



LA MOLINA NATIONAL AGRARIAN UNIVERSITY

FACULTY OF ECONOMICS AND PLANNING
DEPARTMENT OF STATISTICS AND INFORMATICS



Agricolae

version 1.1-8

PRACTICAL MANUAL

Statistical Engineer Felipe de Mendiburu
Master in Systems Engineering

February - 2014

INDEX

	Page
PREFACE	1
1. INSTALLATION OF AGRICOLAE AND USE IN R	1
1.1 INSTALLATION	1
1.2 USE IN R	2
2. DESCRIPTIVE STATISTICS	3
2.1 HISTOGRAM	3
2.2 HANDLING SCALES	5
2.3 FREQUENCY TABLES AND STATISTICS	5
2.4 REPRODUCING HISTOGRAMS AND USE OF hist()	6
2.5 HISTOGRAM BASED ON GROUPED DATA	7
2.6 JOINING CLASSES	7
3. EXPERIMENT DESIGNS	8
3.1 COMPLETELY RANDOMIZED DESIGNS	9
3.2 RANDOMIZED COMPLETE BLOCK DESIGN	9
3.3 LATIN SQUARE DESIGNS	10
3.4 GRAECO-LATIN DESIGNS	11
3.5 BALANCED INCOMPLETE BLOCK DESIGNS	11
3.6 CYCLIC DESIGNS	12
3.7 LATTICE DESIGNS	14
3.8 ALPHA DESIGNS	16
3.9 AUGMENTED BLOCK DESIGNS	17
3.10 SPLIT-PLOT DESIGNS	19
3.11 STRIP-PLOT DESIGNS	20
3.12 FACTORIAL	22

4. MULTIPLE COMPARISONS	23
4.1 THE LEAST SIGNIFICANT DIFFERENCE (LSD)	23
4.2 BONFERRONI	25
4.3 DUNCAN'S NEW MULTIPLE-RANGE TEST	26
4.4 STUDENT-NEWMAN-KEULS	27
4.5 TUKEY'S W PROCEDURE (HSD)	28
4.6 WALLER-DUNCAN'S BAYESIAN K-RATIO t-TEST	29
4.7 SCHEFFE'S TEST	30
4.8 MULTIPLE COMPARISON IN FACTORIAL TREATMENTS	31
4.9 GRAPHICS OF THE MULTIPLE COMPARISON	33
4.10 ANALYSIS OF BALANCED INCOMPLETE BLOCKS	34
4.11 PARTIALLY BALANCED INCOMPLETE BLOCKS	37
4.12 AUGMENTED BLOCKS	43
4.13 NON-PARAMETRIC COMPARISONS	46
4.14 KRUSKAL-WALLIS	47
4.15 FRIEDMAN	48
4.16 WAERDEN	48
4.17 DURBIN	50
5. STABILITY ANALYSIS	51
5.1 PARAMETRIC STABILITY	51
5.2 NON-PARAMETRIC STABILITY	53
5.3 AMMI	54
6. SPECIAL FUNCTIONS	56
6.1 CONSENSUS OF DENDROGRAM	56
6.2 MONTECARLO	58
6.3 RE-SAMPLING IN LINEAR MODEL	59
6.4 SIMULATION IN LINEAR MODEL	60
6.5 PATH ANALYSIS	61
6.6 LINE X TESTER	61
6.7 SOIL UNIFORMITY	63
6.8 CONFIDENCE LIMITS IN BIODIVERSITY INDICES	64
6.9 CORRELATION	65
6.10 OTHER FUNCTIONS	66
REFERENCES	72

PREFACE

R is a functional programming system exclusive to manage data in statistics and related sciences, such as mathematics, in environments like Windows, Linux and MAC. 'Agricolae' is a package of functions for R applied to agricultural research.

The package 'agricolae' offers a broad functionality in the design of experiments, especially for experiments in agriculture and improvements of plants, which can also be used for other purposes. It contains the following designs: lattice, alpha, cyclic, balanced incomplete block designs, complete randomized blocks, Latin, Graeco-Latin, augmented block designs, divided parcels, divided blocks. It also has several procedures of experimental data analysis, such as the comparisons of treatments of Waller-Duncan, Bonferroni, Duncan, Student-Newman-Keuls, Scheffe, or the classic LSD and Tukey; and non-parametric comparisons, such as Kruskal-Wallis, Friedman, Durbin and Waerden, stability analysis, and other procedures applied in genetics, as well as procedures in biodiversity and descriptive statistics.

For more details on the use of 'agricolae', see the reference manual and the aid system in HTML, which can be found in the menu of R.

1 INSTALLATION OF AGRICOLAE AND USE IN R

1.1 INSTALLATION

The main program of R should be already installed in the platform of your computer (Windows, Linux or MAC). If it is not installed yet, you can download it from the R project (www.r-project.org) of a repository CRAN (R Development Core Team, 2013). As it is a free program, no identification is required. The packages can be incorporated through an installation process, directly from the platform of R.

'Agricolae' is a package for R, and as such its installation is just like any other package of R.

For Windows, the R program (version 3.0.0 or higher) is required.

If the R program is already installed in Windows or in another platform, the installation of 'agricolae' can be done directly from the console of R through Internet, that is

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

A repository should be selected and the system is installed automatically.

If there is no Internet connection, it is necessary to copy the file agricolae_x.x-x. zip for Windows from the page of the R project.

The file agricolae_x.x-x. zip (De Mendiburu, 2014) can be downloaded from the R repository in the following addresses: www.r-project.org or <http://cran.at.r-project.org/web/packages/agricolae/index.html>

The file can be directly incorporated into R installing from the console with the following instruction set if the file is located in the address E:
`install.packages("E:/agricolae_x.x-x.zip")`

It can also be installed from the R menu:

Packages, Install package(s) from local zip files.... Selecting the file zip does not require any unpacking.

For a complete functionality, 'agricolae' requires other packages.

MASS: for the generalized inverse used in the function `PBIB.test()`
nlme: for the methods REML and LM in `PBIB.test`
klaR: for the function `triplot()` used in the function `AMMI()`
akima: for the use of the function `interp()` used in `grid3p()` for interpolation
Cluster: for the use of the function `consensus()`

1.2 USE IN R

Since 'agricolae' is a package of functions, these are operational when they are called directly from the console of R and are integrated to all the base functions of R.

The following orders are frequent:

Load the package to the memory: `library(agricolae)`

Download: `detach(package:agricolae)`

Once the package is loaded, you can:

List the database: `data(package="agricolae")`

Load the sweet potato data: `data(sweetpotato)`

List data: `head(sweetpotato)`

See its structure: `str(sweetpotato)`

Edit its content: `fix(sweetpotato)`

In order to continue with the command line, do not forget to close the open windows with any R order.

For help: `help(sweetpotato)`; `? sweetpotato`

To search any functions: `apropos("design")`

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

For the use of symbols that do not appear in the keyboard in Spanish, such as: ~, [,], &, ^, |, <, >, {, }, % or others, use the table 6.10.

2 DESCRIPTIVE STATISTICS

The package 'agricolae' provides some complementary functions to the R program, specifically for the management of the histogram.

2.1 HISTOGRAM

The histogram is constructed with the function `graph.freq()` and is associated to other functions: `polygon.freq`, `table.freq`, `stat.freq`, `intervals.freq`, `sturges.freq`, `join.freq`, `ojiva.freq`, and `normal.freq`.

Example 1.1 Data generated in R. (students' weight). Figure 2.1

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

Load the package 'agricolae':

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

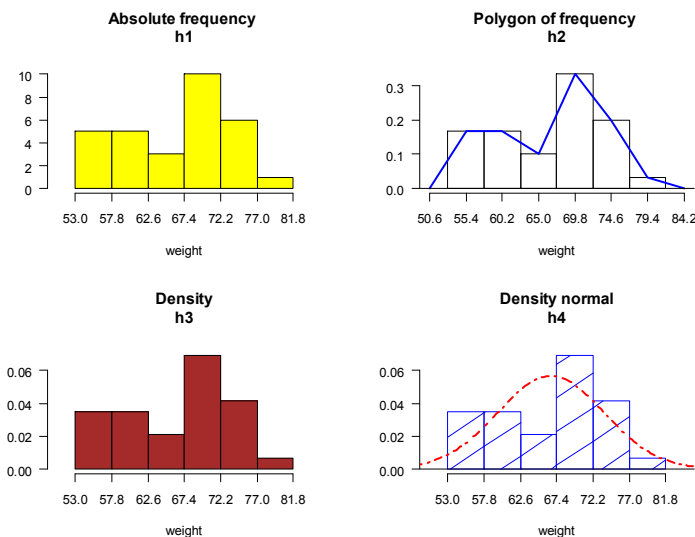


Figure 2.1 Histograms, polygon and density

```
# Comparison graf.freq() with hist(),
# output en Figura 2.2
```

```
set.seed(seed=71)
WEIGHT <- rnorm(30, 66.45, 7.42)
par(mfrow=c(2,1), mar=c(3,3,2,0), cex=0.7)
H1<-graf.freq(WEIGHT, frequency=3, main="R-agricolae\ngraf.freq()")
lines(density(WEIGHT), col="blue")
round(summary(H1), 2)
H2<-hist(WEIGHT, freq=FALSE, main="R-base\nhist()")
lines(density(WEIGHT), col="brown")
round(table.freq(H2), 2)
```

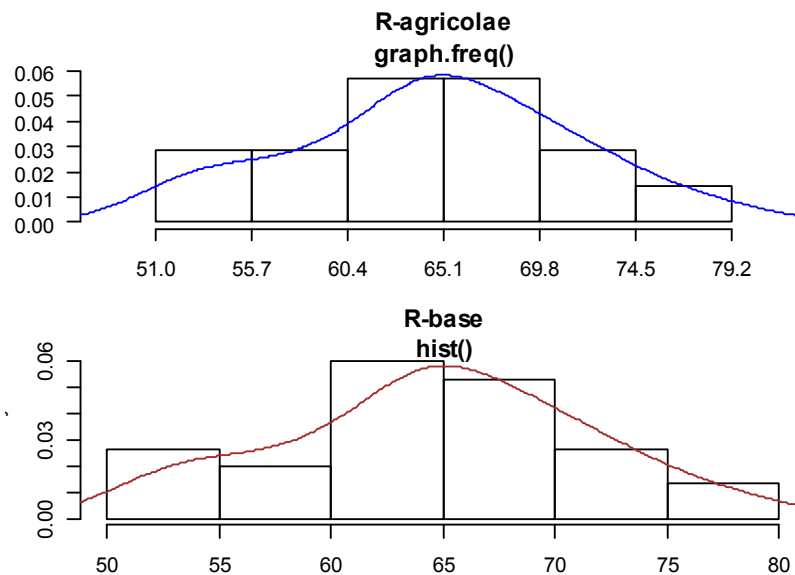


Figure 2.2 Histogram and density

```
> summary(WEIGHT)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 51.74  60.65   64.78   64.69  69.25   78.87

> round(summary(H1), 2)
  Lower Upper  Main freq relative CF  RCF
 51.0  55.7 53.35    4    0.13  4 0.13
 55.7  60.4 58.05    4    0.13  8 0.27
 60.4  65.1 62.75    8    0.27 16 0.53
 65.1  69.8 67.45    8    0.27 24 0.80
 69.8  74.5 72.15    4    0.13 28 0.93
 74.5  79.2 76.85    2    0.07 30 1.00

> round(table.freq(H2), 2)
  Lower Upper  Main freq relative CF  RCF
  50    55 52.5    4    0.13  4 0.13
  55    60 57.5    3    0.10  7 0.23
  60    65 62.5    9    0.30 16 0.53
  65    70 67.5    8    0.27 24 0.80
  70    75 72.5    4    0.13 28 0.93
  75    80 77.5    2    0.07 30 1.00
```

2.2 HANDLING SCALES

It refers to the scale changes in the axes. Figure 2.2

```
par(mfrow=c(2,2))
h5<- graph.freq(weight, frequency =1, main=" Absolute
frequency\nh5")
h6<- graph.freq(weight, nclass=5, main=" frequency with 5 classes
\nh6")
normal.freq(h6,col="red")
h7<- graph.freq(weight, density=6, col="blue", frequency =3, main="
density \nh7")
lines(density(weight),col="brown",lwd=2)
h8<- graph.freq(weight, border=0, frequency =3, main=" polygon and
density \nh8")
polygon.freq(h8,col="blue", frequency =3)
lines(density(weight),col="brown",lwd=2)
```

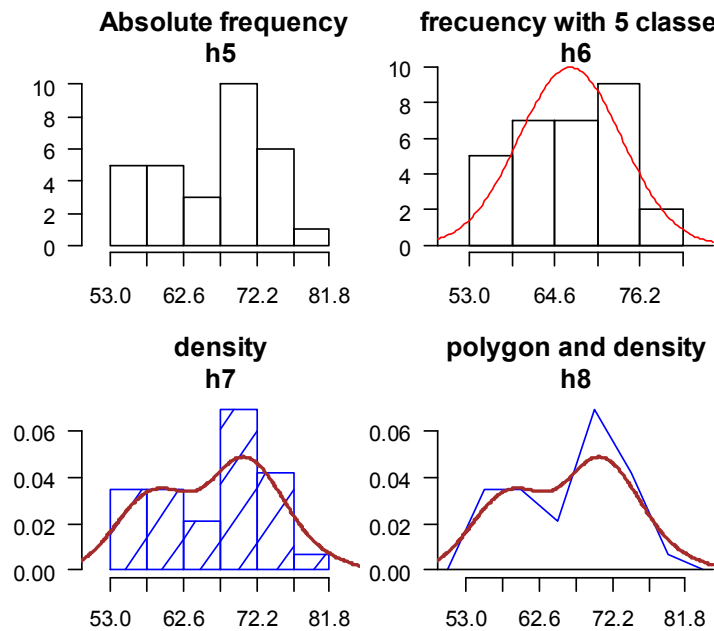


Figure 2.3. Scale change of the axes of coordinates

2.3 FREQUENCY TABLES AND STATISTICS

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

2.4 REPRODUCING HISTOGRAMS AND USE OF hist()

The class of `graph.freq()` is `graph.freq`. Figure 2.3

Reproducing the histogram `h6` (5 classes)

```
round(summary(h6),2)
```

Lower	Upper	Main	freq	relative	CF	RCF
53.0	58.8	55.9	5	0.17	5	0.17
58.8	64.6	61.7	7	0.23	12	0.40
64.6	70.4	67.5	7	0.23	19	0.63
70.4	76.2	73.3	9	0.30	28	0.93
76.2	82.0	79.1	2	0.07	30	1.00

The class types of the functions `hist()` and `graph.freq()` are 'histogram' and 'graph.freq', respectively. However, it is possible to establish compatibility between both functions.

```
par(mfrow=c(1,2))
h9<-ojiva.freq(h5, type="b", main="Ojiva de h5\nh9 ", col="red ")
h10<-plot(h6, main=" Five classes\nh10",xlab="weight")
normal.freq(h6,col="red")
#hh <- hist(weight,nclass=5) # 7 classes
#hh <- graph.freq(weight,nclass=5,density=8) # 5 classes
#h11<-graph.freq(hh, frequency=2,main="relative\nh11",
#xlab="weight", col=colors()[467])
```

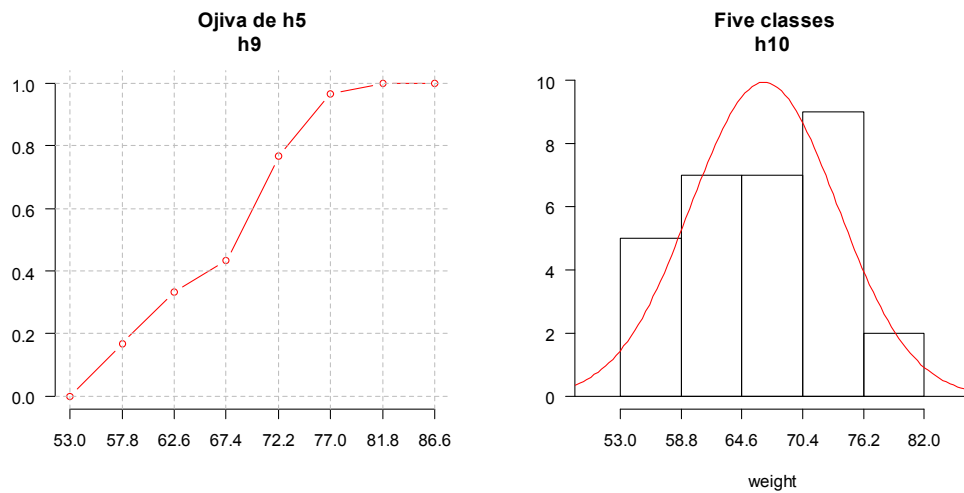


Figure 2.4. Ojiva and `normal.freq()`

In order to show the relative frequencies, you can use `graph.freq()` with the object `hh` created by `hist()`, without modifying the classes.

See the summaries: `summary(hh)`, `summary(h11)`

The functions of 'agricolae' for the management of histograms function correctly on the objects created by the function `hist()` of R.

2.5 HISTOGRAM BASED ON GROUPED DATA

If there are grouped data, you can graphic and obtain the histogram summaries with the function `graph.freq()`, as, for example, in the following table:

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

In R we have:

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

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

All the functions of 'agricolae' can be applied, including `plot()`.

2.6 JOINING CLASSES

Knowing the students' weight, the original intervals can be changed, joining, for example:

intervals.freq(h12)	h13<- join.freq(h12,1:2)
lower upper	intervals.freq(h13)
[1,] 0 10	lower upper
[2,] 10 20	[1,] 0 20
[3,] 20 30	[2,] 20 30
[4,] 30 40	[3,] 30 40
[5,] 40 50	[4,] 40 50

```
par(mfrow=c(1,2))
plot(h12, xlab="Original Class", main="target")
plot(h13, xlab="Changes in class", main="join.freq")
```

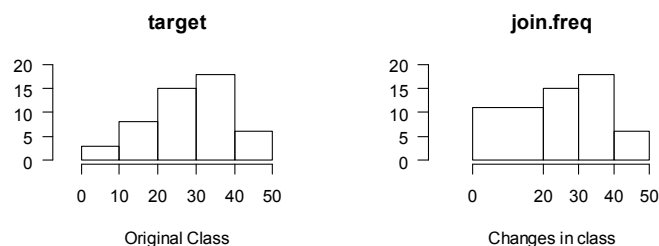


Figure 2.5 Classes new for the histogram

3 EXPERIMENT DESIGNS

The package 'agricolae' presents special functions for the creation of the field book for experimental designs. Due to the random generation, this package is quite used in agricultural research.

For this generation, certain parameters are required, as for example the name of each treatment, the number of repetitions, and others, according to the design (Cochran, 1992; Kuehl, 2000; Montgomery, 2002; LeClerg, 1962). There are other parameters of random generation, as the seed to reproduce the same random generation or the generation method (See the reference manual of agriculture <http://cran.at.r-project.org/web/packages/agricolae/agricolae.pdf>).

Important parameters in the generation of design:

Series : A constant that is used to set numerical tag blocks , eg number = 2, the labels will be : 101, 102, for the first row or block, 201 , 202, for the following , in the case of completely randomized design, the numbering is sequential.

design: Some features of the design requested agricolae be applied specifically to design.ab(factorial) or design.split (split plot) and their possible values are: "rcbd", "crd " and "lsd".

seed : The seed for the random generation and its value is any real value , if the value is zero, it has no reproducible generation, in this case copy of value of the outdesign\$parameters.

Kinds : the random generation method, by default " Super- Duper " .

first: For some designs is not required random the first repetition, especially in the block design , if you want to switch to random , change to TRUE.

Output design:

parameters: the input to generation design, include the seed to generation random, if seed=0, the program generate one value and it is possible reproduce the design.

book: field book

statistics: the information statistics the design for example efficiency index, number of treatments..

sketch: distribution of treatments in the field.

The enumeration of the plots:

zigzag is a function that allows you to place the numbering of the plots in the direction of serpentine:

Before :

After:

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

The zigzag(outdesign) function has one parameter:

outdesign: is output generated by one design: blocks, Latin square, graeco, split plot, strip plot, into blocks factorial, balanced incomplete block, cyclic lattice, alpha and augmented blocks.

fieldbook: output zigzag, contain field book.

3.1 COMPLETELY RANDOMIZED DESIGNS

They only require the names of the treatments and the number of their repetitions and its parameters are:

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

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

	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 RANDOMIZED COMPLETE BLOCK DESIGN

They require the names of the treatments and the number of blocks and its parameters are:

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

trt <- c("A", "B", "C")
repeticion <- 4
outdesign <- design.rcbd(trt,r=repeticion, seed=543, serie=2)
book2 <- outdesign$book
t(matrix(book2[,3],c(3,4)))
      [,1] [,2] [,3]
[1,] "A"  "B"  "C"
[2,] "B"  "A"  "C"
[3,] "B"  "C"  "A"
[4,] "C"  "B"  "A"
```

The plan can be sent to excel as a field book.

La numeración de las parcelas:

```
> t(matrix(book2[,1],c(3,4)))
      [,1] [,2] [,3]
[1,]  101  102  103
[2,]  201  202  203
[3,]  301  302  303
[4,]  401  402  403

> book <- zigzag(outdesign)

> t(matrix(book[,1],c(3,4)))
      [,1] [,2] [,3]
[1,]  101  102  103
[2,]  203  202  201
[3,]  301  302  303
[4,]  403  402  401
> t(matrix(book[,3],c(3,4)))
      [,1] [,2] [,3]
[1,]  "A"  "B"  "C"
[2,]  "B"  "A"  "C"
[3,]  "B"  "C"  "A"
[4,]  "C"  "B"  "A"
```

3.3 LATIN SQUARE DESIGNS

They require the names of the treatments and its parameters are:

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

trt <- c("A", "B", "C", "D")
outdesign <- design.lsd(trt, seed=543, serie=2)
book3 <- outdesign$book
t(matrix(book3[,4],c(4,4)))

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

Serpentine enumeration:

```
book <- zigzag(outdesign)
t(matrix(book,c(4,4)))

      [,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 GRAECO-LATIN DESIGNS

They require the names of the treatments of each factor of study and its parameters are:

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

trt1 <- c("A", "B", "C", "D")
trt2 <- 1:4
outdesign <- design.graeco(trt1, trt2, seed=543, serie=2)
book4 <- outdesign$book
t(matrix(paste(book4[,4], book4[,5]), c(4,4)))

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

Serpentine enumeration:

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

      [,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 BALANCED INCOMPLETE BLOCK DESIGNS

They require the names of the treatments and the size of the block and its parameters are:

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

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

Parameters BIB
=====
Lambda      : 3
treatments  : 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"

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

$k
[1] 4

$serie
[1] 2

$seed
[1] 543

$kind
[1] "Super-Duper"

```

According to the produced information, they are five blocks of size 4, being the matrix:

```

t(matrix(book5[,3],c(4,5)))
      [,1] [,2] [,3] [,4]
[1,] "C"  "B"  "E"  "A"
[2,] "C"  "D"  "A"  "B"
[3,] "B"  "A"  "E"  "D"
[4,] "D"  "C"  "E"  "B"
[5,] "A"  "D"  "E"  "C"

```

It can be observed that the treatments have four repetitions. The parameter lambda has three repetitions, which means that a couple of treatments are together on three occasions. For example, B and E are found in the blocks I, III and V.

Serpentine enumeration:

```

book <- zigzag(outdesign)

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

```

3.6 CYCLIC DESIGNS

They require the names of the treatments, the size of the block and the number of repetitions. This design is used for 6 to 30 treatments. The repetitions are a multiple

of the size of the block; if they are six treatments and the size is 3, then the repetitions can be 6, 9, 12, etc. and its parameters are:

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

```
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
=====
treatments : 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 blocks of 4 treatments each have been generated.

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
```

3.7 LATTICE DESIGNS

They require a number of treatments of a perfect square; for example 9, 16, 25, 36, 49, etc. and its parameters are:

```
> args(design.lattice)
function (trt, r = 3, serie = 2, seed = 0, kinds = "Super-Duper")
```

They can generate a simple lattice (2 rep.) or a triple lattice (3 rep.)

generating a triple lattice design for 9 treatments 3x3

```
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
```

```
$repl
      [,1] [,2] [,3]
[1,]  "F"  "A"  "B"
[2,]  "I"  "E"  "G"
[3,]  "D"  "C"  "H"
```

```

$rep2
      [,1] [,2] [,3]
[1,] "I"  "D"  "F"
[2,] "G"  "H"  "B"
[3,] "E"  "C"  "A"
$rep3
      [,1] [,2] [,3]
[1,] "I"  "H"  "A"
[2,] "E"  "D"  "B"
[3,] "G"  "C"  "F"

```

```
> 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
7	107	1	3	h
8	108	1	3	f
9	109	1	3	g
10	201	2	4	e
11	202	2	4	g
12	203	2	4	a
13	204	2	5	c
14	205	2	5	f
15	206	2	5	d
16	207	2	6	b
17	208	2	6	h
18	209	2	6	i
19	301	3	7	e
20	302	3	7	h
21	303	3	7	d
22	304	3	8	c
23	305	3	8	g
24	306	3	8	i
25	307	3	9	b
26	308	3	9	f
27	309	3	9	a

Serpentine enumeration:

```

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
```

3.8 ALPHA DESIGNS

These designs are generated by the alpha arrangements (Patterson & Williams, 1976). They are similar to the lattice designs, but the tables are rectangular, with s blocks \times k treatments. The number of treatments should be equal to $s \times k$ and all the experimental units, $r \times s \times k$ and its parameters are:

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

```
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"
```

```
# codificacion de las parcelas

> 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
```

Serpentine enumeration:

```
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
```

3.9 AUGMENTED BLOCK DESIGNS

These are designs for two types of treatments: the control treatments (common) and the increased treatments. The common treatments are applied in complete randomized blocks, and the increased treatments, at random. Each treatment should be applied in any block once only. It is understood that the common treatments are of a greater interest; the standard error of the difference is much smaller than when between two increased ones in different blocks. The function `design.dau()` achieves this purpose and its parameters are:

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

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

```
attach(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"
```

```
detach(book9)
```

Serpentine enumeration:

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

```
attach(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"
```

```
detach(book)
```

```
print(book)
```

```
> print(book)  
  plots block trt  
1    101     1  D  
2    102     1  C  
3    103     1  A  
4    104     1  u  
...  
27   505     5  x
```

For augmented randomized complete block designs, use the function `design.crd()`.

3.10 SPLIT-PLOT DESIGNS

These designs have two factors, one is applied in plots and is defined as A in a randomized complete block design; and a second factor, which is applied in the subplots of each plot applied at random. The function `design.split()` permits to find the experimental plan for this design and its parameters are:

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

```
trt1<-c("A","B","C","D")
trt2<-c("a","b","c")
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    B    b
5    102      2     1    B    c
6    102      3     1    B    a
7    103      1     1    C    b
8    103      2     1    C    c
9    103      3     1    C    a
10   104      1     1    D    a
11   104      2     1    D    b
12   104      3     1    D    c
13   201      1     2    A    b
14   201      2     2    A    a
...
36   304      3     3    D    b
```

```
p<-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"  "B"  "C"  "D"
[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"
```

Serpentine enumeration:

```
book <- zigzag(outdesign)
print(book)
```

	plots	splots	block	trt1	trt2
1	101	1	1	A	c
2	101	2	1	A	a
3	101	3	1	A	b
4	102	1	1	B	b
5	102	2	1	B	c
...					
12	104	3	1	D	c
13	204	3	2	A	b
14	204	2	2	A	a
15	204	1	2	A	c
16	203	3	2	C	a
17	203	2	2	C	b
18	203	1	2	C	c
...					
34	304	1	3	D	c
35	304	2	3	D	a
36	304	3	3	D	b

3.11 STRIP-PLOT DESIGNS

These designs are used when there are two types of treatments (factors) and are applied separately in large plots, called bands, in a vertical and horizontal direction of the block, obtaining the divided blocks. Each block constitutes a repetition and its parameters are:

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

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

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

```
t3<-paste(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"
```

Serpentine enumeration:

```
book <- zigzag(outdesign)
print(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
7	107	1	B	a
8	108	1	B	b
...				
36	310	3	A	a

```
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
```


3.12 FACTORIAL

The full factorial of n factors applied to an experimental design (CRD, RCBD and LSD) is common and this pocedimientto in agricolae applies the factorial to one of these three designs and its parameters are:

```
> args(design.ab)
```

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

To generate the factorial, you need to create a vector of levels of each factor, the method automatically generates up to 25 factors and "r" repetitions.

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

to crd and rcbd designs, it is necessary to value "r" as the number of repetitions, this can be a vector if unequal to equal or constant repetition (recommended).

```
library(agricolae)  
trt<-c(3,2) # factorial 3x2  
outdesign <- design.ab(trt, r=3, serie=2)  
book12 <- outdesign$book  
print(book12) # print of the field book  
  plots block A B  
1    101     1 1 1  
2    102     1 1 2  
3    103     1 2 1  
4    104     1 2 2  
5    105     1 3 1  
6    106     1 3 2  
7    201     2 1 2  
8    202     2 3 2  
9    203     2 1 1  
10   204     2 2 2  
.  
.  
.
```

Serpentine enumeration:

```
book <- zigzag(outdesign)  
print(book)
```

```
  plots block A B  
1    101     1 1 1  
2    102     1 1 2  
3    103     1 2 1  
4    104     1 2 2  
5    105     1 3 1  
6    106     1 3 2  
7    206     2 1 2  
8    205     2 2 2  
9    204     2 1 1  
10   203     2 3 1  
11   202     2 2 1  
12   201     2 3 2  
13   301     3 2 2  
14   302     3 3 1
```

```

15  303      3 1 2
16  304      3 1 1
17  305      3 3 2
18  306      3 2 1

```

```

> # factorial 2 x 2 x 2 with 5 replications in completely randomized
design.
> trt<-c(2,2,2)
> crd<-design.ab(trt, r=5, serie=2,design="crd")
> print(crd)
      plots r A B C
1      101 1 2 1 2
2      102 1 2 2 2
3      103 2 2 2 2
4      104 1 1 2 2
...

```

4 MULTIPLE COMPARISONS

For the analyses, the following functions of 'agricolae' are used: `LSD.test()`, `HSD.test()`, `duncan.test()`, `scheffe.test`, `waller.test`, `SNK.test()` (Steel, 1996) and `durbin.test()`, `kruskal()`, `friedman()` and `waerden.test` (Conover, 1999).

For every statistical analysis, the data should be organized in columns. For the demonstration, the 'agricolae' database will be used.

The 'sweetpotato' data correspond to a completely random experiment in field with plots of 50 sweet potato plants, subjected to the virus effect and to a control without virus (See the reference manual of the package).

```

data(sweetpotato)
model<-aov(yield~virus, data=sweetpotato)

cv.model(model)
[1] 17.16660
attach(sweetpotato)
mean(yield)
[1] 27,625

```

Model parameters: Degrees of freedom and variance of the error:

```

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

```

4.1 THE LEAST SIGNIFICANT DIFFERENCE (LSD)

It includes the multiple comparison through the method of the minimum significant difference (Least Significant Difference), (Steel, 1997).

```

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

```

LSD t Test for yield

Mean Square Error: 22.48917

virus, means and individual (95 %) CI

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

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

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

Groups, Treatments and means

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

In the function `LSD.test()`, the multiple comparison was carried out. In order to obtain the probabilities of the comparisons, it should be indicated that groups are not required; thus:

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

LSD t Test for yield

Mean Square Error: 22.48917

virus, means and individual (95 %) CI

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

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

Comparison between treatments means

	Difference	pvalue	sig		LCL	UCL
cc - fc	11.53333333	0.0176377595	*	2.604368	20.462299	
cc - ff	-11.93333333	0.0150730851	*	-20.862299	-3.004368	
cc - oo	-12.50000000	0.0120884239	*	-21.428965	-3.571035	
fc - ff	-23.46666667	0.0003023690	***	-32.395632	-14.537701	
fc - oo	-24.03333333	0.0002574929	***	-32.962299	-15.104368	
ff - oo	-0.56666667	0.8872673216		-9.495632	8.362299	

The significance code "sig" is interpreted as:

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

> comparison

```
$statistics
      Mean      CV  MSerror
27.625 17.1666 22.48917
```

```
$parameters
      Df ntr  t.value
      8   4 2.306004
```

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

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

```
$groups
NULL
```

4.2 BONFERRONI

With the function `LSD.test()` we can make adjustments to the probabilities found, as for example the adjustment by Bonferroni.

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

```
LSD t Test for yield
P value adjustment method: bonferroni
```

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

Comparison between treatments means

	Difference	pvalue	sig	LCL	UCL
cc - fc	11.53333333	0.105827		-1.937064	25.0037305
cc - ff	-11.93333333	0.090439	.	-25.403730	1.5370638
cc - oo	-12.5000000	0.072531	.	-25.970397	0.9703971
fc - ff	-23.4666667	0.001814	**	-36.937064	-9.9962695
fc - oo	-24.0333333	0.001545	**	-37.503730	-10.5629362
ff - oo	-0.5666667	1.000000		-14.037064	12.9037305

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

For “duncan”, use the function `duncan.test()`; for “Student-Newman-Keuls”, the function `SNK.test()`; for “tukey”, the function `HSD.test()`; for “scheffe”, the function `scheffe.test()`; and for “waller-duncan”, the function `waller.test()`. The parameters are the same. “Waller” also requires the value of F-calculated of the ANOVA treatments. If the model is used as a parameter, this is no longer necessary.

4.3 DUNCAN’S NEW MULTIPLE-RANGE TEST

It corresponds to the Duncan’s Test (Steel, 1997).

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

```
Duncan's new multiple range test
for yield
```

```
Mean Square Error: 22.48917
```

```
virus, means
```

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

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

```
Critical Range
```

	2	3	4
	8.928965	9.304825	9.514910

Means with the same letter are not significantly different.

```
Groups, Treatments and means
```

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

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

```
Duncan's new multiple range test
for yield
```

```
Mean Square Error: 22.48917
```

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

```
Critical Range
      2      3      4
8.928965 9.304825 9.514910
```

```
Comparison between treatments means
```

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

4.4 STUDENT-NEWMAN-KEULS

Student, Newman and Keuls helped to improve the Newman-Keuls test of 1939, which was known as the Keuls method (Steel, 1997).

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

```
Student Newman Keuls Test
for yield
```

```
Mean Square Error: 22.48917
```

```
virus, means
```

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

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

```
Critical Range
      2      3      4
8.928965 11.064170 12.399670
```

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

```
Groups, Treatments and means
```

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

```
SNK.test(model, "virus", group=FALSE,console=TRUE)
```

```
Student Newman Keuls Test
for yield
```

```
Mean Square Error: 22.48917
```

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

```
Critical Range
```

```
      2      3      4  
8.928965 11.064170 12.399670
```

```
Comparison between treatments means
```

	Difference	pvalue	sig	LCL	UCL
cc-fc	11.5333333	0.017638	*	2.604368	20.462299
cc-ff	-11.9333333	0.015073	*	-20.862299	-3.004368
cc-oo	-12.5000000	0.029089	*	-23.564170	-1.435830
fc-ff	-23.4666667	0.000777	***	-34.530836	-12.402497
fc-oo	-24.0333333	0.001162	**	-36.433003	-11.633664
ff-oo	-0.5666667	0.887267		-9.495632	8.362299

4.5 TUKEY'S W PROCEDURE (HSD)

This studentized range test, created by Tukey in 1953, is known as the Tukey's HSD (Honestly Significant Differences) Test (Steel, 1997).

```
comparison1 <- HSD.test(model, "virus", console=TRUE)
```

```
HSD Test for yield
```

```
Mean Square Error: 22.48917
```

```
virus, means
```

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

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

```
Critical Value of Studentized Range: 4.52881
```

```
Honestly Significant Difference: 12.39967
```

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

```
Groups, Treatments and means
```

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

```
> comparison1
```

```
$statistics
```

Mean	CV	MSerror	HSD
27.625	17.1666	22.48917	12.39967

```
$parameters
```

Df	ntr	StudentizedRange
8	4	4.52881

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

```
$comparison
NULL
```

```
$groups
      trt      means  M
1  oo 36.90000  a
2  ff 36.33333  ab
3  cc 24.40000  bc
4  fc 12.86667  c
```

4.6 WALLER-DUNCAN'S BAYESIAN K-RATIO t-TEST

In 1975, Duncan continued the multiple comparison procedures, introducing the criterion of minimizing both experimental errors; for this, he used the Bayes' theorem, obtaining one new test called Waller-Duncan (Steel, 1997).

```
# variance analysis:
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
```

The value of calculated F is 17,345. For the comparative treatment analysis, it is required that 'sweetpotato' be active, thus:

```
attach(sweetpotato)
```

then:

```
waller.test(yield,virus,df,MSerror,Fc= 17.345, group=F,console=TRUE)
```

In another case with only invoking the `model` object:

```
comparison2 <- waller.test(model, "virus", group=FALSE,console=TRUE)
```

Waller-Duncan K-ratio t Test for yield

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

```

      . . . . .
K ratio                100.00000
Error Degrees of Freedom    8.00000
Error Mean Square          22.48917
F value                   17.34478
Critical Value of Waller    2.23600
```



```
virus, means
```

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

```
Minimum Significant Difference 8.657906
Comparison between treatments means
```

```
      Difference significant
cc - fc 11.53333333      TRUE
ff - cc 11.93333333      TRUE
oo - cc 12.50000000      TRUE
ff - fc 23.46666667      TRUE
oo - fc 24.03333333      TRUE
oo - ff  0.56666667      FALSE
```

It is indicated that the virus effect "ff" is not significant to the control "oo."

The found object "compare" has information to make other procedures.

```
> names(compare2)
[1] "statistics" "parameters" "means"      "comparison" "groups"
```

```
> compare2$statistics
```

```
      Mean      CV  MSerror  F.Value CriticalDifference
27.625 17.1666 22.48917 17.34478      8.657906
```

4.7 SCHEFFE'S TEST

This method, created by Scheffe in 1959, is very general for all the possible contrasts and their confidence intervals. The confidence intervals for the averages are very broad, resulting in a very conservative test for the comparison between treatment averages (Steel, 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.err r Min. Max.
cc 24.40000 2.084067 3 21.7 28.5
fc 12.86667 1.246774 3 10.6 14.9
ff 36.33333 4.233727 3 28.0 41.8
oo 36.90000 2.482606 3 32.1 40.4
```

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

```
Minimum Significant Difference: 13.52368
```

Means with the same letter are not significantly different.

```
Groups, Treatments and means
```

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

The minimum significant value is very high.

If you require the approximate probabilities of comparison, you can use the option Group=FALSE.

```
comparison3 <- scheffe.test(model,"virus", group=FALSE,
console=TRUE)
```

Study:

Scheffe Test for yield

```
Mean Square Error : 22.48917
```

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

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

Comparison between treatments means

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

4.8 MULTIPLE COMPARISON IN FACTORIAL TREATMENTS

In a factorial combined effects of the treatments. Comparative tests: LSD, HSD, Waller-Duncan, Duncan, Scheffé, SNK can be applied.

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

The comparison is the combination of A:B:C.

Data RCBD design with a factorial clone x nitrogen. The response variable 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
```

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

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	1.187			

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

```
out <-LSD.test (model, 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.err	r	LCL	UCL	Min.	Max.
c1:n1	7.50	0.6454972	4	6.167759	8.832241	6	9
c1:n2	8.50	0.6454972	4	7.167759	9.832241	7	10
c1:n3	9.00	0.4082483	4	8.157417	9.842583	8	10
c2:n1	13.50	0.6454972	4	12.167759	14.832241	12	15
c2:n2	16.75	0.4787136	4	15.761984	17.738016	16	18
c2:n3	20.25	0.8539126	4	18.487611	22.012389	18	22

```

c3:n1  9.50 0.6454972 4  8.167759 10.832241  8  11
c3:n2  9.50 0.8660254 4  7.712611 11.287389  8  12
c3:n3 10.25 0.7500000 4  8.702076 11.797924  9  12

```

```

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

```

```

Least Significant Difference 1.590341
Means with the same letter are not significantly different.
Groups, Treatments and means

```

```

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

```

```

par(mar=c(3,3,2,0))
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="average and range\nclon:nitrogen")

```

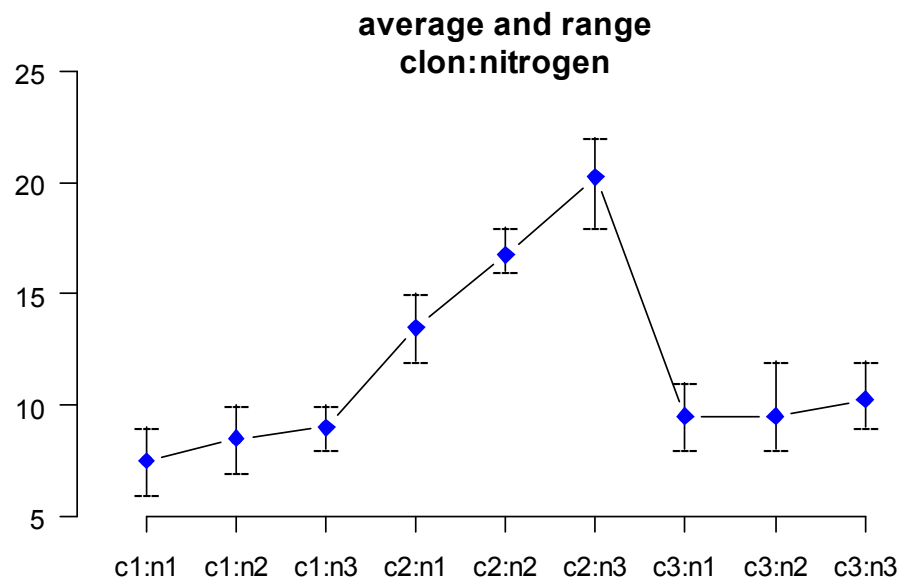


Figure 4.1 Combined clone:nitrogen

4.9 GRAPHICS OF THE MULTIPLE COMPARISON

The results of a comparison can be graphically seen with the functions `bar.group()` and `bar.err()`.

The found object of one comparison is the entry for these functions, Figure 4.2.

The objects `compare1` and `compare2` are used in the following exercise:

comparison1, for the functions bar.group() and bar.err()
 comparison2, for the function bar.err()

```
par(mfrow=c(2,2))
c1<-colors()[480]; c2=colors()[65]; c3=colors()[15];
c4=colors()[140]
G1<-bar.group(comparison1$groups, ylim=c(0,45),
main="Tukey\nG1",col=c1)
G2<-bar.group(comparison1$groups, horiz=T, xlim=c(0,45),
main="Tukey\nG2",col=c2)
G3<-bar.err(comparison2$means, variation="range",ylim=c(0,45),
col=c3, main="Range\nG3")
G4<-bar.err(comparison2$means, horiz=T, xlim=c(0,45), col=c4,
variation="SE",main="Standard error \nG4")
```

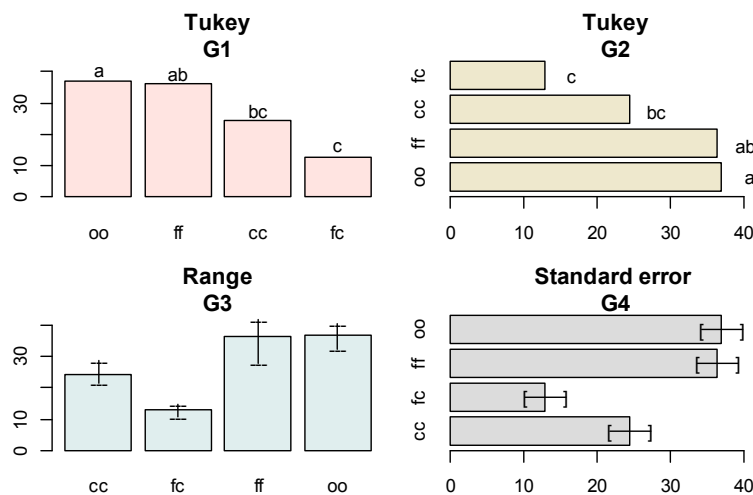


Figure 4.2 Comparison between treatments

4.10 ANALYSIS OF BALANCED INCOMPLETE BLOCKS

This analysis can come from balanced or partially balanced designs. The function BIB.test() is for balanced designs, and PBIB.test(), for partially balanced designs. In the following example, the 'agricolae' data will be used.

```
#Example linear estimation and design of experiments. (Joshi, 1987)
# Profesor de Estadística, Institute of Social Sciences Agra, India
# 6 variedades de trigo en 10 bloques de 3 parcelas cada una.
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
```

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

LSD test

Std.diff : 5.363111

Alpha : 0.05

LSD : 11.4312

Parameters BIB

Lambda : 2

treatmeans : 6

Block size : 3

Blocks : 10

Replication: 5

Efficiency factor 0.8

<<< Book >>>

Means with the same letter are not significantly different.

Comparison of treatments

Groups, Treatments and means

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

function (block, trt, y, test = c("lsd", "tukey", "duncan", "waller", "snk"), alpha = 0.05, group = TRUE). LSD, Tukey Duncan, Waller-Duncan and SNK, can be used. The probabilities of the comparison can also be obtained. It should only be indicated: group=FALSE, thus:

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

```
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 0.070509 .
1 - 3 16.5833333 0.066649 .
1 - 4 20.1666667 0.019092 *
1 - 5 15.0833333 0.109602
1 - 6 20.7500000 0.015510 *
2 - 3  0.1666667 1.000000
2 - 4  3.7500000 0.979184
2 - 5 -1.3333333 0.999840
```

```

2 - 6  4.3333333 0.961588
3 - 4  3.5833333 0.982927
3 - 5 -1.5000000 0.999715
3 - 6  4.1666667 0.967375
4 - 5 -5.0833333 0.927273
4 - 6  0.5833333 0.999997
5 - 6  5.6666667 0.890815

```

The found "outl" object can be used for the functions `bar.group()` and `bar.err()` for the bar graphics, in the same way as previously.

```
names(out)
```

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

```
bar.group: out$groups
```

```
bar.err: out$means
```

4.11 PARTIALLY BALANCED INCOMPLETE BLOCKS

The function `PBIB.test()` (Joshi, 1987) can be used for the lattice and alpha designs.

Consider the following case: Construct the alpha design with 30 treatments, 2 repetitions, and a block size equal to 3.

```

library(agricolae)
library(MASS)
library(nlme)
# alpha design
Genotype<-paste("geno",1:30,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 >>>
```

The generated plan is `plan$book`.

Suppose that the corresponding observation to each experimental unit is:

```

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)

```


The data table is constructed for the analysis. In theory, it is presumed that a design is applied and the experiment is carried out; subsequently, the study variables are observed from each experimental unit.

```
data<-data.frame(plan$book,yield)
rm(yield,Genotype)
```

The analysis:

```
attach(data)
model <- PBIB.test(block, Genotype, replication, yield, k=3,
group=TRUE,console=TRUE)
detach(data)
```

ANALYSIS PBIB: yield

Class level information

block : 20

Genotype : 30

Number of observations: 60

Estimation Method: REML

Parameter Estimates

	Variance
block:replication	2.834033e+00
replication	8.045359e-09
Residual	2.003098e+00

Fit Statistics

-2 Res Log Likelihood	-73.82968
AIC	213.65937
BIC	259.89888

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
Means with the same letter are not significantly different.

Groups, Treatments and means

a	20	7.729
ab	13	6.715
ab	1	6.505
abc	8	6.192
abcd	24	6.032
abcd	23	5.735
abcd	10	5.473
abcd	16	5.455
abcd	21	5.14
abcd	22	5.069
abcd	4	4.874
abcd	3	4.794
abcd	14	4.742
abcd	15	4.587
abcd	27	4.563
abcd	7	4.424
abcd	5	4.286
abcd	19	4.198
abcd	6	4.165
abcd	25	3.978
abcd	17	3.943
bcd	2	3.628
bcd	28	3.495
bcd	11	3.379
bcd	26	3.341
bcd	9	3.052
bcd	30	3
bcd	12	2.879
cd	29	2.44
d	18	2.186

Comparison between treatments means and its name

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

The adjusted averages can be extracted from the model.

model\$means

	yield	trt	mean.adj	SE	r	std.err	Min.	Max.
geno1	7.5	1	6.504753	1.313644	2	1.5	6	9
geno10	4.5	2	3.628197	1.313644	2	0.5	4	5
geno11	5.5	3	4.793620	1.310727	2	0.5	5	6
geno12	4.0	4	4.873878	1.313644	2	3.0	1	7
...								
geno8	3.0	29	2.439879	1.310727	2	1.0	2	4
geno9	3.5	30	2.999639	1.310727	2	1.5	2	5

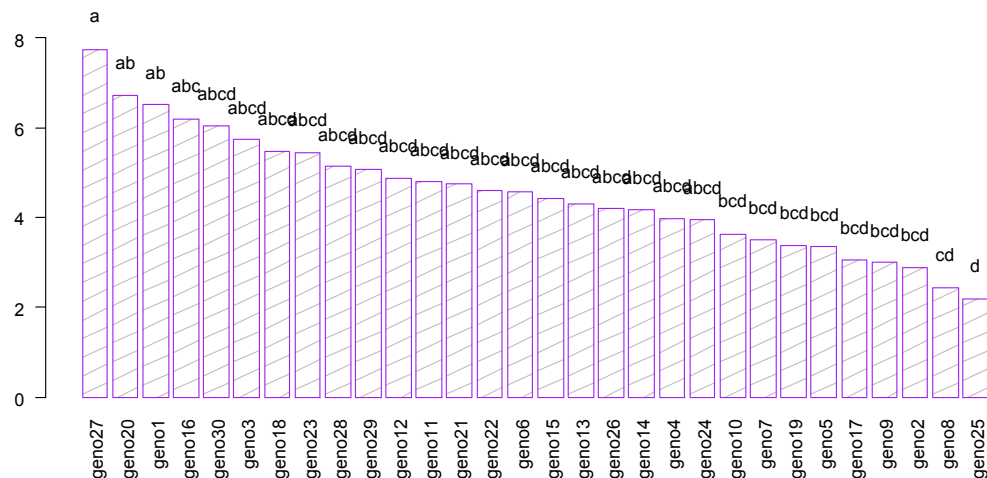


Figure 4.3. Treatment Groups.

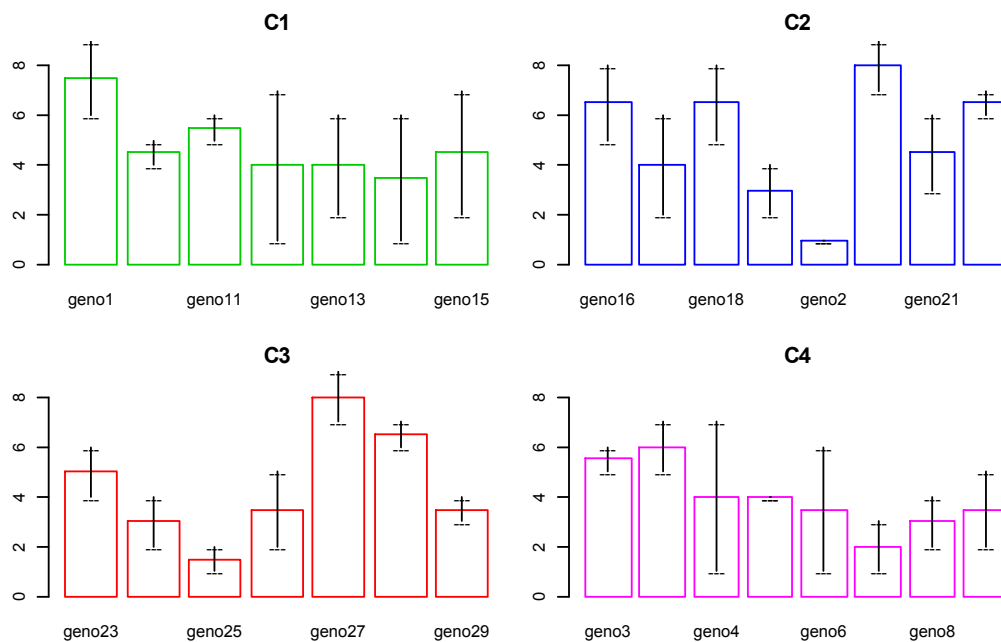
The comparisons:

`model$comparison`

	Difference	stderr	pvalue
1 - 2	2.87655609	1.844369	0.147134
1 - 3	1.71113347	1.576447	0.300944
1 - 4	1.63087546	1.727017	0.365280
1 - 5	2.21879737	1.853044	0.256324
....			
27 - 30	1.56380843	1.812352	0.406634
28 - 29	1.05522249	1.850096	0.579896
28 - 30	0.49546251	1.847787	0.793554
29 - 30	-0.55975998	1.587130	0.730988

The data on the adjusted averages and their standard error can be illustrated (figure 4.2), since the created object is very similar to the objects generated by the multiple comparisons.

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



Analysis of balanced lattice 3x3, 9 treatments, 4 repetitions.

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

ANALYSIS PBIB: yield

```

Number of observations:  36

Estimation Method:  Residual (restricted) maximum likelihood

Parameter Estimates
      Variance
block:sqr 1.604257e-08
sqr        1.668375e-07
Residual   5.693724e+01

                        Fit Statistics
-2 Res Log Likelihood    -99.11599
AIC                      222.23197
BIC                      237.78201

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

Means with the same letter are not significantly different.

Groups, Treatments and means
a      1      45.89
ab     9      39.61
ab     4      38.09
bc     7      31.9
bc     6      30.24
cd     2      25.67
d      8      18.71
d      5      16.9
d      3      15.45

Comparison between treatments means and its name

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

par(mar=c(3,3,0,0))

```

```
bar.group(model2$group,ylim=c(0,52),density=10)
```

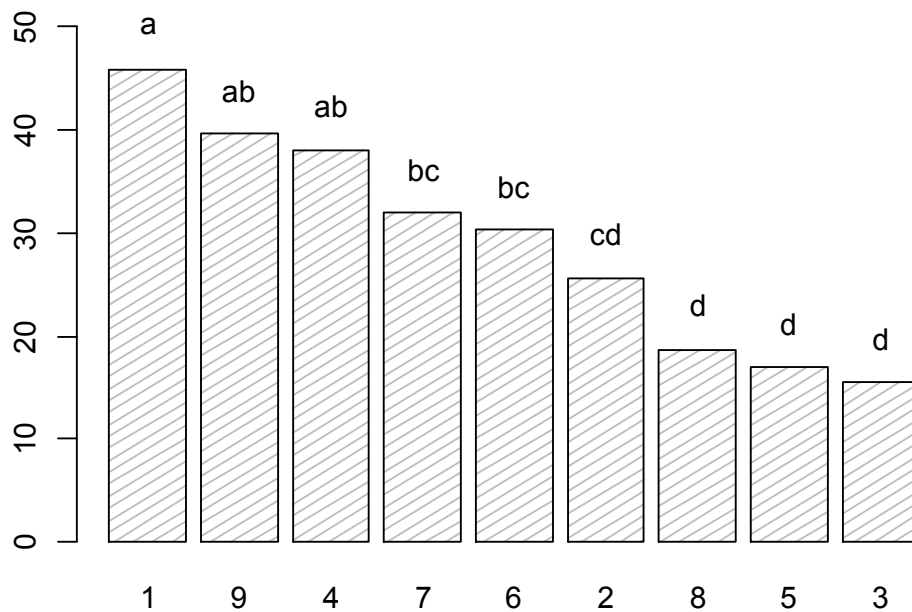


Figure 4.5. Treatment Groups.

```
model2$means
```

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

```
model2$comparison
```

	Difference	stderr	pvalue
1 - 2	20.2250	5.335599	0.001604
1 - 3	30.4475	5.335599	0.000032
1 - 4	7.8050	5.335599	0.162884
....			
8 - 9	-20.9025	5.335599	0.001228

4.12 AUGMENTED BLOCKS

The function `DAU.test()` can be used for the analysis of the augmented block design.

The data should be organized in a table, containing the blocks, treatments, and the response.

```

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

The treatments are in each block:

```

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"

```

With their respective responses:

```

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

```

```

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

```

```

ANALYSIS DAU:  yield
Class level information

```

```

Block:  I II III

```

Trt : A B C D e f g h i j k l

Number of observations: 20

ANOVA, Treatment Adjusted
Analysis of Variance Table

Response: yield

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

ANOVA, Block Adjusted
Analysis of Variance Table

Response: yield

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

coefficient of variation: 6.4 %
yield Means: 81.5

Critical Differences (Between) Std Error Diff.

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

Means with the same letter are not significantly different.

Groups, Treatments and means

a	h	93.5
ab	f	86.5
ab	A	84.67
ab	D	83.33
ab	C	82
ab	j	79.5
ab	B	79
ab	e	78.25
ab	k	78.25
ab	i	77.25
ab	l	77.25
b	g	73.25

Comparison between treatments means

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


```
model$means
```

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

```
model<- DAU.test(block,trt,yield,method="lsd",group=F,console=TRUE)
> head(model$comparison,8)
```

	Difference	pvalue	sig
A - B	5.666667	0.229886	
A - C	2.666667	0.552612	
A - D	1.333333	0.763840	
A - e	6.416667	0.352008	
A - f	-1.833333	0.782870	
A - g	11.416667	0.122820	
A - h	-8.833333	0.214268	
A - i	7.416667	0.287856	

4.13 NON-PARAMETRIC COMPARISONS

The functions for non-parametric multiple comparisons included in 'agricolae' are: `kruskal()`, `waerden.test()`, `friedman()` and `durbin.test()` (Conover, 1999).

The function `kruskal()` is used for N samples (N>2), populations or data coming from a completely random experiment (populations = treatments).

The function `waerden.test()`, similar to kruskal-wallis, uses a normal score instead of ranges as `kruskal` does.

The function `friedman()` is used for organoleptic evaluations of different products, made by judges (every judge evaluates all the products). It can also be used for the analysis of treatments of the randomized complete block design, where the response cannot be treated through the analysis of variance.

The function `durbin.test()` for the analysis of balanced incomplete block designs is very used for sampling tests, where the judges only evaluate a part of the treatments.

Montgomery book data (Montgomery, 2002)
Included in the 'agricolae' package

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

```
str(corn)
```

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

For the examples, the ‘agricolae’ package data will be used.

4.14 KRUSKAL-WALLIS

```
out<-kruskal(observation,method,group=TRUE, main="corn",
console=TRUE)
```

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

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

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

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

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

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

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

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

The object “compares” has the same structure of the comparisons (figure 4.3).

```
par(cex=0.8,mar=c(3,3,1,0))
bar.group(out$groups,ylim=c(0,35),col=colors()[55],las=1)
```

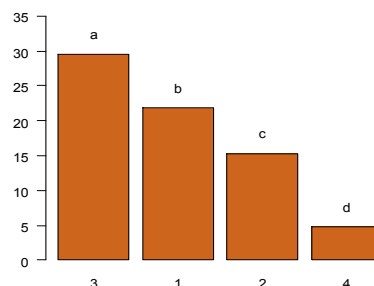


Figure 4.6. Comparison according to Kruskal-Wallis.

4.15 FRIEDMAN

```
friedman()

rm(list=ls())
library(agricolae)
data(grass)
attach(grass)
out<-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

Value: 8.097345

Pvalue chisq : 0.04404214

F value : 3.192198

Pvalue F: 0.03621547

Alpha : 0.05

t-Student : 2.034515

Comparison between treatments

Sum of the ranks

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

4.16 WAERDEN

waerden.test(), with the sweet potato data in the 'agricolae' basis.

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

Study:

Van der Waerden (Normal Scores) test's

Value : 8.409979

Pvalue: 0.03825667

Degrees of freedom: 3

virus, means of the normal score

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

t-Student: 3.355387

Alpha : 0.01

LSD : 1.322487

Means with the same letter are not significantly different

Groups, Treatments and means of the normal score

a	ff	0.6886
a	oo	0.6044
ab	cc	-0.2328
b	fc	-1.06

The comparison probabilities are obtained with the parameter group=FALSE.

> names(out)

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

To see out\$comparison

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

Study:

Van der Waerden (Normal Scores) test's

Value : 8.409979

Pvalue: 0.03825667

Degrees of freedom: 3

virus, means of the normal score

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

Comparison between treatments means
mean of the normal score

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

4.17 DURBIN

`durbin()`; example: Myles Hollander (p. 311) Source: W. Moore and C.I. Bliss. (1942)

```
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=F,console=TRUE,
main="Logarithm of the toxic dose")
```

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

5 STABILITY ANALYSIS

In 'agricolae' there are two methods for the study of stability and the AMMI model. These are: a parametric model for a simultaneous selection in yield and stability "SHUKLA'S STABILITY VARIANCE AND KANG'S", and a non-parametric method of Haynes, based on the data range.

5.1 PARAMETRIC STABILITY

Use the parametric model, function `stability.par()`.

Prepare a data table where the rows and the columns are the genotypes and the environments, respectively. The data should correspond to yield averages or to another measured variable. Determine the variance of the common error for all the environments and the number of repetitions that was evaluated for every genotype. If the repetitions are different, find a harmonious average that will represent the set. Finally, assign a name to each row that will represent the genotype. We will consider five environments in the following example:

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

For 17 genotypes, the identification is made by letters.

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

An error variance of 2 and 4 repetitions is assumed.

```
stability <- stability.par(study, rep=4, MSerror=2)
```

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

YIELD - STABILITY (YSi) STATISTICS

Environmental index - covariate

Analysis of Variance

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

Genotype. Stability statistics

	Mean	Sigma-square	.	s-square	.	Ecovalence
A	7.86	1.671833	ns	2.209084	ns	6.567031
B	7.22	1.822233	ns	1.977299	ns	7.097855
C	7.44	0.233967	ns	0.134103	ns	1.492208
D	7.42	4.079567	ns	1.443859	ns	15.064913
E	7.64	2.037967	ns	2.369090	ns	7.859266
F	7.14	5.161967	*	6.763106	*	18.885149
G	7.90	1.759300	ns	1.058092	ns	6.875737
H	7.68	1.757167	ns	2.028880	ns	6.868208
I	7.34	5.495300	*	0.423680	ns	20.061619
J	7.54	4.129967	ns	5.125514	ns	15.242796
K	7.12	3.848900	ns	4.360772	ns	14.250796
L	7.30	2.675300	ns	3.610982	ns	10.108678
M	7.66	3.473167	ns	2.198229	ns	12.924678
N	7.92	0.806233	ns	1.097156	ns	3.511972
O	8.30	1.951300	ns	1.459578	ns	7.553384
P	6.08	3.647833	ns	4.919102	ns	13.541149
Q	5.88	3.598500	ns	4.353030	ns	13.367031

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

Simultaneous selection for yield and stability (++)

	Yield	Rank	Adj.rank	Adjusted	Stab.var	Stab.rating	YSi	...
A	7.86	14	1	15	1.671833	0	15	+
B	7.22	5	-1	4	1.822233	0	4	
C	7.44	9	1	10	0.233967	0	10	+
D	7.42	8	1	9	4.079567	-2	7	
E	7.64	11	1	12	2.037967	0	12	+
F	7.14	4	-1	3	5.161967	-4	-1	
G	7.90	15	1	16	1.759300	0	16	+
H	7.68	13	1	14	1.757167	0	14	+
I	7.34	7	-1	6	5.495300	-4	2	
J	7.54	10	1	11	4.129967	-2	9	+
K	7.12	3	-1	2	3.848900	0	2	
L	7.30	6	-1	5	2.675300	0	5	
M	7.66	12	1	13	3.473167	0	13	+
N	7.92	16	1	17	0.806233	0	17	+
O	8.30	17	2	19	1.951300	0	19	+
P	6.08	2	-2	0	3.647833	0	0	
Q	5.88	1	-3	-2	3.598500	0	-2	

Yield Mean: 7.378824

YS Mean: 8.352941

LSD (0.05): 0.7384513

- - - - -

+ selected genotype

++ Reference: Kang, M. S. 1993. Simultaneous selection for yield and stability: Consequences for growers. Agron. J. 85:754-757.

The selected genotypes are: A, C, E, G, H, J, M, N and O. These genotypes have a higher yield and a lower variation. According to the ANOVA, the interaction is significant.

If for example there is an environmental index, it can be added as a covariate. For this case, the altitude of the localities is included.

```
altitude<-c(1200, 1300, 800, 1600, 2400)
stability <- stability.par(study,rep=4,MSerror=2, cova=TRUE,
name.cov= "altitude", file.cov= altitude)
```

5.2 NON-PARAMETRIC STABILITY

For non-parametric stability, the function in ‘agricolae’ is `stability.nonpar()`. The names of the genotypes should be included in the first column, and in the other columns, the response by environments.

```
data <- data.frame(name=row.names(study), study)
out<-stability.nonpar(data, "YIELD", ranking=TRUE)
```

Nonparametric Method for Stability Analysis

Estimation and test of nonparametric measures
Variable: YIELD

Ranking...

	v1	v2	v3	v4	v5
A	16.0	8.0	9	14.0	8.0
B	7.5	14.0	5	5.5	10.0
C	7.5	8.0	9	9.0	6.0
D	9.5	6.0	5	5.5	17.0
E	12.5	8.0	11	14.0	3.0
F	2.0	17.0	7	7.0	11.0
G	6.0	13.0	16	16.0	15.0
H	12.5	3.0	15	10.5	6.0
I	4.0	4.5	17	12.0	2.0
J	14.0	15.0	9	2.5	9.0
K	9.5	12.0	13	4.0	1.0
L	5.0	4.5	5	14.0	14.0
M	15.0	16.0	3	8.0	12.5
N	11.0	10.5	12	17.0	12.5
O	17.0	10.5	14	10.5	16.0
P	1.0	2.0	2	1.0	6.0
Q	3.0	1.0	1	2.5	4.0

Statistics...

	Mean	Rank	s1	Z1	s2	Z2
A	7.86	14	5.4	0.02	21.5	0.04
B	7.22	5	6.2	0.12	25.7	0.02
C	7.44	9	3.0	2.73	7.5	1.83
D	7.42	8	7.4	1.20	36.5	1.05
E	7.64	11	5.6	0.00	21.8	0.03
F	7.14	4	7.8	1.81	39.2	1.55
G	7.90	15	5.2	0.08	18.7	0.19
H	7.68	13	6.2	0.12	25.3	0.01
I	7.34	7	8.8	3.87	51.5	5.08
J	7.54	10	7.2	0.94	34.3	0.71
K	7.12	3	7.8	1.81	43.0	2.43
L	7.30	6	7.4	1.20	34.7	0.77
M	7.66	12	7.6	1.49	38.2	1.36


```

N 7.92    16 4.2 0.82 14.8 0.57
O 8.30    17 7.0 0.71 31.7 0.40
P 6.08     2 6.6 0.35 27.7 0.09
Q 5.88     1 7.0 0.71 32.3 0.46

```

```

-----
Sum of Z1:  17.97158
Sum of Z2:  16.59462
-----

```

Test...

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

```

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

```

expectation and variance: es1, es2, vs1, vs2

5.3 AMMI

The model AMMI uses the biplot constructed through the principal components generated by the interaction environment-genotype. If there is such interaction, the percentage of the two principal components would explain more than the 50% of the total variation; in such case, the biplot would be a good alternative to study the interaction environment-genotype.

The data for AMMI should come from similar experiments conducted in different environments. Homogeneity of variance of the experimental error, produced in the different environments, is required. The analysis is done by combining the experiments.

The data can be organized in columns, thus: environment, genotype, repetition, and variable.

The data can also be the averages of the genotypes in each environment, but it is necessary to consider a harmonious average for the repetitions and a common variance of the error. The data should be organized in columns: environment, genotype, and variable.

When performing AMMI, this generates the BIPLLOT graphics; see figure 5.1.

For the application, we consider the data used in the example of parametric stability (study):

```

par(mar=c(4,4,0,0))
rdto <- c(study[,1], study[,2], study[,3], study[,4], study[,5])
environment <- gl(5,17)
genotype <- rep(rownames(study),5)
model<-AMMI(ENV=environment, GEN=genotype, REP=4, Y=rdto, MSE=2,
ylim=c(-2,2), xlim=c(-2,2), number=FALSE)

```

ANALYSIS AMMI: rdto
Class level information

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

Number of means: 85

Dependent Variable: rdto

Analysis of variance

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

Coeff var	Mean rdto
19.16584	7.378824

Analysis

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

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

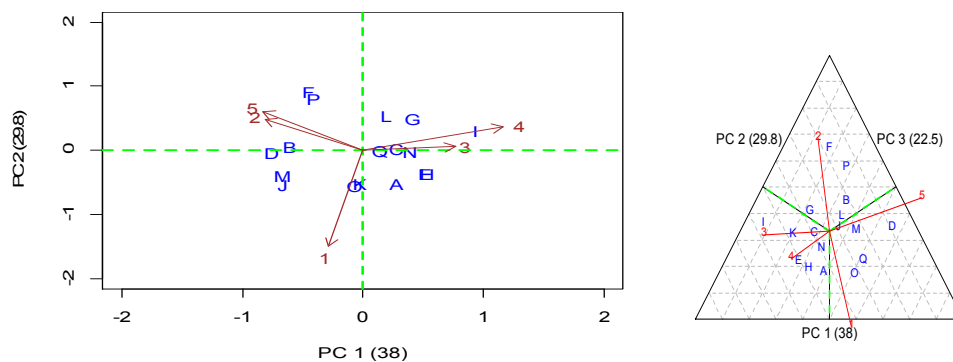


Figure 5.1. Biplot and Triplot

In this case, the interaction is significant. The first two components explain 67.8%; then the biplot can provide information about the interaction genotype-environment. With the triplot, 90.3% would be explained.

6 SPECIAL FUNCTIONS

6.1 CONSENSUS OF DENDROGRAM

Consensus is the degree or similarity of the vertexes of a tree regarding its branches of the constructed dendrogram. The function to apply is `consensus()`.

The data correspond to a table, with the name of the individuals and the variables in the rows and columns respectively. For the demonstration, we will use the “pamCIP” data of ‘agricolae’, which correspond to molecular markers of 43 entries of a germplasm bank (rows) and 107 markers (columns).

The program identifies duplicates in the rows and can operate in both cases. The result is a dendrogram, in which the consensus percentage is included, figure 6.1.

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

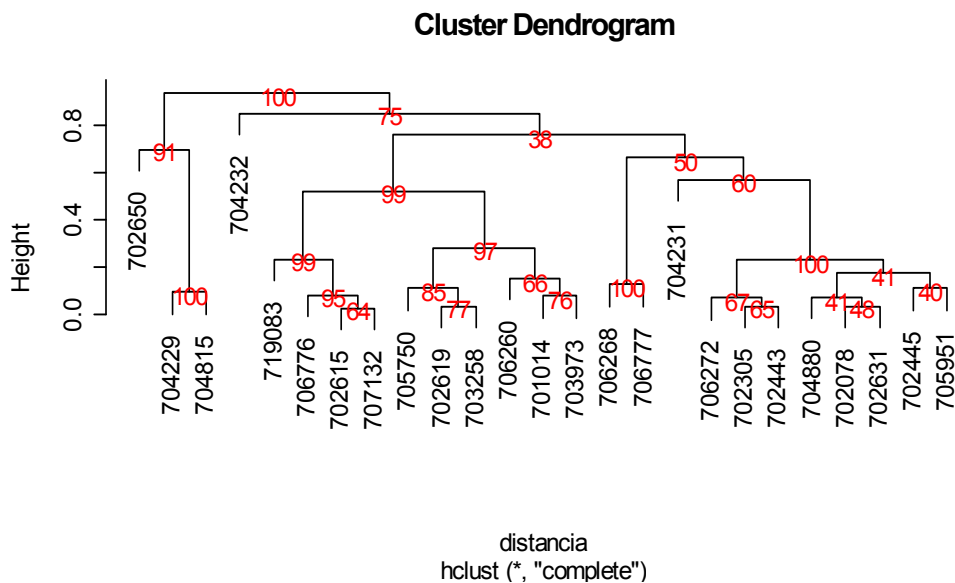


Figure 6.1. Dendrogram, production by consensus()

```
Duplicates: 18
New data   : 25 Records
```

```
Consensus hclust
```

```
Method distance: binary
Method cluster  : complete
rows and cols   : 25 107
n-bootstrap     : 500
Run time        : 20.469 secs
```

When the dendrogram is complex, it is convenient to extract part of it with the function `hcut()`, figure 6.2.

```
hcut(output,h=0.4,group=8,type="t",edgePar = list(lty=1:2,
col=colors()[c(42,84)]),main="group 8",
,col.text="blue",cex.text=1,las=1)
```

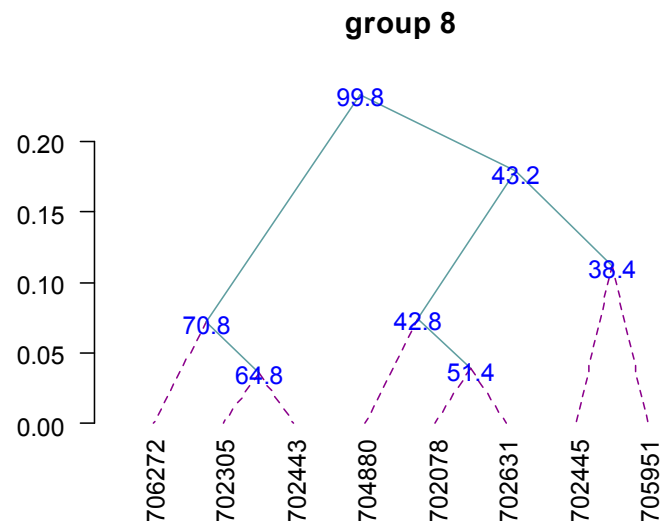


Figure 6.2. Dendrogram, production by `hcut()`

The obtained object "output" contains information about the process:

```
names(output)
```

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

This means that we can know the duplicates, reconstruct the tree diagram and maintain the interactions.

```
output$table.dend
```

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

Reproduce the dendrogram:

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

Construct a classic dendrogram, figure 6.3

```

par(mar=c(4,3,1,1))
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",cex=1)

```

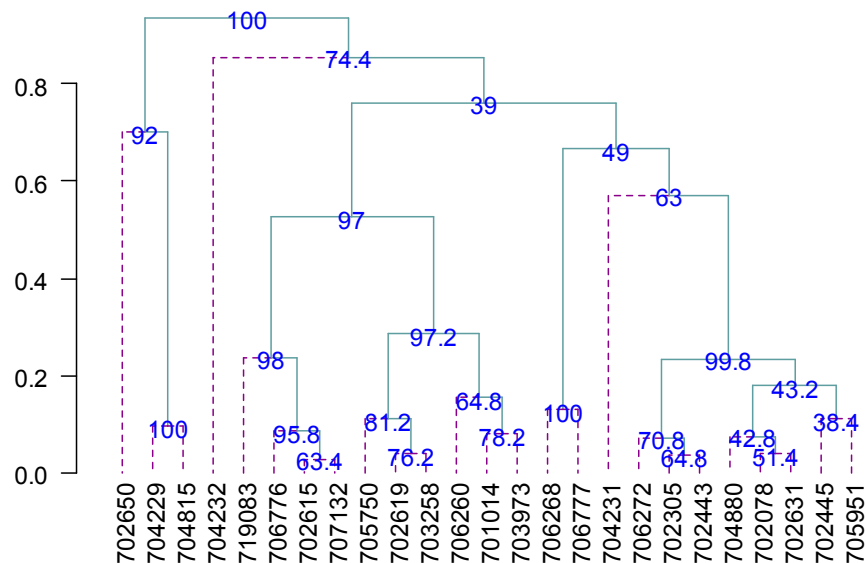


Figure 6.3. Classic dendrogram

6.2 MONTECARLO

It is a method for generating random numbers of an unknown distribution. It uses a data set and, through the cumulative behavior of its relative frequency, generates the possible random values that follow the data distribution. These new numbers are used in some simulation process.

The probability density of the original and simulated data can be compared, figure 6.4.

```

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

```

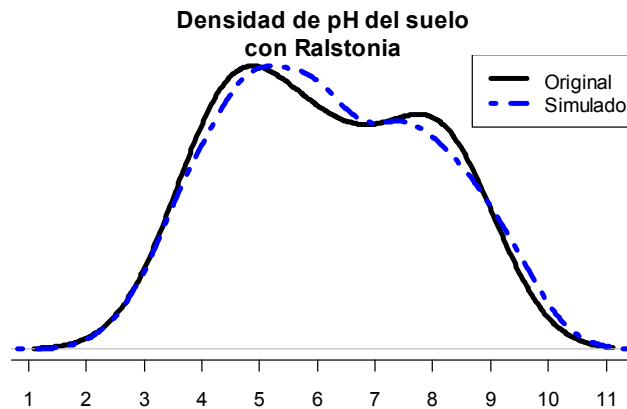


Figure 6.4. Distribution of the simulated and the original data

1000 data have been generated, being the frequency table:

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

Lower	Upper	Main	freq	relative	CF	RCF
2.00	2.79	2.40	12	0.01	12	0.01
2.79	3.58	3.19	50	0.05	62	0.06
...
9.11	9.90	9.51	49	0.05	989	0.99
9.90	10.69	10.30	11	0.01	1000	1.00

Some statistics:

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

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

```
summary(simulated)
```

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

6.3 RE-SAMPLING IN LINEAR MODEL

It uses the permutation method for the calculation of the probabilities of the sources of variation of ANOVA according to the linear regression model or the design used. The principle is that the Y response does not depend on the averages proposed in the model; hence, the Y values can be permuted and many model estimates can be constructed. On the basis of the patterns of the random variables of the elements under study, the probability is calculated in order to measure the significance.

For a variance analysis, the data should be prepared similarly. The function to use is: `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=1000)
```

```

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

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

```

The function `resampling.model()` can be used when the errors have a different distribution from normal.

6.4 SIMULATION IN LINEAR MODEL

Under the assumption of normality, the function generates pseudo experimental errors under the proposed model, and determines the proportion of valid results according to the analysis of variance found.

The function is: `simulation.model()`. The data are prepared in a table, similarly to an analysis of variance.

Considering the example proposed in the previous procedure:

```

model <- simulation.model(model, potato, k=1000)

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

Response: cutting
      Df Sum Sq Mean Sq F value    Pr(>F)
variety    1 25.087  25.087  7.2580 0.01952 *
date       2 13.892   6.946  2.0096 0.17671
variety:date 2  4.853   2.427  0.7020 0.51484
Residuals 12 41.477   3.456
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
---
Validation of the analysis of variance for the proposed model
Simulations: 1000

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

```

The validation is referred to the percentage of decision results equal to the result of the ANOVA decision. Thus, 67.5% of the results simulated on the interaction variety*date gave the same result of acceptance or rejection obtained in the ANOVA.

6.5 PATH ANALYSIS

It corresponds to the “path analysis” method. The data correspond to correlation matrices of the independent ones with the dependent matrix (XY) and between the independent ones (XX).

It is necessary to assign names to the rows and columns in order to identify the direct and indirect effects.

```
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 0.33333333 0.26666667
X2 0.16666667 0.53333333
```

Residual Effect^2 = 0.42666667

> output

```
$Coeff
              X1          X2
X1 0.33333333 0.26666667
X2 0.16666667 0.53333333
```

```
$Residual
[1] 0.42666667
```

6.6 LINE X TESTER

It corresponds to a crossbreeding analysis of a genetic design. The data should be organized in a table. Only four columns are required: repetition, females, males, and response. In case it corresponds to progenitors, the females or males field will only be filled with the corresponding one. See the heterosis data.

Example with the heterosis data, locality 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

where <NA> is empty.

If it is a progeny, it comes from a "Female" and a "Male."

If it is a progenitor, it will only be "Female" or "Male."

The following example corresponds to data of the locality 2:

24 progenies
8 females
3 males
3 repetitions

They are 35 treatments (24, 8, 3) applied to three blocks.

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

ANALYSIS LINE x TESTER: v2

ANOVA with parents and crosses

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

ANOVA for line X tester analysis

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

ANOVA for line X tester analysis including parents

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

```

GCA Effects:
=====
Lines Effects:
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

F = 0, Additive genetic variance: 0.01546921
F = 1, Additive genetic variance: 0.007734604
F = 0, Variance due to Dominance: 0.1128228
F = 1, Variance due to Dominance: 0.05641141

Proportional contribution of lines, testers
and their interactions to total variance
=====
Contributions of lines : 59.48026
Contributions of testers: 7.763104
Contributions of lxt : 32.75663

```

6.7 SOIL UNIFORMITY

The Smith index is an indicator of the uniformity, used to determine the parcel size for research purposes. The data correspond to a matrix or table that contains the response per basic unit, a number of n rows x m columns, and a total of n*m basic units.

For the test, we will use the rice file. The graphic is a result with the adjustment of a model for the parcel size and the coefficient of variation, figure 6.5.

```
data(rice)
table<-index.smith(rice,
  main="Interaction between the CV and the parcel size" ,col="red",
  type="l",xlab="Size")
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
...						
40	162	9	18	4	4009.765	8.6

The size is the product of the width x the length of the parcel, and the rectangle size is the product of the width x the length.

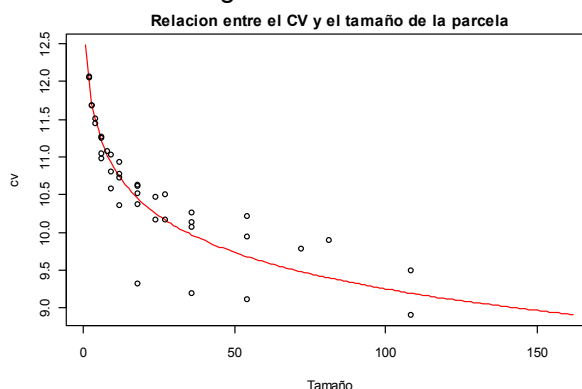


Figure 6.5. Adjustment curve for the optimal size of parcel

6.8 CONFIDENCE LIMITS IN BIODIVERSITY INDICES

The biodiversity indices are widely used for measuring the presence of living things in an ecological area. Many programs indicate their value. The function of 'agricolae' is also to show the confidence intervals, which can be used for a statistical comparison. Use the bootstrap procedure. The data are organized in a table; the species are placed in a column; and in another one, the number of individuals. The indices that can be calculated with the function `index.bio()` of 'agricolae' are: "Margalef", "Simpson.Dom", "Simpson.Div", "Berger.Parker", "McIntosh", and "Shannon."

In the example below, we will use the data obtained in the locality of Paracsho, district of Huasahuasi, province of Tarma in the department of Junín.

The evaluation was carried out in the parcels on 17 November 2005, without insecticide application. The counted specimens were the following:

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

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

The Shannon index is:

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

```
Method: Shannon
Index: 3.52304
```

```
95 percent confidence interval:
 3.088775 ; 4.286088
```

6.9 CORRELATION

The function `correlation()` of 'agricolae' makes the correlations through the methods of Pearson, Spearman and Kendall for vectors and/or matrices. If they are two vectors, the test is carried out for one or two lines; if it is a matrix one, it determines the probabilities for a difference, whether it is greater or smaller.

For its application, consider the soil data: `data(soil)`

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

Correlation Analysis

```
Method      : pearson
Alternative: two.sided
```

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

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

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

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

Correlation Analysis

```
Method      : pearson
Alternative: two.sided
```

```
$correlation
      EC CaCO3
pH 0.55  0.73
$pvalue
      EC  CaCO3
pH 0.0525 0.0048
$n.obs
[1] 13
```

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

Pearson's product-moment correlation

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

6.10 OTHER FUNCTIONS

Desirability functions that facilitate the data management:

`tapply.stat()` Calculation of statistical and mathematical operations in columns of a table in relation to grouped factors.

Factor and variable table

Application with 'agricolae' data:

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

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

It corresponds to the range of variation in the farmers' yield.

The function "tapply" can be used directly or with function.

If A is a table with columns 1,2 and 3 as category, and 5,6 and 7 as variables, then the following procedures are valid:

```
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))
```

Coefficient of variation of an experiment

If “model” is the object resulting from an analysis of variance of the function aov() or lm() of R, then the function cv.model() calculates the coefficient of variation.

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

Skewness and curtosis

The skewness and curtosis results, obtained by ‘agricolae’, are equal to the ones obtained by SAS, MiniTab, SPSS, InfoStat, and Excel.

If x represents a data set:

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

skewness is calculated with:

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

and curtosis with:

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

Tabular value of Waller-Duncan

The function Waller determines the tabular value of Waller-Duncan. For the calculation, value F is necessary, calculated from the analysis of variance of the study factor, with its freedom degrees and the estimate of the variance of the experimental error. Value K, parameter of the function is the ratio between the two types of errors (I and II). To use it, a value associated with the alpha level is assigned. When the alpha level is 0.10, 50 is assigned to K; for 0.05, K=100; and for 0.01, K=500. K can take any value.

Figure 6.6 illustrates the function for different values of K with freedom degrees of 5 for the numerator and 15 for the denominator, and values of calculated F, equal to 2, 4, and 8.

```
q<-5
f<-15
K<-seq(10,1000,100)
n<-length(K)
y<-rep(0,3*n)
dim(y)<-c(n,3)
for(i in 1:n) y[i,1]<-waller(K[i],q,f,Fc=2)
for(i in 1:n) y[i,2]<-waller(K[i],q,f,Fc=4)
```

```

for(i in 1:n) y[i,3]<-waller(K[i],q,f,Fc=8)
plot(K,y[,1],type="l",col="blue",ylab="waller")
lines(K,y[,2],type="l",col="red",lty=2,lwd=2)
lines(K,y[,3],type="l",col="green",lty=4,lwd=2)
legend("topleft",c("2","4","8"),col=c("blue","red","green"),lty=c(1,
8,20),lwd=2,title="Fc")
title(main="Waller in function of K")

```

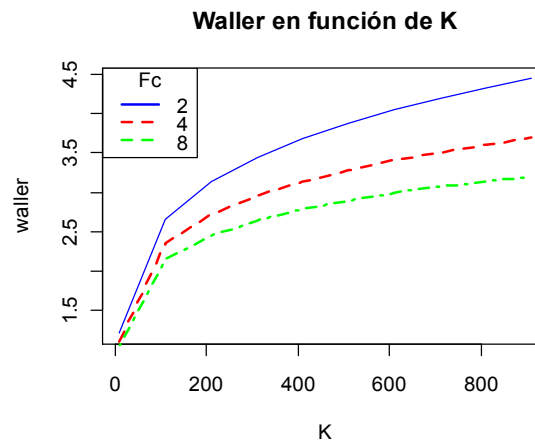


Figure 6.6. Function of Waller to different value of parameters K and Fc

AUDPC

The area under the disease progress curve (AUDPC), (see figure 6.7), calculates the absolute and relative progress of the disease. It is required to measure the disease in percentage terms during several dates, preferably equidistantly.

```

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=F,col=
colors()[42],xlab="Days", ylab="Evaluation")
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)
audpc(evaluation,days,"relative")
text(15,80,"Audpc Absolute = 2030")
text(15,70,"Audpc Relative = 0.58")

```

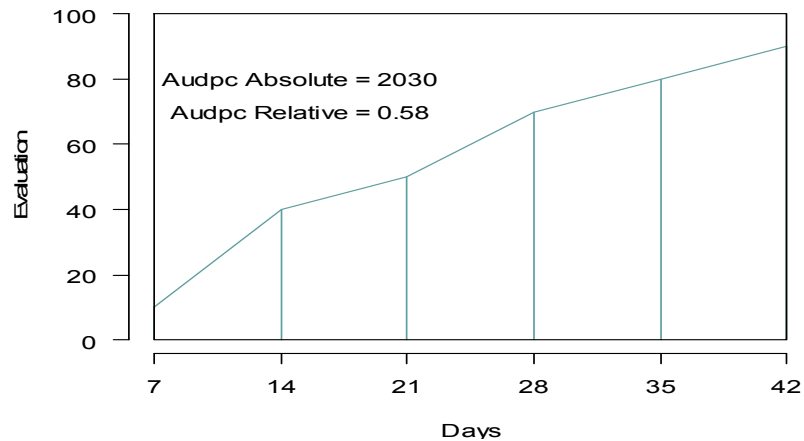


Figure 6.7. AUDPC: Area under the curve

NON-ADDITIVITY

Tukey's test for non-additivity is used when there are doubts about the additivity veracity of a model. This test confirms such assumption and it is expected to accept the null hypothesis of the non-additive effect of the model.

For this test, all the experimental data used in the estimation of the linear additive model are required.

Use the function `nonadditivity()` of 'agricolae'. For its demonstration, the experimental data "potato", of the package 'agricolae', will be used. In this case, the model corresponds to the randomized complete block design, where the treatments are the varieties.

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

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

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

```
Analysis of Variance Table
```

```
Response: residual
```

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

According to the results, the model is additive because the p.value 0.35 is greater than 0.05.

LATEBLIGHT

LATEBLIGHT is a mathematical model that simulates the effect of weather, host growth and resistance, and fungicide use on asexual development and growth of *Phytophthora infestans* on potato foliage.

LATEBLIGHT Version LB2004 was created in October 2004 (Andrade-Piedra et al., 2005a, b and c), based on the C-version written by B.E. Ticknor ('BET 21191 modification of cbm8d29.c'), reported by Doster et al. (1990) and described in detail by Fry et al. (1991) (This version is referred as LB1990 by Andrade-Piedra et al. [2005a]). The first version of LATEBLIGHT was developed by Bruhn and Fry (1981) and described in detail by Bruhn et al. (1980).

```
library(agricolae)
f <- system.file("external/weather.csv", package="agricolae")
weather <- read.csv(f,header=FALSE)
f <- system.file("external/severity.csv", package="agricolae")
severity <- read.csv(f)
weather[,1]<-as.Date(weather[,1],format = "%m/%d/%Y")
# Parameters dates
dates<-c("2000-03-25","2000-04-09","2000-04-12","2000-04-16","2000-
04-22")
dates<-as.Date(dates)
EmergDate <- as.Date('2000/01/19')
EndEpidDate <- as.Date("2000-04-22")
dates<-as.Date(dates)
NoReadingsH<- 1
RHthreshold <- 90
WS<-weatherSeverity(weather,severity,dates,EmergDate,EndEpidDate,
NoReadingsH,RHthreshold)
# Parameters Lateblight
InocDate<-"2000-03-18"
LGR <- 0.00410
IniSpor <- 0
SR <- 292000000
IE <- 1.0
LP <- 2.82
InMicCol <- 9
Cultivar <- 'NICOLA'
ApplSys <- "NOFUNGICIDE"
main<-"Cultivar: NICOLA"
#-----
model<-lateblight(WS, Cultivar,ApplSys, InocDate, LGR,IniSpor,SR,IE,
LP,MatTime='LATESEASON',InMicCol,main=main,type="l",xlim=c(65,95),lwd=1.5,xlab="Time (days after emergence)", ylab="Severity
(Percentage)")

> head(model$Gfile)
```

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

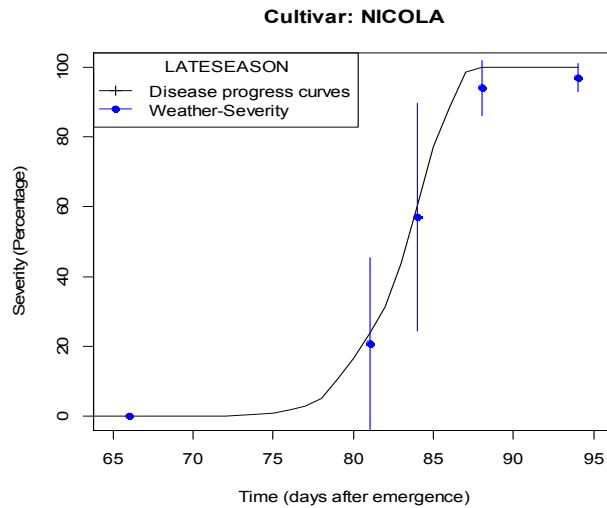


Figure 6.8. lateblight: LATESEASON

```
# reproduce graph
x<- model$Ofile$nday
y<- model$Ofile$SimSeverity
w<- model$Gfile$nday
z<- model$Gfile$MeanSeverity
Min<-model$Gfile$MinObs
Max<-model$Gfile$MaxObs
plot(x,y,type="l",xlim=c(65,95),lwd=1.5,xlab="Time (days after
emergence)",
ylab="Severity (Percentage)")
points(w,z,col="red",cex=1,pch=19)
npoints <- length(w)
for ( i in 1:npoints){
segments(w[i],Min[i],w[i],Max[i],lwd=1.5,col="red")
}
legend("topleft",c("Disease progress curves","Weather-Severity"),
title="Description",lty=1,pch=c(3,19),col=c("black","red"))
```

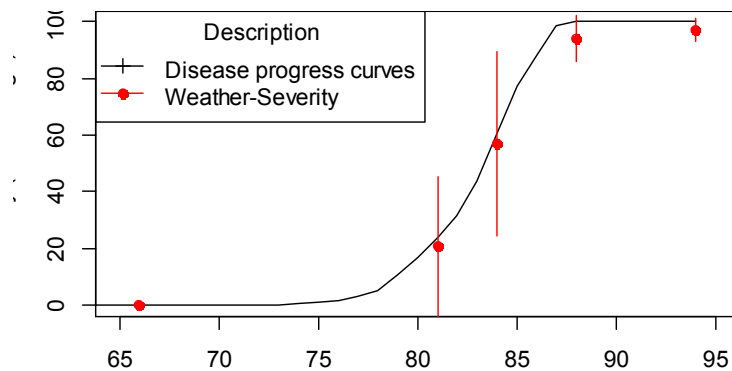


Figure 6.9. lateblight: LATESEASON

Table 6.10. ASCII Character Code Reference for the use of symbols

ASCII Codes Table used in R					
Code	Symbol	Code	Symbol	Code	Symbol
92	}	124	 	64	@
47	/	60	<	94	^
91	[62	>	35	#
93]	61	=	36	\$
40	(34	"	37	%
41)	126	~	38	&
123	{	58	:	39	'
125	}	59	;		

REFERENCES

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