1<-bar.err(out$means,variation="range",ylim=c(5,25), bar=FALSE,col=0,las=1)
> points(pic1$index,pic1$means,pch=18,cex=1.5,col="blue")
> axis(1,pic1$index,labels=FALSE)
> title(main="promedio y rango\nclon:nitrogen")
```

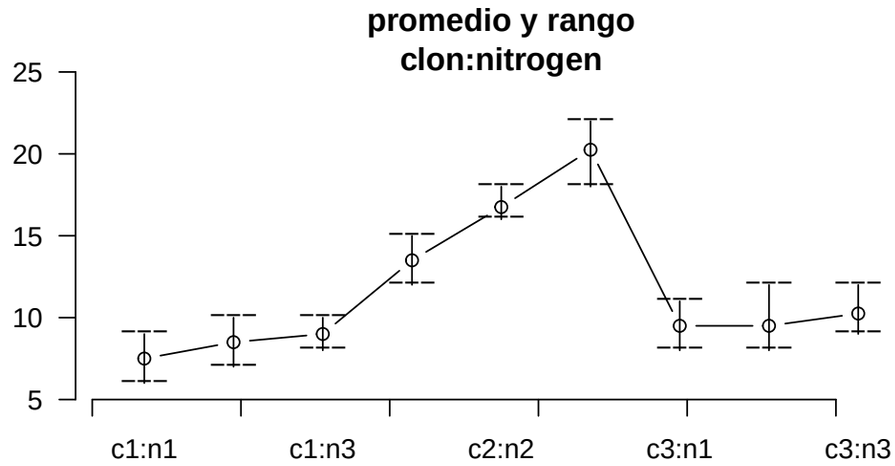

Figura 8: combinado clone:nitrogen

```
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

variety,  statistics

     y mean.adj       SE r       std Min Max
1 70.2 75.13333 3.728552 5  5.069517  63  77
2 60.0 58.71667 3.728552 5  4.898979  54  65
3 59.4 58.55000 3.728552 5 12.381438  45  75
```

```
4 55.0 54.96667 3.728552 5  9.848858  38  62
5 61.4 60.05000 3.728552 5  4.505552  54  65
6 55.8 54.38333 3.728552 5 10.756393  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 >>>

Comparison between treatments means
      Difference pvalue sig.
1 - 2 16.4166667 0.0080   **
1 - 3 16.5833333 0.0074   **
1 - 4 20.1666667 0.0018   **
1 - 5 15.0833333 0.0132    *
1 - 6 20.7500000 0.0016   **
2 - 3  0.1666667 0.9756
2 - 4  3.7500000 0.4952
2 - 5 -1.3333333 0.8070
2 - 6  4.3333333 0.4318
3 - 4  3.5833333 0.5142
3 - 5 -1.5000000 0.7836
3 - 6  4.1666667 0.4492
4 - 5 -5.0833333 0.3582
4 - 6  0.5833333 0.9148
5 - 6  5.6666667 0.3074

Treatments with the same letter are not significantly different.

         y groups
1 75.13333      a
5 60.05000      b
2 58.71667      b
3 58.55000      b
4 54.96667      b
6 54.38333      b
```

function (block, trt, y, test = c("lsd", "tukey", "duncan", "waller", "snk"), alpha = 0.05, group = TRUE) LSD, Duncan de Tukey, Waller-Duncan y SNK, se pueden utilizar. Las pro-

babilidades de la comparación también se pueden obtener. Sólo se debe indicar: group=FALSE, así:

```
> out <-BIB.test(block, trt=variety, y, test="tukey", group=FALSE, console=TRUE)

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

variety,  statistics

     y mean.adj       SE r       std Min Max
1 70.2 75.13333 3.728552 5  5.069517  63  77
2 60.0 58.71667 3.728552 5  4.898979  54  65
3 59.4 58.55000 3.728552 5 12.381438  45  75
4 55.0 54.96667 3.728552 5  9.848858  38  62
5 61.4 60.05000 3.728552 5  4.505552  54  65
6 55.8 54.38333 3.728552 5 10.756393  39  67

Tukey
Alpha      : 0.05
Std.err    : 3.792292
HSD        : 17.42458
Parameters BIB
Lambda     : 2
treatmeans : 6
Block size : 3
Blocks     : 10
Replication: 5

Efficiency factor 0.8
```

```
<<< Book >>>

Comparison between treatments means
      Difference pvalue sig.
1 - 2 16.4166667 0.0705    .
1 - 3 16.5833333 0.0666    .
1 - 4 20.1666667 0.0191    *
1 - 5 15.0833333 0.1096
1 - 6 20.7500000 0.0155    *
2 - 3  0.1666667 1.0000
2 - 4  3.7500000 0.9792
2 - 5 -1.3333333 0.9998
2 - 6  4.3333333 0.9616
3 - 4  3.5833333 0.9829
3 - 5 -1.5000000 0.9997
3 - 6  4.1666667 0.9674
4 - 5 -5.0833333 0.9273
4 - 6  0.5833333 1.0000
5 - 6  5.6666667 0.8908

> names(out)

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

> rm(block,variety)
```

bar.group: out$groups
plot.group: out
bar.err: out$means

4.11. Bloques incompletos parcialmente balanceados

La funcón PBIB.test(), Joshi (1987), puede ser aplicado para un latice o diseño alfa.

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

```
> library(MASS)
> library(nlme)
> # alpha design
> Genotype<-c(paste("gen0",1:9,sep=""),paste("gen",10:30,sep=""))
> r<-2
> k<-3
> plan<-design.alpha(Genotype,k,r,seed=5)

Alpha Design (0,1) - Serie  I

Parameters Alpha Design
```

```
=======================
Treatmeans : 30
Block size : 3
Blocks     : 10
Replication: 2

Efficiency factor
(E ) 0.6170213

<<< Book >>>

> rm(Genotype)
```

El plan generado es plan$book.
Supongamos que la observación correspondiente a cada unidad experimental es:

```
> yield <-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)
```

La tabla de datos se construye para el análisis. En teoría, se supone que se aplica un diseño y el experimento se lleva a cabo; posteriormente, se observan las variables de estudio de cada unidad experimental.

```
> dplan<-data.frame(plan$book,yield)
> # The analysis:
> modelPBIB <- with(dplan,PBIB.test(block, Genotype, replication, yield, k=3,
+ group=TRUE,console=TRUE))

ANALYSIS PBIB:  yield

Class level information
block : 20
Genotype : 30

Number of observations:  60

Estimation Method:  Residual (restricted) maximum likelihood

Parameter Estimates
                     Variance
block:replication 2.834033e+00
replication       8.045349e-09
Residual          2.003098e+00

                     Fit Statistics
AIC                       213.65937
BIC                       259.89888
-2 Res Log Likelihood     -73.82968
```

```
Analysis of Variance Table

Response: yield
          Df Sum Sq Mean Sq F value Pr(>F)
Genotype  29 72.006  2.4830  1.2396 0.3668
Residuals 11 22.034  2.0031

Coefficient of variation: 31.2 %
yield Means: 4.533333

Parameters PBIB
                   .
Genotype          30
block size         3
block/replication 10
replication        2

Efficiency factor 0.6170213

Comparison test lsd

Treatments with the same letter are not significantly different.

      yield.adj groups
gen27  7.728746      a
gen20  6.714886     ab
gen01  6.504753     ab
gen16  6.192469    abc
gen30  6.032066   abcd
gen03  5.734592   abcd
gen18  5.472736   abcd
gen23  5.454719   abcd
gen28  5.140446   abcd
gen29  5.069293   abcd
gen12  4.873878   abcd
gen11  4.793620   abcd
gen21  4.741512   abcd
gen22  4.586968   abcd
gen06  4.563448   abcd
gen15  4.424032   abcd
gen13  4.285956   abcd
gen26  4.197617   abcd
gen14  4.165424   abcd
gen04  3.978318   abcd
gen24  3.943457   abcd
gen10  3.628197    bcd
gen07  3.495102    bcd
```

```
> par(cex=0.6,mar=c(4,3,1,1))
> plot(modelPBIB,las=2)
```

```
Warning  values plot is not adjusted
```

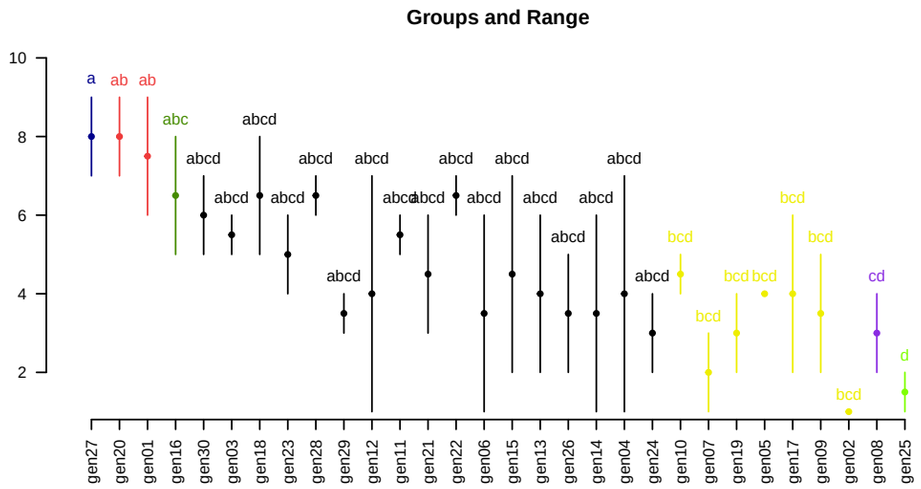

Figura 9: Tratamientos agrupados

```
gen19  3.378992    bcd
gen05  3.341404    bcd
gen17  3.052108    bcd
gen09  2.999639    bcd
gen02  2.879459    bcd
gen08  2.439879     cd
gen25  2.186282      d

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

Los promedios ajustados se pueden extraer a partir del modelo.

```
> head(modelPBIB$means)
```

```
      yield yield.adj       SE r       std Min Max  Q25 Q50  Q75
gen01   7.5  6.504753 1.313644 2 2.1213203   6   9 6.75 7.5 8.25
gen02   1.0  2.879459 1.310727 2 0.0000000   1   1 1.00 1.0 1.00
gen03   5.5  5.734592 1.310727 2 0.7071068   5   6 5.25 5.5 5.75
gen04   4.0  3.978318 1.313644 2 4.2426407   1   7 2.50 4.0 5.50
gen05   4.0  3.341404 1.310727 2 0.0000000   4   4 4.00 4.0 4.00
gen06   3.5  4.563448 1.310727 2 3.5355339   1   6 2.25 3.5 4.75
```

La comparación:

```
> head(modelPBIB$comparison,60)
```

```
              Difference   stderr pvalue
gen01 - gen02  3.6252940 1.770133 0.0652
gen01 - gen03  0.7701618 1.770133 0.6720
gen01 - gen04  2.5264354 1.816445 0.1918
gen01 - gen05  3.1633495 1.846640 0.1148
gen01 - gen06  1.9413054 1.770133 0.2962
gen01 - gen07  3.0096514 1.851959 0.1324
gen01 - gen08  4.0648738 1.576447 0.0256
gen01 - gen09  3.5051139 1.576447 0.0480
gen01 - gen10  2.8765561 1.844369 0.1472
gen01 - gen11  1.7111335 1.576447 0.3010
gen01 - gen12  1.6308755 1.727017 0.3652
gen01 - gen13  2.2187974 1.853044 0.2564
gen01 - gen14  2.3393290 1.828368 0.2270
gen01 - gen15  2.0807215 1.855004 0.2858
gen01 - gen16  0.3122845 1.851959 0.8692
gen01 - gen17  3.4526453 1.844369 0.0880
gen01 - gen18  1.0320169 1.770133 0.5716
gen01 - gen19  3.1257616 1.576447 0.0730
gen01 - gen20 -0.2101325 1.851959 0.9118
gen01 - gen21  1.7632411 1.846640 0.3602
gen01 - gen22  1.9177848 1.828368 0.3168
gen01 - gen23  1.0500345 1.853044 0.5824
gen01 - gen24  2.5612960 1.828368 0.1888
gen01 - gen25  4.3184716 1.851959 0.0398
gen01 - gen26  2.3071359 1.846640 0.2374
gen01 - gen27 -1.2239927 1.727017 0.4932
gen01 - gen28  1.3643068 1.846640 0.4754
gen01 - gen29  1.4354599 1.816445 0.4460
gen01 - gen30  0.4726870 1.828368 0.8008
gen02 - gen03 -2.8551322 1.838267 0.1486
gen02 - gen04 -1.0988586 1.576447 0.5002
gen02 - gen05 -0.4619445 1.849685 0.8074
gen02 - gen06 -1.6839886 1.832612 0.3778
gen02 - gen07 -0.6156426 1.832612 0.7432
gen02 - gen08  0.4395799 1.812352 0.8128
gen02 - gen09 -0.1201801 1.756207 0.9466
gen02 - gen10 -0.7487379 1.851959 0.6938
gen02 - gen11 -1.9141605 1.782236 0.3058
gen02 - gen12 -1.9944185 1.576447 0.2320
gen02 - gen13 -1.4064966 1.828368 0.4580
gen02 - gen14 -1.2859650 1.846673 0.5006
gen02 - gen15 -1.5445725 1.846640 0.4208
gen02 - gen16 -3.3130095 1.847787 0.1004
gen02 - gen17 -0.1726487 1.770133 0.9240
gen02 - gen18 -2.5932771 1.718127 0.1594
gen02 - gen19 -0.4995324 1.587130 0.7588
gen02 - gen20 -3.8354265 1.838267 0.0610
```

```
gen02 - gen21 -1.8620529 1.850096 0.3358
gen02 - gen22 -1.7075092 1.756207 0.3518
gen02 - gen23 -2.5752595 1.851959 0.1918
gen02 - gen24 -1.0639980 1.587130 0.5164
gen02 - gen25  0.6931776 1.846673 0.7146
gen02 - gen26 -1.3181581 1.782236 0.4750
gen02 - gen27 -4.8492867 1.828368 0.0224
gen02 - gen28 -2.2609872 1.812352 0.2382
gen02 - gen29 -2.1898341 1.846640 0.2606
gen02 - gen30 -3.1526070 1.847787 0.1160
gen03 - gen04  1.7562737 1.846640 0.3620
gen03 - gen05  2.3931878 1.812352 0.2134
gen03 - gen06  1.1711437 1.718127 0.5096
```

Los datos sobre los promedios ajustados y su variación se pueden ilustrar véase Figura 10. puesto que el objeto creado es muy similar a los objetos generados por las múltiples comparaciones. Análisis de 3x3 celosía equilibrada, 9 tratamientos, 4 repeticiones.
Cree los datos en un archivo de texto: latice3x3.txt y leer con R:

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

```
> trt<-c(1,8,5,5,2,9,2,7,3,6,4,9,4,6,9,8,7,6,1,5,8,3,2,7,3,7,2,1,3,4,6,4,9,5,8,1)
> yield<-c(48.76,10.83,12.54,11.07,22,47.43,27.67,30,13.78,37,42.37,39,14.46,
+          30.69,42.01,22,42.8,28.28,50,24,24,15.42,30,23.8,19.68,31,23,41,12.9,
+          49.95,25,45.57,30,20,18,43.81)
> sqr<-rep(gl(4,3),3)
> block<-rep(1:12,3)
> modelLatice <-PBIB.test(block,trt,sqr,yield,k=3,console=TRUE)

ANALYSIS PBIB:  yield

Class level information
block : 12
trt : 9

Number of observations:  36
```

```
> par(mfrow=c(2,2),cex=0.6,mar=c(3,2,3,1))
> C1<-bar.err(modelPBIB$means[1:7, ], ylim=c(0,9), col=0, main="C1",
+             variation="range", border=3,las=2)
> C2<-bar.err(modelPBIB$means[8:15,], ylim=c(0,9), col=0, main="C2",
+             variation="range", border =4,las=2)
> C3<-bar.err(modelPBIB$means[16:22,], ylim=c(0,9), col=0, main="C3",
+             variation="range", border =2,las=2)
> C4<-bar.err(modelPBIB$means[23:30,], ylim=c(0,9), col=0, main="C4",
+             variation="range", border =6,las=2)
```

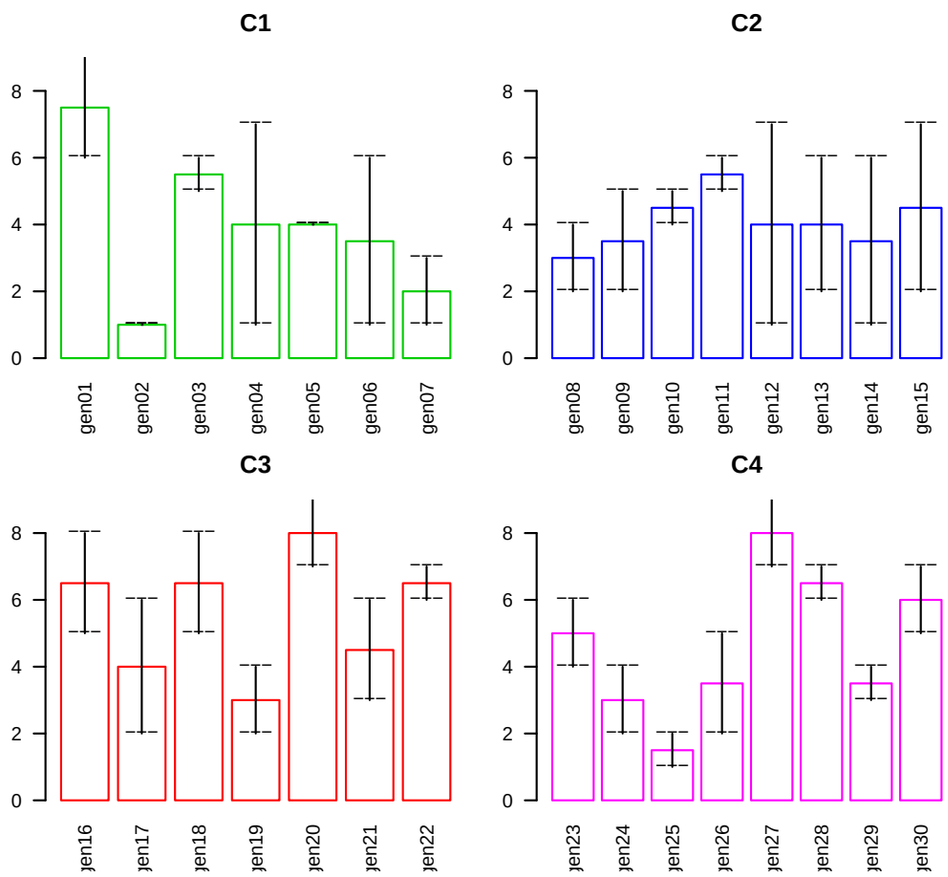

Figura 10: Rango en cada tratamiento

```
Estimation Method:  Residual (restricted) maximum likelihood

Parameter Estimates
              Variance
block:sqr 1.735327e-08
sqr       1.783279e-07
Residual  5.693724e+01

                     Fit Statistics
AIC                       222.23197
```

```
BIC                         237.78201
-2 Res Log Likelihood       -99.11599

Analysis of Variance Table

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

Treatments with the same letter are not significantly different.

  yield.adj groups
1   45.8925      a
9   39.6100     ab
4   38.0875     ab
7   31.9000     bc
6   30.2425     bc
2   25.6675     cd
8   18.7075      d
5   16.9025      d
3   15.4450      d

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


> modelLatice$means

    yield yield.adj       SE r       std   Min   Max     Q25    Q50     Q75
1 45.8925   45.8925 3.772839 4  4.217720 41.00 50.00 43.1075 46.285 49.0700
2 25.6675   25.6675 3.772839 4  3.801170 22.00 30.00 22.7500 25.335 28.2525
3 15.4450   15.4450 3.772839 4  3.010266 12.90 19.68 13.5600 14.600 16.4850
```

```
> par(mar=c(2,2,0,1),cex=0.6)
> plot(modelLatice)

Warning  values plot is not adjusted
```

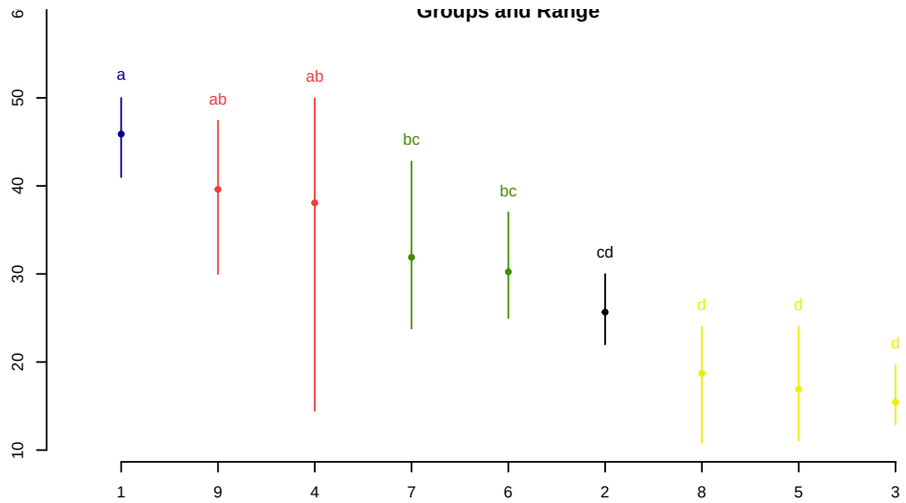

Figura 11: Grupos de Tratamiento

```
4 38.0875   38.0875 3.772839 4 16.055168 14.46 49.95 35.3925 43.970 46.6650
5 16.9025   16.9025 3.772839 4  6.137819 11.07 24.00 12.1725 16.270 21.0000
6 30.2425   30.2425 3.772839 4  5.072779 25.00 37.00 27.4600 29.485 32.2675
7 31.9000   31.9000 3.772839 4  7.933894 23.80 42.80 28.4500 30.500 33.9500
8 18.7075   18.7075 3.772839 4  5.813968 10.83 24.00 16.2075 20.000 22.5000
9 39.6100   39.6100 3.772839 4  7.294669 30.00 47.43 36.7500 40.505 43.3650
```

```
> head(modelLatice$comparison,10)

      Difference   stderr pvalue
1 - 2    20.2250 5.335599 0.0016
1 - 3    30.4475 5.335599 0.0000
1 - 4     7.8050 5.335599 0.1628
1 - 5    28.9900 5.335599 0.0000
1 - 6    15.6500 5.335599 0.0098
1 - 7    13.9925 5.335599 0.0184
1 - 8    27.1850 5.335599 0.0002
1 - 9     6.2825 5.335599 0.2562
2 - 3    10.2225 5.335599 0.0734
2 - 4   -12.4200 5.335599 0.0334
```

4.12. Bloques Aumentados

La función DAU.test() se puede utilizar para el análisis del diseño de bloque aumentados. Los datos deben ser organizados en una tabla, que contiene los bloques, los 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
```

Los tratamientos son en cada bloque:

```
> 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"
```

Analysis:

```
> modelDAU<- 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             7 505.87  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


Treatments with the same letter are not significantly different.

     yield groups
h 93.50000      a
f 86.50000     ab
A 84.66667     ab
D 83.33333     ab
C 82.00000     ab
j 79.50000     ab
B 79.00000     ab
e 78.25000     ab
k 78.25000     ab
i 77.25000     ab
l 77.25000     ab
g 73.25000      b

Comparison between treatments means

<<< to see the objects: comparison and means  >>>
```

```
> modelDAU$means
```

```
     yield      std r Min Max  Q25 Q50  Q75 mean.adj       SE block
A 84.66667 6.658328 3  79  92 81.0  83 87.5 84.66667 2.998456
B 79.00000 2.000000 3  77  81 78.0  79 80.0 79.00000 2.998456
C 82.00000 4.582576 3  78  87 79.5  81 84.0 82.00000 2.998456
D 83.33333 6.806859 3  78  91 79.5  81 86.0 83.33333 2.998456
e 79.00000       NA 1  79  79 79.0  79 79.0 78.25000 5.193479    II
f 89.00000       NA 1  89  89 89.0  89 89.0 86.50000 5.193479   III
g 70.00000       NA 1  70  70 70.0  70 70.0 73.25000 5.193479     I
h 96.00000       NA 1  96  96 96.0  96 96.0 93.50000 5.193479   III
i 78.00000       NA 1  78  78 78.0  78 78.0 77.25000 5.193479    II
j 82.00000       NA 1  82  82 82.0  82 82.0 79.50000 5.193479   III
k 75.00000       NA 1  75  75 75.0  75 75.0 78.25000 5.193479     I
l 74.00000       NA 1  74  74 74.0  74 74.0 77.25000 5.193479     I
```

```
> modelDAU<- DAU.test(block,trt,yield,method="lsd",group=FALSE,console=FALSE)
> head(modelDAU$comparison,40)
```

```
      Difference pvalue sig.
A - B   5.666667 0.2298
A - C   2.666667 0.5526
A - D   1.333333 0.7638
A - e   6.416667 0.3520
```

```
A - f  -1.833333 0.7828
A - g  11.416667 0.1228
A - h  -8.833333 0.2142
A - i   7.416667 0.2878
A - j   5.166667 0.4476
A - k   6.416667 0.3520
A - l   7.416667 0.2878
B - C  -3.000000 0.5058
B - D  -4.333333 0.3462
B - e   0.750000 0.9100
B - f  -7.500000 0.2830
B - g   5.750000 0.4008
B - h -14.500000 0.0628    .
B - i   1.750000 0.7924
B - j  -0.500000 0.9400
B - k   0.750000 0.9100
B - l   1.750000 0.7924
C - D  -1.333333 0.7638
C - e   3.750000 0.5770
C - f  -4.500000 0.5058
C - g   8.750000 0.2180
C - h -11.500000 0.1206
C - i   4.750000 0.4834
C - j   2.500000 0.7078
C - k   3.750000 0.5770
C - l   4.750000 0.4834
D - e   5.083333 0.4546
D - f  -3.166667 0.6364
D - g  10.083333 0.1640
D - h -10.166667 0.1610
D - i   6.083333 0.3758
D - j   3.833333 0.5688
D - k   5.083333 0.4546
D - l   6.083333 0.3758
e - f  -8.250000 0.3538
e - g   5.000000 0.5650
```

5. Comparaciones no paramétricas

Las funciones para comparaciones múltiples no paramétricas en agricolae son: kruskal(), waerden.test(), Friedman() Median() y durbin.test(), Conover (1999).

La función de Kruskal() se usa para tamaño de muestras >2 de una poblaciones o datos procedentes de un experimento completamente al azar (poblaciones = tratamientos).

Las prueba post hov noparametrico (kruskal, friedman, durbin y waerden) Utilizan el critero de Fisher (LSD).

La función waerden.test(), similar a la de Kruskal-Wallis, utiliza una puntuación normal en

lugar de rangos como Kruskal.

La función friedman() se utiliza para las evaluaciones organolépticas de los diferentes productos, hechos por los jueces (cada juez evalúa todos los productos). También se puede utilizar para el análisis de los tratamientos del diseño de bloques completos al azar, donde la respuesta no puede ser tratada en un análisis de la varianza.

La función durbin.test() para el análisis de diseños de bloques incompletos balanceados es muy utilizado para las pruebas de muestreo, donde los jueces evalúan sólo una parte de los tratamientos.

La función Median.test() para el análisis de la distribución es aproximada con la distribución chi-cuadrada con grados de libertad igual al número de grupos menos uno. En cada comparación se forma una tabla de 2x2 (par de grupos) y el criterio de mayor o menor valor que la mediana de ambos, se aplica la prueba de ji cuadrado para el cálculo de la probabilidad de error que ambos son independientes. Este valor se compara con el nivel alfa para la formación de grupos.

Datos de Montgomery, Montgomery (2002). Incluido en el paquete agricolae

```
> data(corn)
> str(corn)

'data.frame':        34 obs. of  3 variables:
 $ method     : int  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 utilizarán los datos del paquete agricolae

5.1. Kruskal-Wallis

Hace la comparación múltiple con Kruskal-Wallis. Los parámetros por defecto son alpha = 0.05.

```
> str(kruskal)

function (y, trt, alpha = 0.05, p.adj = c("none", "holm", "hommel", "hochberg",
    "bonferroni", "BH", "BY", "fdr"), group = TRUE, main = NULL, console = FALSE)
```

Analysis

```
> data(corn)
> outKruskal<-with(corn,kruskal(observation,method,group=TRUE, main="corn",
+ console=TRUE))

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

Critical Value: 25.62884
Degrees of freedom: 3
```

```
> par(cex=0.7,mar=c(2,2,1,1))
> bar.group(outKruskal$groups,ylim=c(0,35),col=colors()[55],las=1)
```

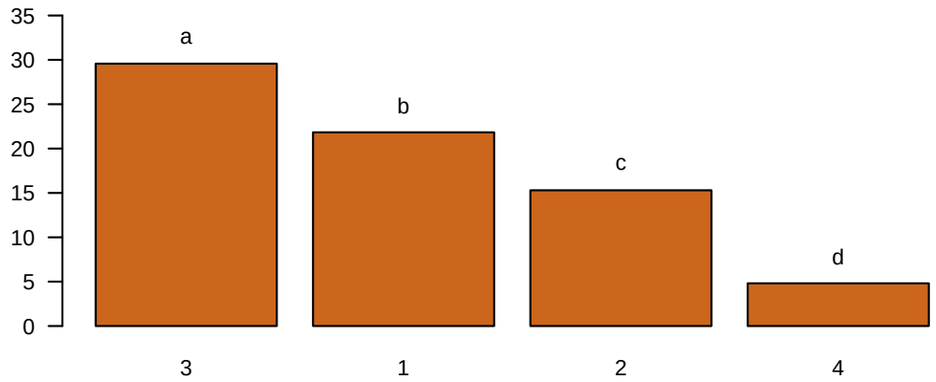

Figura 12: Comparación según Kruskal-Wallis

```
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

Post Hoc Analysis

t-Student: 2.042272
Alpha    : 0.05
Groups according to probability of treatment differences and alpha level.

Treatments with the same letter are not significantly different.

  observation groups
3    29.57143      a
1    21.83333      b
2    15.30000      c
4     4.81250      d
```

El objeto de salida tiene la misma estructura de las comparaciones ver Figura 12.

5.2. Friedman

Los datos consisten en b-bloques y k-variables aleatorias Xij, mutuamente independiente, i=1,...,b; j=1,...,k. La variable aleatoria X está en el bloque i y está asociada con el tratamiento j. La

comparación múltiple de la prueba de Friedman es con o sin empates. Un primer resultado es obtenido por friedman.test de R.

```
> str(friedman)

function (judge, trt, evaluation, alpha = 0.05, group = TRUE, main = NULL,
    console = FALSE)
```

Analysis

```
> data(grass)
> out<-with(grass,friedman(judge,trt, evaluation,alpha=0.05, group=FALSE,
+ main="Data of the book of Conover",console=TRUE))

Study: Data of the book of 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
Critical Value: 8.097345
P.Value Chisq: 0.04404214
F Value: 3.192198
P.Value F: 0.03621547

Post Hoc Analysis

Comparison between treatments
Sum of the ranks

        difference pvalue signif.    LCL   UCL
t1 - t2       14.5 0.0149       *   3.02 25.98
t1 - t3       13.5 0.0226       *   2.02 24.98
t1 - t4        4.0 0.4834          -7.48 15.48
t2 - t3       -1.0 0.8604         -12.48 10.48
t2 - t4      -10.5 0.0717       . -21.98  0.98
t3 - t4       -9.5 0.1017         -20.98  1.98
```

5.3. Waerden

Una prueba no paramétrica para varias muestras independientes, la funcion es waerden.test(). Ejemplo con los datos de camote de agricolae.

```
> str(waerden.test)

function (y, trt, alpha = 0.05, group = TRUE, main = NULL, console = FALSE)
```

Analysis

```
> data(sweetpotato)
> out<-with(sweetpotato,waerden.test(yield,virus,alpha=0.01,group=TRUE, console=TRUE))

Study: yield ~ virus
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 r
cc -0.2328353 0.3028832 3
fc -1.0601764 0.3467934 3
ff  0.6885684 0.7615582 3
oo  0.6044433 0.3742929 3

Post Hoc Analysis

Alpha: 0.01 ; DF Error: 8

Minimum Significant Difference: 1.322487

Treatments with the same letter are not significantly different.

Means of the normal score

        score groups
ff  0.6885684      a
oo  0.6044433      a
cc -0.2328353     ab
fc -1.0601764      b
```

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

```
> names(out)

[1] "statistics" "parameters" "means"      "comparison" "groups"
```

Véase out$comparison

```
> out<-with(sweetpotato,waerden.test(yield,virus,group=FALSE,console=TRUE))

Study: yield ~ virus
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 r
cc -0.2328353 0.3028832 3
fc -1.0601764 0.3467934 3
ff  0.6885684 0.7615582 3
oo  0.6044433 0.3742929 3

Post Hoc Analysis

Comparison between treatments
mean of the normal score

        difference pvalue signif.         LCL         UCL
cc - fc  0.8273411 0.0690       . -0.08154345  1.73622564
cc - ff -0.9214037 0.0476       * -1.83028827 -0.01251917
cc - oo -0.8372786 0.0664       . -1.74616316  0.07160593
fc - ff -1.7487448 0.0022      ** -2.65762936 -0.83986026
fc - oo -1.6646197 0.0029      ** -2.57350426 -0.75573516
ff - oo  0.0841251 0.8363         -0.82475944  0.99300965
```

5.4. Mediana

Una prueba no paramétrica para varias muestras independientes. La prueba mediana está diseñada para examinar si varias muestras provienen de poblaciones que tienen la misma mediana. Conover (1999). ver también Figura 13.

Análisis

```
> str(Median.test )

function (y, trt, alpha = 0.05, correct = TRUE, simulate.p.value = FALSE,
    group = TRUE, main = NULL, console = TRUE)

> data(sweetpotato )
> outMedian<-with(sweetpotato,Median.test(yield,virus,console=TRUE ))

The Median Test for yield ~ virus
```

```
Chi Square = 6.666667   DF = 3   P.Value 0.08331631
Median = 28.25

   Median r  Min  Max   Q25   Q75
cc   23.0 3 21.7 28.5 22.35 25.75
fc   13.1 3 10.6 14.9 11.85 14.00
ff   39.2 3 28.0 41.8 33.60 40.50
oo   38.2 3 32.1 40.4 35.15 39.30

Post Hoc Analysis

Groups according to probability of treatment differences and alpha level.

Treatments with the same letter are not significantly different.

   yield groups
ff  39.2      a
oo  38.2      a
cc  23.0      a
fc  13.1      b
```

```
> names(outMedian )
```

```
[1] "statistics" "parameters" "medians"    "comparison" "groups"
```

```
> outMedian$statistics
```

```
     Chisq Df    p.chisq Median
  6.666667  3 0.08331631  28.25
```

```
> outMedian$Medians
```

```
NULL
```

5.5. Durbin

Una comparación múltiple de la prueba de Durbin para los bloques incompletos balanceados para la evaluación sensorial o categórica. Forma grupos de acuerdo a los requeridos para el nivel de significación (alfa), por defecto es 0.05.

durbin.test(); ejemplo: Myles Hollander (p.311) fuente: W. Moore and C.I. Bliss.(1942)

```
> str(durbin.test)
```

```
function (judge, trt, evaluation, alpha = 0.05, group = TRUE, main = NULL,
    console = FALSE)
```

```
> par(mfrow=c(2,2),mar=c(3,3,1,1),cex=0.8)
> # Graphics
> bar.group(outMedian$groups,ylim=c(0,50))
> bar.group(outMedian$groups,xlim=c(0,50),horiz = TRUE)
> plot(outMedian)
> plot(outMedian,variation="IQR",horiz = TRUE)
```

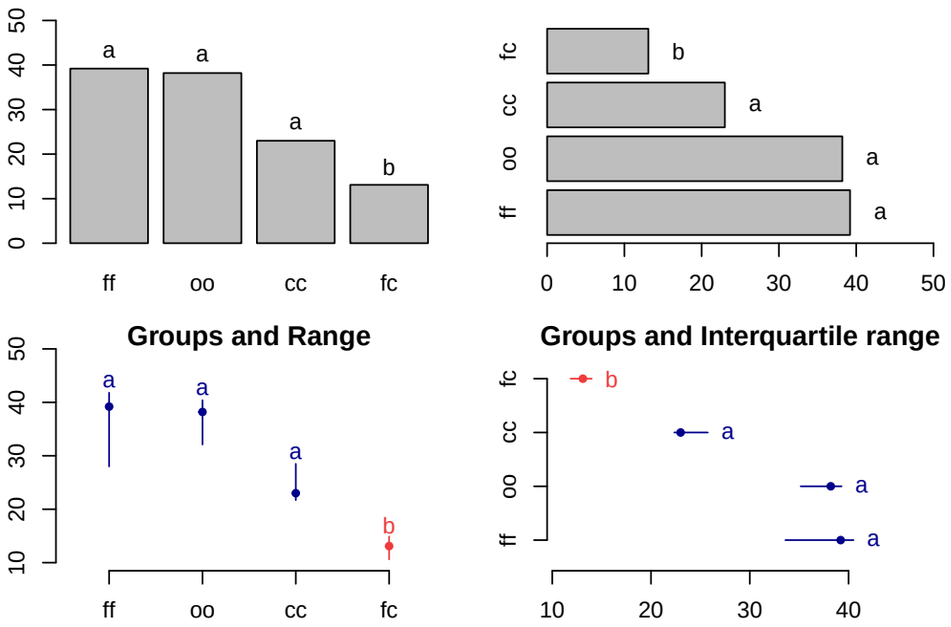

Figura 13: Agrupación de tratamientos y su variación, Método mediana

Análisis

```
> days <-gl(7,3)
> chemical<-c("A","B","D","A","C","E","C","D","G","A","F","G", "B","C","F",
+             "B","E","G","D","E","F")
> toxic<-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)
> out<-durbin.test(days,chemical,toxic,group=FALSE,console=TRUE,main="Logarithm of the

Study: Logarithm of the toxic dose
chemical,  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 signif.
A - B          0 1.0000
A - C         -4 0.1026
A - D          0 1.0000
A - E          0 1.0000
A - F         -3 0.2044
A - G          0 1.0000
B - C         -4 0.1026
B - D          0 1.0000
B - E          0 1.0000
B - F         -3 0.2044
B - G          0 1.0000
C - D          4 0.1026
C - E          4 0.1026
C - F          1 0.6574
C - G          4 0.1026
D - E          0 1.0000
D - F         -3 0.2044
D - G          0 1.0000
E - F         -3 0.2044
E - G          0 1.0000
F - G          3 0.2044
```

6. Gráficos de comparación multiple

Los resultados de una comparación se puede ver gráficamente con las funciones *bar.group*, *bar.err* and *diffograph*.

6.1. bar.group

La función presenta el diagrama de barras horizontal o vertical con las letras de los grupos de tratamientos. La función se aplica a todos los tratamientos de la comparación. Se requiere la funcion group = TRUE.

ejemplo:

```
> # model <-aov (yield ~ fertilizer, data = field)
> # out <-LSD.test (model, "fertilizer", group = TRUE)
> # bar.group (out $ group)
> str(bar.group)

function (x, horiz = FALSE, ...)
```

Ver Figura 13. La prueba de la mediana con la opcion group=TRUE (defecto) es usado en el ejercicio.

6.2. bar.err

Una función para el diagrama de barra horizontal o vertical, donde la variación del error se expresa en cada tratamientos. La función se aplica a todos los tratamientos de comparación. El objeto aplicado es calculado previamente con alguna funcion de comparacion multiple de tratamientos.

```
> # model <-aov (yield ~ fertilizer, data = field)
> # out <-LSD.test (model, "fertilizer", group = TRUE)
> # bar.err(out$means)
> str(bar.err)

function (x, variation = c("SE", "SD", "range", "IQR"), horiz = FALSE,
    bar = TRUE, ...)
```

variation
SE: Standard error
SD: standard deviation
range: max-min)

6.3. plot.group

Formación de grupos y variación de los tratamientos a comparar. Utiliza los objetos generados por un procedimiento de comparación como LSD (Fisher), Duncan, Tukey (HSD), el estudiante

```
> par(mfrow=c(2,2),mar=c(3,3,2,1),cex=0.7)
> c1<-colors()[480]; c2=colors()[65]
> bar.err(outHSD$means, variation="range",ylim=c(0,50),col=c1,las=1)
> bar.err(outHSD$means, variation="IQR",horiz=TRUE, xlim=c(0,50),col=c2,las=1)
> plot(outHSD, variation="range",las=1)
> plot(outHSD, horiz=TRUE, variation="SD",las=1)
```

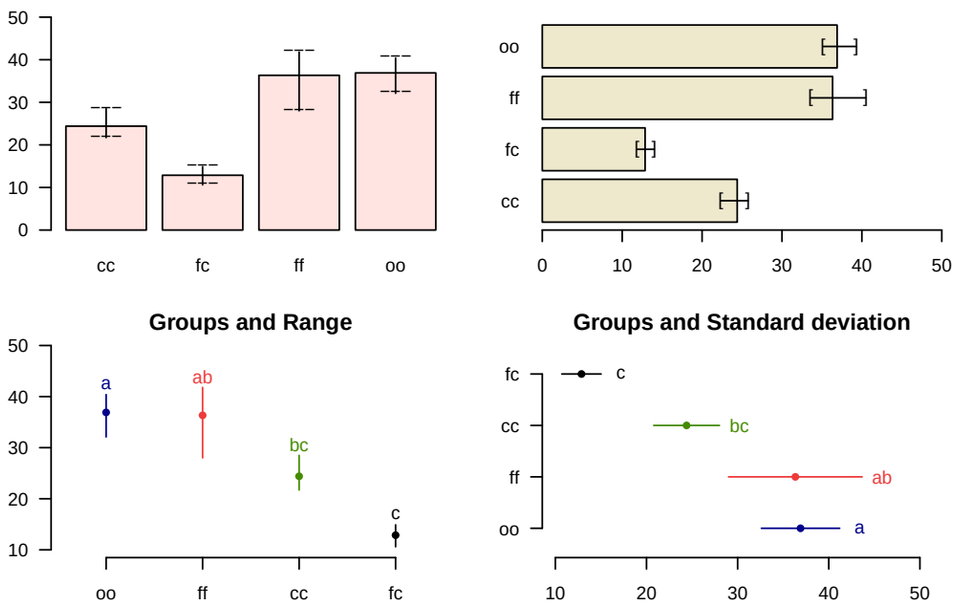

Figura 14: Comparación entre los tratamientos

Newman Keul (SNK), Scheffe, Waller-Duncan, Ryan, Einot y Gabriel y Welsch (REGW), Kruskal Wallis, Friedman, mediana, Waerden y otras pruebas como Durbin, DAU, BIB, PBIB. Los tipos de variación son rango (maximun y minimun), IQR (rango intercuartíl), SD (desviación estándar) y SE (error estándar), véase la Figura 15.

La función: plot.group () y sus argumentos x (salida de prueba de comparación), variación = c (rango", ÏQR", "SE", "SD"), horiz (TRUE o FALSE), xlim, ylim Y main son parámetros opcionales y otros parámetros de la función plot().

6.4. diffograph

Grafica de las diferencias entre tratamientos, cada linea tiene un punto de referencia que corresponde al par de tratamientos que se compara, la longitud del vector es equivalente a la magnitud del limite de confianza de la diferencia del par de tratamientos comparados, Hsu (1996), ver Figura 16

```
> # model : yield ~ virus
> # Important group=TRUE
> par(mfrow=c(1,2),mar=c(3,3,1,1),cex=0.8)
> x<-duncan.test(model, "virus", group=TRUE)
> plot(x,las=1,ylim=c(0,50))
> plot(x,variation="IQR",horiz=TRUE,las=1,xlim=c(0,50))
```

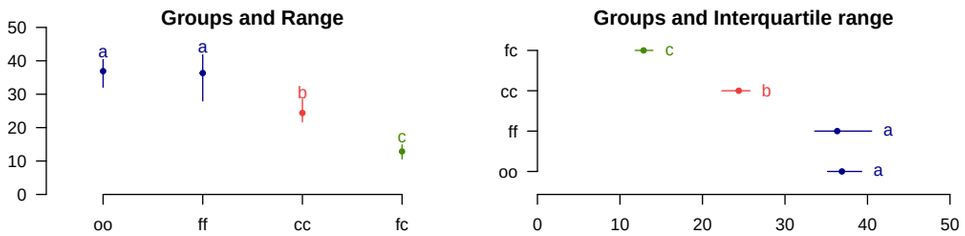

Figura 15: Agrupamiento de tratamientos y su variación, Método de Duncan

```
> # model : yield ~ virus
> # Important group=FALSE
> x<-HSD.test(model, "virus", group=FALSE)
> diffograph(x,cex.axis=0.9,xlab="Yield",ylab="Yield",cex=0.9)
```

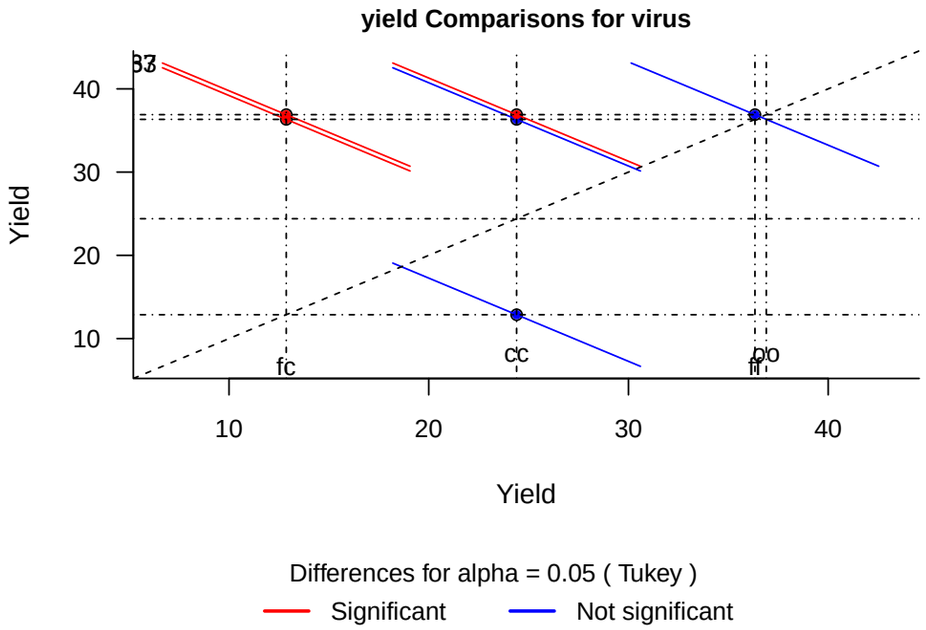

Figura 16: Mean-Mean de la comparacion entre tratamientos

7. Análisis de Estabilidad

En agricolae hay dos métodos para el estudio de la estabilidad y el modelo AMMI. Estos son: un modelo paramétrico para una selección simultánea en el rendimiento y la estabilidad "SHU-

KLA'S STABILITY VARIANCE AND KANG'S", Kang (1993) y un método no paramétrico de Haynes, con base en el rango de datos..

7.1. Estabilidad paramétrico

Utilice el modelo paramétrico, la función stability.par().

Preparar una tabla de datos, donde las filas y las columnas son los genotipos y los entornos, respectivamente. Los datos deben corresponder a producir medias oa otra variable medida. Determinar la varianza del error común para todos los ambientes y el número de repeticiones que se evaluó para cada genotipo. Si las repeticiones son diferentes, encontrar un promedio armónico que representará al conjunto. Por último, asignar un nombre a cada fila que representará al genotipo, Kang (1993). Consideraremos cinco ambientes en el siguiente ejemplo:

```
> options(digit=0)
> f <- system.file("external/dataStb.csv", package="agricolae")
> dataStb<-read.csv(f)
> stability.par(dataStb, rep=4, MSerror=2, alpha=0.1, main="Genotype",console=TRUE)

 INTERACTIVE PROGRAM FOR CALCULATING SHUKLA'S STABILITY VARIANCE AND KANG'S
                       YIELD - STABILITY (YSi) STATISTICS
 Genotype
 Environmental index  - covariate

 Analysis of Variance

               Df    Sum Sq  Mean Sq F value  Pr(>F)
Total         203 2964.1716
Genotypes      16  186.9082  11.6818    4.17  <0.001
Environments   11 2284.0116 207.6374  103.82  <0.001
Interaction   176  493.2518   2.8026     1.4   0.002
Heterogeneity  16   44.8576   2.8036       1   0.459
Residual      160  448.3942   2.8025     1.4  0.0028
Pooled Error  576                 2

 Genotype. Stability statistics

      Mean Sigma-square  . s-square  . Ecovalence
A 7.383333     2.474081 ns 2.449076 ns  25.826563
B 6.783333     1.600869 ns 1.434734 ns  17.351269
C 7.250000     0.567657 ns 0.633936 ns   7.323033
D 6.783333     2.611778 ns 2.134731 ns  27.163033
E 7.066667     1.862364 ns 2.047627 ns  19.889308
F 6.916667     3.575818 ns 3.951442  *  36.519896
G 7.808333     3.580929 ns 3.957319  *  36.569504
H 7.908333     2.723717 ns 2.118116 ns  28.249504
I 7.275000     4.248566  * 3.936130  *  43.049504
J 7.083333     2.273838 ns 2.506382 ns  23.883033
K 6.433333     2.560384 ns 2.551518 ns  26.664210
```

```
L 6.891667     1.558061 ns 1.732557 ns  16.935779
M 6.791667     3.483879 ns 3.275985 ns  35.627543
N 7.491667     5.164848 ** 4.875189 **  51.942837
O 7.658333     2.380202 ns 2.635025 ns  24.915386
P 6.425000     3.445414 ns 3.713885  *  35.254210
Q 6.158333     3.531232 ns 3.688232 ns  36.087151


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 7.383333   13        1       14 2.474081           0  14   +
B 6.783333    4       -1        3 1.600869           0   3
C 7.250000   11        1       12 0.567657           0  12   +
D 6.783333    4       -1        3 2.611778           0   3
E 7.066667    9        1       10 1.862364           0  10   +
F 6.916667    8       -1        7 3.575818          -2   5
G 7.808333   16        2       18 3.580929          -2  16   +
H 7.908333   17        2       19 2.723717           0  19   +
I 7.275000   12        1       13 4.248566          -4   9   +
J 7.083333   10        1       11 2.273838           0  11   +
K 6.433333    3       -2        1 2.560384           0   1
L 6.891667    7       -1        6 1.558061           0   6
M 6.791667    6       -1        5 3.483879          -2   3
N 7.491667   14        1       15 5.164848          -8   7
O 7.658333   15        2       17 2.380202           0  17   +
P 6.425000    2       -2        0 3.445414          -2  -2
Q 6.158333    1       -2       -1 3.531232          -2  -3

 Yield Mean: 7.065196
 YS    Mean: 7.705882
 LSD (0.05): 0.4755932
 - - - - - - - - - - -
 +   selected genotype
 ++  Reference: Kang, M. S. 1993. Simultaneous selection for yield
 and stability: Consequences for growers. Agron. J. 85:754-757.
```

Durante 17 genotipos, la identificación se hace por letras. Se calcula un error de la varianza de 2 y 4 repeticiones.

Analysis

```
> output <- stability.par(dataStb, rep=4, MSerror=2)
> names(output)

[1] "analysis"   "statistics" "stability"

> print(output$stability)
```

```
     Yield Rank Adj.rank Adjusted Stab.var Stab.rating YSi ...
A 7.383333   13        1       14 2.474081           0  14   +
B 6.783333    4       -1        3 1.600869           0   3
C 7.250000   11        1       12 0.567657           0  12   +
D 6.783333    4       -1        3 2.611778           0   3
E 7.066667    9        1       10 1.862364           0  10   +
F 6.916667    8       -1        7 3.575818          -2   5
G 7.808333   16        2       18 3.580929          -2  16   +
H 7.908333   17        2       19 2.723717           0  19   +
I 7.275000   12        1       13 4.248566          -4   9   +
J 7.083333   10        1       11 2.273838           0  11   +
K 6.433333    3       -2        1 2.560384           0   1
L 6.891667    7       -1        6 1.558061           0   6
M 6.791667    6       -1        5 3.483879          -2   3
N 7.491667   14        1       15 5.164848          -8   7
O 7.658333   15        2       17 2.380202           0  17   +
P 6.425000    2       -2        0 3.445414          -2  -2
Q 6.158333    1       -2       -1 3.531232          -2  -3
```

Los genotipos seleccionados son: A, C, E, G, H, I, J y O. Estos genotipos tienen un mayor rendimiento y una menor variación. De acuerdo con el análisis de varianza, la interacción es significativa.

Si por ejemplo hay un índice del medio ambiente, se puede añadir como una covariable. Para este caso, se incluye la altitud de las localidades.

```
> data5<-dataStb[,1:5]
> altitude<-c(1200, 1300, 800, 1600, 2400)
> stability <- stability.par(data5,rep=4,MSerror=2, cova=TRUE, name.cov= "altitude",
+ file.cov=altitude)
```

7.2. Estabilidad no paramétrico

Para la estabilidad no paramétrica, la función en agricolae es stability.nonpar(). Los nombres de los genotipos deben ser incluidos en la primera columna, y en las otras columnas, la respuesta de los entornos, Haynes and Lambert and Christ and Weingartner and Douches and Backlund and Secor and Fry and Stevenson (1998). **Analysis**

```
> data <- data.frame(name=row.names(dataStb), dataStb)
> output<-stability.nonpar(data, "YIELD", ranking=TRUE)
> names(output)

[1] "ranking"    "statistics"

> output$statistics

      MEAN      es1 es2       vs1      vs2  chi.ind  chi.sum
1 7.065196 5.647059  24 0.7247475 46.72727 8.843605 27.58711
```

7.3. AMMI

El modelo AMMI utiliza el biplot construida a través de los principales componentes generados por la interacción ambiente - genotipo. Si existe esa interacción , el porcentaje de los dos componentes principales explicaría más que el 50 % de la variación total ; en tal caso , la biplot sería una buena alternativa para estudiar la interacción ambiente - genotipo, Crossa (1990).

Los datos para AMMI deben provenir de experimentos similares llevados a cabo en diferentes entornos . La homogeneidad de la varianza del error experimental , producido en los diferentes ambientes, se requiere. El análisis se realiza mediante la combinación de los experimentos.

Los datos pueden ser organizados en columnas, así: el medio ambiente, el genotipo, la repetición y variable.

Los datos también pueden ser los promedios de los genotipos en cada ambiente , pero es necesario tener en cuenta una media armónica de las repeticiones y una varianza común del error. Los datos deben ser organizados en columnas : el medio ambiente , el genotipo y variable.

Al realizar AMMI , esto genera los gráficos Biplot ; Vea las Figuras 17, 18.

Para la aplicación, tenemos en cuenta los datos utilizados en el ejemplo de estabilidad paramétrica:

Estructura de AMMI

```
> str(AMMI)

function (ENV, GEN, REP, Y, MSE = 0, console = FALSE, PC = FALSE)
```

Estructura de plot.AMMI, plot()

```
> str(plot.AMMI )

function (x, first = 1, second = 2, third = 3, type = 1, number = FALSE,
    gcol = NULL, ecol = NULL, icol = NULL, angle = 25, lwd = 1.8, length = 0.1,
    xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL, ...)
```

type: 1=biplot, 2= triplot 3=influencia del genotipo

```
> data(plrv)
> model<-with(plrv,AMMI(Locality, Genotype, Rep, Yield, console=FALSE))
> names(model)

[1] "ANOVA"    "genXenv"  "analysis" "means"    "biplot"   "PC"

> model$ANOVA

Analysis of Variance Table

Response: Y
           Df Sum Sq Mean Sq  F value    Pr(>F)
ENV         5 122284 24456.9 257.0382  9.08e-12 ***
```

```
> par(mfrow=c(1,2),cex=0.8,mar=c(4,4,1,1))
> plot(model,type=1)
> plot(model,type=2)
```

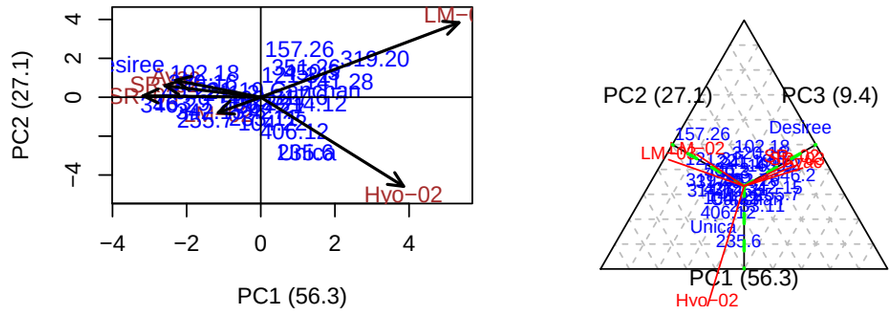

Figura 17: Biplot y Triplot

```
REP(ENV)    12   1142    95.1   2.5694  0.002889 **
GEN         27  17533   649.4  17.5359 < 2.2e-16 ***
ENV:GEN    135  23762   176.0   4.7531 < 2.2e-16 ***
Residuals  324  11998    37.0
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> model$analysis
```

```
     percent  acum Df     Sum.Sq   Mean.Sq F.value   Pr.F
PC1     56.3  56.3 31 13368.5954 431.24501   11.65 0.0000
PC2     27.1  83.3 29  6427.5799 221.64069    5.99 0.0000
PC3      9.4  92.7 27  2241.9398  83.03481    2.24 0.0005
PC4      4.3  97.1 25  1027.5785  41.10314    1.11 0.3286
PC5      2.9 100.0 23   696.1012  30.26527    0.82 0.7059
```

```
> pc <- model$analysis[, 1]
> pc12<-sum(pc[1:2])
> pc123<-sum(pc[1:3])

> require(klaR)
> require(spdep)
```

En este caso, la interacción es significativa. Los dos primeros componentes explican 83.4 %; entonces el biplot puede proporcionar información sobre la interacción genotipo-ambiente. Con el gráfico triespacial, se explicaría 92.8 %.

7.4. indice de estabilidad de AMMI y rendimiento

Halla los indices de estabilidad de AMMI (ASV) y rendimiento (YSI), Sabaghnia and Sabaghpour and Dehghani (2008); Purchase (1997).

```
> par(mar=c(4,4,1,1),cex=0.5)
> plot(model,type=3,number=TRUE)
```

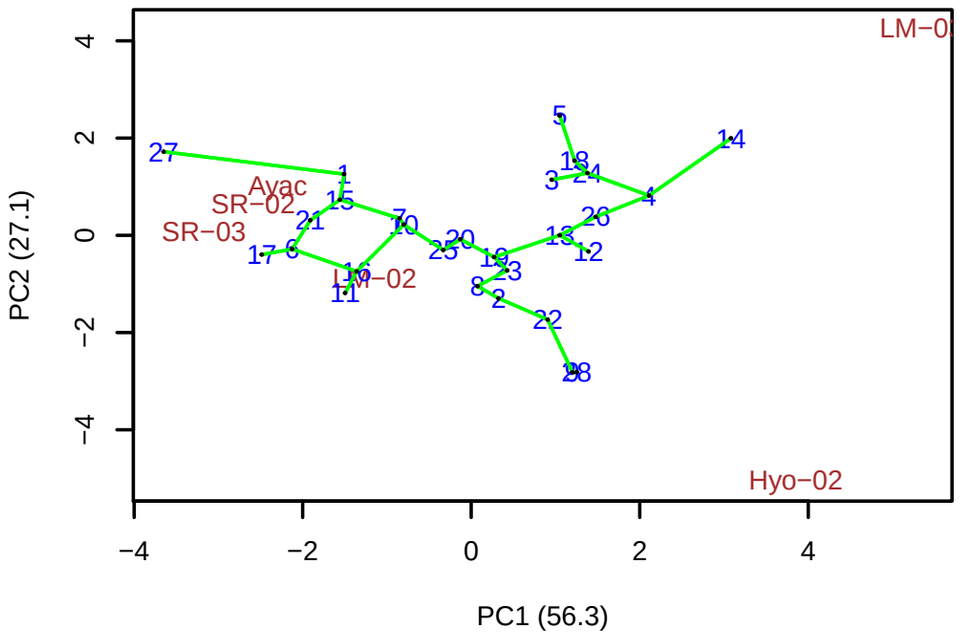

Figura 18: Influencia del genotipo

```
> data(plrv)
> with(plrv,model<- AMMI(Locality, Genotype, Rep, Yield, console=FALSE))
> index<-index.AMMI(model)
> # Crops with improved stability according AMMI.
> print(index[order(index[,3]),])
```

	ASV	YSI	rASV	rYSI	means
402.7	0.2026430	20	1	19	27.47748
506.2	0.5646275	13	2	11	33.26623
364.21	0.5966506	13	3	10	34.05974
427.7	0.9507170	11	4	7	36.19020
233.11	1.0521529	22	5	17	28.66655
241.2	1.1739456	28	6	22	26.34039
221.19	1.2740344	33	7	26	22.98480
104.22	1.3792025	21	8	13	31.28887
317.6	1.5167528	18	9	9	35.32583
121.31	1.7912464	25	10	15	30.10174
314.12	2.0368354	29	11	18	28.17335
342.15	2.0954103	36	12	24	26.01336
Canchan	2.1652861	33	13	20	27.00126
406.12	2.1722949	26	14	12	32.68323
351.26	2.3436592	23	15	8	36.11581
320.16	2.3623790	37	16	21	26.34808
450.3	2.3663500	23	17	6	36.19602

```
255.7   2.4615460  32   18   14 30.58975
102.18  2.5131813  42   19   23 26.31947
405.2   2.7709324  36   20   16 28.98663
157.26  2.8907699  26   21    5 36.95181
163.9   3.0764673  49   22   27 21.41747
141.28  3.1531170  24   23    1 39.75624
235.6   3.3065468  28   24    4 38.63477
Unica   3.3470545  27   25    2 39.10400
346.2   3.6050812  51   26   25 23.84175
319.20  4.8741897  30   27    3 38.75767
Desiree 5.5374138  56   28   28 16.15569
```

```
> # Crops with better response and improved stability according AMMI.
> print(index[order(index[,4]),])
```

```
              ASV YSI rASV rYSI    means
141.28  3.1531170  24   23    1 39.75624
Unica   3.3470545  27   25    2 39.10400
319.20  4.8741897  30   27    3 38.75767
235.6   3.3065468  28   24    4 38.63477
157.26  2.8907699  26   21    5 36.95181
450.3   2.3663500  23   17    6 36.19602
427.7   0.9507170  11    4    7 36.19020
351.26  2.3436592  23   15    8 36.11581
317.6   1.5167528  18    9    9 35.32583
364.21  0.5966506  13    3   10 34.05974
506.2   0.5646275  13    2   11 33.26623
406.12  2.1722949  26   14   12 32.68323
104.22  1.3792025  21    8   13 31.28887
255.7   2.4615460  32   18   14 30.58975
121.31  1.7912464  25   10   15 30.10174
405.2   2.7709324  36   20   16 28.98663
233.11  1.0521529  22    5   17 28.66655
314.12  2.0368354  29   11   18 28.17335
402.7   0.2026430  20    1   19 27.47748
Canchan 2.1652861  33   13   20 27.00126
320.16  2.3623790  37   16   21 26.34808
241.2   1.1739456  28    6   22 26.34039
102.18  2.5131813  42   19   23 26.31947
342.15  2.0954103  36   12   24 26.01336
346.2   3.6050812  51   26   25 23.84175
221.19  1.2740344  33    7   26 22.98480
163.9   3.0764673  49   22   27 21.41747
Desiree 5.5374138  56   28   28 16.15569
```

8. Funciones especiales

```
> par(cex=0.6,mar=c(3,3,2,1))
> data(pamCIP)
> rownames(pamCIP)<-substr(rownames(pamCIP),1,6)
> output<-consensus(pamCIP,distance="binary", method="complete", nboot=5)

Duplicates: 18
New data  : 25 Records

Consensus hclust

Method distance: binary
Method cluster : complete
rows and cols  : 25 107
n-bootstrap    : 5
Run time       : 2.192126 secs
```

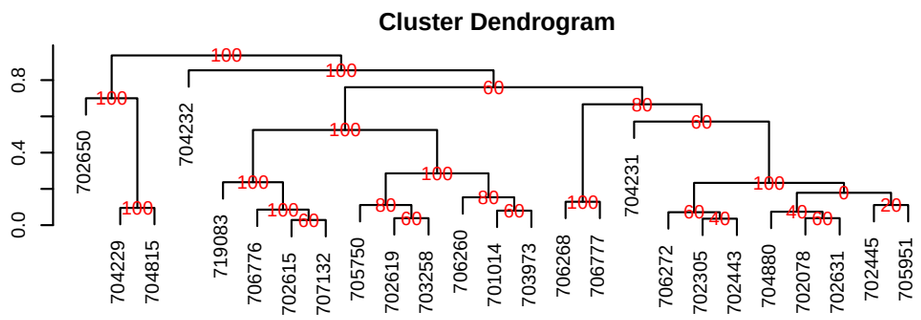

Figura 19: Dendrograma, la producción por consenso

8.1. Consenso de dendrograma

El consenso es el grado o la similitud de los vértices de un árbol sobre sus ramas del dendrograma construido. La función para aplicar es consensus().

Los datos corresponden a una tabla, con el nombre de los individuos y las variables en las filas y columnas respectivamente. Para la demostración, se utilizará los datos *pamCIP* de agricolae, que corresponden a los 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, véase la Figura 19.

Cuando el dendrograma es compleja, es conveniente para extraer parte de él con la función hcut(), consulte la Figura 20. El objeto salida contiene información sobre el proceso:

```
> names(output)

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

Esto significa que podemos conocer los duplicados, reconstruir el diagrama de árbol y mantener

```
> par(cex=0.6,mar=c(3,3,1.5,1))
> out1<- hcut(output,h=0.4,group=8,type="t",edgePar = list(lty=1:2, col=colors()[c(42,
+ main="group 8" ,col.text="blue",cex.text=1,las=1)
```

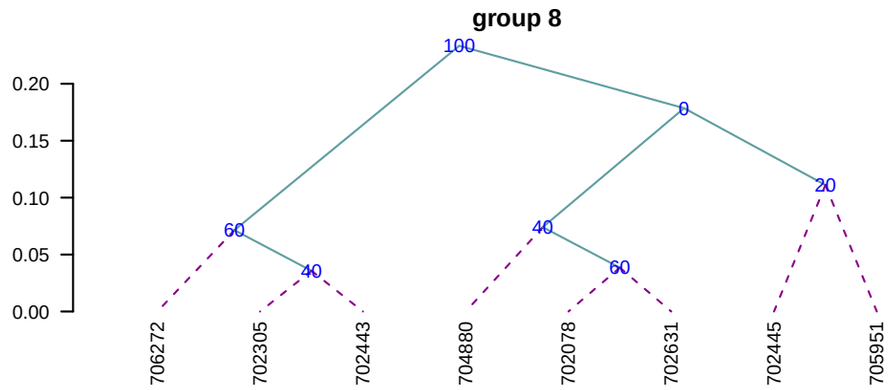

Figura 20: Dendrograma, producción por hcut()

```
> #
> # Repitiendo dendrogramas
> #
> par(mar=c(0,2,1,1),cex=0.7)
> dend<-output$dendrogram
> data<-output$table.dend
> plot(dend)
> text(data[,3],data[,4],data[,5])
```

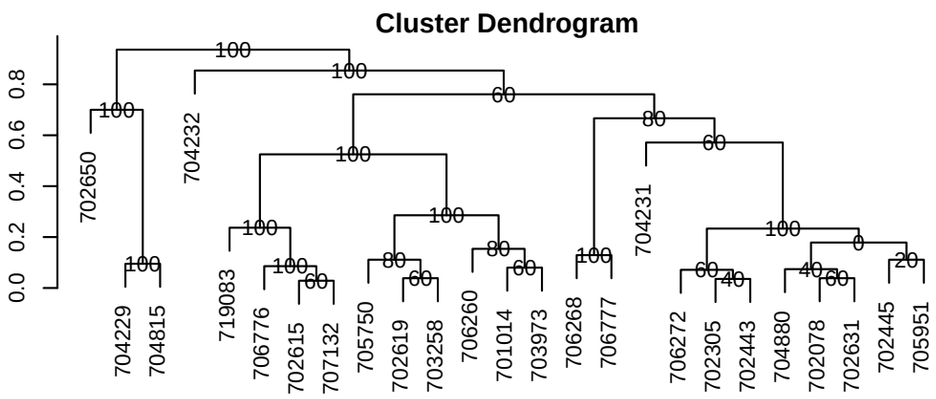

Figura 21: dendrograma Clásico

la interacción, véase Figura 21. Tambien se puede construir un dendrograma clásico, véase Figura 22.

```
> head(output$table.dend)
```

```
   X1  X2 xaxis     height percentage groups
1  -6 -24  7.50 0.02857143         60   6-24
```

```
> #
> # Construir un dendrograma clásico
> #
> par(mar=c(3,3,1,1),cex=0.6)
> dend<-as.dendrogram(output$dendrogram)
> plot(dend,type="r",edgePar = list(lty=1:2, col=colors()[c(42,84)]) ,las=1)
> text(data[,3],data[,4],data[,5],col="blue")
```

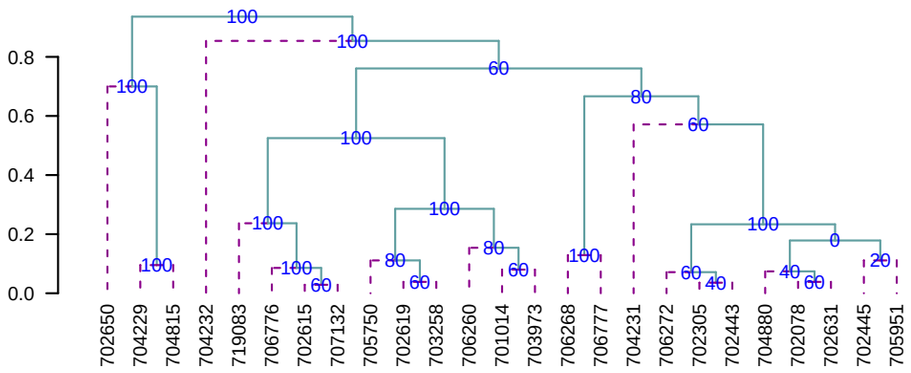

Figura 22: dendrograma Clasico

```
2  -3  -4 19.50 0.03571429          40    3-4
3  -2  -8 22.50 0.03846154          60    2-8
4  -7 -10 10.50 0.03846154          60   7-10
5 -21   2 18.75 0.07142857          60 3-4-21
6 -16   3 21.75 0.07407407          40 2-8-16
```

8.2. Montecarlo

Es un método para generar números aleatorios de una distribución desconocida. Se utiliza un conjunto de datos y, a través del comportamiento acumulado de su frecuencia relativa, genera los posibles valores aleatorios que siguen la distribución de datos. Estos nuevos números se utilizan en un proceso de simulación.

La densidad de probabilidad de los datos originales y simulados se puede comparar, véase la Figura 23.

```
> data(soil)
> set.seed(9473)
> simulated <- montecarlo(soil$pH,1000)
> h<-graph.freq(simulated,nclass=7,plot=FALSE)
```

Datos de 1000 se han generado, siendo la tabla de frecuencias:

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

```
  Lower Upper Main Frequency Percentage   CF   CPF
1  1.40  2.67 2.04        22        2.2   22   2.2
```

```
> par(mar=c(2,0,2,1),cex=0.6)
> plot(density(soil$pH),axes=FALSE,main="pH density of the soil\ncon Ralstonia",xlab="
> lines(density(simulated), col="blue", lty=4,lwd=4)
> axis(1,0:12)
> legend("topright",c("Original","Simulated"),lty=c(1,4),col=c("black", "blue"), lwd=4
```

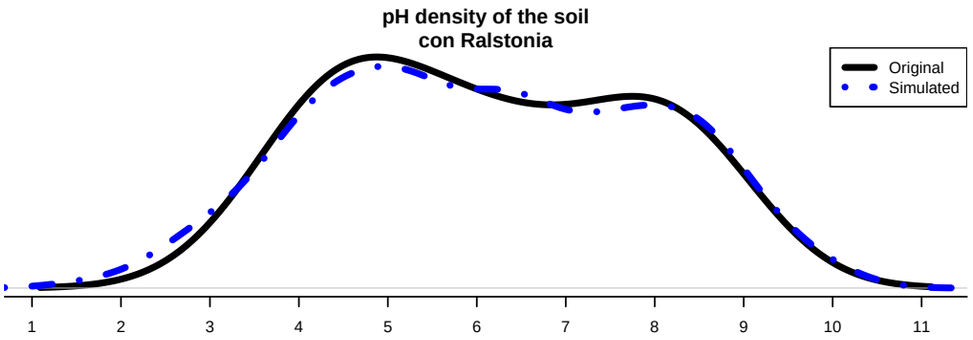

Figura 23: Distribución de la simulación y los datos originales

```
2  2.67  3.94 3.30       101       10.1  123  12.3
3  3.94  5.21 4.58       235       23.5  358  35.8
4  5.21  6.48 5.85       216       21.6  574  57.4
5  6.48  7.75 7.12       183       18.3  757  75.7
6  7.75  9.02 8.38       187       18.7  944  94.4
7  9.02 10.29 9.65        56        5.6 1000 100.0
```

Algunas estadísticas, datos originales:

```
> summary(soil$pH)

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

Algunas estadísticas, montecarlo simular datos:

```
> summary(simulated)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.462   4.659   6.090   6.124   7.703  10.287
```

El objeto h es un histograma que representa la distribución de la población. Con la función montecarlo (), podemos generar un nuevo conjunto de datos representativos de esta población. 1000 datos fueron simulados, vea la Figura 24.

new<-montecarlo(h, k=1000)

8.3. Re-muestreo en el modelo lineal

Se utiliza el método de permutación para el cálculo de las probabilidades de las fuentes de variación de ANOVA de acuerdo con el modelo de regresión lineal o el diseño utilizado. El

```
> par(mfrow=c(1,2),cex=0.8,mar=c(2,3,1,1))
> h<-graph.freq(soil$pH,nclass=4,frequency=3,plot=FALSE)
> breaks<-h$breaks
> r<-montecarlo(h, k=1000)
> plot(h,breaks=breaks,frequency = 3,ylim=c(0,0.5))
> text(6,0.4,"Population\n100 obs.")
> graph.freq(r,breaks,frequency = 3,ylim=c(0,0.5))
> lines(density(r),col="blue")
> text(6,0.4,"Montecarlo\n1000 obs.")
```

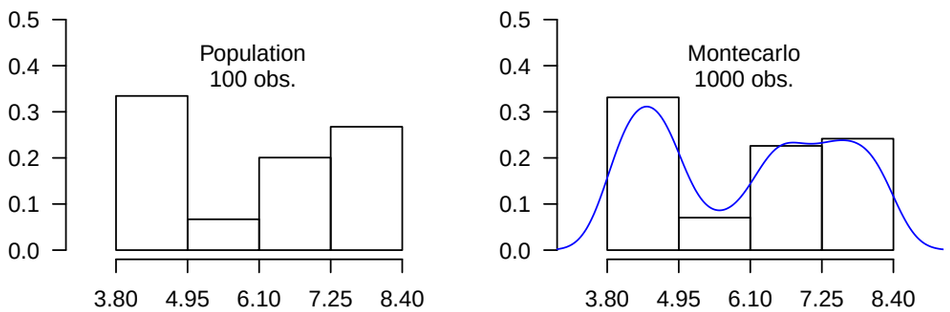

Figura 24: Histograma de los datos originales y con datos simulados

principio es que la respuesta Y no depende de los promedios propuestos en el modelo; por lo tanto, los valores de Y pueden ser permutación y muchas estimaciones de los modelos pueden ser construidos. Sobre la base de los patrones de las variables aleatorias de los elementos objeto de estudio, la probabilidad se calcula con el fin de medir la significación. Para un análisis de la varianza, los datos deben ser preparadas de manera similar. 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(model, potato, k=100)
```

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

8.4. Simulación en el modelo lineal

Bajo el supuesto de normalidad, la función genera errores experimentales seudo según el modelo propuesto, y determina la proporción de resultados válidos de acuerdo con el análisis de la varianza encontrada.

La función es: simulation.model(). Los datos se preparan en una tabla, de manera similar a un análisis de varianza.

Teniendo en cuenta el ejemplo propuesto en el procedimiento anterior:

```
> model <- simulation.model(model, potato, k=100)
```

La validación se conoce el porcentaje de resultados de toma igual al resultado de la decisión de ANOVA. Por lo tanto, 67,5 % de los resultados simulados de la interacción variedad * fecha dio el mismo resultado de la aceptación o rechazo obtenido en el ANOVA.

8.5. Análisis Path

Se corresponde con el método de análisis de ruta. Los datos corresponden a las matrices de correlación de variables dependientes e independientes (XY) y entre 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")
> output<-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

> output

$Coeff
          X1        X2
X1 0.3333333 0.2666667
X2 0.1666667 0.5333333

$Residual
[1] 0.4266667
```

8.6. Linea X Probador

Se corresponde a un análisis de cruzamiento de un diseño genético. Los datos deben ser organizados en una tabla. Sólo se requieren cuatro columnas: la repetición, las mujeres, los hombres, y la respuesta. En caso de que corresponde a los progenitores, las mujeres o los hombres de campo sólo se llenarán con el correspondiente. Consulte los datos de 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
 ...
 140           1 Achirana   <NA> 3.35
 ...
 215           3     <NA> TPS-67 2.91
```

donde <NA>es vacio.

Si se trata de una progenie, es una hembra(Female) y macho (Male) Si es progenitor, solamente es hembra (Female) o (Male).

El ejemplo siguiente corresponde a los datos de la localidad 2:

24 progenies 8 hembras 3 machos 3 repeticiones

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

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

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

Testers Effects:
TPS-13 TPS-67  TS-15
 0.087  0.046 -0.132

SCA Effects:
===========
          Testers
Lines      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
```

```
> par(mar=c(3,3,4,1),cex=0.6)
> data(rice)
> table<-index.smith(rice,pch=19, col="blue",
+  main="Interaction between the CV and the parcel size",type="l",xlab="Size")
```

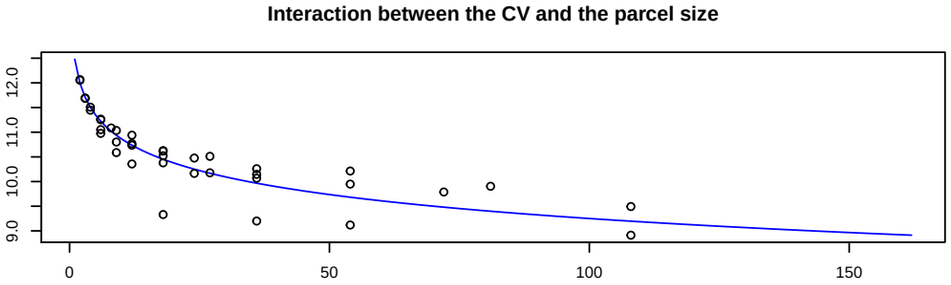

Figura 25: Curva de ajuste para el tamaño óptimo de parcela

```
F = 0, Adittive genetic variance: 0.01546921
F = 1, Adittive 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
```

8.7. La uniformidad del suelo

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

Para la prueba, vamos a utilizar el archivo de arroz. El gráfico es un resultado con el ajuste de un modelo para el tamaño de paquete y el coeficiente de variación, véase la Figura 25.

```
> uniformity <- data.frame(table$uniformity)
> uniformity
```

```
   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
```

```
6     4     1     4   162 7047.717 11.4
7     4     2     2   162 7123.973 11.5
8     6     1     6   108 6571.278 11.1
9     6     2     3   108 6826.889 11.3
10    6     3     2   108 6812.627 11.3
11    6     6     1   108 6480.927 11.0
12    8     2     4    81 6611.042 11.1
13    9     1     9    72 6276.257 10.8
14    9     3     3    72 6550.445 11.0
15    9     9     1    72 6029.794 10.6
16   12     1    12    54 5769.816 10.4
17   12     2     6    54 6239.270 10.8
18   12     3     4    54 6437.818 10.9
19   12     6     2    54 6199.754 10.7
20   18     1    18    36 4683.748  9.3
21   18     2     9    36 6063.466 10.6
22   18     3     6    36 6074.789 10.6
23   18     6     3    36 5954.826 10.5
24   18     9     2    36 5794.902 10.4
25   24     2    12    27 5558.070 10.2
26   24     6     4    27 5904.217 10.5
27   27     3     9    24 5943.106 10.5
28   27     9     3    24 5571.450 10.2
29   36     2    18    18 4552.791  9.2
30   36     3    12    18 5452.777 10.1
31   36     6     6    18 5662.243 10.3
32   36     9     4    18 5537.279 10.1
33   54     3    18    12 4473.072  9.1
34   54     6     9    12 5611.097 10.2
35   54     9     6    12 5323.471  9.9
36   72     6    12     9 5153.500  9.8
37   81     9     9     8 5276.787  9.9
38  108     6    18     6 4272.874  8.9
39  108     9    12     6 4847.973  9.5
40  162     9    18     4 4009.765  8.6
```

8.8. Límites de confianza de los índices de biodiversidad

Los índices de biodiversidad son ampliamente utilizados para la medición de la presencia de los seres vivos en una zona ecológica. Muchos programas indican su valor. La función de agricolae es también para mostrar los intervalos de confianza, que pueden ser utilizados para una comparación estadística. Utilice el procedimiento de arranque. Los datos se organizan en una tabla; la especie se colocan en una columna; y en otra el número de individuos. Los índices que se pueden calcular con la función index.bio() de agricolae son: "Margalef", "Simpson.Dom", "Simpson.Div", "Berger.Parker", "McIntosh", and "Shannon."

En el siguiente ejemplo, vamos a utilizar los datos obtenidos en la localidad de Paracsho, distrito de Huasahuasi, provincia de Tarma, en el departamento de Junín.

La evaluación se llevó a cabo en las parcelas, el 17 de noviembre de 2005, sin la aplicación de insecticidas. Los especímenes fueron contados los siguientes:

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

```
         Orden        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                  3
```

El índice de Shannon es:

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

```
Method: Shannon

The index: 3.52304

95 percent confidence interval:
 3.089167 ; 4.26084
```

8.9. Correlación

La función correlation() de agricolae calcula las correlaciones a través de los métodos de Pearson, Spearman y Kendall para vectores y/o matrices. Si son dos vectores, la prueba se lleva a cabo para una o dos colas; si se trata de una matriz, determina las probabilidades de una diferencia si es mayor o menor que cero.

Para su aplicación, tenga en cuenta los datos del suelo: data(soil)

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

```
$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    1.000000000 0.0525330 0.004797027
EC    0.052532997 1.0000000 0.294159813
CaCO3 0.004797027 0.2941598 1.000000000

$n.obs
[1] 13
```

```
> with(soil,correlation(pH,soil[,3:4],method="pearson"))
```

```
$correlation
     EC CaCO3
pH 0.55  0.73

$pvalue
       EC  CaCO3
pH 0.0525 0.0048

$n.obs
[1] 13
```

```
> with(soil,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
```

8.10. Otras funciones

Otras funciones que facilitan las operaciones de datos y estadísticas:

tapply.stat()

Obtiene un cálculo funcional de las variables agrupadas por factores de estudio.

Tabla de factores y variables

Aplicación con los datos de agricolae

```
> data(RioChillon)
> with(RioChillon$babies,tapply.stat(yield,farmer,function(x) max(x)-min(x)))
```

```
            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 del rendimiento de los agricultores.

La función "Tapply"se puede utilizar directamente o con la función.

Si A es una tabla con columnas de 1,2 y 3 (categorí) y las columnas 5,6 y 7 las variables, los siguientes procedimientos son válidos en agricolae:
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 del experimento

Si "modelo.es el objeto resultante de un análisis de la varianza de la función AOV o LM() de R, entonces la función de cv.model() calcula del coeficiente de variacion.

```
> data(sweetpotato)
> model <- model<-aov(yield ~ virus, data=sweetpotato)
> cv.model(model)

[1] 17.1666
```

Asimetría curtosis

Los resultados de la asimetría y la curtosis, obtenidos por *agricolae*, son iguales a los obtenidos por SAS, Minitab, SPSS, InfoStat y Excel.

If x represents a data set:

```
> 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
```

```
> par(mar=c(3,3,4,1),cex=0.6)
> plot(K,y[,1],type="l",col="blue",ylab="waller",bty="l")
> lines(K,y[,2],type="l",col="brown",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","brown","green"),lty=c(1,8,20),
+        lwd=2,title="Fc")
> title(main="Waller en funci\'on de K")
```

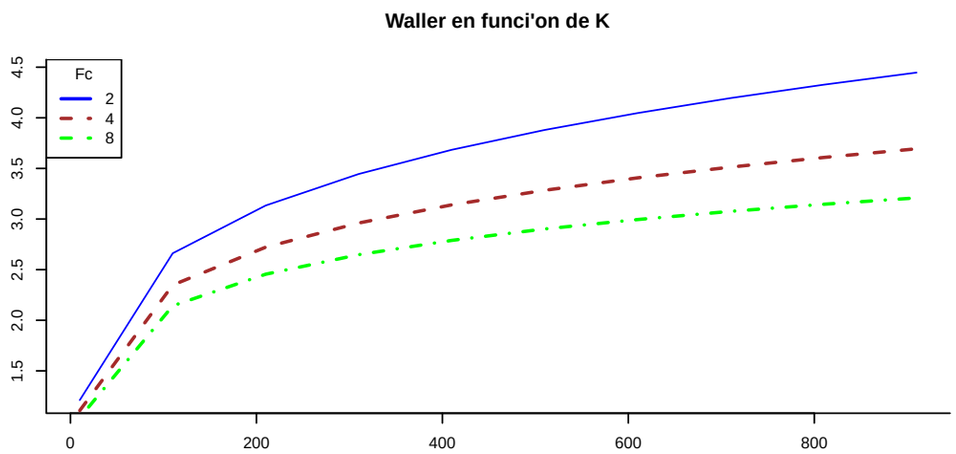

Figura 26: Función de Waller para diferentes valores de los parámetros K y Fc

Valor tabular de Waller-Duncan

La función Waller() determina el valor tabular de Waller-Duncan. Para el cálculo se requiere el valor F calculada a partir del análisis de la varianza del factor de estudio, con sus grados de libertad y la estimación de la varianza del error experimental. El valor K es parámetro de la función y es la relación entre los dos tipos de errores (I y II). Para aplicar, se le asigna un valor asociado con el nivel alfa, por ejemplo, cuando alfa es 0,10, k=50; para 0,05, K = 100; y para alfa 0.01, K = 500. K puede tomar cualquier valor.

La Figura 26 ilustra la función para diferentes valores de K con grados de libertad de 5 para el numerador y el 15 para el denominador y los valores de F calculado, igual 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)
```

Generando la tabla Waller-Duncan

```
> K<-100
> Fc<-1.2
```

```
> q<-c(seq(6,20,1),30,40,100)
> f<-c(seq(4,20,2),24,30)
> n<-length(q)
> m<-length(f)
> W.D <-rep(0,n*m)
> dim(W.D)<-c(n,m)
> for (i in 1:n) {
+ for (j in 1:m) {
+ W.D[i,j]<-waller(K, q[i], f[j], Fc)
+ }}
> W.D<-round(W.D,2)
> dimnames(W.D)<-list(q,f)
> cat("tabla: Waller Duncan k=100, F=1.2")

tabla: Waller Duncan k=100, F=1.2

> print(W.D)

       4    6    8   10   12   14   16   18   20   24   30
6   2.85 2.87 2.88 2.89 2.89 2.89 2.89 2.88 2.88 2.88 2.88
7   2.85 2.89 2.92 2.93 2.94 2.94 2.94 2.94 2.94 2.94 2.94
8   2.85 2.91 2.94 2.96 2.97 2.98 2.99 2.99 2.99 3.00 3.00
9   2.85 2.92 2.96 2.99 3.01 3.02 3.03 3.03 3.04 3.04 3.05
10  2.85 2.93 2.98 3.01 3.04 3.05 3.06 3.07 3.08 3.09 3.10
11  2.85 2.94 3.00 3.04 3.06 3.08 3.09 3.10 3.11 3.12 3.14
12  2.85 2.95 3.01 3.05 3.08 3.10 3.12 3.13 3.14 3.16 3.17
13  2.85 2.96 3.02 3.07 3.10 3.12 3.14 3.16 3.17 3.19 3.20
14  2.85 2.96 3.03 3.08 3.12 3.14 3.16 3.18 3.19 3.21 3.23
15  2.85 2.97 3.04 3.10 3.13 3.16 3.18 3.20 3.21 3.24 3.26
16  2.85 2.97 3.05 3.11 3.15 3.18 3.20 3.22 3.24 3.26 3.29
17  2.85 2.98 3.06 3.12 3.16 3.19 3.22 3.24 3.25 3.28 3.31
18  2.85 2.98 3.07 3.13 3.17 3.21 3.23 3.25 3.27 3.30 3.33
19  2.85 2.98 3.07 3.13 3.18 3.22 3.25 3.27 3.29 3.32 3.35
20  2.85 2.99 3.08 3.14 3.19 3.23 3.26 3.28 3.30 3.33 3.37
30  2.85 3.01 3.11 3.19 3.26 3.31 3.35 3.38 3.41 3.45 3.50
40  2.85 3.02 3.13 3.22 3.29 3.35 3.39 3.43 3.47 3.52 3.58
100 2.85 3.04 3.17 3.28 3.36 3.44 3.50 3.55 3.59 3.67 3.76
```

AUDPC

El área bajo la curva de progreso de la enfermedad (AUDPC), véase la Figura 27, calcula el progreso absoluto y relativo de la enfermedad. Es necesario medir la enfermedad en términos porcentuales durante varias fechas, de preferencia de manera equidistante.

AUDPS

The Area Under the Disease Progress Stairs (AUDPS), ver Figura 28. A better estimate of disease progress is the area under the disease progress stairs (AUDPS). The AUDPS approach improves the estimation of disease progress by giving a weight closer to optimal to the first and last observations..

```
> par(mar=c(3,3,4,1),cex=0.6)
> days<-c(7,14,21,28,35,42)
> evaluation<-data.frame(E1=10,E2=40,E3=50,E4=70,E5=80,E6=90)
> par(mar=c(4,4,1,1))
> plot(days, evaluation,type="h",ylim=c(0,100),axes=FALSE,col= colors()[42],
+      xlab="Dias", ylab="Evaluacion")
> lines(days,evaluation,col= colors()[42])
> axis(1,days)
> axis(2,seq(0,100,20),las=2)
> rect(7,0,42,100)
> audpc(evaluation,days)

[1] 2030

> audpc(evaluation,days,"relative")

[1] 0.58

> text(15,80,"Audpc Absoluto = 2030")
> text(15,70,"Audpc Relativo = 0.58")
```

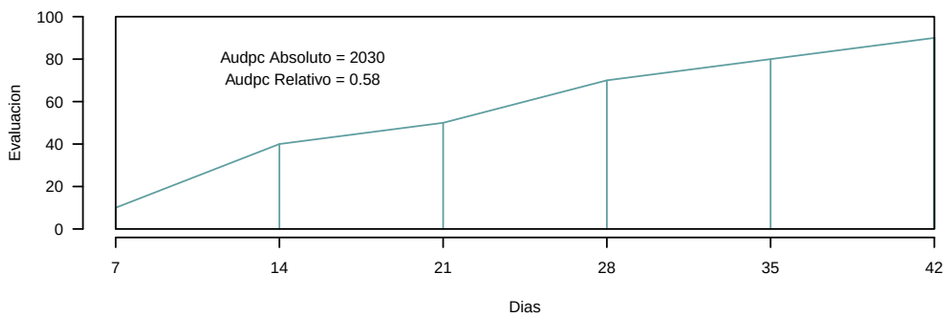

Figura 27: AUDPC: Área bajo la curva

```
> absolute1 <-audpc(evaluation,days)
> relative1 <-audpc(evaluation,days,"relative")
> absolute2 <-audps(evaluation,days)
> relative2 <-round(audps(evaluation,days,"relative"),2)
```

No Aditividad

La prueba de Tukey para la no aditividad se utiliza cuando existen dudas sobre la veracidad de adición de un modelo. Esta prueba confirma tal hipótesis y se espera para aceptar la hipótesis nula de efecto no aditivo del modelo.

Para esta prueba se requiere todos los datos experimentales utilizados en la estimación del modelo lineal aditivo.

La función nonadditivity() de *agricolae* y los datos experimentales de *potato* del paquete agricolae, se utilizaran para la prueba; en este caso, el modelo corresponde al diseño de bloques completos para el estudio comparativo de variedades.

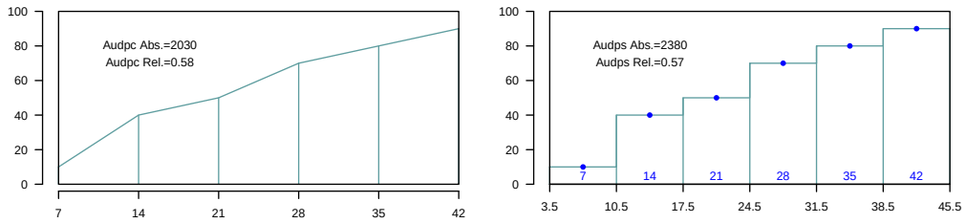

Figura 28: Area under the curve (AUDPC) and Area under the Stairs (AUDPS)

```
> data(potato)
> potato[,1]<-as.factor(potato[,1])
> model<-lm(cutting ~ date + variety,potato)
> df<-df.residual(model)
> MSerror<-deviance(model)/df
> analysis<-with(potato,nonadditivity(cutting, date, variety, df, MSerror))

Tukey's test of nonadditivity
cutting

P : 15.37166
Q : 77.44441

Analysis of Variance Table

Response: residual
             Df Sum Sq Mean Sq F value Pr(>F)
Nonadditivity  1  3.051  3.0511   0.922 0.3532
Residuals     14 46.330  3.3093
```

Según los resultados, el modelo es aditivo porque el p.value 0.35 es mayor que 0.05.

LATEBLIGHT

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

LATEBLIGHT Versión LB2004 fue creado en octubre de 2004 (Andrade-Piedra et al., 2005a, byc), basado en la versión C escrito por BE Ticknor ('BET 21191 modification of cbm8d29.c'), escrito por Doster et al. (1990) y descrito en detalle por Fry et al. (1991) (Esta versión se denomina 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). El programa fue implementado como una función en R por Felipe de mendiburu en agricolae. Aplicación, véase la Figura 29

```
> 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"
```

Repitiendo el gráfico

Con el siguiente script puede reproducir el gráfico:

```
> x<- model$Ofile$nday
> y<- model$Ofile$SimSeverity
> w<- model$Gfile$nday
> z<- model$Gfile$MeanSeverity
> Min<-model$Gfile$MinObs
> Max<-model$Gfile$MaxObs
> par(mar=c(3,2,0,1),cex=0.6)
> plot(x,y,type="l",xlim=c(65,95),lwd=1.5,xlab="Time (days after emergence)",
+ ylab="Severidad (Porcentaje)")
> 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("Curva del progreso de la enfermedad","Tiempo-Severidad"),
+ title="Descripcion",lty=1,pch=c(3,19),col=c("black","red"))
```

Informacion Gfile

```
> 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
```

```
> par(mar=c(3,2,0,1),cex=0.6)
> #-------------------------
> model<-lateblight(WS, Cultivar,ApplSys, InocDate, LGR,IniSpor,SR,IE, LP,
+ MatTime='LATESEASON',InMicCol,type="l", xlim=c(65,95),lwd=1.5,
+ xlab="Tiempo (d\'ias despu\'es de la emergencia)",ylab="Severidad (Porcentaje)")
```

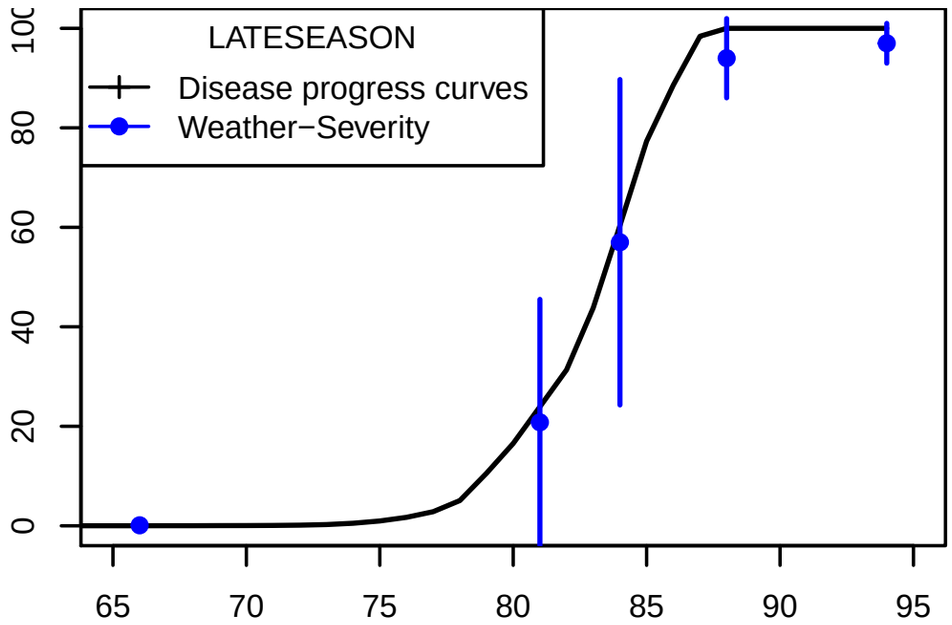

Figura 29: lateblight: LATESEASON

```
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
```

Referencia código ASCII					
Codigo	Simbolo	Codigo	Simbolo	Codigo	Simbolo
92	\	124	|	64	@
47	/	60	<	94	^
91	[	62	>	35	#
93	]	61	=	36	$
40	(	34	”	37	%
41	)	126	~	38	&
123	{	58	:	39	’
125	}	59	;		

Referencias

Gertrude M. and W. G. Cochran. (1992). *Experimental designs* Wiley, New York.

Conover, W.J (1999). *P*ractical Nonparametrics Statistics John Wiley & Sons, INC, New York.

Crossa, J. (1990). *S*tatistical analysis of multilocation trials Advances in Agronomy 44:55-85.

De Mendiburu, . (2009). *U*na herramienta de análisis estadístico para la investigación agrícola Universidad Nacional de Ingeniería (UNI).

Haynes, K.G. and Lambert, D.H. and Christ, B.J. and Weingartner, D.P. and Douches, D.S. and Backlund, J.E. and Secor, G. and Fry, W. and Stevenson, W. (1998). 75 5 211–217 Springer.

Hsu, J. (1996). *M*ultiple comparisons: theory and methods CRC Press Printed in the United States of America.

Joshi, D.D.(1987). *Linear estimation and design of experiments* WILEY EASTERN LIMITED New Delhi, India.

Kang, M.S. (1993). *Simultaneous Selection for Yield and Stability: Consequences for Growers* Agron. J. 85:754-757.

Kang, M.S. (1993). *Simultaneous Selection for Yield and Stability: Consequences for Growers* Agron. J. 85:754-757.

Kuehl, R.O. (2000). *Designs of experiments: statistical principles of research design and analysis* Duxbury Press.

LeClerg, E. (1962). *Field Plot Technique* Burgess Publishing Company.

Montgomery, D.C. (2002). *Design and Analysis of Experiments* John Wiley and Sons.

Patterson, H.D. and E.R. Williams, (1976). *Design and Analysis of Experiments Biometrika* Printed in Great Britain.

Purchase, J. L. (1997). *Parametric analysis to describe genotype environment interaction and yield stability in winter wheat* Department of Agronomy, Faculty of Agriculture of the University of the Free State Bloemfontein, South Africa.

R Core Team (2017). *A language and environment for statistical computing R Foundation for Statistical Computing* Department of Agronomy, Faculty of Agriculture of the University of the Free State Vienna, Austria. www.R-project.org.

Sabaghnia N. and S.H.Sabaghpour and H. Dehghani (2008). *The use of an AMMI model and its parameters to analyse yield stability in multienvironment trials Journal of Agricultural Science* Department of Agronomy, Faculty of Agriculture of the University of the Free State Cambridge University Press 571 doi:10.1017/S0021859608007831 Printed in the United Kingdom 146, 571-581.

Singh R.K. and B.D. Chaudhary (1979). *Biometrical Methods in Quantitative Genetic Analysis* Kalyani Publishers.

Steel and Torry and Dickey, (1997). *Principles and Procedures of Statistic a Biometrical Approach* Third Edition The Mc Graw Hill companies, Inc.