# agricolae tutorial (Version 1.2-1)

*Felipe de Mendiburu*[(1)]

2014-09-01

## Contents

# List of Figures



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

# Preface

The following document was developed to facilitate the use of agricolae package in R, it is understood that the user knows the statistical methodology for the design and analysis of experiments and through the use of the functions programmed in agricolae facilitate the generation of the field book experimental design and their analysis. The first part document describes the use of graph.freq role is complementary to the *hist* function of R functions to facilitate the collection of statistics and frequency table, statistics or grouped data histogram based training grouped data and graphics as frequency polygon or ogive; second part is the development of experimental plans and numbering of the units as used in an agricultural experiment; a third part corresponding to the comparative tests and finally provides agricolae miscellaneous additional functions applied in agricultural research and stability functions, soil consistency, late blight simulation and others.

# 1 Introduction

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, split plot and strip plot. 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, Median and Waerden, stability analysis, and other procedures applied in genetics, as well as procedures in biodiversity and descriptive statistics. reference [4]

## 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. Reference [13]

`> install.packages("agricolae")` Once the `agricolae` package is installed, it needs to be made accessible to the current **R** session by the command:

```
> library(agricolae)
```

For online help facilities or the details of a particular command (such as the function `waller.test`) you can type:

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

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*
**Cluster:** for the use of the function *consensus*
**spdep:** for the between genotypes spatial relation in biplot of the function *AMMI*

### 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:

```
> detach(package:agricolae) # detach package agricole
> library(agricolae) # Load the package to the memory
> designs<-apropos("design")
> print(designs[substr(designs,1,6)=="design"], row.names=FALSE)

 [1] "design.ab"      "design.alpha"   "design.bib"
 [4] "design.crd"     "design.cyclic"  "design.dau"
 [7] "design.graeco"  "design.lattice" "design.lsd"
[10] "design.rcbd"    "design.split"   "design.strip"
[13] "design.youden"

For the use of symbols that do not appear in the keyboard in Spanish, such as:

~, [, ], &, ^, |. <, >, {, }, \% or others, use the table ASCII code.

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

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

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

### 1.3 Data set in agricolae

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

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

## 2 Descriptive statistics

The package **agricolae** provides some complementary functions to the **R** program, specifically for the management of the histogram and function *hist*.

## 2.1 Histogram

The histogram is constructed with the function *graph.freq* and is associated to other functions: *polygon.freq, table.freq, stat.freq*. See Figures: 1, 2 and 3 for more details.

Example. Data generated in **R** . (students' weight).

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

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

> par(mfrow=c(1,2),mar=c(4,3,0,1),cex=0.6)
> h1<- graph.freq(weight,col="yellow",frequency=1,las=2,xlab="h1")
> h2<- graph.freq (weight, frequency =2, axes= FALSE,las=2,xlab="h2")
> 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)
```

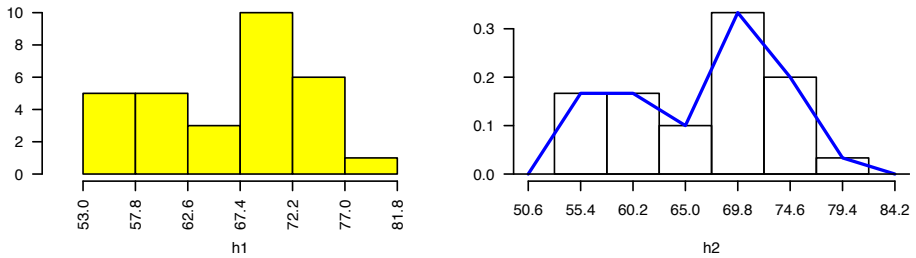


Figure 1: Absolute and relative frequency with polygon.

## 2.2 Statistics and Frequency tables

Statistics: mean, median, mode and standard deviation of the grouped data.

```
> stat.freq(h1)

$variance
[1] 51.37655

$mean
[1] 66.6

$median
[1] 68.36

$mode
```

```
       [-   -]     mode
[1,] 67.4 72.2 70.45455
```

Frequency tables: Use *table.freq*, *stat.freq* and *summary*

The table.freq is equal to summary()

Limits class: **Lower and Upper**
Class point: **Main**
Frequency: **freq**
Relative frequency: **relative**
Cumulative frequency: **CF**
Cumulative relative frequency: **RCF**

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

```
     Lower Upper Main freq   relative CF       RCF
[1,]  53.0  57.8 55.4    5 0.16666667  5 0.1666667
[2,]  57.8  62.6 60.2    5 0.16666667 10 0.3333333
[3,]  62.6  67.4 65.0    3 0.10000000 13 0.4333333
[4,]  67.4  72.2 69.8   10 0.33333333 23 0.7666667
[5,]  72.2  77.0 74.6    6 0.20000000 29 0.9666667
[6,]  77.0  81.8 79.4    1 0.03333333 30 1.0000000
```

## 2.3 Histogram manipulation functions

You can extract information from a histogram such as class intervals *intervals.freq*, attract new intervals with the *sturges.freq* function or to join classes with *join.freq* function. It is also possible to reproduce the graph with the same creator *graph.freq* or function *plot* and overlay normal function with *normal.freq* be it a histogram in absolute scale, relative or density . The following examples illustrates these properties.

```
> sturges.freq(weight)
```

```
$maximum
[1] 81.5

$minimum
[1] 53

$amplitude
[1] 29

$classes
[1] 6

$interval
[1] 4.8

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

```
> intervals.freq(h1)

     lower upper
[1,]  53.0  57.8
[2,]  57.8  62.6
[3,]  62.6  67.4
[4,]  67.4  72.2
[5,]  72.2  77.0
[6,]  77.0  81.8

> join.freq(h1,1:3) -> h3
> print(summary(h3))

     Lower Upper Main freq   relative CF       RCF
[1,]  53.0  67.4 60.2   13 0.43333333 13 0.4333333
[2,]  67.4  72.2 69.8   10 0.33333333 23 0.7666667
[3,]  72.2  77.0 74.6    6 0.20000000 29 0.9666667
[4,]  77.0  81.8 79.4    1 0.03333333 30 1.0000000

> par(mfrow=c(1,2),mar=c(4,3,0,1),cex=0.6)
> plot(h3, frequency=2,col="magenta",ylim=c(0,0.6))
> normal.freq(h3,frequency=2,col="green")
> ogive.freq(h3,col="blue")

     x    RCF
1 53.0 0.0000
2 67.4 0.4333
3 72.2 0.7667
4 77.0 0.9667
5 81.8 1.0000
6 86.6 1.0000
```

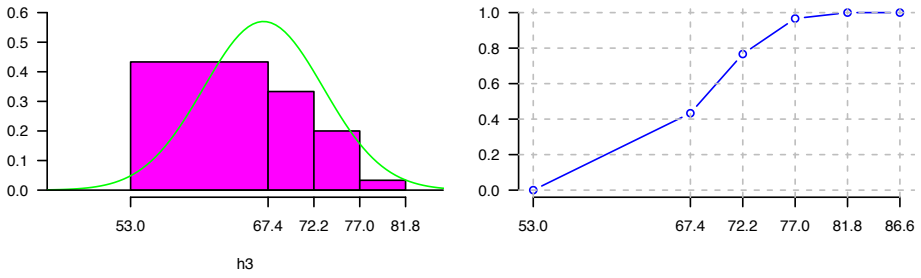


Figure 2: Join frequency and relative frequency with normal and Ogive.

## 2.4 hist() and graph.freq() based on grouped data

The *hist* and *graph.freq* have the same characteristics, only f2 allows build histogram from grouped data.

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

```
> par(mfrow=c(1,2),mar=c(4,3,2,1),cex=0.6)
> h4<-hist(weight,xlab="Classes (h4)")
> table.freq(h4)
> # this is possible
> # hh<-graph.freq(h4,plot=FALSE)
> # summary(hh)
> # new class
> classes <- c(0, 10, 20, 30, 40, 50)
> freq <- c(3, 8, 15, 18, 6)
> h5 <- graph.freq(classes,counts=freq, xlab="Classes (h5)",main="Histogram grouped data")
```

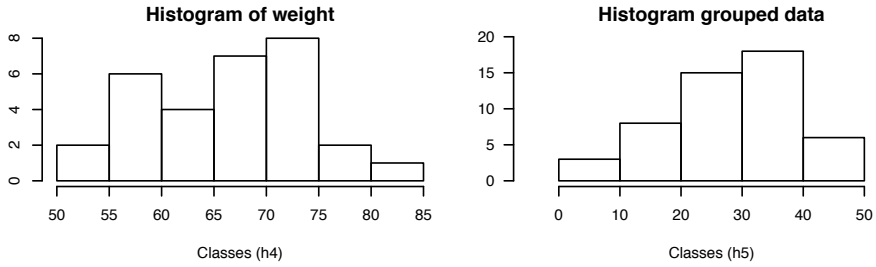


Figure 3: hist() function and histogram defined class

```
> print(summary(h5))
```

```
     Lower Upper Main freq relative CF  RCF
[1,]     0    10    5    3     0.06  3 0.06
[2,]    10    20   15    8     0.16 11 0.22
[3,]    20    30   25   15     0.30 26 0.52
[4,]    30    40   35   18     0.36 44 0.88
[5,]    40    50   45    6     0.12 50 1.00
```

# 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 refrerences [1, 8, 9, 10]. 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 **agricolae** .

*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 sequencial.

**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: The zigzag 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 design

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

```
> str(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
> head(book1)

  plots r trt
1     1 1   B
2     2 1   A
3     3 2   A
4     4 1   C
5     5 2   C
6     6 3   A
```

Excel:write.csv(book1,**"book1.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:

```
> str(design.rcbd)

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

> trt <- c("A", "B", "C","D","E")
> repeticion <- 4
> outdesign <- design.rcbd(trt,r=repeticion, seed=-513, serie=2)
> # book2 <- outdesign$book
> book2<- zigzag(outdesign) # zigzag numeration
> print(t(matrix(book2[,3],c(5,4))))

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

> print(t(matrix(book2[,1],c(5,4))),digits=0)

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

## 3.3 Latin square design

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

```
> str(design.lsd)

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

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

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

**Serpentine enumeration:**

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

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

## 3.4 Graeco-Latin designs

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

```
> str(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
> print(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)
> print(t(matrix(book[,1],c(4,4))),digit=0)

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

## 3.5 Youden design

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

```
> str(design.youden)

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

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

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

> plots <-as.numeric(youden[,1])
> trt <-as.character(youden[,4])
> dim(plots)<-c(3,4)
> dim(trt) <-c(3,4)
> print(t(plots))

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

> print(t(trt))

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

**Serpentine enumeration:**

```
> book <- zigzag(outdesign)
> print(t(matrix(book[,1],c(3,4))),digit=0)

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

## 3.6 Balanced Incomplete Block Designs

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

```
> str(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
treatmeans : 5
Block size : 4
Blocks     : 5
Replication: 4

Efficiency factor 0.9375

<<< Book >>>

> book5 <- outdesign$book
> outdesign$statistics

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

> outdesign$parameters

$design
[1] "bib"

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

$k
[1] 4

$serie
[1] 2

$seed
[1] 543

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

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,]  204  203  202  201
[3,]  301  302  303  304
[4,]  404  403  402  401
[5,]  501  502  503  504
```

## 3.7 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:

```
> str(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
==================
treatmeans : 6
Block size : 3
Replication: 6
```

```
> book6 <- outdesign$book
> outdesign$sketch[[1]]

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

> outdesign$sketch[[2]]

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

12 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
```

## 3.8 Lattice designs

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

```
> str(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

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

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

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

```
> head(book7)
```

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

**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
```

## 3.9 Alpha designs

These designs are generated by the alpha arrangements reference [11]. They are similar to the lattice designs, but the tables are rectangular, with s blocks x k treatments. The number of treatments should be equal to s*k and all the experimental units, r*s*k and its parameters are:

```
> str(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"

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

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

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

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

**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
```

## 3.10 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:

```
> str(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)
> head(book)
```

```
  plots block trt
1   101     1   D
2   102     1   C
3   103     1   A
4   104     1   u
5   105     1   B
6   106     1   t
```

For augmented ompletely randomized design, use the function design.crd().

## 3.11 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:

```
> str(design.split)

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

**Aplication**

```
> 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
> head(book10)

  plots splots block trt1 trt2
1   101      1     1    A    c
2   101      2     1    A    a
3   101      3     1    A    b
4   102      1     1    D    b
5   102      2     1    D    c
6   102      3     1    D    a

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

**In plots:**

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

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

**Ind sub plots (split plot)**

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

```
     [,1]    [,2]    [,3]    [,4]
[1,] "c a b" "b c a" "b c a" "a b c"
[2,] "b a c" "a b c" "a c b" "b c a"
[3,] "a b c" "a c b" "a c b" "c a b"
```

**Serpentine enumeration:**

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

```
  plots splots block trt1 trt2
1   101      1     1    A    c
2   101      2     1    A    a
3   101      3     1    A    b
4   102      1     1    D    b
5   102      2     1    D    c
```

## 3.12 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:

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

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

**Aplication**

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

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

```
> t3<-paste(book11$trt1, book11$trt2)
> B1<-t(matrix(t3[1:12],c(4,3)))
> B2<-t(matrix(t3[13:24],c(3,4)))
> B3<-t(matrix(t3[25:36],c(3,4)))
> print(B1)
```

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

```
> print(B2)
```

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

```
> print(B3)
```

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

**Serpentine enumeration:**

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

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

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

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

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

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

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

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

## 3.13 Factorial

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

```
> str(design.ab)

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

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).

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

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

**Serpentine enumeration:**

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

  plots block A B
1   101     1 3 1
2   102     1 2 2
```

```
3   103     1 1 1
4   104     1 1 2
5   105     1 3 2
6   106     1 2 1
```

factorial 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")
> names(crd)

[1] "parameters" "book"

> crd$parameters

$design
[1] "factorial"

$trt
[1] "1 1 1" "1 1 2" "1 2 1" "1 2 2" "2 1 1" "2 1 2" "2 2 1"
[8] "2 2 2"

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

$serie
[1] 2

$seed
[1] 970386955

$kinds
[1] "Super-Duper"

$applied
[1] "crd"

> head(crd$book)

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

# 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* reference [16] and *durbin.test, kruskal, friedman, waerden.test and*

*Median.test* reference [2].

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] 17.1666

> attach(sweetpotato)
> mean(yield)

[1] 27.625

> detach(sweetpotato)
```

**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), reference [16].

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

Study: model ~ "virus"

LSD t Test for yield

Mean Square Error:  22.48917

virus,  means and individual ( 95 %) CI

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

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

```
Least Significant Difference 8.928965
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)
> outLSD <-LSD.test(model, "virus", group=F,console=TRUE)

Study: model ~ "virus"

LSD t Test for yield

Mean Square Error:  22.48917

virus,  means and individual ( 95 %) CI

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

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

Comparison between treatments means

         Difference       pvalue 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
```

Signif. codes:

**0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1**

```
> print(outLSD)

$statistics
    Mean      CV  MSerror
```

```
  27.625 17.1666 22.48917

$parameters
  Df ntr  t.value
   8   4 2.306004

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

$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)

Study: model ~ "virus"

LSD t Test for yield
P value adjustment method: bonferroni

Mean Square Error:  22.48917

virus,  means and individual ( 95 %) CI

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

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

```
Comparison between treatments means

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

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 reference [16].

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

Study: model ~ "virus"

Duncan's new multiple range test
for yield

Mean Square Error:  22.48917

virus,  means

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

alpha: 0.05 ; Df Error: 8

Critical Range
       2        3        4
8.928965 9.304825 9.514910

Means with the same letter are not significantly different.

Groups, Treatments and means
a          oo          36.9
a          ff          36.33
```

```
b          cc          24.4
c          fc          12.87
```

## 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 reference [16]

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

Study: model ~ "virus"

Student Newman Keuls Test
for yield

Mean Square Error:  22.48917

virus,  means

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

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 reference [16]

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

```
Study: model ~ "virus"

HSD Test for yield

Mean Square Error:  22.48917

virus,  means

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

alpha: 0.05 ; Df Error: 8
Critical Value of Studentized Range: 4.52881

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

> outHSD

$statistics
    Mean      CV  MSerror      HSD
  27.625 17.1666 22.48917 12.39967

$parameters
  Df ntr StudentizedRange
   8   4          4.52881

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

$comparison
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 reference [16]

```
> # 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 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

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

Study: yield ~ virus

Waller-Duncan K-ratio t Test for yield

This test minimizes the Bayes risk under additive
loss and certain other assumptions.
                            ......
K ratio                   100.00000
Error Degrees of Freedom    8.00000
Error Mean Square          22.48917
F value                    17.34500
Critical Value of Waller    2.23600

virus,  means

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

Minimum Significant Difference 8.657906
Comparison between treatments means

         Difference significant
```

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

```
> detach(sweetpotato)
```

In another case with only invoking the model object:

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

The found object *outWaller* has information to make other procedures.

```
> names(outWaller)
```

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

```
> print(outWaller$comparison)
```

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

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

```
> outWaller$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 reference [16]

```
> # analysis of variance:
> scheffe.test(model,"virus", group=TRUE,console=TRUE,
+ main="Yield of sweetpotato\nDealt with different virus")
```

```
Study: Yield of sweetpotato
Dealt with different virus
```

```
Scheffe Test for yield

Mean Square Error  : 22.48917

virus,  means

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

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

Minimum Significant Difference: 13.52368

Means with the same letter are not significantly different.

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*.

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

Study: model ~ "virus"

Scheffe Test for yield

Mean Square Error  : 22.48917

virus,  means

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

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

Comparison between treatments means

         Difference   pvalue sig        LCL         UCL
```

```
cc - fc  11.5333333 0.097816   .  -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. Comparetive tests: *LSD, HSD, Waller-Duncan, Duncan, Scheffé, SNK* can be applied.

```
> # modelABC <-aov (y ~ A * B * C, data)
> # compare <-LSD.test (modelABC, 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
```

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

```
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 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

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

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

LSD t Test for yield

Mean Square Error:  1.1875

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

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

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

Least Significant Difference 1.590341
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(outFactorial$means,variation="range",ylim=c(5,25), bar=FALSE,col=0,las=1)
> points(pic1$index,pic1$means,pch=18,cex=1.5,col="blue")
> axis(1,pic1$index,labels=FALSE)
> title(main="average and range\nclon:nitrogen")
```

## 4.9 Analysis of Balanced Incomplete Blocks

This analysis can come from balanced or partially balanced designs. The function *BIB.test* is for balanced designs, and *BIB.test*, for partially balanced designs. In the following example, the **agricolae** data will be used, reference [6].

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

ANALYSIS BIB:  y
Class level information

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 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

coefficient of variation: 12.6 %
y Means: 60.3

variety,  statistics

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

LSD test
Std.diff   : 5.363111
Alpha      : 0.05
LSD        : 11.4312
```

```
Parameters BIB
Lambda     : 2
treatmeans : 6
Block size : 3
Blocks     : 10
Replication: 5

Efficiency factor 0.8

<<< Book >>>

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 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

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

```
variety,  statistics

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

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

Efficiency factor 0.8

<<< Book >>>

Comparison between treatments means
      Difference   pvalue sig.
1 - 2 16.4166667 0.070509    .
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

> names(out)

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

> rm(block,variety)
```

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

## 4.10 Partially Balanced Incomplete Blocks

The function *PBIB.test*, reference [6], 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.

```
> # 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)
> modelPBIB <- PBIB.test(block, Genotype, replication, yield, k=3, group=TRUE,
+ console=TRUE)

ANALYSIS PBIB:  yield

Class level information
block : 20
Genotype : 30

Number of observations:  60
```

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

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

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

Analysis of Variance Table

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

coefficient of variation: 31.2 %
yield Means: 4.533333

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

Efficiency factor 0.6170213

Comparison test lsd

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

> detach(data)
```

**The adjusted averages can be extracted from the modelPBIB.**

head(modelPBIB$means)

**The comparisons:**

head(modelPBIB$comparison)

The data on the adjusted averages and their variation can be illustrated see Figure 6. since the created object is very similar to the objects generated by the multiple comparisons.

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

Create the data in a text file: latice3x3.txt and read with R:

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

```
> rm(trt)
> A<-read.table("lattice3X3.txt", header=T)
> attach(A)
> modelLattice<-PBIB.test(block,trt,sqr,yield,k=3,console=TRUE)

ANALYSIS PBIB:  yield

Class level information
block : 12
trt : 9

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
AIC                        222.23197
BIC                        237.78201
-2 Res Log Likelihood      -99.11599

Analysis of Variance Table

Response: yield
          Df Sum Sq Mean Sq F value    Pr(>F)
trt        8 3749.4  468.68  8.2315 0.0001987 ***
Residuals 16  911.0   56.94
---
Signif. codes:
0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 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

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

> detach(A)
```

**The adjusted averages can be extracted from the modelLattice.**

print(modelLattice$means)

**The comparisons:**

head(modelLattice$comparison)

## 4.11 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)
> head(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
```

**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"
```

**Analysis:**

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

ANALYSIS DAU:  yield
Class level information

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

Number of observations:  20

ANOVA, Treatment Adjusted
Analysis of Variance Table

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

ANOVA, Block Adjusted
Analysis of Variance Table

Response: yield
                     Df Sum Sq Mean Sq F value Pr(>F)
trt.unadj            11 575.67  52.333
block.adj             2  69.50  34.750  1.2884 0.3424
Control               3  52.92  17.639  0.6540 0.6092
Augmented             7 505.88  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: comparison and means  >>>

> modelDAU$means

     yield      std r Min Max mean.adj       SE block
A 84.66667 6.658328 3  79  92 84.66667 2.998456
B 79.00000 2.000000 3  77  81 79.00000 2.998456
C 82.00000 4.582576 3  78  87 82.00000 2.998456
D 83.33333 6.806859 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
```

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

# 5 Non-parametric comparisons

The functions for non-parametric multiple comparisons included in **agricolae** are: *kruskal, waerden.test, friedman and durbin.test*, reference [2].

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, reference [10]. Included in the **agricolae** package

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

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

**For the examples, the agricolae** package data will be used

## 5.1 Kruskal-Wallis

It makes the multiple comparison with Kruskal-Wallis. The parameters by default are alpha = 0.05.

```
> str(kruskal)

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

**Analysis**

```
> attach(corn)
> outKruskal<-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
Minimum difference changes for each comparison

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

> detach(corn)
```

The object output has the same structure of the comparisons see Figure 8.

## 5.2 Friedman

```
> str(friedman)

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

**Analysis**

```
> rm(trt)
> 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
```

```
> detach(grass)
```

## 5.3 Waerden

A nonparametric test for several independent samples. Example applied with the sweet potato data in the **agricolae** basis.

```
> str(waerden.test)

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

**Analysis**

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

```
Study: yield ~ virus
Van der Waerden (Normal Scores) test's

Value : 8.409979
Pvalue: 0.03825667
Degrees of freedom:  3

virus,  means of the normal score

        yield       std r
cc -0.2328353 0.3028832 3
fc -1.0601764 0.3467934 3
ff  0.6885684 0.7615582 3
oo  0.6044433 0.3742929 3

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(outWaerden)

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

**To see outWaerden$comparison**

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

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

Value : 8.409979
Pvalue: 0.03825667
Degrees of freedom:  3

virus,  means of the normal score

        yield       std r
cc -0.2328353 0.3028832 3
fc -1.0601764 0.3467934 3
ff  0.6885684 0.7615582 3
```

```
oo  0.6044433 0.3742929 3

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
```

```
> detach(sweetpotato)
```

## 5.4 Median test

A nonparametric test for several independent samples. The median test is designed to examine whether several samples came from populations having the sam median, reference [2].

```
> str(Median.test)
```

```
function (y, trt, correct = TRUE, simulate.p.value = FALSE,
    console = TRUE)
```

**Analysis**

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

```
The Median Test for yield ~ virus

Chi-square = 6.666667   DF = 3   P.value 0.08331631
Median = 28.25

          Median     Chisq     pvalue sig
cc and fc  18.30 6.0000000 0.01430588   *
cc and ff  28.25 0.6666667 0.41421618
cc and oo  30.30 6.0000000 0.01430588   *
fc and ff  21.45 6.0000000 0.01430588   *
fc and oo  23.50 6.0000000 0.01430588   *
ff and oo  38.70 0.6666667 0.41421618
```

```
> detach(sweetpotato)
> names(outMedian)
```

```
[1] "statistics" "parameters" "Medians"    "comparison"
[5] "data"
```

```
> outMedian$statistics

     Chisq    p.chisq Median
  6.666667 0.08331631  28.25

> outMedian$Medians

  trt Median grather lessEqual
1  cc   23.0       1         2
2  fc   13.1       0         3
3  ff   39.2       2         1
4  oo   38.2       3         0
```

## 5.5 Durbin

*durbin.test*; example: Myles Hollander (p. 311) Source: W. Moore and C.I. Bliss. (1942) A multiple comparison of the Durbin test for the balanced incomplete blocks for sensorial or categorical evaluation. It forms groups according to the demanded ones for level of significance (alpha); by default, 0.05.

```
> str(durbin.test)

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

**Analysis**

```
> 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
B - C         -4 0.102688
B - D          0 1.000000
B - E          0 1.000000
B - F         -3 0.204420
B - G          0 1.000000
C - D          4 0.102688
C - E          4 0.102688
C - F          1 0.657370
C - G          4 0.102688
D - E          0 1.000000
D - F         -3 0.204420
D - G          0 1.000000
E - F         -3 0.204420
E - G          0 1.000000
F - G          3 0.204420
```

```
> names(out)

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

> out$statistics

  chisq.value   p.value  t.value     LSD
     7.714286 0.2597916 2.306004 5.00689
```

# 6 Graphics of the multiple comparison

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

## 6.1 bar.group

A function to plot horizontal or vertical bar, where the letters of groups of treatments is expressed. The function applies to all functions comparison treatments. Each object must use the group object previously generated by comparative function in indicating that group = TRUE.

example:

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

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

The found object of one comparison is the entry for these functions, see Figure 4. The objects outHSD and outWaller are used in the following exercise:
outHSD, for the functions *bar.group* and *bar.err*
outWaller, for the function *bar.err*

## 6.2 bar.err

A function to plot horizontal or vertical bar, where the variation of the error is expressed in every treatments. The function applies to all functions comparison treatments. Each object must use the means object previously generated by the comparison function, see Figure 4

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

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

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

```
> par(mfrow=c(2,2),cex=0.7,mar=c(3.5,1.5,3,0))
> C1<-bar.err(modelPBIB$means[1:7, ], ylim=c(0,9), col=0, main="C1",
+ variation="range",border=3,las=2)
> C2<-bar.err(modelPBIB$means[8:15,], ylim=c(0,9), col=0, main="C2",
+ variation="range", border =4,las=2)
> # Others graphic
> C3<-bar.err(modelPBIB$means[16:22,], ylim=c(0,9), col=0, main="C3",
+ variation="range",border =2,las=2)
```

```
> par(mfrow=c(1,2),mar=c(3,3,2,0),cex=0.7)
> c1<-colors()[480]; c2=colors()[65]; c3=colors()[15]; c4=colors()[140]
> G1<-bar.group(outHSD$groups, ylim=c(0,45), main="Tukey\nG1",col=c1,las=1)
> # G2<-bar.group(outHSD$groups, horiz=T, xlim=c(0,45), main="Tukey\nG2",col=c2,las=1)
> # G3<-bar.err(outWaller$means, variation="range",ylim=c(0,45), col=c3,main="Range\nG3",las=1)
> G4<-bar.err(outWaller$means, horiz=T, xlim=c(0,45), col=c4, variation="SE",
+ main="Standard error \nG4",las=1)
```

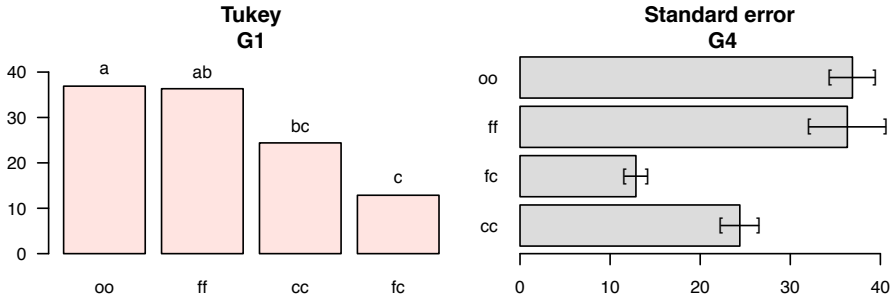


Figure 4: Comparison between treatments

```
> C4<-bar.err(modelPBIB$means[23:30,], ylim=c(0,9), col=0, main="C4",
+ variation="range", border =6,las=2)
> # Lattice graphics
> par(mar=c(2.5,2.5,1,0),cex=0.6)
> bar.group(modelLattice$group,ylim=c(0,55),density=10,las=1)
```

# 7 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", reference [7] and a non-parametric method of Haynes, based on the data range.

## 7.1 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. Reference [7] We will consider five environments in the following example:

```
> options(digit=2)
> 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.

**Analysis**

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

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

> print(output$stability)

  Yield Rank Adj.rank Adjusted Stab.var Stab.rating YSi ...
A  7.86   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
```

The selected genotypes are: A, C, E, G, H, J, M, N and O. These genotypes have a higher yield and a lower variation. to see output$analysis, 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)
```

## 7.2 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. Reference [5]

**Analysis**

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

[1] "ranking"    "statistics"

> output$statistics

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

## 7.3 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. Reference [3]

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 Biplot, Triplot and Influence graphics, see Figures 5

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

**AMMI structure**

```
> str(AMMI)

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

**plot.AMMI structure, plot()**

```
> str(plot.AMMI)

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

**type: 1=biplot, 2= triplot 3=influence genotype**

```
> 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, console=TRUE)
```

```
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  90.4 15 40.84756 2.723170    1.36 0.1680
PC4     9.6 100.0 13 17.43370 1.341054    0.67 0.7915
```

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

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.

**To triplot require klaR package. in R execute:**

plot(model,type=2,las=1)

**To Influence graphics genotype require spdep package, in R execute:**

plot(model,type=3,las=1)

## 7.4 AMMI index and yield stability

Calculate AMMI stability value (ASV) and Yield stability index (YSI). References [14, 12]

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

```
> par(cex=0.8,mar=c(4,4,1,0))
> plot(model,type=1,las=1)
```

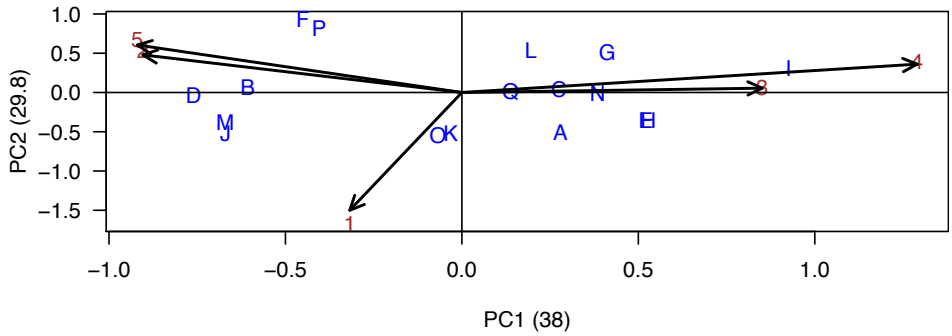


Figure 5: Biplot

```
> detach(plrv)
> index<-index.AMMI(model)
> # Crops with improved stability according AMMI.
> print(index[order(index[,3]),])
```

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

```
Desiree 5.5374138  56   28   28 16.15569
```

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

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

# 8 Special functions

## 8.1 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, see Figure 6.

When the dendrogram is complex, it is convenient to extract part of it with the function hcut(), see Figure 7.

```
> par(cex=0.6,mar=c(3,3,2,0))
> data(pamCIP)
> rownames(pamCIP)<-substr(rownames(pamCIP),1,6)
> output<-consensus(pamCIP,distance="binary", method="complete", nboot=5)

Duplicates: 18
New data  : 25 Records

Consensus hclust

Method distance: binary
Method cluster : complete
rows and cols  : 25 107
n-bootstrap    : 5
Run time       : 1.021002 secs
```

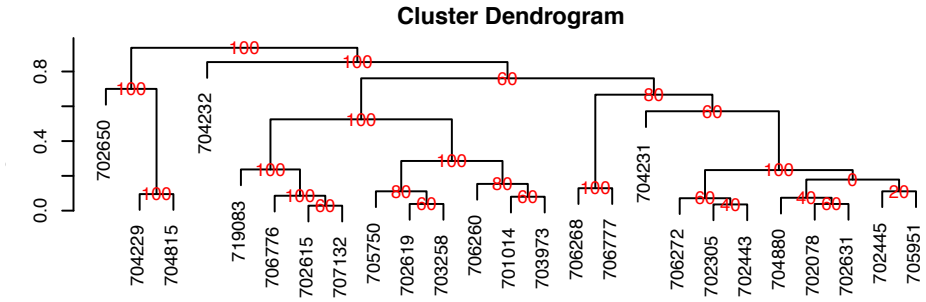


Figure 6: Dendrogram, production by consensus

```
> par(cex=0.6,mar=c(3,3,1.5,0))
> out1<- 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)
```

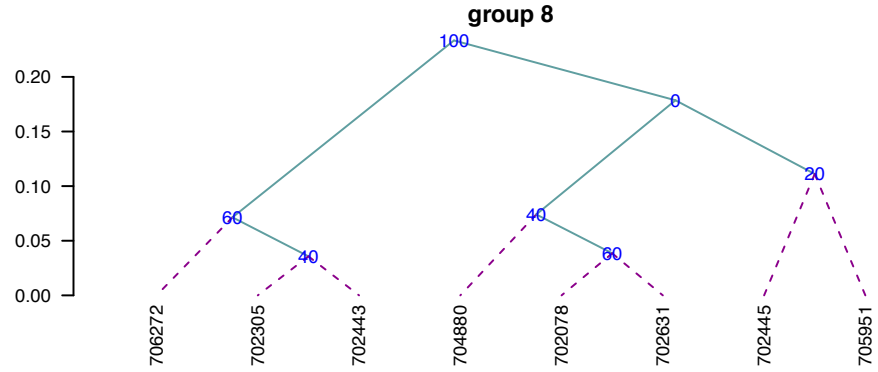


Figure 7: Dendrogram, production by hcut()

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

```
> names(output)
```

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

**Construct a classic dendrogram, execute procedure in R**

use the previous result 'output'

```
> dend <- as.dendrogram(output$dendrogram)
> data <- output$table.dend
> head(output$table.dend)
```

```
   X1  X2 xaxis     height percentage groups
1  -6 -24  7.50 0.02857143         60   6-24
2  -3  -4 19.50 0.03571429         40    3-4
3  -2  -8 22.50 0.03846154         60    2-8
4  -7 -10 10.50 0.03846154         60   7-10
5 -21   2 18.75 0.07142857         60 3-4-21
6 -16   3 21.75 0.07407407         40 2-8-16
```

```
> par(mar=c(3,3,1,1),cex=0.6)
> 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)
```

## 8.2 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, see Figure 8.

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

1000 data was simulated, being the frequency table:

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

```
     Lower Upper  Main freq relative   CF  RCF
[1,]  1.50  2.81  2.16   20     0.02   20 0.02
[2,]  2.81  4.12  3.47  120     0.12  140 0.14
[3,]  4.12  5.43  4.78  238     0.24  378 0.38
[4,]  5.43  6.74  6.09  225     0.22  603 0.60
[5,]  6.74  8.05  7.40  198     0.20  801 0.80
[6,]  8.05  9.36  8.70  168     0.17  969 0.97
[7,]  9.36 10.67 10.02   31     0.03 1000 1.00
```

```
> par(mar=c(2,0,2,1),cex=0.6)
> 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)
> axis(1,0:12)
> legend("topright",c("Original","Simulated"),lty=c(1,4),col=c("black", "blue"), lwd=4)
```

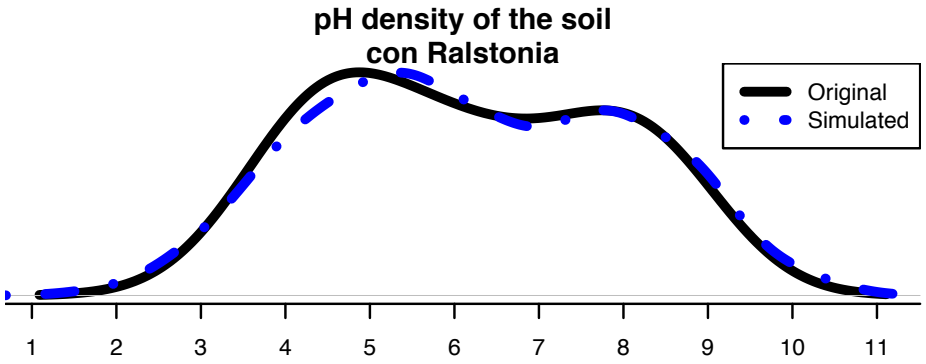


Figure 8: Distribution of the simulated and the original data

**Some statistics, original data:**

```
> summary(soil$pH)

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

**Some statistics, montecarlo simulate data:**

```
> summary(simulated)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.600   4.776   6.090   6.218   7.737  10.660
```

## 8.3 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 permutated 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=100)
> Xsol<-as.matrix(round(analysis$solution,2))
> print(Xsol,na.print = "")

             Df Sum Sq Mean Sq F value Pr(>F) Resampling
variety       1  25.09   25.09    7.26   0.02       0.01
date          2  13.89    6.95    2.01   0.18       0.16
variety:date  2   4.85    2.43    0.70   0.51       0.61
Residuals    12  41.48    3.46
```

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

## 8.4 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:

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

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 25.0868  7.2580 0.01952 *
date          2 13.892  6.9459  2.0096 0.17671
variety:date  2  4.853  2.4265  0.7020 0.51484
Residuals    12 41.477  3.4564
---
Signif. codes:
0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1
---
Validation of the analysis of variancia for the proposed model
Simulations: 100

             Df   F value % Acceptance % Rejection
variety       1 7.2580377           49          51
date          2 2.0095604           60          40
variety:date  2 0.7020312           61          39
                 Criterion
variety      nonacceptable
date            acceptable
```

```
variety:date    acceptable
---
```

The validation is referred to the percentage of decision results equal to the result of the ANOVA decision. Thus, 61% of the results simulated on the interaction variety*date gave the same result of acceptance or rejection obtained in the ANOVA.

## 8.5 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 0.3333333 0.2666667
X2 0.1666667 0.5333333

Residual Effect^2 =  0.4266667

> output

$Coeff
          X1        X2
X1 0.3333333 0.2666667
X2 0.1666667 0.5333333

$Residual
[1] 0.4266667
```

## 8.6 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. Reference [15].

**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.

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

ANALYSIS LINE x TESTER:  v2

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

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

GCA Effects:
============
Lines Effects:
Achirana     LT-8     MF-I    MF-II  Serrana    TPS-2
   0.022   -0.338    0.199   -0.449    0.058   -0.047
  TPS-25    TPS-7
   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, Adittive genetic variance: 0.01546921
F = 1, Adittive genetic variance: 0.007734604
F = 0, Variance due to Dominance: 0.1128228
F = 1, Variance due to Dominance: 0.05641141

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

> detach(site2)
```

## 8.7 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 plot size and the coefficient of variation, see Figure 9.

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

  Size Width Length plots       Vx   CV
1    1     1      1   648 9044.539 13.0
2    2     1      2   324 7816.068 12.1
3    2     2      1   324 7831.232 12.1
4    3     1      3   216 7347.975 11.7
5    3     3      1   216 7355.216 11.7
6    4     1      4   162 7047.717 11.4
```

```
> par(mar=c(3,3,4,0),cex=0.7)
> data(rice)
> table<-index.smith(rice,pch=19, col="blue",
+  main="Interaction between the CV and the plot size",type="l",xlab="Size")
```

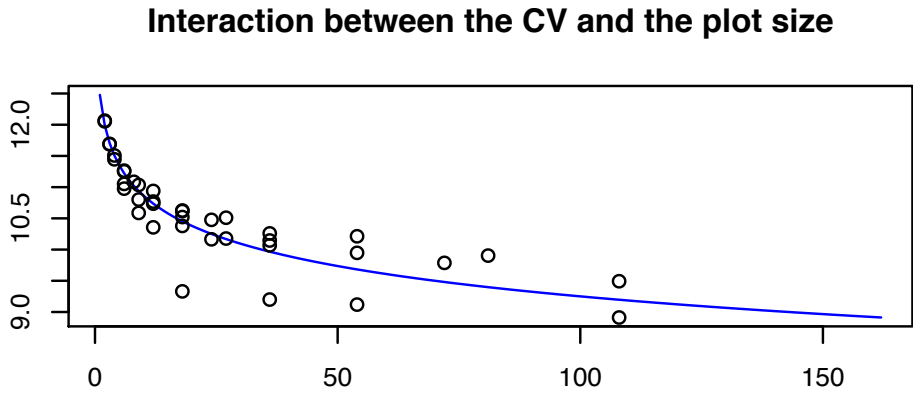


Figure 9: Adjustment curve for the optimal size of plot

## 8.8 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 Junin.

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
```

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

**The Shannon index is:**

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

Method: Shannon

The index: 3.52304

95 percent confidence interval:
 3.180131 ; 4.260501
```

## 8.9 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
```

```
> detach(soil)
```

## 8.10 tapply.stat()

Gets a functional calculation of variables grouped by study factors.

**Application with 'agricolae' data:**

**max(yield)-min(yield) by farmer**

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

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

```
> detach(RioChillon$babies)
```

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))

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

[1] 17.1666
```

## 8.12 Skewness and kurtosis

The skewness and kurtosis 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 kurtosis with:**

```
> kurtosis(x)

[1] -0.1517996
```

## 8.13 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.

```
> 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)
```

**Function of Waller to different value of parameters K and Fc** The next procedure 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.

```
> par(mar=c(3,3,4,0),cex=0.7)
> plot(K,y[,1],type="l",col="blue",ylab="waller",bty="l")
> lines(K,y[,2],type="l",col="brown",lty=2,lwd=2)
> lines(K,y[,3],type="l",col="green",lty=4,lwd=2)
> legend("topleft",c("2","4","8"),col=c("blue","brown","green"),lty=c(1,8,20),
+ lwd=2,title="Fc")
> title(main="Waller in function of K")
```

**Generating table Waller-Duncan**

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

table: Waller Duncan k=100, F=1.2

> print(W.D)

       4    6    8   10   12   14   16   18   20   24   30
6   2.85 2.87 2.88 2.89 2.89 2.89 2.89 2.88 2.88 2.88 2.88
7   2.85 2.89 2.92 2.93 2.94 2.94 2.94 2.94 2.94 2.94 2.94
8   2.85 2.91 2.94 2.96 2.97 2.98 2.99 2.99 2.99 3.00 3.00
9   2.85 2.92 2.96 2.99 3.01 3.02 3.03 3.03 3.04 3.04 3.05
10  2.85 2.93 2.98 3.01 3.04 3.05 3.06 3.07 3.08 3.09 3.10
11  2.85 2.94 3.00 3.04 3.06 3.08 3.09 3.10 3.11 3.12 3.14
```

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

## 8.14 AUDPC

The area under the disease progress curve (AUDPC), see Figure 10 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)
> print(evaluation)

  E1 E2 E3 E4 E5 E6
1 10 40 50 70 80 90

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

## 8.15 AUDPS

The Area Under the Disease Progress Stairs (AUDPS), see Figure 10. A better estimate of disease progress is the area under the disease progress stairs (AUDPS). The AUDPS approach improves the estimation of disease progress by giving a weight closer to optimal to the first and last observations..

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

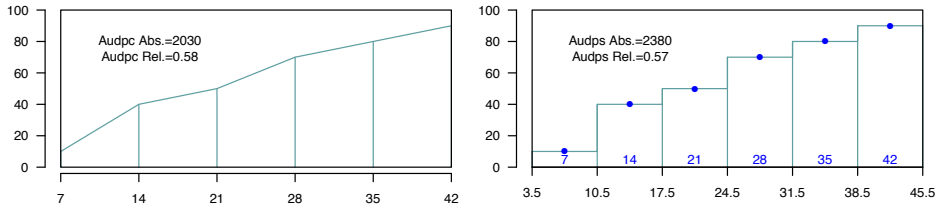


Figure 10: Area under the curve (AUDPC) and Area under the Stairs (AUDPS)

## 8.16 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)

Tukey's test of nonadditivity
cutting

P : 15.37166
Q : 77.44441

Analysis of Variance Table

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

> detach(potato)
```

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

## 8.17 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, see Figure 11

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).

```
> 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 to Lateblight function
> 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"

> par(mar=c(3,3,4,0),cex=0.7)
> #-------------------------
> 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)")
```

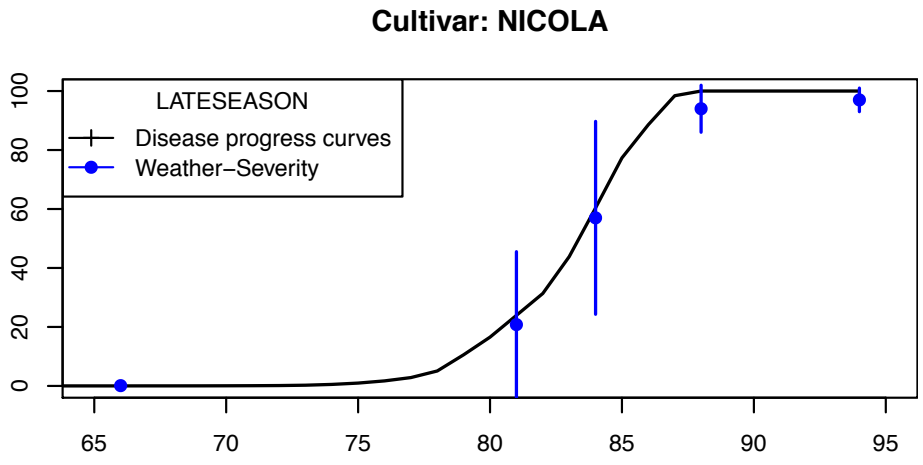


Figure 11: lateblight: LATESEASON

```
> head(model$Gfile)

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

> str(model$Ofile)

'data.frame':        94 obs. of  13 variables:
 $ Date       : Date, format: "2000-01-20" ...
 $ nday       : num  1 2 3 4 5 6 7 8 9 10 ...
 $ MicCol     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ SimSeverity: num  0 0 0 0 0 0 0 0 0 0 ...
 $ LAI        : num  0.01 0.0276 0.0384 0.0492 0.06 0.086 0.112 0.138 0.164 0.19 ...
 $ LatPer     : num  0 2 2 2 2 2 2 2 2 2 ...
 $ LesExInc   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ AttchSp    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ AUDPC      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ rLP        : num  0 0 0 0 0 0 0 0 0 0 ...
 $ InvrLP     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ BlPr       : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Defol      : num  0 0 0 0 0 0 0 0 0 0 ...

> head(model$Ofile[,1:7])

        Date nday MicCol SimSeverity    LAI LatPer LesExInc
1 2000-01-20    1      0           0 0.0100      0        0
2 2000-01-21    2      0           0 0.0276      2        0
3 2000-01-22    3      0           0 0.0384      2        0
4 2000-01-23    4      0           0 0.0492      2        0
5 2000-01-24    5      0           0 0.0600      2        0
6 2000-01-25    6      0           0 0.0860      2        0
```

**Repeating graphic**

```
> x<- model$Ofile$nday
> y<- model$Ofile$SimSeverity
> w<- model$Gfile$nday
> z<- model$Gfile$MeanSeverity
> Min<-model$Gfile$MinObs
> Max<-model$Gfile$MaxObs
```

```
> par(mar=c(3,2.5,1,0),cex=0.7)
> 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"))
```

# 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] Crossa, J. 1990. *Statistical analysis of multilocation trials*. Advances in Agronomy 44:55-85.

[4] De Mendiburu, Felipe, 2009. *Una herramienta de análisis estadístico para la investigación agrícola*, Universidad Nacional de Ingeniería (UNI).

[5] Haynes K G, Lambert D H, Christ B J, Weingartner D P, Douches D S, Backlund J E, Fry W and Stevenson W. 1998. *Phenotypic stability of resistance to late blight in potato clones evaluated at eight sites in the United States American*, Journal Potato Research 75, pag 211-21.

[6] Joshi, D.D., 1987. *Linear Estimation and Design of Experiments*, WILEY EASTERN LIMITED, New Delhi, India.

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

[8] Kuehl, Robert, 2000. *Design of Experiments*, 2nd ed., Duxbury.

[9] LeClerg, Erwin, 1962. *Field Plot Technique*, Burgess Publishing Company.

[10] Montgomery, 2002. *Diseño y Análisis de Experimentos*, 2nd Ed, WILEY.

[11] Patterson, H.D. and Williams, E.R., 1976. *A New Class of Resolvable Incomplete Block Designs*, Biometrika,Printed in Great Britain.

[12] Purchase, J. L. 1997. *Parametric analysis to describe genotyperenvironment interaction and yield stability in winter wheat*. Ph.D. Thesis, Department of Agronomy, Faculty of Agriculture of the University of the Free State, Bloemfontein, South Africa.

[13] R Core Team, 2014. *A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria. www.R-project.org

[14] Sabaghnia N., Sabaghpour S.H. and Dehghani H. 2008.*The use of an AMMI model and its parameters to analyse yield stability in multienvironment trials*. Journal of Agricultural Science, 146, 571-581. f 2008 Cambridge University Press 571 doi:10.1017/S0021859608007831. Printed in the United Kingdom.

[15] Singh R. K., Chaudhary B. D., 1979. *Biometrical Methods in Quantitative Genetic Analysis*.Kalyani Publishers

[16] Steel & Torry & Dickey, 1997. *Principles and Procedures of Statistic a Biometrical Approach*. Third Edition. The Mc Graw Hill companies, Inc