Package ‘plantbreeding’

September 2, 2012

Type Package

Title Analysis and visualization of data from plant breeding and genetics experiments

Version 1.1.0

Date 2012-06-25

Author Umesh R. Rosyara

Maintainer Umesh R. Rosyara <rosyara@msu.edu>

Depends R (>= 1.15.1), qtl, lattice, ggplot2, onemap, grid, agricolae, reshape, lme4, boot, plyr, pvclust

Suggests qtl, onemap, agricolae

Description The package contains different functionalities relevant to analysis of data from both conventional and molecular plant breeding and genetics experiments.

License GPL (>= 2)

URL https://r-forge.r-project.org/projects/plantbreeding/

R topics documented:

plantbreeding-package ........................................... 3
adesign .................................................................. 6
alphasim ................................................................ 7
ammi.full ................................................................. 8
assamby.plot ............................................................. 10
assoc.unr ................................................................. 11
AUDPC.cal ............................................................... 12
aug.rcb ................................................................. 13
aug.rowcol ............................................................. 15
augblock ................................................................. 16
augumentdesign ..................................................... 16
balincom ................................................................. 17
carolina1 ................................................................. 18
carolina2 ................................................................. 20
datapbib ................................................................. 21
diallele1 ................................................................. 22
diversity ................................................................. 23
fulldial ................................................................. 25
R topics documented:

gencor.lm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
geno.convert . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
genotype2alleles . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
graphicalgeno . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
gs2joinmap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
histlab . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
hsq.single . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
line.tester . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
linetester . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
manhatton.circos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
manhatton.plot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
map.fill.gplot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
map.plot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
mapbar.plot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
mapone . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
multienv . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
multiloc . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
northcaro1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
northcaro2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
onemap2mapchart . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
onfarm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
parentoffs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
peanut . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
phenosim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
plotblock . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
plotgen . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
plotwith.map . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
polar.genome . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
popvisd . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
rcbsingle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
respdataf . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
rowcoldata . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
rqtl2mapchart . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
rqtldata . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
seletion.index . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
selindex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
shaded.normal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
stability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
table.creator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
variability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
wintwheat . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78

Index 80
Description

Plant breeding is science of altering the genetics of plants in order to create desired characteristics for food, energy, medicine, industry and environmental purposes in cultivars. Plant breeding is interdisciplinary applied science involve application of diverse disciplines including genetics, statistics, plant pathology, entomology, plant physiology and other agricultural and biological sciences.

Data analysis and visualization is very important in plant breeding research area and this package is developed with objective of analysis of conventional and molecular data using different functions implemented in the robust R statistical analysis environment. In addition to development of new functions, examples are provided with analysis command to demonstrate how R can be used in analysis and visualization of data from plant breeding and genetics experiments.

This adds-on package contains functionality for analysis and visualization data from plant breeding experiments. Analysis includes both conventional quantitative genetics as well as molecular breeding tools. The library also consists of example datasets and codes to perform different analysis relevant to plant breeders depending upon other R packages.

Details

Package: plantbreeding
Type: Package
Version: 1.0
Date: 2012-04-21
License: GPL (>= 2)

The package contains different functionalities relevant to analysis of data from both conventional and molecular plant breeding and genetics experiments. The functionalities include analysis of designs specific to plant breeding needs such as Augmented block designs, Genotype x Environment and stability, variance component and combining ability estimations (eg. Diallel analysis, North Carolina designs, LinexTester), Heritability and Genetic correlation estimation, selection index. Beside classical breeding tools functionalities and examples provide different molecular analysis tools such as genetic map construction, - QTL mapping, association mapping and genomic selection.

There are other relevant utilities relevant to handling of moderate to large datasets. Also the package includes functions for visualization of population and genetic gain under selection as well as genome or chromosome wide visualization tools fitted to needs of molecular breeding tools. General R functions are also integrated with to guide new user who have limited experience of using R.

Author(s)

Umesh R Rosyara

Maintainer: Umesh Rosyara <rosyara@msu.edu>, <rosyaraur@gmail.com>
References


See Also

onemap
qtl
agricolae

Examples

# load the package
library(plantbreeding)
require (plantbreeding)

# seek help about the package
help(plantbreeding)
library(help = plantbreeding)

# list of dataset in the package
data(package="plantbreeding")

# list all objects in the package
ls("package:plantbreeding")

# load a dataset from the library - for example dataset nassociation
data (nassociation)

# seek help on particular function, example map.plot
help (map.plot)
?map.plot
# Example of applying a function

# Example 1: Diallele analysis
require(plantbreeding)
data(fulldial)
out <- diallele(dataframe = fulldial, male = "MALE", female = "FEMALE", progeny = "TRT", replication = "REP", yvar = "YIELD")
print(out)
out$anovout # analysis of variance
out$anova.mod1 # analysis of variance for GCA and SCA effects
out$components.model1 # model1 GCA, SCA and reciprocal components
out$gca.effmat # GCA effects
out$sca.effmat # SCA effect matrix
heatmap(out$sca.effmat, labRow = rownames(out$sca.effmat), labCol = colnames(out$sca.effmat)) # heatmap plot of SCA matrix
out$reciprocal.effmat # reciprocal effect matrix

# Example 2: Stability, AMMI analysis and heatmap plot
# stability analysis
require(plantbreeding)
data(multienv)
out <- stability(dataframe = multienv, yvar = "yield", genotypes = "genotypes", environments = "environments", replication = "replication")
out
# AMMI analysis
results <- ammi.full(dataframe = multienv, environment = "environments", genotype = "genotypes", replication = "replication", yvar = "yield")
results

# heatmap plot
heatmap(results$means, col = cm.colors(10))

# Example 3: Analysis of Augmented row column block designs
data(rowcoldata)
outp <- aug.rowcol(dataframe = rowcoldata, rows = "rows", columns = "columns", genotypes = "genotypes", yield = "yield")
outp$ANOVA # analysis of variance
outp$Adjustment # adjusted values

#### Example 4: Mahattan plots for association mapping results
set.seed(1234)
data12 <- data.frame(snp = rnorm(2000*20), chr = c(rep(1:20, each = 2000)), pos = rnorm(2000*20, 0.001, 0.005))
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos", pval = "pval", ymax = "maximum", ymin = 0, gapbp = 500, type = "polar", colour = "multicolor", geom = "area")
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos", pval = "pval", ymax = "maximum", ymin = 0, gapbp = 1000,
### adesign

**Design Augmented Block Design experiment**

**Description**

The function generates randomized plans for augmented block design.

**Usage**

```r
adesign(checks, newtrt, block.size = block.size, r, seed = 999)
```

**Arguments**

- **checks**: A vector with names of checks to be included in the experiment
- **newtrt**: A vector with names of new treatments to be included in the experiment
- **block.size**: A vector of block size (maximum number of entries allowed in each block consequetively)
- **r**: total number of replications (single number)
- **seed**: Random seed

```r
type = "polar", colour = "multicolor", geom = "point"

manhattan.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", ymax = "maximum", yimn = 0, gapbp = 1000,
type = "regular", colour = "multicolor", geom = c("line","point"))

manhattan.plot(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", ymax = "maximum", yimn = 0, gapbp = 500, color=c("hotpink3","dodgerblue4"),
line1 = 3, line2 = 5, pch = c(1,20))

### Example 5: plot maps with additional informations
lab1 <- paste("SNP_", 1:3, sep = "")
mapdat <- data.frame (chr = rep(1:3, each = length(lab1)/3), label= lab1,
position= c(0, 1, 4, 5, 6, 8, 10, 11, 12, 13,
0, 4, 5, 9, 12, 18, 20, 21, 22, 33,
0, 2, 6, 9, 12, 14, 18, 21, 24, 28 ))

# positions must start from zero
# data 2 filling avariable data
set.seed (1234)
fillcol <-rnorm(3*(length(lab1)-1), 0.5, 0.2)
filld <- data.frame(chr1 = rep(1:3, each = length(fillcol)/3), fillcol)

mapbar.plot (mapdat = mapdat, chr = "chr" ,position = "position",label = "label",
colorpalvec = heat.colors, size = 1, filld = filld, chr1 = "chr1")

# Brewing own color palette
colvec1 <- colorRampPalette(c("red", "yellow", "green"))
mapbar.plot (mapdat = mapdat, chr = "chr" ,position = "position",label = "label",
colorpalvec = colvec1, size = 10, filld = filld, chr1 = "chr1")
```
**alphasim**

**Details**

block.size and r can not be zero or NA. Total of block size should be equal to number of checks \(\times r\) + number of new treatments.

**Value**

Returns dataframe with randomization with plot number, blocks and treatments

**Author(s)**

Umesh R. Rosyara

**References**


**See Also**

augblock

**Examples**

```r
## example 1
ntrt = paste("EL", 1:60, sep=" ")
checks = c("A", "B", "C", "D", "E", "F")
bsize = c(20, 12, 16, 16, 10, 22)
ado <- adesign (checks = checks, newtrt = ntrt, block.size = bsize, r = 6, seed = 3246)
print(ado)
# example 2
checks1 = c("Rampur", "Elice", "Lansing", "Glover")
r1 = 4
block.size1 = c(16, 16, 16, 16)
print(EX2 <- augmentdesign (checks = checks1, newtrt = newtrt1,
block.size = block.size1, r = r1, seed = 124))
```
Format
A data frame with 60 observations on the following 6 variables.

- plotn: a numeric vector - plot number
- column: a numeric vector - column number
- block: a factor with levels 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
- genotype: a factor with levels 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
- replication: a numeric vector - replication number
- height: a numeric vector - trait variable with plant height data

Details
The following example code use function from agricolae package developed by Mendiburu F (2010). Please cite its use as:


Source
simulated dataset

References

Examples

data(alphasim)
summary(alphasim)
# not run
# require(agricolae)
# attach(alphasim)
# model <- PBIB.test(block, genotype, replication, height, k=3)
# model$comparison
# model$means

---

ammi.full Additive Main Effects and Multiplicative Interaction (AMMI) analysis

Description
The function implements Additive Main Effects and Multiplicative Interaction (AMMI) analysis for multiple environment replicated data. AMMI analysis (Gauch 1992) is one of popular tool in GE analysis and particularly effective for depicting adaptive responses. In this process, after genotype and environment main effects in the model, the interaction is retained as multiplicative term in the statistically significant GE-interaction principal-component (PC) axes.

The results of AMMI can be visualized as biplot (Gower and Hand 1996).
Usage

ammi.full(dataframe, environment, genotype, replication, yvar)

Arguments

dataframe dataframe objet
environment Name of environment (location or year) variable
 genotype Name of genotype variable
 replication Name of replication variable
 yvar Name of Y variable to be used in the analysis

Author(s)

Umesh Rosyara

References


Examples

# Example: AMMI analysis
data(multienv)
results <- ammi.full(dataframe = multienv , environment = "environments", genotype = "genotypes", replication = "replication", yvar = "yield")

# plot means
myd <- melt(results$means)
require(ggplot2)
d <- ggplot(myd, aes(genotype, value)) + geom_bar(fill = c("cadetblue"))
d + facet_wrap(~ environment) + theme_bw()

# plot PCA scores
myd2 <- data.frame (results$pc.scrs)
# genotype
mydgen <- myd2[myd2$category=="genotype",]
d1 <- ggplot(mydgen, aes(PC1, PC2)) + geom_point() + geom_text (aes (label = row.names (mydgen)), colour = "blue", hjust=1.2, vjust=0) + ylab ("PC2") + xlab ("PC1") + theme_bw()
print(d1)

#environment
mydenv <- myd2[myd2$category=="environment",]
d2 <- ggplot(mydenv, aes(PC1, PC2)) + geom_point() + geom_text (aes (label = row.names (mydenv)), colour = "red",hjust=1.2, vjust=0) +
assambly.plot

Description

The function develops assambly plot of Displaying multiple assembled line segments such as plotting of chromosomes or genome segments.

X axis consists of distance whereas Y axis consists of serial number for fragments (scaffold) or even quantitative data (if required).

Both start point and end point for drawing lines need to be supplied.

Usage

assambly.plot(data = dat, yvar = "yvar", xstart = "start", xend = "end", id = "id", xlab = "position", ylab = "", linecol = "cyan4", lwd = 6, lend = 2)

Arguments

data Dataframe
yvar The Y axis object - normally serial number otherwise any quantitative data
xstart Name of variable with start point for the segments
xend Name of variable with end points for the segments
id ID variable for the segments (for example fragment names)
xlab Label on X axis for example physical position, default is position
ylab Label of Y axis, default is none
linecol Color of line, default is cyan4
lwd Width of line (see graphical par in R / graphics)
lend End style of line (see graphical par in R / graphics)

Author(s)

Umesh Rosyara

Examples

#Example data
dat <- data.frame(yvar = c(1:10), id = paste("Frag-", 1:10, sep=""),
                 start = c(1,3,4,5,6,8,9,12,13,15,20),
                 end = c(5,9,6,15,19,11,13,19,20,25))

# two assambly plot in shame window
par(mfrow = c(2,1))
assoc.unr

Association mapping in unrelated or unstructured mapping population
using linear model

Description

The function performs mapping in unrelated or without clear population structure population using linear model. Both binomial (eg. susceptible or resistance, diseased or disease free, present or absent) variable are also possible using glm chi-square test. Also covariate can be fitted to model. The different genetic model (Additive or dominance) can be fitted.

Usage

assoc.unr(dataframe, yvar, xvars, covariate = FALSE, cvar, binomial = FALSE, model = "ADD")

Arguments

dataframe: dataframe with at least one y variable (yvar), x variable(s) (SNP markers) while covariate (cvar)

yvar: Name of Y variable to be used in association mapping, when binomial = TRUE the value must be 0 to 1.

xvars: Name of X variable (SNPs). vector of names of x variables (SNPs) (eg. c("SNP1", "xLoci12", "SNP1-3-4")) or column number (for example - c(6, 8, 10) or 6:100, or 6:length(dataframe))

covariate: Logical (TRUE or FALSE) whether we need to fit covariate in the model

cvar: While covariate = TRUE, we need name of covariate variable

binomial: Logical whether the y variable is used is binomial

model: Whether to fit additive model - "ADD", or dominance - "DOM" or regular anova - "NONE".

Value

The function output a dataframe with markers, pvalue (for SNPs) and cprob (for covariate), if covariate is fitted in the model.

Author(s)

Umesh Rosyara
Examples

```r
# simulated example
set.seed(3456)
id <- 1:115
snpmat <- data.frame(matrix(sample(c("AA","AB","BB"), 115, replace = TRUE), ncol = 115))
names(snpmat) <- c(paste("SNP",1:115, sep=''))
trait1 <- rnorm(115, 30, 5)
covtrait <- rnorm(115, 25, 3)
status <- sample(c(1,0), 115, replace = TRUE)
snpdata <- data.frame(id, trait1, covtrait, status, snpmat)

# x variable in range
out1.add <- assoc.unr(dataframe = snpdata, yvar = "trait1", xvars = 6:115, covariate = FALSE, cvar = NA, binomial = FALSE, model = "ADD")
out1.add

out1.dom <- assoc.unr(dataframe = snpdata, yvar = "trait1", xvars = 6:115, covariate = FALSE, cvar = NA, binomial = FALSE, model = "DOM")
out1.dom

out1.dom <- assoc.unr(dataframe = snpdata, yvar = "trait1", xvars = 6:115, covariate = FALSE, cvar = NA, binomial = FALSE, model = "NONE")
out1.dom

# X variables
xout <- assoc.unr(dataframe = snpdata, yvar = "trait1", xvars = 5:length(snpdata), covariate = FALSE, cvar = NA, binomial = FALSE)
plot(xout[,1], xout[,2], pch = 19, xaxt="n")
labdf <- xout[seq(1, nrow(xout), 1),]
axis(1, at=labdf[,1], label= labdf[,1])

# selected x vars
vars <- c("SNP3", "SNP4", "SNP10")
out4 <- assoc.unr(dataframe = snpdata, yvar = "trait1", xvars = vars, covariate = FALSE, cvar = NA, binomial = FALSE)
out4

# with covariates
out5 <- assoc.unr(dataframe = snpdata, yvar = "trait1", xvars = vars, covariate = TRUE, cvar = "covtrait", binomial = FALSE)
out5

# binomial is true, uses Chi-square test probabilities
out6 <- assoc.unr(dataframe = snpdata, yvar = "status", xvars = vars, covariate = FALSE, cvar = NA, binomial = TRUE)
out6

# binomial with covariates
out7 <- assoc.unr(dataframe = snpdata, yvar = "status", xvars = 5:length(snpdata), covariate = TRUE, cvar = "covtrait", binomial = TRUE)
out7
```

AUDPC.cal

Calculation of Area Under Disease / Pest Progress Curve
**Description**

The function calculates area under disease / pest progress curve (Jeger and Viljanen-Rollinson 2001; Madden et al. 2007). The area under the disease / pest progress curve (AUDPC) is a useful quantitative summary of disease or pest intensity over time. The trapezoidal method is the most frequently used method for estimating the AUDPC. This method discretize the time in different units (hours, days, weeks, months, or years) and then calculate the average disease or pest intensity between each pair of adjacent time points which are summed over time intervals (Madden et al. 2007).

**Usage**

```r
AUDPC.cal(reading.dates, severity.data)
```

**Arguments**

- `reading.dates` Vector disease reading dates
- `severity.data` Matrix of severity data, first column is ID of the individuals

**Author(s)**

Umesh Rosyara

**References**


**Examples**

```r
# Example
reading.dates <- as.Date(c("2012-02-13","2012-02-20","2012-02-28"))

mydat <- data.frame (ID = c("A", "B", "C", "D"), Date1 = c(1:4), Date2 = c(5:8), Date3 = c(11:14))

cd <- AUDPC.cal (reading.dates, mydat)
print(cd)
```

---

**Description**

The function implements analysis of augmented random block design. The function assumes that checks (controls) are replicated r times making complete blocks while other treatments (new treatments) are unreplicated. Once the desired block size is determined, the checks are completely randomized making complete blocks and remaining plots / experimental units are also completely randomized however new treatments are unreplicated.
Usage

aug.rcb(dataframe, genotypes, block, yvar)

Arguments

dataframe  Dataframe object with at least variable containing genotypes, blocks and one response variable to be analyzed

genotypes  Name of column consisting of genotype or treatments (use "nameofcolumn" format)

block       Name of column consisting of block (use "nameofblock" format)

yvar        Name of response variable column (use "yvar" format)

Value

A list consisting of the following items :

anova     Analysis of variance object

adjusted_values     dataframe Table with raw and adjusted values

se_check    Difference between check means

se_within   Difference adjusted yield of two varieties/entries in same block

SE_siff     Difference between two varieties/entries in different blocks

se_geno     Difference between two varieties/entries and a check mean

Author(s)

Umesh R. Rosyara

Examples

# Example
data(augblock)
out <- aug.rcb(dataframe = augblock, genotypes = "var", block = "blk", yvar = "gw")
out$anova  # analysis of variance
out$adjusted_values # yield observed and expected value table

# calculation of means
stab <- aggregate( gw ~ var, data=augblock, FUN= mean)

hist(stab$gw, col = "cadetblue", xlab = "Grain Yield",
main = "Mean yields from Augmented Yield Trial")
**Description**

The function implements analysis of augmented random row and column design.

**Usage**

`aug.rowcol(dataframe, rows, columns, genotypes, yield)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dataframe</code></td>
<td>dataframe object with at least columns with information of rows, columns, genotypes / entries / varieties / or treatments and yield (y variable)</td>
</tr>
<tr>
<td><code>rows</code></td>
<td>name of numeric variable with rows number</td>
</tr>
<tr>
<td><code>columns</code></td>
<td>name of numeric variable with column number</td>
</tr>
<tr>
<td><code>genotypes</code></td>
<td>name of column with with treatments / genotypes (factor)</td>
</tr>
<tr>
<td><code>yield</code></td>
<td>name of column with yield or any y variable</td>
</tr>
</tbody>
</table>

**Value**

- **ANOVA**: Analysis of Variance Table
- **Adjustment**: Original and Adjusted Phenotypic value
- **se_check**: Difference between check means
- **se_within**: Difference adjusted yield of two genotypes / varieties / entries in same row or column
- **se_diff**: Difference between two genotypes / varieties / entries in different rows or blocks
- **se_geno_check**: Difference between two genotypes / varieties / entries and a check mean

**Author(s)**

Umesh Rosyara

**Examples**

```r
# example 1
data(rowcoldata)
outp <- aug.rowcol(dataframe = rowcoldata, rows = "rows", columns = "columns", genotypes = "genotypes", yield = "yield")
outp$ANOVA # analysis of variance
outp$Adjustment # adjusted values

# calculation of means
stab <- aggregate( yield ~ genotypes, data=rowcoldata, FUN= mean)

hist(stab$yield, col = "cadetblue", xlab = "Grain Yield", main = "Mean yields from Augmented Yield Trial")
```
augblock  

### Augmented block data

**Description**

The example dataset for Augmented randomized block design

**Usage**

```r
data(augblock)
```

**Format**

A data frame with 78 observations on the following 4 variables.

- **var** a factor
- **blk** a numeric vector - blocks
- **trt** a numeric vector - treatments
- **gw** a numeric vector - grain weight

**Examples**

```r
data(augblock)
out <- aug.rcb(dataframe = augblock, genotypes = "var", block = "blk", yvar = "gw")
out$anova  # analysis of variance
out$adjusted_values  # yield observed and expected value table

# calculation of means
stab <- aggregate( gw ~ var, data=augblock, FUN= mean)

hist(stab$gw, col = "cadetblue", xlab = "Grain Yield",
     main = "Mean yields from Augmented Yield Trial")
```

auugmentdesign  

### Randomization of augmented block design

**Description**

Generating randomized augmented block design

**Usage**

```r
auugmentdesign(checks, newtrt, block.size = block.size, r, seed = 999)
```

**Arguments**

- **checks** vector with name of checks (replicated)
- **newtrt** vector for names of new treatments (unreplicated)
- **block.size** block size
- **r** replications
- **seed** random seed
Note

block.size and r cannot be zero or NA. Total of block size should be equal to number of checks x r + number of new treatments.

Author(s)

Umesh Rosyara

References


Examples

```r
## example 1
ntrt = paste("EL", 1:6, sep="")
checks = c("A", "B", "C", "D", "E", "F")
bsize = c(2, 12, 16, 16, 1, 22)
ado <- adesign(checks = checks, newtrt = ntrt, block.size = bsize, r = 6, seed = 3246)
print(ado)

# example 2
checks1 = c("Rampur", "Elice", "Lansing", "Glover")
r1 = 4
block.size1 = c(16, 16, 16, 16)
print(EX2 <- augmentdesign(checks = checks1, newtrt = newtrt1, block.size = block.size1, r = r1, seed = 124))
```

Data from a balanced incomplete block design

balincom

Description

Data from a balanced incomplete block design

Usage

data(balincom)
Format

A data frame with 24 observations on the following 5 variables.

- Block: a factor with levels 1 2 3 4 5 6 7 8
- Treatment: a factor with levels 1 2 3 4
- y: a numeric vector
- x: a numeric vector
- Grp: a factor with levels 13 24

References


Examples

data(balincom)
str(balincom)
# analysis of variance using mixed model
require("lme4")
print(mod1Bal <- lmer(y ~ Treatment * x + (1 | Block), balincom))
print(anova(mod1Bal))

# with Grp in the model
print(mod2Bal <- lmer(y ~ Treatment + x : Grp + (1 | Block), balincom))
print(anova(mod2Bal))

carolina1

Analysis of North Carolina Design I

description

The function performs both conventional and restricted (or residual, or reduced) maximum likelihood (REML) analysis of North Carolina I design (Comstock and Rosbinson 1952).

Usage

carolina1(dataframe, set, male, female, progeny, replication, yvar, REML = TRUE)

Arguments

dataframe: Dataframe should consist of variables set, male, female, progeny and replication along with at least one y variable (yvar)
set: name of numeric variable for set
male: name of numeric variable with male
female: name of numeric variable with female
progeny: name of numeric variable with progeny
replication: name of numeric variable with replication
yvar: name of name of y variable to be analyzed
REML: TRUE or FALSE depending upon if you want to fit REML model or not
Value

The following values as list are returned

- **model**
  - model - use anova (model) to see analysis of variables
  - `'variance male'`
    - Male variance
  - `'BULP estimates'`
    - BLUP estimates
  - `'variance female'`
    - Female variance
  - `'additive variance'`
    - Additive variance
  - `'dominance variance'`
    - Dominance variance
  - `'female:male:set:replication'`
    - female:male:set:replication
  - `'female:male:set'`
    - female:male:set
  - `'male:set'`
    - male:set
  - `'set:replication'`
    - set:replication

Author(s)

Umesh R. Rosyara

References


Examples

data(northcaro1)

# using general linear model
p1 <- carolina1(dataframe = northcaro1, set = "set", male = "male", female = "female",
progeny = "progeny", replication = "replication", yvar = "yield", REML = FALSE )
print(p1)

anova(p1[[1]]) # anova

p1[[1]]$coefficients ## coefficients

p1$var.m # male variance
p1$ var.f # female variance
p1$ var.A # variance additive
p1$ var.D # variance dominance
# using REML estimation
require(lme4)
p2 <- carolina1(dataframe = northcarol, set = "set", male = "male", female = "female",
progeny = "progeny", replication = "replication", yvar = "yield", REML = TRUE)

p2
p2$model
p2$'BULP estimates'

---

**Description**

The function performs analysis of North Carolina II design (Comstock and Rosbinson 1952).

**Usage**

carolina2(dataframe, set, male, female, replication, yvar)

**Arguments**

dataframe  | Dataframe with the variables set, male, female, replication and other response variables
set        | Name of column with set variables
male       | Name of column with male parent information
female     | Name of column with female parent information
replication| Name of column with replication column
yvar       | Name of response variable to be used for the analysis

**Value**

The following values as list are returned

model       | model - use anova (model) to see analysis of variables
var.m       | Male variance
var.f       | Female variance
var.mf      | Male*Female variance
var.AM      | Additive male variance
var.Af      | Additive female variance
var.D       | Dominance variance

**Author(s)**

Umesh Rosyara
References


Examples

data(northcaro2)
# for trait yield
myo <- carolina2(dataframe = northcaro2, set = "set", male = "male", female = "female",
  replication = "rep", yvar = "yield")
anova(myo$model) # anova
myo$var.m
myo$var.f
myo$var.mf
myo$var.Af
myo$var.D

# for trait tuber
tum <- carolina2(dataframe = northcaro2, set = "set", male = "male", female = "female",
  replication = "rep", yvar = "tuber")
anova(tum$model)
anova(tum$model) # anova
tum$var.m
tum$var.f
tum$var.mf
tum$var.Af
tum$var.D

datapbib

A partially balanced incomplete block experiment

Description

A partially balanced incomplete block experiment

Usage

data(datapbib)

Format

A data frame with 60 observations on the following 3 variables.

response  a numeric vector
Treatment  a factor with levels 1 10 11 12 13 14 15 2 3 4 5 6 7 8 9
Block     a factor with levels 1 10 11 12 13 14 15 2 3 4 5 6 7 8 9
References


Examples

data(datapbib)
help(datapbib)

require("lme4")
print(mod1 <- lmer(response ~ Treatment + (1|Block), data = datapbib,
                contrasts = c(unordered = "contr.SAS", ordered = "contr.poly")))
print(anova(mod1))

diallele1

Description

Calculates general and specific combining ability and other estimates for Diallel mating design using Griffings I method (Griffing, 1956).

Usage

diallele1(dataframe, yvar = "yvar", progeny = "progeny", male = "male",
          female = "female", replication = "replication")

Arguments

dataframe  Dataframe object
yvar       Name of yvariable to be used in analysis
progeny    Name of progeny variable to be used in the analysis
male       Name of male variable to be used in the analysis
female     Name of female variable to be used in the analysis
replication Name of replication variable

Author(s)

Umesh Rosyara

References

Examples

```r
data(fulldial)
out <- diallele!(dataframe = fulldial, male = "MALE", female = "FEMALE",
progeny = "TRT", replication = "REP", yvar = "YIELD")
print(out)
out$anova # analysis of variance
out$anova.mod1 # analysis of variance for GCA and SCA effects
out$components.model1 # model1 GCA, SCA and reciprocal components
out$gca.effmat # GCA effects
out$sca.effmat # SCA effect matrix
out$reciprocal.effmat # reciprocal effect matrix
out$varcompare # SE for comparisions
out$anovadf.mod2 # ANOVA for model 2
out$varcomp.model2 # variance components for model 2
```

---

**diversity**  
**Diversity analysis**

### Description

This function principal component analysis, cluster analysis and development of heatmap plot.

### Usage

```r
diversity(dataframe, varcol = 2:length(dataframe), xvar = "genotype",
yvarlab = "marker", dendocol = "blue4", cor = TRUE, heatcol = cm.colors(256),
rc = rainbow(nrow(tvar), start = 0, end = 0.3), cc = rainbow(ncol(tvar),
start = 0, end = 0.3), method = "ward", scale = "column", ...)
```

### Arguments

- **dataframe**: Name of dataframe
- **varcol**: Name of numerical variable column used in principal component analysis, cluster analysis and heatmap plot. The default is second column to end column of the dataframe (2:length(dataframe))
- **xvar**: Name of Xvar column, name of observation column
- **yvarlab**: Name of y variable label
- **dendocol**: Line colour for dendogram, default is "blue4"
- **cor**: logical Correlation
- **heatcol**: colour for main area of heatmap plot
- **rc**: vector with row colour for heatmap
- **cc**: vector with column colour for heatmap
diversity

method  the agglomeration method to be used for creating dendogram. This should be (an
unambiguous abbreviation of) one of "ward", "single", "complete", "average",
"mcquitty", "median" or "centroid".

scale for heatmap plot: character indicating if the values should be centered and scaled
in either the row direction or the column direction, or none. The default is "row"
if symm FALSE, and "none" otherwise.

Other heatmap plot parameters can be passed as argument, see help(heatmap)

Value

In addition to four plots: screeplot, biplot, dendogram and heatmap plot, the output object will be
list of following components:

pca.results output of Principal component results, for detail help(prcomp)
Hclust output of cluster analysis, for details help(hclust)
hv output of heatmap plot, for details help(heatmap)

Author(s)

Umesh Rosyara

Examples

# default setting
data(variability)
out <- diversity(variability)

# or something as above with detail code
out <- diversity(dataframe = variability, varcol = 2:length(variability),
    xvar = "genotype",
yvarlab = "marker", dendocol = "blue4", cor = TRUE, heatcol = cm.colors(256),
    RowSideColors = rainbow(100, start = 0, end = 0.3),
    ColSideColors = rainbow(10, start = 0, end = 0.3), method = "ward",
    scale = "column")

# some variation, using subset of markers, different clustering method
out1 <- diversity(dataframe = variability, varcol = 2:21, xvar = "genotype",
yvarlab = "marker", dendocol = "blue4", cor = TRUE, heatcol = cm.colors(256),
    RowSideColors = rainbow(20, start = 0, end = 0.3), ColSideColors = rainbow(10,
    start = 0, end = 0.3), method = "single", scale = "column")

# random selected columns
out2 <- diversity(dataframe = variability, varcol = sample(2:101, 20),
    xvar = "genotype", yvarlab = "marker", dendocol = "black", cor = TRUE,
    heatcol = cm.colors(256), RowSideColors = rainbow(20, start = 0, end = 0.3),
    ColSideColors = rainbow(10, start = 0, end = 0.3),
    method = "ward", scale = "column")

# heatmap row and column color based on user defined categories
cc1 <- c(rep(c("red", "purple", "green", "blue", "pink"), each = 2))
rc1 <- colors(3)[2:21]
out3 <- diversity(dataframe = variability, varcol = 2:21, xvar = "genotype",
yvarlab = "marker", dendocol = "black", cor = TRUE, heatcol = heat.colors(256),
    RowSideColors = rc1, ColSideColors = cc1, method = "ward", scale = "column")
# heatmap plot without side colors and dendograms, no scaling
out4 <- diversity(dataframe = variability, Rowv = NA, Colv = NA, varcol = 2:21,
xvar = "genotype", yvarlab = "marker", dendocol = "black", cor = TRUE,
heatcol = topo.colors(256), method = "ward", keep.dendro = FALSE, scale = "none")

# heatmap plot without side colors and dendograms at column only, no scaling
cc2 <- c("#F0FFFF", "#00008B", "#006400", "#FF7256", "#80A500", "#696969",
"#ADFF2F", "#BEBEBE", "#BB008B", "#FFB900")

out4 <- diversity(dataframe = variability, Rowv = NA, ColSideColors = cc2,
varcol = 2:21, xvar = "genotype", yvarlab = "marker", dendocol = "black", cor = TRUE,
heatcol = cm.colors(256), method = "ward", keep.dendro = FALSE, scale = "none")

---

**fulldial**  
*Full 8 x 8 diallel data*

**Description**

Data for a full 8 x 8 diallel for grain yield from Singh and Chaudhary (1979)

**Usage**

data(fulldial)

**Format**

A data frame with 256 observations on the following 7 variables.

- **FAMILY** Name of family
- **TRT** Treatments i.e. genotype
- **FAMQC** FAMQC
- **MALE** Male
- **FEMALE** Female
- **REP** Replication
- **YIELD** Grain yield

**Source**


**References**


Examples

data(fulldial)
out <- diallele1(dataframe = fulldial, male = "MALE", female = "FEMALE", progeny = "TRT", replication = "REP", yvar = "YIELD")

print(out)
out$anovout # analysis of variance
out$anova.mod1 # analysis of variance for GCA and SCA effects

out$components.model1 # model1 GCA, SCA and reciprocal components
out$gca.effmat # GCA effects
out$sca.effmat # SCA effect matrix
out$reciprocal.effmat # reciprocal effect matrix

out$varcompare # SE for comparisions
out$anovadf.mod2 # ANOVA for model 2
out$varcomp.model2 # variance components for model 2

---

**gencor.lm**

*Computing genetic correlation using linear model from single site replicated experiments*

Description

The genetic correlation between two traits is calculated by fitting linear model as outlined by Singh and Chaudhary (1985).

Usage

gencor.lm(dataframe, yvar1, yvar2, genovar, replication = replication, exout = F)

Arguments

dataframe
yvar1 name of first Y variable
yvar2 name of second Y variable
genovar name of genotype variable
replication name of replication variable
exout logical if extended output should be provided

Value

The result consists of a list of following components:

genetic.corr Genetic correlation
modelV1 linear model for variable 1
modelV2 linear model for variable 2
modelV1V2 linear model for variable 1 and 2
Author(s)
Umesh R Rosyara

References

Examples
# mydata
mydf <- data.frame (replication = rep(1:4,times = 8), genovar = rep(1:8,each = 4),
tgw = c(39,40,38,39, 37,36,36,37,45,46,46,47, 43,44,42,41,41, 40,
42, 41, 42,43,45,43,43,42,43,42,40,43,41), grw = c(104.9,84.3,77.0,76.5, 88.0,
106.5, 89.8,108.7, 80.0,71.3,77.5,69.5, 80.8,106.5,83.3, 95.9,
60.0,52.5,53.0,51.0, 96.4,98.8,99.1,107.2, 91.4,99.7,83.3,89.5, 91.8,84.8,70.0,81.5))

ot <- gencor.lm(dataframe = mydf, yvar1 = "tgw", yvar2 = "grw", genovar = "genovar",
replication = "replication", exout = FALSE)
print (ot)

# with extended output printed to screen and output saved
ct <- gencor.lm(dataframe = mydf, yvar1 = "tgw", yvar2 = "grw", genovar = "genovar",
replication = "replication", exout = TRUE)
print(ct)
anova(ct$modelV1) # analysis variance of variable 1
anova(ct$modelV2) # analysis of variance of variable 2
anova(ct$modelV1V2) # analysis of variance of variable 1 and variable 2

---

genotype.convert

Recoding genotypes

Description
The function converts recoding from DNA base pair (A/C/G/T) to number or other preferred forms.

Usage
genotype.convert(dataframe, tranvec, ownvec = "ACGT", output.file, outsep = ",",
na.strings = "NA")

Arguments
dataframe Input dataframe, the text or other documents can read to create dataframe using read.table function
tranvec What type of recoding needed: "ACGT" to recode basepair to number, A = 1, C = 2, G = 3, T = 4, D = 5, I = 6, else missing string defined in na.string or default = "NA" "AB" to recode basepair to number, A = 1, B = 2, else missing string defined in na.string or default = "NA" "num2base" to recode number of basepair, 1 = A, 2 = C, 3 = G, 4 = T, 5 = D, 6=I, else missing string defined
in na.string or default = "NA" If "OWN", we need to define the ownvec with recoding information

**ownvec**

If defined as "OWN" in tranvec, ownvec must be defined as list with list (in data = recoded to), example, ove <- c("AA" = "A", "AB" = "H", "BB" = "B"), here AA will be recoded to A, AB recoded to H and BB recoded to B.

**output.file**

The output file output files to be produced.

**outsep**

The output file separator. Use "," for comma deliniated file, " " space deliniated and other delimiter is also supported

**na.strings**

what string should be used for missing data.

---

**Author(s)**

Umesh Rosyara

---

**Examples**

# Example 1, convert number base (A, C, G, T, D, I) to number (1, 2, 3, 4, 5, 6)

```r
X1 <- c(sample(c("AA", "AC", "CA", "CC", "--"), 200, replace = TRUE))
X2 <- c(sample(c("II", "ID", "DI", "DD", "--"), 200, replace = TRUE))
X3 <- c(sample(c("TT", "GG", "TG", "GT", "--"), 200, replace = TRUE))
mydf1 <- data.frame(X1, X2, X3)
p1 <- geno.convert(dataframe = mydf1, tranvec="ACGT", output.file = "p2.csv",
      outsep = ",", na.strings = "-")
p1 <- geno.convert(dataframe = mydf1, tranvec="ACGT", output.file = "p2.txt",
      outsep = " ", na.strings = "NA")
print(p1)
```

# Example 2, convert number (1, 2, 3, 4, 5, 6) to base (A, C, G, T, D, I)

```r
var1 <- c(sample(c(11, 13, 31, 33, "--"), 100, replace = TRUE))
var2 <- c(sample(c(11, 12, 21, 22, "--"), 100, replace = TRUE))
var3 <- c(sample(c(55, 56, 65, 66, "--"), 100, replace = TRUE))
ex2 <- data.frame(var1, var2, var3)
p2 <- geno.convert(dataframe = ex2, tranvec="num2base", output.file = "p.csv",
      outsep = ",", na.strings = "-")
print(p2)
```

# Example 3, convert A, B to number 1, 2

```r
V1 <- c(sample(c("AA", "AB", "BA", "BB", "--"), 100, replace = TRUE))
V2 <- c(sample(c("AA", "AB", "BA", "BB", "--"), 100, replace = TRUE))
V3 <- c(sample(c("AA", "AB", "BA", "BB", "--"), 100, replace = TRUE))
ex3 <- data.frame(V1, V2, V3)
p3 <- geno.convert(dataframe = ex3, tranvec="AB", output.file = "p3.csv",
      outsep = ",", na.strings = "-")
print(p3)
```

# Example 4: recoding the data with own vector

```r
ove <- c("AA" = "A", "AB" = "H", "BA" = "H", "BB" = "B")
p4 <- geno.convert(dataframe = ex3, tranvec = "OWN", ownvec = ove, output.file = "p4.csv",
      outsep = ",", na.strings = "-")
print(p4)
```
genotype2alleles

Converting SNP or other two letter genotype format to allele format

Description

Converting SNP or other two letter genotype format to allele format

Usage

```r
genotype2alleles(input.file, input.sep = ",", column.num = "all", allele.sep = "/", comment.char = "]", na.strings = "NA", output.file, output.spe = ",")
```

Arguments

- `input.file`: Name of input file - eg csv or txt tabdelimited files
- `input.sep`: Input separator - eg "," for csv file input or " " space delimitated (this is separator used in file)
- `column.num`: Whether to use all column ("all") or range of column to be converted from genotype to allele format
- `allele.sep`: If thee is separator between allele characters (eg. for A/B for the seperator is ",", A-B format the seperator is "/".
- `comment.char`: The comment character used in the file to be read. The default is "]", it can be any special characters such as "];"
- `na.strings`: What is missing value string used in the file to be read - default is NA
- `output.file`: Desired name of output file
- `output.spe`: Desired seperator for the output file (use "," for csv, " " for space delimited files)

Author(s)

Umesh Rosyara

Examples

```r
# Example 1
B1 <- c("B/B", "C/C", "C/B", "D/A")
C1 <- c("B/B", "C/C", "/-/", "D/A")
mydf <- data.frame(A1, B1, C1)
write.table(mydf, file = "mycsv22.csv", sep = ",")
p <- genotype2alleles(input.file = "mycsv22.csv", input.sep = ",", column.num = "all", allele.sep = "/", comment.char = "]", na.strings = "NA", output.file = "out_mycsv22.csv", output.spe = ",")
print(p)
```
```r
# Example 2
ID <- 1:4
pos <- c(0, 245, 567, 871)
B1 <- c("B/B", "C/C", "C/B", "D/A")
C1 <- c("B/B", "C/C", "/-/", "D/A")
```
```r
mydf2 <- data.frame(ID, A1, B1, C1, pos)
write.table(mydf2, file = "mycsv26.csv", sep = ",")
p <- genotype2alleles(input.file = "mycsv26.csv", input.sep = ",", column.num = 2:4,
                    allele.sep = "/", comment.char = "#", na.strings = "NA", output.file = "out_mycsv26.csv",
                    output.spe = ",")
print(p)
```

### graphicalgeno

**Graphical genotype plot with shaded regions**

**Description**

The function produce graphical genotype plot with multiple chromosomes - with whole chromosomes or certain regions shaded for whole or subset genomic regions. The function can plot multiple faceted plots per chromosome, where X axis consists of the position in chromosome and Y axis consists of Individual ID. Heatmap plot is prepared with additional variable and text labeling can be done with genotype codes. Color can be the genotype itself if is numerically inputted such as A = 1, C = 2, G = 3, or T = 4.

**Usage**

```r
graphicalgeno(dataframe = dataframe, group = "group", position = "position",
               yvar = "yvar", ycat = "ycat", namevar = "namevar", subset = TRUE,
               subsetdata, panel.margin = /zero.noslash.1, strip.background = "lightpink",
               filllow = "white", fillhigh = "darkgreen", textlab = TRUE,
               textcolor = "blue", chr.arrange = "LR")
```

**Arguments**

- **dataframe**
  - The dataframe with group (= chromosome), position (map position), yvar (color coding variable, numeric), ycat (text variable for example genotype), and namevar (individual identification variable).

- **group**
  - groups (= chromosomes) or segment names

- **position**
  - Position on X axis

- **yvar**
  - Name of numeric variable to be color coded.

- **ycat**
  - Name of variable to used as text

- **namevar**
  - Name of variable for individual id

- **subset**
  - Logical variable whether to subset data or not

- **subsetdata**
  - Name of subset dataframe. Whenever subsetted data need only be plotted, there should be two datasets - full dataset specified in dataframe and subsetdata specified here.

- **panel.margin**
  - Width of panel margin, if you do not want to have panel margin use 0, otherwise use a numerical value, is interpreted as number of lines

- **strip.background**
  - Color of strip background

- **filllow**
  - Lower color for the heatmap

- **fillhigh**
  - Higher color for the heatmap
graphicalgeno

| textlab | Logical, whether to label text or not |
| textcolor | Color of the text |
| chr.arrange | How to arrange chromosomes, "LR" is single row with left to right, "TR" a single column from top to bottom, "DF" is ggplot is allowed to decide number of rows or columns depending upon the open plot area. |

Author(s)

Umesh Rosyara

Examples

# Example 1

```r
Id = paste("ID-", 1:5, sep = "")
position <- rep(seq(1, 100, 10), each = 5)
group = rep(rep(rep(1:5, each = length(Id)), each = length(position)))
set.seed(1234)
yvar <- rnorm(length(position), 0.5, 0.1)
ycat <- c(sample(c("A", "B", "H"), length(yvar), replace = TRUE))
namevar <- rep(Id, length(group)/length(Id))
dataframe <- data.frame(namevar, group, position, yvar, ycat)

# subset the data
datas = subset(dataframe,(group == 1 & position >= 3 & position <= 5) |
(group == 3 & position >= 2 & position <= 6))
datas1 = subset(dataframe,(group == 1 & position >= 3 & position <= 5) |
(group == 3 & position >= 2 & position <= 6) |
(group == 3 & position >= 8 & position <= 85))

dataframe <- data.frame(namevar, group, position, yvar, ycat)

dataframe <- subset(dataframe,(group == 1 & position >= 3 & position <= 5) |
(group == 3 & position >= 2 & position <= 6) |
(group == 3 & position >= 8 & position <= 85))

dataframe <- subset(dataframe,(group == 1 & position >= 3 & position <= 5) |
(group == 3 & position >= 2 & position <= 6) |
(group == 3 & position >= 8 & position <= 85))

# Implementation

graphicalgeno (dataframe = dataframe, group = "group",position = "position", yvar = "yvar", 
ycat = "ycat",namevar = "namevar",subset = TRUE, subsetdata = datas, panel.margin = 0.1 , 
strip.background = "lightpink", filllow = "white", fillhigh = "darkgreen", 
textlab = TRUE, textcolor = "blue", chr.arrange = "LR")

graphicalgeno (dataframe = dataframe, group = "group",position = "position", yvar = "yvar", 
ycat = "ycat",namevar = "namevar",subset = FALSE, subsetdata = datas, panel.margin = 0.2 , 
strip.background = "lightpink", filllow = "white", fillhigh = "darkgreen", textcolor = "blue", 
chr.arrange = "LR")

data1 = subset(dataframe,(group == 1 & position >= 3 & position <= 5) |
(group == 3 & position >= 2 & position <= 6) |
(group == 5 & position >= 8 & position <= 85))

graphicalgeno (dataframe = dataframe, group = "group",position = "position", yvar = "yvar", 
ycat = "ycat",namevar = "namevar",subset = TRUE, subsetdata = data1, panel.margin = 1 , 
strip.background = "lightpink", filllow = "white", fillhigh = "darkgreen", 
textcolor = "blue", chr.arrange = "LR")

# full data

graphicalgeno (dataframe = dataframe, group = "group",position = "position", yvar = "yvar", 
ycat = "ycat",namevar = "namevar",subset = FALSE, subsetdata = data, panel.margin = 0.2 , 
strip.background = "lightpink", filllow = "yellow", fillhigh = "blue", textcolor = "blue", 
chr.arrange = "DF")

graphicalgeno (dataframe = dataframe, group = "group",position = "position", yvar = "yvar", 

gs2joinmap

Convert Conversion of Ggenostudio matrix output to cross pollinated or self-pollinated Joinmap codes.

Description

The function convert the Genostudio matrix output to cross pollinated or self-pollinated Joinmap code formats. Genomestudio software is Illumina proprietary software for visualization and scoring of single nucleotide polymorphism (SNP) run in golden gate and iscan platforms. Joinmap is proprietary software from Kyazma commonly used for creating linkage / genetic maps. Manual conversion of genome studio output matrix to Joinmap code can be crumble some if need to be done manually, thus this function automate the process.

Usage

gs2joinmap(dataframe, type = "CP")
Arguments

dataframe  Dataframe should consist of first two rows for parent 1 and parent 2 followed by all columns with the genotype data (without any other columns)
type  Type of population to be coded "CP" for cross pollinated fullsib family or "F2" for F2 or RIL of early or advanced generations

Author(s)

Umesh Rosyara

References


http://www.illumina.com/support/array/array_software/genomestudio.ilmn

Examples

# Cross pollinated (CP) population example
mark1 <- c("AB", "BB", "AB", "BB", "BB", "AB", "--", "BB")
mark2 <- c("AB", "AB", "AA", "BB", "BB", "AA", "--", "BB")
mark3 <- c("BB", "AB", "AA", "BB", "BB", "AA", "--", "BB")
mark4 <- c("AA", "AB", "AA", "BB", "BB", "AA", "--", "BB")
mark5 <- c("AB", "AB", "AA", "BB", "BB", "AA", "--", "BB")
mark6 <- c("--", "BB", "AA", "BB", "BB", "AA", "--", "BB")
mark7 <- c("AB", "--", "AA", "BB", "BB", "AA", "--", "BB")
mark8 <- c("BB", "AA", "BB", "BB", "AA", "--", "BB")
loctype <- c(4, 3, 5, 5, 3, 6, 6, 0)

f2.pop <- data.frame (mark1, mark2, mark3, mark4, mark5, mark6, mark7, mark8)
outjoinF2 <- gs2joinmap(dataframe = f2.pop, type = "F2")
write.table(outjoinF2, file = "outjoinF2.csv", sep = ",", col.names = NA, qmethod = "double")
histlab

Plotting histograms with labelled with arrows

Description

The function generates one or multiple faceted histograms where arrows can be added to certain frequency class defined by user. This graph is useful to identify certain individual belongs to. For example, position on X where parents of population belong to. One or multiple arrows can be added to histogram.

Usage

histlab(dataframe, classvar = "class", yvar = "yvar", arrow_yvar, arrow_label, arrow_class, bwidth, colour = "cyan4", fill = "cyan4")

Arguments

dataframe: Dataframe with at least class variable and y variable column (whose frequency distribution need to be plotted).
classvar: Class variable (for example population name) is required (even with single class variable mention a single name in the class variable column)
yvar: Name of Y variable (whose frequency distribution need to be plotted)
arrow_yvar: Y variable vector that whose position in histogram need to be potted with an arrow
arrow_label: Label for vector whose position in histogram is potted with an arrow
arrow_class: Class variable vector whose position in histogram is potted with an arrow
bwidth: Bin width for the histogram
colour: Color for lines of the histogram
fill: Colour need to be filled in the histogram.

Author(s)

Umesh R Rosyara

Examples

# example 1
set.seed(123)
myd <- data.frame (class = rep(1:4, each = 100), yvar = rnorm(400, 50, 30))

# arrow label
class = c(2,3,3,4)
name = c ("geno4", "P3", "P1", "P2", "S1")
yvar = c(104.0, 8.5,80.0,40.0, 115.0)

histlab(dataframe = myd, classvar = "class", yvar = "yvar", arrow_yvar = yvar, arrow_label = name, arrow_class = class, bwidth = 20, colour = "blue", fill = "red")

# example 2
set.seed(123)
hsq.single

Bread sense Mixed model analysis of Heritability estimation

Description

Mixed model analysis of Heritability from single replicated RCB experiment using general linear model (Singh and Chaudhary 1985) or using restricted (or residual, or reduced) maximum likelihood method (Saxton 2004).

Usage

hsq.single(dataframe, yvars, genovar, replication, exout = F, REML = F)

Arguments

dataframe  
dataframe object

yvars  
ame of Y variable used in the model

genovar  
ame of genotype variable used in the analysis

replication  
ame of replication variable used in the analysis

exout  
logical variable (TRUE or FALSE), depending upon whether extended output is to be printed to screen

REML  
logical variable (TRUE or FALSE), depending upon whether REML be used to fit the model
Value

Depending upon REML (REML = TRUE or REML = FALSE) and extensive output (exout = TRUE or exout = FALSE), different output are returned.

When REML = FALSE

- model: The model
- heritiability: Heritability

When REML = TRUE

- heritiability: Heritability
- genovar: Genotypic variance
- totalvar: Total variance
- randomeffects: Random effects

When exout is FALSE only heritability is returned.

Author(s)

Umesh Rosyara

References


Examples

data (rcbsingle)
p1 <- hsq.single (dataframe = rcbsingle, yvars = "tgw", genovar = "genovar", replication = "replication", exout= TRUE, REML = FALSE)
print(p1)
anova(p1$model)

otGrw <- hsq.single (dataframe = rcbsingle, yvars = "grw", genovar = "genovar", replication = "replication", exout= TRUE, REML = FALSE)
print (otGrw)
anova(otGrw$model)

p2 <- hsq.single (dataframe = rcbsingle, yvars = "tgw", genovar = "genovar", replication = "replication", exout= TRUE, REML = TRUE)
print(p2)

hsq.single (dataframe = rcbsingle, yvars = "grw", genovar = "genovar", replication = "replication", exout= FALSE, REML = TRUE)

hsq.single (dataframe = rcbsingle, yvars = "tgw", genovar = "genovar", replication = "replication", exout= FALSE, REML = FALSE)

hsq.single (dataframe = rcbsingle, yvars = "grw", genovar = "genovar", replication = "replication", exout= FALSE, REML = FALSE)
Description

The function performs line x tester analysis as outlined by Singh and Chaudhary (1985).

Usage

```r
line.tester(dataframe, yvar, genotypes = genotypes, replication, Lines = Lines, Testers, gclass = gclass)
```

Arguments

- `dataframe`: Dataframe object with genotype, replication, lines, testers, gclass and at least of Y variable
- `yvar`: Name of Y variable
- `genotypes`: Name of genotype variable
- `replication`: Name of replication variable
- `Lines`: Name of lines variable
- `Testers`: Names of testers variables
- `gclass`: Name of gclass variable

Author(s)

Umesh R. Rosyara

References


Examples

```r
data(linetester)
pls <- line.tester(dataframe = linetester, yvar = "trait1", genotypes = "genotypes", replication = "replication", Lines = "Lines", Testers = "Tester", gclass = "gclass" )
print(pls)
```
**linetester**  
*Line x Tester analysis data*

**Description**

The line x tester analysis dataset is taken from Single and Chaudhari and Singh (1985).

**Usage**

```r
data(linetester)
```

**Format**

A data frame with 92 observations on the following 6 variables.

- **genotypes**: genotypes and parents 1 1x6 1x7 1X8 2 2x6 2x7 2X8 3 3x6 3x7 3X8 4 4x6 4x7 4X8 5 5x6 5x7 5X8 6 7 8
- **gclass**: codes to show whether is parent (P) or children (C), a factor with levels C P
- **Lines**: Lines
- **Tester**: Tester
- **replication**: Replication
- **trait1**: trait 1 - Y variable

**Source**


**Examples**

```r
data(linetester)
pls <- line.tester(dataframe = linetester, yvar = "trait1", genotypes = "genotypes", replication = "replication", Lines = "Lines", Testers = "Tester", gclass = "gclass")
print(pls)
```

---

**manhatton.circos**  
*Polar and Cartesian Manhattan plots*

**Description**

This function plots polar or Cartesian Manhattan plots. Different variations can be obtained by changing color or point type.

**Usage**

```r
manhatton.circos(dataframe, SNPname, chromosome, position, pvcol, colour = "seablue", ymax = "maximum", ymin = "minimum", gapbp = 1000, type = "polar", geom = "point")
```
Arguments

dataframe Name of dataframe
SNPname Name of column with SNP name
chromosome Name of column with chromosome
position Name of column with physical / genetic positions
pvcol Name of column with p-value
colour Colour
ymax Upper limit to y axis (i.e. -log10 p value)
ymin Lower limit to y axis (i.e. -log10 p value)
gapbp Gap between consecutive groups, 0 if no gap is required
type Value "polar" is polar plot need to be produced, "regular" if regular cartesian plot
geom type of geom to be plotted - example "point" (conventional) and "line" for line

Author(s)

Umesh Rosyara

Examples

data12 <- data.frame(snp = 1:2000*2, chr = c(rep(1:20, each = 2000)),
pos= rep(1:2000/20, pval= rnorm(2000*20, 0.001, 0.005))
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr",
position = "pos", pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 500,
type = "polar", colour = "multicolor", geom = "area")
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 1000,
type = "polar", colour = "multicolor", geom = "point")
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 1000,
type = "polar", colour = "multicolor", geom = "line")
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 1000,
type = "polar", colour = "multicolor", geom = "jitter")
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 1000,
type = "polar", colour = "multicolor", geom = "path")
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 1000,
type = "polar", colour = "multicolor", geom = "step")
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 1000,
type = "polar", colour = "multicolor", geom = c("line","point"))
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", yaxs = "maximum")
manhatton.circos(dataframe = data12, SNPname = "snp", chromosome = "chr", position = "pos",
pvcol = "pval", yaxs = "maximum", yaxs = "maximum")
manhatton.plot

**Manhattan plot of p-values**

**Description**

The function develops Mahnattan plot of p-values scaled to -log10(p). If polar type of Manhattan plot is desired use the function manhatton.circos. Manhattan plot (Gibson 2010) are popular in plotting association mapping results, however can be used to plot other results genome-wide.

**Usage**

```r
manhatton.plot(dataframe, SNPname, chromosome, position, pvcol, ymax = "maximum", ymim = "minimum", gapbp = 500, pch = c(18, 19, 2), color = c("midnightblue", "lightpink4", "blue"), line1, line2)
```

**Arguments**

- **dataframe**: dataframe with SNP name (SNPname), chromosome, physical position (position), p-value columns.
- **SNPname**: Name of variable consisting of SNP name - (eg. "SNPN")
- **chromosome**: Name of variable column consisting of chromosome - ( eg. "chr")
- **position**: Name of variable column consisting of physical position of SNPs - (eg. "physicaldis")
- **pvcol**: Name of p-value column to be used for plotting, dataframe can consists of multiple p-value column, can be plotted one by one. Note that p-value should not contain zero or Inf or NaNs
- **ymax**: Maximum value to be plotted in Y axis, if ymax is less than 8, the plot will set the maximum to 8 otherwise user defined maximum.
- **ymin**: Minimum value to be plotted in X axis.
- **gapbp**: Gap between two adjacent chromosome for plotting. Should be specified to scale of distances provided for X axis (ie. base pair). The default value is 500.
- **pch**: The list of symbol type used to plot in the plot, maximum allowed is equal to number of chromosomes plotted. If the number is less than total number of chromosomes, the pch is recycled till end.
- **color**: The list of color type used to plot in the plot, maximum allowed is equal to number of chromosomes plotted. If the number is less than total number of chromosomes, the color is recycled till end. The number of color should be equal to number of pch.
- **line1**: Value at the point where you need to Horizental threshold line 1. NULL for no line
- **line2**: Value at the point where you need to Horizental threshold line 2. NULL for no line
Details

Most of plot parameters (not conflicting with specified here) can be applied to plot.

Value

Produce Manhattan plot

Author(s)

Umesh Rosyara

Examples

```r
# Example 1
data12 <- data.frame (snp = 1: 2000*20, chr = c(rep(1:20, each = 2000)),
pos= rep(1:2000, 20), pval= rnorm(2000*20, 0.001, 0.005))
manhatton.plot(dataframe = data12, SNPname = "snp", chromosome = "chr",
position = "pos", pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 500,
color=c("hotpink3","dodgerblue4"), line1 = 3, line2 = 5, pch = c(1:20)
)
manhatton.plot(dataframe = data12, SNPname = "snp", chromosome = "chr",
position = "pos", pvcol = "pval", ymax = 10, ymin = 2, gapbp = 500, color=c("dodgerblue4"),
line1 = 3, line2 = 5, pch = 20)
manhatton.plot(dataframe = data12, SNPname = "snp", chromosome = "chr",
position = "pos", pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 500,
color=c("midnightblue", "lightpink4", "blue"),
line1 = 3, line2 = 5, pch = c("*", "+", "a")
)
manhatton.plot(dataframe = data12, SNPname = "snp", chromosome = "chr",
position = "pos", pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 500, color= "cadetblue",
line1 = 3, line2 = 5, pch = 19)
# all different color and pch example
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#0009E73", 
"#F0E442", ",#0072B2", ",#0D55E0", ",#3CC79A7", ",#CD661D", ",#FF00FF", ",#B86508", 
"#88B5A00", ",#366488")
manhatton.plot(dataframe = data12, SNPname = "snp", chromosome = "chr",
position = "pos", pvcol = "pval", ymax = "maximum", ymin = 0, gapbp = 500, color= cbPalette,
line1 = 3, line2 = 5, pch = 1:20)
# Example 2
set.seed(123)
data22 <- data.frame (snp = 1: 20000*5, chr = c(rep(1:5, each = 20000)),
pos= rep(1:20000, 5), pval1= rnorm(20000*5, 0.2, 0.3),
pval2= rnorm(20000*5, 0.2, 0.3 )
# the above simulation produce negative values so the following will replace
# negative values with NA
# data22$pval1[data22$pval1 < 0] <- NA
# removal of negative values
data2 <- data22[!is.na(data22$pval1),]
```
map.fill.gplot

Chromosome bar plot or LD block plot using ggplot2

Description

The function fills color between adjacent markers where color is scaled by continuous variable such as linkage disequilibrium (LD block plot). The output is similar to that produced from function mapbar.plot however use ggplot2 instead of R/graphics.

Usage

map.fill.gplot(mapd = mapd, chr = "chr", label = "label", position = "position", filld = filld, fillcol = "fillcol", fcol1 = "blue", fcol2 = "red")

Arguments

mapd  
Map data frame

chr  
Name of chromosome variable within the map data frame

label  
Name of label variable in the map data frame

position  
Name of map position variable within the map data frame

filld  
Name of dataframe with phenotypic data from n-1 intervals in each chromosomes from n markers

fillcol  
Name of variables used to fill chromosome segments

fcol1  
The first color to be used to brew color gradient within ggplot2

fcol2  
The second color to be used to brew color gradient within ggplot2

Author(s)

Umesh Rosyara

Examples

# Example 1
#data 1:
lab1 <- 1:10
chr <- rep(1:3, each = length (lab1))
label <- rep(lab1, 3)
position <- rep(c(0, 1, 4, 5, 6, 8, 10, 11, 12, 13), 3)
mapd<- data.frame (chr, label, position)
# data 2

fillcol <- rep(rnorm(length(lab1)-1, 0.5, 0.2), 3)
chr <- rep(1:3, each = length(fillcol)/3)
# this variable will be used to fill color in bars
filld <- data.frame(chr, fillcol)
map.fill.gplot(mapd = mapd, chr = "chr", label = "label", position = "position",
filld = filld, fillcol = "fillcol", fcol1 = "green", fcol2 = "red")

---

**map.plot**

**Chromosomal maps with or without scaled ticks**

**Description**

The plot develops maps with specified maker positions and labels. The length of ticks can be constant over the map or can be scaled to other variables of interest showing property of markers (for example polymorphism indicated by minor allele frequency).

**Usage**

```r
map.plot(mapdata, chr = "chr", markname = "markname", position = "position",
mbar.col = c("lightseagreen"), tick.size = FALSE, tvar = NULL, marklab = TRUE,
poslab = TRUE)
```

**Arguments**

- `mapdata`: dataframe with map information
- `chr`: Chromosome information
- `markname`: Marker name information
- `position`: Position of markers
- `mbar.col`: Color of the marker bar
- `tick.size`: Size of ticks
- `tvar`: Variable to be used as ticks
- `marklab`: Marker labels
- `poslab`: Marker position labels

**Author(s)**

Umesh Rosyara

**Examples**

```r
### Example 1
# map
nmar <- seq (1, 100, 5)
position= rep(nmar, 5)
n = length (nmar )
chr = rep(1:5, each = n )
set.seed (1234)
```
mapminor <- rnorm (length (chr), 0.5, 0.2)
mapdata <- data.frame (chr = chr, position = position, snpname = paste("SNP-", 1:length (position), sep = ""), mapminor = mapminor)

#mapdata
# with constant tick size
map.plot(mapdata = mapdata, chr = "chr", markname = "snpname", position = "position", mbar.col = c("lightseagreen"), tick.size = FALSE, tvar = FALSE, marklab = TRUE, poslab = TRUE)

# with tick size scaled to minor allele frequency
map.plot(mapdata = mapdata, chr = "chr", markname = "snpname", position = "position", mbar.col = c("lightseagreen"), tick.size = TRUE, tvar = "mapminor", marklab = TRUE, poslab = TRUE)

#### Example 2
nmar <- seq (1, 1000, 5)
position= rep(nmar, 5)
n = length (nmar )
chr = rep(1:5, each = n )
set.seed (456)
pval <- rnorm (length (chr), 0.5, 0.5)
mapdata1 <- data.frame (CHRM = chr, position = position, snpname = paste("SNP-", 1:length (position), sep = ""), pval = pval)

map.plot(mapdata = mapdata1, chr = "CHRM", markname = "snpname", position = "position", tick.size = TRUE, tvar = "pval", mbar.col = c("darkblue"), marklab = FALSE, poslab = FALSE)

---

mapbar.plot

Chromosome bar plot or LD block plot

Description

The plot is useful to plot map where each interval is filled with different color scaled to user specified variables such as linkage disequilibrium. For example we can plot linkage disequilibrium information between adjacent markers to identify linkage blocks.

Usage

mapbar.plot(mapdat = mapdat, chr = "chr", position = "position", label = "label", colorpalvec = heat.colors, size = 10, filld = filld, chr1 = "chr1", fillcol = "fillcol")

Arguments

mapdat  Map dataframe
chr     Name of chromosome variable in the map dataframe
position Name of variable for position in map dataframe
label   Name of label variable in map dataframe
colorpalvec The colors to be used for filling - the default is heat.colors. User can develop own color scaling using color brewer or use other build-in color pallettes.
size    Size of color vector - number of colors in the scale
filld   Dataframe with color filling information
mapone

chr1

Name for chromosome in the dataframe filld

fillcol

Name of variable in dataframe for filling colors (for example linkage disequilibrium) - for n markers n-1 colors corresponding to the interval plotting should provided.

Details

The bar size can be changed by changing dimension of plot area (reduced height smaller bar, increased height larger bar)

Author(s)

Umesh Rosyara

Examples

#Example:

#data 1: map data
lab1 <- paste("SNP_", 1:30, sep = "")
mapdat <- data.frame(chr = rep(1:3, each = length(lab1)/3), label= lab1,
  position= c(0, 1, 4, 5, 6, 8, 10, 11, 12, 13,
    0, 4, 5, 9, 12, 18, 20, 21, 22, 33,
    0, 2, 6, 9, 12, 14, 18, 21, 24, 28 ))
  # positions must start from zero
# data 2 filling avariable data
fillcol <- rnorm(3*(length(lab1)-1), 0.5, 0.2)
filld <- data.frame(chr1 = rep(1:3, each = length(fillcol)/3), fillcol)

mapbar.plot (mapdat = mapdat, chr = "chr",position = "position",label = "label",
  colorpalvec = heat.colors, size = 1,
  filld = filld, chr1 = "chr1")

# Brewing own color palette
colvec1 <- colorRampPalette(c("red", "yellow", "green"))
mapbar.plot (mapdat = mapdat, chr = "chr",position = "position",label = "label",
  colorpalvec = colvec1, size = 10,
  filld = filld, chr1 = "chr1")

# using build in color brewer
mapbar.plot (mapdat = mapdat, chr = "chr",position = "position",label = "label",
  colorpalvec = cm.colors, size = 20,
  filld = filld, chr1 = "chr1")

mapone

Genetic mapping dataset

Description

This is example data for creating genetic map using onemap (Margarido et al. 2007). Simulated F2 dataset with 250 individuals with 75 markers.

Usage

data(mapone)
Format

"f2.onemap" object with 250 individuals with 75 markers. There are two traits - "QT1" and "QT2".

Source

A simulated dataset

References


The following examples make use of onemap package by Margarido et al. (2007), please cite above reference if you make use of the following codes.

Examples

# load onemap package
require(onemap)
# load the data from plantbreeding library
data(mapone)
ls(mapone)
# to know in detail about how to read mapmaker style file into onemap
help(read.mapmaker)

# Estimating two-point recombination fractions
tw.mapone <- rf.2pts(mapone, LOD = 2, max.rf = 0.4)

# Assigning markers to linkage groups
mark.all.mapone <- make.seq(tw.mapone, "all")

# You can assign markers to linkage groups using the function "group".
LGs.mapone <- group (mark.all.mapone, LOD = 3, max.rf=0.5)
LGs.mapone

# estimation of marker order and map distance
# linkage group 1.
LG1.mapone <- make.seq(LGs.mapone, 1)
LG1.mapone

# using different algorithms
LG1.rcd <- rcd(LG1.mapone) # Rapid Chain Delineation
LG1.rec <- record(LG1.mapone) # order obtained using RECORD algorithm:

# compare different sequence
subsam <- rf.2pts(mapone)
markers <- make.seq(subsam,c(1, 2, 3,4,5))
markers.comp <- compare(markers)
markers.comp

# setting mapping function
## set.map.fun(type=c("kosambi"))

# using order.seq function
LG1.mapone.ord <- order.seq(input.seq=LG1.mapone, n.init = 5, subset.search = "twopt", twopt.alg = "rcd", THRES = 3, draw.try = TRUE, wait = 1, touchdown=TRUE)
multienv

# force order
LG1.mapone.final <- make.seq(LG1.mapone.ord,"force")
ripple.seq(ws=5, LG1.mapone.final)

LG1.mapone.final # to display the map for group 1

# performing the task in batch mode for all
twpt<-rf.2pts(mapone)
lgrp <- group(make.seq(twpt, "all"))
mapslist<-vector("list", lgrp$n.groups)

for(i in 1:lgrp$n.groups){
    ##create linkage group i
    LGcur <- make.seq(lgrp,i)
    ##ordering
    mapcur<-order.seq(LGcur, subset.search = "sample")
    ##assign the map of the i-th group to the maps.list
    mapslist[[i]]<-make.seq(mapcur, "force")
}

##write maps.list to "mapone_vs1.map" file
write.map(mapslist, "mapone_vs1.map")

---

multienv  Multi-environment data

Description

Simulated Multi-environment data for stability and Additive Main Effects and Multiplicative Interaction (AMMI) analysis.

Usage

data(multienv)

Format

A data frame with 150 observations on the following 4 variables.

- yield  yield - Y variable
- replication  replication
- genotypes  genotype: G1 G10 G2 G3 G4 G5 G6 G7 G8 G9
- environments  environments: CA CB CC MN SD

References


**Examples**

```r
# stability analysis
data(multienv)
out <- stability(dataframe = multienv, yvar = “yield”, genotypes = “genotypes”,
environments = “environments”, replication = “replication”)
out
# AMMI analysis
results <- ammi.full(dataframe = multienv, environment = “environments”,
genotype = “genotypes”, replication = “replication”, yvar = “yield”)

# heatmap plot
heatmap(results$means)

# plot bar plot
myd <- melt(results$means)
require(ggplot2)
d <- ggplot(myd, aes(genotype, value)) + geom_bar() + facet_wrap(~ environment) + theme_bw()

# plot PCA scores
myd2 <- data.frame(results$pc.scrs)
# genotype
mydgen <- myd2[myd2$category == “genotype”,]
d1 <- ggplot(mydgen, aes(PC1, PC2)) + geom_point() + geom_text(aes(label = row.names(mydgen)), colour = “blue”, hjust=1.2, vjust=0) + ylab(“PC2”) + xlab(“PC1”) + theme_bw()
print(d1)

# environment
mydenv <- myd2[myd2$category == “environment”,]
d2 <- ggplot(mydenv, aes(PC1, PC2)) + geom_point() + geom_text(aes(label = row.names(mydenv)), colour = “red”, hjust=1.2, vjust=0) + ylab(“PC2”) + xlab(“PC1”) + theme_bw()
print(d2)
```

---

**multiloc**

*Multi-location data*

**Description**

The multi-location data from SAS for mixed model package
Usage

data(multloc)

Format

A data frame with 108 observations on the following 7 variables.

- obs observations
- Location: Locations: A B C D E F G H I
- Block: Blocks: a factor with levels 1 2 3
- Trt: Treatments: a factor with levels 1 2 3 4
- Adj: Adj: a numeric vector
- Fe: Fe: a numeric vector
- Grp: Groups

Source


References


Examples

# stability analysis
data(multloc)
out <- stability (dataframe = multloc , yvar = "Adj", genotypes = "Trt", environments = "Location", replication = "Block")
out
# AMMI analysis
results <- ammi.full(dataframe = multloc , environment = "Location", genotype = "Trt", replication = "Block", yvar = "Adj")

# heatmap plot of means
heatmap (results$means)

# plot PCA scores
myd2 <- data.frame (results$pc.scrs)
# genotype
require(ggplot2)
mydgen <- myd2[myd2$category =="genotype",]
d1 <- ggplot(mydgen, aes(PC1, PC2)) + geom_point() +
  geom_text (aes (label = row.names (mydgen)), colour = "blue", hjust=1.2, vjust=0) +
  ylab ("PC2") + xlab ("PC1") + theme_bw()
print(d1)
#environment
mydenv <- myd2[myd2$category =="environment",]
d2 <- ggplot(mydenv, aes(PC1, PC2)) + geom_point() +
  geom_text (aes (label = row.names (mydgen)), colour = "red",hjust=1.2, vjust=0) +
  ylab ("PC2") + xlab ("PC1") + theme_bw()
print(d2)

northcarol

North Carolina Design I

Description
Data for analysis of north Carolina design I (Comstock and Rosbinson 1952).

Usage
data(northcarol)

Format
A data frame with 72 observations on the following 6 variables.
set set
male male
female female
progeny progeny
replication replication
yield yield - Y variable

Source
Publishers

References
Comstock R.F., Rosbinson F.F (1952). Estimation of average dominance of genes. In Heterosis,
Iowa State College Press, Iowa City, Iowa, chapter 30.
Publishers
Examples

data(northcaro1)
# using general linear model
p1 <- carolina1(dataframe = northcaro1, set = "set", male = "male", female = "female", 
progeny = "progeny", replication = "replication", yvar = "yield", REML = FALSE )
print(p1)

anova(p1[[1]]) # anova

p1[[1]]$coefficients ## coefficients

p1$var.m # male variance
p1$ var.f # female variance
p1$ var.A # variance additive
p1$ var.D # variance dominance

# using REML estimation
require(lme4)
p2 <- carolina1(dataframe = northcaro1, set = "set", male = "male", female = "female", 
progeny = "progeny", replication = "replication", yvar = "yield", REML = TRUE )
print(p2)

Description

Example data for analysis of North Carolina design II (Comstock and Rosbinson 1952).

Usage

data(northcaro2)

Format

A data frame with 300 observations on the following 9 variables.

Loc Loc
set Set
rep replication
female femial
male male
plrv plrv
yield yield
tuber tuber
weight weight
References


Examples

data(northcaro2)
# for trait yield
myo <- carolina2(dataframe = northcaro2, set = "set", male = "male", female = "female", replication = "rep", yvar = "yield")
anova(myo$model) # anova
myo$var.m
myo$var.f
myo$var.mf
myo$var.Af
myo$var.D

# for trait tuber
tum <- carolina2(dataframe = northcaro2, set = "set", male = "male", female = "female", replication = "rep", yvar = "tuber")
anova(tum$model)
anova(tum$model) # anova
tum$var.m
tum$var.f
tum$var.mf
tum$var.Af
tum$var.D

onemap2mapchart

Convert onemap map output to Marchart readable *mct" format

Description

Convert onemap map output to Marchart readable *mct" format

Usage

onemap2mapchart(mapfile, outprefix = ".")

Arguments

mapfile Name of mapfile outputed by write.map function
outprefix Prefix of output mapchart file

Author(s)

Umesh Rosyara
References


Examples

data(mapone) require(onemap) twpt<-rf.2pts(mapone) lgrp <-group(make.seq(twpt, "all")) mapslist<-vector("list", lgrp$n.groups) 

for(i in 1:lgrp$n.groups){
##create linkage group i
LGcur <- make.seq(lgrp,i)
##ordering
mapcur<order.seq(LGcur, subset.search = "sample")
##assign the map of the i-th group to the maps.list
mapslist[[i]]<-make.seq(mapcur, "force")
}
write.map(mapslist, "mapone.map")

onfarm

Data from on farm experiments

Description

Simulated data from on-farm experiments to demonstrate application of mixed model to analyze data from on-farm experiments.

Usage

data(onfarm)

Format

A data frame with 200 observations on the following 5 variables.

year year - a factor with levels 1 2
village village - a factor with levels 1 2 3 4
farm farm - a factor with levels 1 2 3 4 5
genotype genotype - a factor with levels 1 2 3 4 5
trait1 trait1 - Y variable

Source

Simulated data set
Examples

data(onfarm)
require(lme4)
# analysis using lme4, REML criterion used instead of log-likelihood
require(lme4)
modelf <- lmer(trait1 ~ (year + village + genotype)^3 + (year + village +
genotype|farm:village), data= onfarm, REML = TRUE)# full model
anova(modelf)
modelr = lmer(trait1 ~ year+village+year:village+(1|farm:village:year)+ genotype, data=onfarm, REML = TRUE)
# reduced model without genotype:year, genotype:village, genotype:year:village
anova(modelr)
anova(modelf, modelr)# comparison of two models

<table>
<thead>
<tr>
<th>parentoffs</th>
<th>Parent offspring regression example data</th>
</tr>
</thead>
</table>

Description

The data provide example to calculate parent offspring regression data.

Usage

data(parentoffs)

Format

A data frame with 100 observations on the following 4 variables.

- parent1  parent 1
- parent2  parent 2
- midparent  mid parent value
- offspring  offspring

References

Examples

```r
data(parentoffs)

# parent offspring regression
model <- lm(parentoffs$offspring ~ parentoffs$midparent)
heritability <- coef(model)[2]
heritability

# plotting
par(fig=c(0,0.8,0.8))
plot(parentoffs$midparent, parentoffs$offspring, xlab="Mid parent value", ylab="Offspring",
     col = "cadetblue", pch= 19)
abline(model, col = "red")

par(fig=c(0.65,1,0.8), new=TRUE)
boxplot(parentoffs$midparent, horizontal=TRUE, col = "red", axes=FALSE)
par(fig=c(0.0,0.8,0.55,1), new=TRUE)
boxplot(parentoffs$offspring, col = "blue", axes=FALSE)
mtext(paste("Parent of offspring regression \\
                 heritability = ", round(heritability,2), sep = ""),
      side=3, outer=TRUE, line=-3)

# bootstrap analysis of the heritability (regression coefficient)
# Need to install package boot
require(boot)
hsq.function <- function(data, i){
  d <- data[i,]
  fit <- lm(d$offspring ~ d$midparent, data=d)
  return(coef(fit)[2])
}
boot.results <- boot(parentoffs, hsq.function, R=1000)
boot.results
plot(boot.results)
```

peanut

---

**Description**

peanut multi-location data from genetic analysis using SAS

**Usage**

data(peanut)

**Format**

A data frame with 590 observations on the following 5 variables.

- `geno` genotypes - a factor with levels Florman manf393 mf447 mf478 mf480 mf484 mf485 mf487 mf489 Tegua
- `rep` replications
yield - Y variable
env  environment
gen  genotypes

**Source**

**References**

**Examples**
```r
data(peanut)
peanut$rep <- as.factor (peanut$rep)
peanut$env <- as.factor (peanut$env)

# stability analysis
out_peanut <- stability (dataframe = peanut , yvar = "yield", genotypes = "geno",
environments = "env", replication = "rep")
out_peanut

# AMMI analysis
test_p <- ammi.full(dataframe = peanut, environment = "env", genotype = "geno",
replication = "rep", yvar = "yield")
test_p

# heatmap plot
heatmap (test_p$means)

# plot bar plot
myd <- melt(test_p$means)
require(ggplot2)
d <- ggplot(myd, aes(genotype, value)) + geom_bar()
d + facet_wrap(~ environment) + theme_bw()

# plot pc scores (biplot)
myd2 <- data.frame (test_p$pc.scrs)

# genotype
mydgen <- myd2[myd2$category =="genotype",]
d1 <- ggplot(mydgen, aes(PC1, PC2)) + geom_point() +
```
phenosim

Simulation of phenotypic data

Description
Simulation of phenotypic data with supplied alpha, delta, mean and variance.

Usage
phenosim(n, p = 0.5, alpha = 10, delta = -10, sig = 4, mu = 50, plot = TRUE)

Arguments
n Number of individuals simulated
p p value
alpha alpha
delta delta
sig variance
mu grant mean
plot logical variable (TRUE or FALSE) depending upon whether to plot cross or not

Value
returns vector of phenotypic values. If plot "TRUE" density plot will be displayed.

Author(s)
Umesh Rosyara

References
Examples

```r
# example 1
plt <- phenosim (n = 1000 , p = 0.4, alpha = 10, delta = -5, sig = 3, mu = 60, plot = TRUE)
print(plt)
hist(plt, col = "blue")

# example 2
plt1 <- phenosim (n = 1000 , p = 0.4, alpha = 0, delta = -10, sig = 3, mu = 60, plot = TRUE)
print(plt1)
```

---

**plotblock**

*Plot complete block designs*

**Description**

The function create map (graph) for plot layout of complete block designs

**Usage**

```r
plotblock(label,plotn, nrow, ncol, g.col = 0.49, g.row = 0.45, l.pos = -0.2, fill = "azure2", h = c(0,360), psize = 3, lsize = 3)
```

**Arguments**

- `label` Vector with label for each plot (name of treatments)
- `plotn` Vector with plot number
- `nrow` Number of rows (plots per plots)
- `ncol` Number of column (number of blocks)
- `g.col` gap between two columns (a value between 0.0 and 0.5 (0.5 being no gap), option depend upon the output plot window size and shape
- `g.row` gap between two rows (a value between 0.0 and 0.5 (0.5 being no gap), option depend upon the output plot window size and shape
- `l.pos` determines whether the plot levels in comparision to treatment levels. The suggested value -0.3 to -0.1 or 0.3 to 0.1 negative value puts level below the name of treatments positive values places the plot name above the name of treaments.
- `fill` Color need to filled in plot areas, if value is "Treatment", then each treatment will have different color depending upon hue defined by h
- `h` hue value for color, 0 to 360, applicable when fill = "Treatment"
- `psize` size of plot number text
- `lsize` size of label size text

**Author(s)**

Umesh Rosyara
Examples

# example 1
genotypes <- paste("EL", 1:2, sep="")
treatment <- c(sample(genotypes), sample(genotypes), sample(genotypes), sample(genotypes), sample(genotypes))
plot.number <- 1:length (treatment)
dev.new(width= 12, height= 6)
plotblock(label = treatment, plotn = plot.number, nrow = 5 ,
ncol = length (genotypes), g.col = 0.49, g.row = 0.49, fill = "azure2", l.pos = -0.2)

# color coded
plotblock(label = treatment, plotn = plot.number, nrow = 5 ,
ncol = length (genotypes), g.col = 0.49, g.row = 0.49, fill = "treatment", l.pos = -0.2, h = c(0, 200))

plotblock(label = treatment, plotn = plot.number, nrow = 5 ,
ncol = length (genotypes), g.col = 0.49, g.row = 0.49, fill = "treatment", l.pos = -0.2, h = c(90, 180))

plotblock(label = treatment, plotn = plot.number, nrow = 5 ,
ncol = length (genotypes), g.col = 0.49, g.row = 0.49, fill = "gray8", l.pos = 0.2)

plotblock(label = treatment, plotn = plot.number, nrow = 5 ,
ncol = length (genotypes), g.col = 0.49, g.row = 0.49, fill = "gray8", l.pos = 0.2)

plotblock(label = treatment, plotn = plot.number, nrow = 5 ,
ncol = length (genotypes), g.col = 0.45, g.row = 0.45, fill = "antiquewhite", l.pos = 0.2)

plotblock(label = treatment, plotn = plot.number, nrow = 5 ,
ncol = length (genotypes), g.col = 0.45, g.row = 0.45, fill = "cornsilk", l.pos = 0.2)

plotblock(label = treatment, plotn = plot.number, nrow = 5 ,
ncol = length (genotypes), g.col = 0.45, g.row = 0.49, fill = "cadetblue1", l.pos = 0.2)

# example 2

# randomization
set.seed(1)
ntrt = LETTERS[seq( from = 1, to = 10 )]
repl <- rep (1:4, each = length (ntrt))
nsam = as.vector(replicate(4, sample(ntrt)))
plot.number <- 1:length (nsam)
newd <- data.frame (repl, nsam, plot.number)
plotblock(label = nsam, plotn = plot.number, nrow = 4 ,
ncol = length (ntrt), g.col = 0.49, g.row = 0.49, fill = "azure2", l.pos = -0.2)

plotgen

Plot genetic gain

Description

The function plots response to selection and genetic gain over generations.
Usage

`plotgen(dataframe, classvar, phenovar, selint = 0.1)`

Arguments

- `dataframe`: dataframe object
- `classvar`: factor column with generation class - for example: "F2", "F3" etc..
- `phenovar`: phenotypic data column
- `selint`: selection intensity applied

Author(s)

Umesh R. Rosyara

References


Examples

```r
set.seed (1234)
require(ggplot2)
mydf1 <- data.frame (class = c(rep("F2", 1/zero.noslash/zero.noslash/zero.noslash), rep("F3", 1/zero.noslash/zero.noslash)),
yield = c(rnorm (1/zero.noslash/zero.noslash/zero.noslash, 5/zero.noslash, 2/zero.noslash),rnorm (5/zero.noslash, 65, 5),rnorm (5/zero.noslash, 25, 5)))
plotgen(dataframe = mydf1, classvar = "class", phenovar = "yield", selint = 0.1)
```

---

`plotwith.map`  
Chromosomal maps with or without scaled ticks

Description

The plot develops single chromosome map with specified maker positions and labels. In addition to map, aligned scatter plot will be produced for additional variable (such as LOD score, minor allele frequency). The scatter plot can have points or lines or area as user specified.

Usage

```r
plotwith.map(mapdata, ydata, yvar, position, marker, type = "$1\$, ycol = "blue4", mbar.col = "gray20", ylab = "", cex.lab = 1, chr.lab = 1, ...)
```
Arguments

- mapdata: dataframe with map information
- ydata: dataframe with information for y variable (such as LOD score, minor allele frequency)
- marker: name of marker column in mapdata
- position: name of column with position of markers in mapdata
- yvar: name of yvar column in ydata
- type: type of additional information plot (type used in graphical parameters from R base). Use p for points, l for lines, b for both, h for histogram like (or high-density) vertical lines.
- ycol: Y variable colour
- mbar.col: Map bar colour
- ylab: Y axis label
- cex.lab: The magnification to be used for x and y labels relative to the current setting of cex, in scatter plot
- chr.lab: The magnification to be used for x and y labels relative to the current setting of cex, in map plot

Examples

```r
# Example 1
#minor allele frequency
position= seq(1, 100, 0.1)
mapminor <- data.frame (position, minorallele = rnorm(length(position), 0.5, 0.2))

# map
position= seq (1, 100, 5)
mapdata <- data.frame (position, snpname = paste("SNP-1-", position, sep = ""))

plotwith.map(mapdata = mapdata, ydata = mapminor, yvar = "minorallele",
position = "position", marker = "snpname", type = "l", ycol = "blue4",
mbar.col = "gray2", ylab = "Minor Alele Frequency")

plotwith.map(mapdata = mapdata, ydata = mapminor, yvar = "minorallele",
position = "position", marker = "snpname", type = "p", pch = "+",
ycol = "red4", mbar.col = "gray20", ylab = "Minor Alele Frequency")

plotwith.map(mapdata = mapdata, ydata = mapminor, yvar = "minorallele",
position = "position", marker = "snpname", type = "b", pch = 19, ycol = "red4",
mbar.col = "gray20", ylab = "Minor Alele Frequency")

plotwith.map(mapdata = mapdata, ydata = mapminor, yvar = "minorallele",
position = "position", marker = "snpname", type = "h", pch = 19, ycol = "pink",
mbar.col = "gray20", ylab = "Minor Alele Frequency")

plotwith.map(mapdata = mapdata, ydata = mapminor, yvar = "minorallele",
position = "position", marker = "snpname", type = "c", pch = 19,
ycol = "cadetblue", mbar.col = "gray20", ylab = "Minor Alele Frequency")

plotwith.map(mapdata = mapdata, ydata = mapminor, yvar = "minorallele",
Polar genome plot

Circular (polar) genome plot with markers represented in points outer circle and the genes/qtl positions identified in inner circle.

\textbf{Usage}

\texttt{polar.genome(mapdataframe, mapsubset, groupvar = "group", position = "position", gapbp = 10, pt.pch = 19, sub.pch = 17, pt.size = 4, sub.size = 6)}

\textbf{Arguments}

- \texttt{mapdataframe}: Map dataframe, with group (chromosome), position (physical or genetic position)
- \texttt{mapsubset}: Subset of map dataframe with only exactly same columns in mapdataframe, however the position of each QTL / genes is indicated
- \texttt{groupvar}: Name of group (chromosome) variable (same for both mapdataframe and mapsubset)
- \texttt{position}: Name of position (chromosome) variable (same for both mapdataframe and mapsubset)
- \texttt{gapbp}: Gap between adjacent groups
- \texttt{pt.pch}: Pch for marker circle
- \texttt{sub.pch}: Pch for qtl or gene circle
- \texttt{pt.size}: Size of pch for marker circle
- \texttt{sub.size}: Size of pch for qtl / gene circle

\textbf{Author(s)}

Umesh Rosyara

\textbf{Examples}

\begin{verbatim}
g1 <- c(1, 5, 15, 20, 30, 40)
g2 <- c(1, 15, 25, 30, 40)
g3 <- c(1, 5, 10, 25, 40, 60, 80)
mapdataframe <- data.frame(group = c(rep(1, length(g1)), rep(2, length(g2)), rep(3, length(g3))), position = c(g1, g2, g3))
mapsubset <- data.frame(group = c(1,1,2, 2,3,3, 3), position = c(25, 35, 5, 35, 8, 50, 65, 75))
\end{verbatim}
popvisd

Multiple population dataset

Description

Example multiple population dataset for frequency distribution visualization.

Usage

data(popvisd)

Format

A data frame with 3000 observations on the following 3 variables.

population populations: L100 x L134 L189 x L564 L452 x L564
trait1 trait 1 - a numeric vector
trait2 trait 2 - a numeric vector

Source

Simulated dataset

References


Examples

data(popvisd)
require(lattice)
histogram(~ trait1|factor(population), data= popvisd, nint = 10,
xlab = "trait1(measuring unit)", type = "density",
panel = function(x, ...) {
  panel.histogram(x, col = "cadetblue",...)
  panel.mathdensity(dmath = dnorm, col = "red",
                   args = list(mean=mean(x),sd=sd(x)))
})
# similar histogram using ggplot2
require(ggplot2)
qplot( trait1, data = popvisd, geom = "histogram", fill= population)+ theme_bw()
rcbsingle

Single location randomized complete block design data

Description

The single location randomization data of two traits is used to demonstrate in calculation of broad-sense heritability and genetic correlation.

Usage

data(rcbsingle)

Format

A data frame with 32 observations on the following 4 variables.

- replication a numeric vector
- genovar a numeric vector
- tgw a numeric vector
- grw a numeric vector

Examples

data(rcbsingle)

# broad sense heritability
hsq.single(dataframe = rcbsingle, yvars = "tgw", genovar = "genovar",
repairation = "replication", exout = TRUE, REML = FALSE)

# genetic correlation
gencor.lm(dataframe = rcbsingle, yvar1 = "tgw", yvar2 = "grw", genovar = "genovar",
repairation = "replication", exout = FALSE)
out <- gencor.lm(dataframe = rcbsingle, yvar1 = "tgw", yvar2 = "grw",
genovar = "genovar", repairation = "replication", exout = TRUE)
out

respdataf

Example data to visualize response to selection and heritability

Description

Example data to visualize response to selection and heritability

Usage

data(respdataf)
rowcoldata

Analysis of Row Column Experimental Design Data

Description

The data are example of row-column augmented design.

Usage

data(rowcoldata)
**Format**

A data frame with 75 observations on the following 4 variables.

- **rows** a numeric vector
- **columns** a numeric vector
- **genotypes** gentypes 50 + 5 checks
- **yield** yield - a numeric vector

**Examples**

```r
data(rowcoldata)
outp <- aug.rowcol(dataframe = rowcoldata, rows = "rows", columns = "columns",
genotypes = "genotypes", yield = "yield")
outp$ANOVA # analysis of variance
outp$Adjustment # adjusted values

# calculation of means
stab <- aggregate( yield ~ genotypes, data=rowcoldata, FUN= mean)
hist(stab$yield, col = "cadetblue", xlab = "Grain Yield",
main = "Mean yields from Augmented Yield Trial")
```

**rqt12mapchart**

Convert R/qtl object to mapchart

**Description**

The function converts R/qtl (Broman and Sen 2009) object to mapchart file (Voorrips, 2002). Mapchart is one of popular free (license can be requested) software.

**Usage**

```r
rqtl2mapchart(crossobj, outobj=, trait = "1", chr = c(1, 2, 3))
```

**Arguments**

- **crossobj** R/qtl cross object
- **outobj** R/qtl output object
- **trait** Trait name to be used to produce mapchart chart
- **chr** Chromosomes to be plotted for QTL, those chromosomes with QTL present

**Author(s)**

Umesh Rosyara

**References**


Examples

```r
#example 1
require(qtl)
data(hyper)
hyper <- calc.genoprob(hyper, step=2.5)
out.em <- scanone(hyper, method="em")
rqtl2mapchart(crossobj = hyper, outobj=out.em, trait = "obs", chr = c(1, 2, 5, 9))

#example 2
data(rqtldata)
mydata <- calc.genoprob(rqtldata, step=1, error.prob=0.001)
# standard interval mapping using EM algorithm
out.em <- scanone(rqtldata, method="em")
summary(out.em, threshold=3)
plot(out.em, chr=c(1,2))
rqtl2mapchart(rqtldata, out.em, trait = "obs", chr = c(1))
getwd() # the output file will be in the working directory
```

---

**rqtldata**

*QTL mapping example data*

Description

The simulated dataset consists of two traits and 2 chromosomes for qtl mapping using R/qtl (Broman et al. 2003, Broman and Sen 2009). R/qtl is meta-package include several functions to create maps and qtl mapping (interval, composite interval and multiple QTL mapping).

Usage

```r
data(rqtldata)
```

Format

The data is R/qtl of class "f2" and "cross".

Source

Simulated data

References


Please cite Broman et al. (2003) and Broman and Sen (2009), if you use qtl in mapping.

Examples

data(rqtlda)
require(qtl)
??qtl # for help
summary(rqtlda) # provide brief summary of the dataset
# visualizing the dataset
plot.pheno(rqtlda, pheno.col=1, col = "blue") # plot histogram of trait 1
plot(rqtlda)
# graphical genotype
plot(rqtlda)
require(qtl)

# marker regression
out.mr <- scanone(rqtlda, method="mr")
out.mr
mydata <- calc.genoprob(rqtlda, step=1, error.prob=0.001)
# standard interval mapping using EM algorithm
out.em <- scanone(rqtlda, method="em")
summary(out.em, threshold=3)
plot(out.em, chr=c(1,2))

# Haley-Knott regression
mydata <- calc.genoprob(rqtlda, step=1, error.prob=0.001)
out.hk <- scanone(rqtlda, method="hk")

# Extended Haley-Knott regression
mydata <- calc.genoprob(rqtlda, step=1, error.prob=0.001)
out.ehk <- scanone(rqtlda, method="ehk")

# Multiple imputation
mydata <- sim.geno(rqtlda, step=1, n.draws=64, error.prob=0.001)
out.imp <- scanone(rqtlda, method="imp")

# Interval estimates of QTL location
# 1.5-LOD support for chromosome 4 QTL
lodint(out.em, 2, 1.5)
# 95% Bayes credible intervals chromosome 4 QTL
bayesint(out.em, 2, 0.95)
# bootstrap-based confidence interval
out.boot <- scanoneboot(rqtlda, chr=2, n.boot=1000)
out.boot
plot(out.boot)
summary(out.boot)

# Multiple QTL model
# Composite interval mapping
out.cim.20 <- cim(rqtlda, n.marcovar=3, window=20)
out.cim.20

# Two-dimensional, two-QTL scans
mydata <- calc.genoprob(rqtlda, step=2.5, err=0.001)
out2 <- scantwo(rqtlda, verbose=FALSE, method = "em")
# multi-dimensional, multiple-QTL scans

mq.rqtldata <- mqmaugment(rqtldata, minprob=0.001) # data augmentation
mq.rqtldata.out <- mqmscan(mq.rqtldata)
summary(mq.rqtldata.out, lod =3)
plot(mq.rqtldata.out$out.cim.2/out, col = c("red","blue"))

# there are many functions available please refer to R/qtl documentation

---

**seletion.index**

### Construction of selection index

**Description**

The function implements development of selection index outlined by Smith (1936) which is based on genetic and economic worth. The detail computation procedure is outlined by Singh and Chaudhary (1985).

**Usage**


```r
seletion.index(phenodf, pcovmat, gcovmat, ecovmat, exout = TRUE, selectint = 0.01)
```

**Arguments**

- **phenodf**
  - Matrix of phenotypic data
- **pcovmat**
  - phenotypic covariance matrix
- **gcovmat**
  - genotypic covariance matrix
- **ecovmat**
  - matrix of economic value
- **exout**
  - Whether to produce extended output to screen
- **selectint**
  - Selection intensity

**Author(s)**

Umesh Rosyara

**References**


**Examples**

```r
data(selindex)
p <- seletion.index (phenodf = selindex$phenodf, pcovmat = selindex$X,
gcovmat = selindex$G, ecovmat = selindex$A)
print(p)
```
Data for selection index

Description


Usage

data(selindex)

Format

The data is list of threee matrix and a phenotypic dataframe.

- $X$ : phenotypic covariance matrix
- $G$ : genotypic covariance matrix
- $A$ : economic covariance matrix
- phenodf: 'data.frame': 8 obs. of 5 variables:
  - parents: parents
  - trait1 : trait 1
  - trait2 : trait 2
  - trait3 : trait 3
  - trait4 : trait 4

Details

List of X, G, A, phenodf

References


Examples

data(selindex)
p <- selection.index (phenodf = selindex$phenodf, pcovmat = selindex$X, gcovmat = selindex$G, ecovmat = selindex$A)
print(p)

shaded.normal

Shading regions in theoretical normal curves or sample density curves for quantitative traits

Description

The function is useful for teaching and publication purpose.
shaded.normal

Usage

shaded.normal(type = "TH", trait = NULL, avg = 0, sdev = 1, shade = "percent", lowrp, uprp = 0, Lcolfill = "lightgreen", Fcolfill = "pink", lincolor = "blue", lat = NULL)

Arguments

type "TH" if theoretical distribution with specified average (avg) and standard deviation (sdev). If type is "TR", distribution for trait datapoints provided in trait column

trait If type is "TR", trait is vector of trait values, otherwise NULL

avg mean of population if type is "TH", else NULL

sdev standard deviation of population if type is "TH", else NULL

shade "percent" - Whether to shade upper or lower percent, "trunp" - when defined is upper or lower truncation point

lowrp Lower truncation point or percent in the distribution

uprp Upper truncation point or percent in the distribution

Lcolfill Color to fill lower area (polygon)

Fcolfill Color to fill upper area (polygon)

lincolor Color of additional vertical lines added to plot

lat Point of additional vertical lines added to plot

Value

The function will output shaded normal or density curves with user defined shaded regions on the trails of the density plot of observed or theoretical distribution

Author(s)

Umesh Rosyara

Examples

# plot with mean 0 and sd = 1 , percent in fraction highlighted
shaded.normal(lowrp = 0.1, uprp = 0.1, avg = 50, sdev= 40)

# plotting density
shaded.normal (type = "TH", trait = NULL, avg = 20, sdev= 5, shade = "percent", lowrp = 0.10, uprp = 0.2, Fcolfill = "lightgreen", Lcolfill = "aquamarine3", lincolor = "blue", lat = NULL)

shaded.normal (type = "TH", trait = NULL, avg = 20, sdev= 5, shade = "percent", lowrp = 0.3, uprp = 0.05, Fcolfill = "#F5F5DC", Lcolfill = "#FF7F50", lincolor = "blue", lat = NULL)

shaded.normal (type = "TH", trait = NULL, avg = 20, sdev= 5, shade = "percent", lowrp = 0.10, uprp = 0.2, Fcolfill = "lightgreen", Lcolfill = "aquamarine3", lincolor = "blue", lat = NULL)

# plot with mean 0 and sd = 1 , percent in fraction highlighted
par(mfrow=c(3,1))
stability

Stability analysis based on Eberhart and Russell (1966) model

Description

The function implements the Eberhart and Russell (1966) model for stability analysis.

Usage

stability(dataframe, yvar, genotypes, environments, replication)

Arguments

dataframe : dataframe with Y variables, genotype, environment, and replication
yvar : Name of Y variable
genotypes : Name of genotype variable
environments : Name of environments variable
replication : Name of replication variable

Author(s)

Umesh R Rosyara

References


Examples

yvar <- c( 36.4, 40.0, 32.4, 33.5, 41.3, 27.9, 38.5, 38.6, 41.6, 22.6, 41.3, 38.9, 30.9, 40.1, 43.6, 36.3, 43.0, 29.6, 34.4, 35.1, 51.7, 37.1, 25.5, 47.4, 39.5, 39.1, 36.1, 40.6, 28.6, 32.8, 33.0, 22.6, 42.6, 52.8, 20.3, 38.3, 39.4, 36.5, 31.7, 22.8, 33.2, 39.4, 28.2, 45.8, 28.6, 35.4, 36.5, 37.4, 21.0, 25.4, 28.3, 30.2, 29.5, 32.9, 29.5, 47.6, 40.3, 30.8, 30.1, 34.5, 35.8, 21.8, 27.1, 28.6, 25.5, 28.5, 24.5, 27.1, 25.4, 22.4, 24.9, 32.4,
replication <- c( rep(c(rep(1, 10), rep(2,10), rep(3,10)),5))
genotypes <- c(rep(paste("G", 1:10, sep= ""), 15))
environments <- c(rep(paste("CB","CA", "CC", "MN","SD"), each = 30))
mydf1 <- data.frame (yvar, replication, genotypes, environments)
out <- stability (dataframe = mydf1 , yvar = "yvar", genotypes = "genotypes",
environments = "environments", replication = "replication")
# print out
out

table.creator  Table creator

Description  Creates tables for categorical variable or categorizing quantitative variable.

Usage  

\texttt{table.creator(mydata, yvar = FALSE, classvars, classy = FALSE, ycut = NULL)}

Arguments

- **mydata**: Name of dataframe
- **yvar**: Name of variable
- **classvars**: Name of class variable
- **classy**: Name of class variable
- **ycut**: cut vector with information how to categorize a quantitative variable

Author(s)

Umesh Rosyara

Examples

\texttt{# Example 1:}
\texttt{mydata1 <- data.frame (yvar1 = rnorm(2000, 15, 5), xv1 = rep(1:5, each = 400),}
\texttt{xv2 = rep(1:10, 200), xv3 = rep(1:2, 1000), xv4 = rep(1:2, 1000))}
\texttt{table.creator (mydata = mydata1, yvar = NA, classvars = c("xv1", "xv2", "xv3"),
ycut = FALSE)}
\texttt{table.creator (mydata = mydata1, yvar = NA, classvars = c("xv2", "xv3"), ycut = FALSE)}
table.creator (mydata = mydata1, yvar = "yvar1", classvars = c("xv2", "xv3"),
classy = TRUE, ycut = c(-Inf,10,14,16,Inf))

table.creator (mydata = mydata1, yvar = "yvar1", classvars = c("xv2", "xv3", "xv1"),
classy = TRUE, ycut = c(-Inf,10,14,16,Inf))

outv <- table.creator (mydata = mydata1, yvar = "yvar1",
classvars = c("xv2", "xv3", "xv1", "xv4"), classy = TRUE, ycut = c(-Inf,10,14,16,Inf))

# Example 2
snpprop <- data.frame (SN = 1:4, chromosome = as.factor (rep(1:10, each = 4)),
genome = sample (c("A", "B", "C"), 4000, replace = "TRUE"),
snpsource = sample (c("Nap", "Kat"), 4000, replace = "TRUE"),
minorAF = rnorm (4000, 0.5, 0.1), GenTrain = rnorm(4000, 0.8, 0.05))

summary (snpprop)
af1 <- table.creator (mydata = snpprop, yvar = FALSE,
classvars = c("chromsome","genome"), classy = FALSE, ycut = NULL)

snpout <- table.creator (mydata = snpprop, yvar = "minorAF",
classvars = c("chromosome","genome"), classy = TRUE, ycut = c(-Inf,0.15,0.35,0.75,Inf))

snpout1 <- table.creator (mydata = snpprop, yvar = "minorAF",
classvars = c("snpsource","chromsome","genome"),
classy = TRUE, ycut = c(-Inf,0.5,Inf))
snpout1[["(-Inf,0.5]"]]

# cateogrizing numerical variables and cross tabling it
snpprop$GTcategory <- cut(snpprop$GenTrain,
breaks = c(-Inf,0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9,Inf))

snpout2 <- table.creator (mydata = snpprop, yvar = "minorAF",
classvars = c("GTcategory"),
classy = TRUE, ycut = c(-Inf,0.15,0.35,0.75,Inf))

snpout2

snpout3 <- table.creator (mydata = snpprop, yvar = "minorAF",
classvars = c("chromsome","GTcategory"),
classy = TRUE, ycut = c(-Inf,0.15,0.35,0.75,Inf))
snpout3[["(0.75, Inf]"]]

variability

Example data for plotting genetic diversity

Description

Ten genotypes were characterized by 100 markers. This data will be used to demonstrate some R functionalities to perform diversity analysis and plotting.
Usage

data(variability)

Format

A data frame with 10 observations on the following 101 variables.

genotype  a numeric vector
MR1      a numeric vector
MR2      a numeric vector
MR3      a numeric vector
MR4      a numeric vector
MR5      a numeric vector
MR6      a numeric vector
MR7      a numeric vector
MR8      a numeric vector
MR9      a numeric vector
MR10     a numeric vector
MR11     a numeric vector
MR12     a numeric vector
MR13     a numeric vector
MR14     a numeric vector
MR15     a numeric vector
MR16     a numeric vector
MR17     a numeric vector
MR18     a numeric vector
MR19     a numeric vector
MR20     a numeric vector
MR21     a numeric vector
MR22     a numeric vector
MR23     a numeric vector
MR24     a numeric vector
MR25     a numeric vector
MR26     a numeric vector
MR27     a numeric vector
MR28     a numeric vector
MR29     a numeric vector
MR30     a numeric vector
MR31     a numeric vector
MR32     a numeric vector
MR33     a numeric vector
MR34     a numeric vector
variability

MR35  a numeric vector
MR36  a numeric vector
MR37  a numeric vector
MR38  a numeric vector
MR39  a numeric vector
MR40  a numeric vector
MR41  a numeric vector
MR42  a numeric vector
MR43  a numeric vector
MR44  a numeric vector
MR45  a numeric vector
MR46  a numeric vector
MR47  a numeric vector
MR48  a numeric vector
MR49  a numeric vector
MR50  a numeric vector
MR51  a numeric vector
MR52  a numeric vector
MR53  a numeric vector
MR54  a numeric vector
MR55  a numeric vector
MR56  a numeric vector
MR57  a numeric vector
MR58  a numeric vector
MR59  a numeric vector
MR60  a numeric vector
MR61  a numeric vector
MR62  a numeric vector
MR63  a numeric vector
MR64  a numeric vector
MR65  a numeric vector
MR66  a numeric vector
MR67  a numeric vector
MR68  a numeric vector
MR69  a numeric vector
MR70  a numeric vector
MR71  a numeric vector
MR72  a numeric vector
MR73  a numeric vector
MR74  a numeric vector
variability

MR75 a numeric vector
MR76 a numeric vector
MR77 a numeric vector
MR78 a numeric vector
MR79 a numeric vector
MR80 a numeric vector
MR81 a numeric vector
MR82 a numeric vector
MR83 a numeric vector
MR84 a numeric vector
MR85 a numeric vector
MR86 a numeric vector
MR87 a numeric vector
MR88 a numeric vector
MR89 a numeric vector
MR90 a numeric vector
MR91 a numeric vector
MR92 a numeric vector
MR93 a numeric vector
MR94 a numeric vector
MR95 a numeric vector
MR96 a numeric vector
MR97 a numeric vector
MR98 a numeric vector
MR99 a numeric vector
MR100 a numeric vector

Source
Simulated data

Examples

data(variability)
attach(variability)
lf <- paste("MR",1:100, sep=' ', collapse = " + ")
formula <- as.formula(paste("genotype", lf, sep = " ~"))

# cluster analysis
HClust.2 <- hclust(dist(model.matrix(formula, variability)) , method= "ward")
plot(HClust.2, main= "Cluster Dendrogram for Solution HClust.2",
xlab= "Observation Number in Data Set variability", sub="Method=ward; Distance=euclidian")

# Calculate probability of cluster in dendogram using bootstrap method
# transposing the dataset
tvariability = data.frame(t(variability)[-1,])
wintwheat

Winter wheat data from mixed model data SAS

Description

Winter wheat data from mixed model data SAS

Usage

data(wintwheat)

Format

A data frame with 60 observations on the following 3 variables.

- **Variety**: a factor with levels 1 1 2 3 4 5 6 7 8 9
- **Yield**: a numeric vector
- **Moisture**: a numeric vector

Source


Examples

data(wintwheat)
par(mfrow = c(1,4))
hist(wintwheat$Yield)
qqnorm(wintwheat$Yield)
qqline(wintwheat$Yield)
boxplot(wintwheat$Yield)
boxplot (wintwheat$Yield ~ wintwheat$Variety)

# ANOVA
model<-aov(Yield~Variety,data=wintwheat)
plot (model)# Check assumption of anova
plot.design (wintwheat$Yield ~ wintwheat$Variety)# Effect sizes graphically
model.tables (model, "means", se=TRUE)# Standard error of means
summary.lm (aov (model))# Another-way to see effect size

# mean comparisons
# Tukey test
modelTukey=TukeyHSD(model, "Variety", ordered = TRUE)
modelTukey
Index

*Topic.datasets*
  augblock, 16
  datapbib, 21
  fulldial, 25
  linetester, 38
  northcaro1, 50
  northcaro2, 51
  onfarm, 53
  parentoffs, 54
  peanut, 55
  popvisd, 63
  rcbsingle, 64
  respdataf, 64
  rowcoldata, 65
  rqtlddata, 67
  selindex, 70
  variability, 74
  wintwheat, 78

*Topic.package*
  plantbreeding-package, 3

  adesign, 6
  agricolae, 4
  alphasim, 7
  ammi.full, 8
  assambly.plot, 10
  assoc.unr, 11
  AUDPC.cal, 12
  aug.rcb, 13
  aug.rowcol, 15
  augblock, 7, 16
  augmentdesign, 16
  balincom, 17
  carolina1, 18
  carolina2, 20
  datapbib, 21
  dialele1, 22
  diversity, 23
  fulldial, 25
  gencor.lm, 26
  geno.convert, 27
  genotype2alleles, 29
  graphicalgeno, 30
  gs2joinmap, 32
  histlab, 34
  hsq.single, 35
  line.tester, 37
  linetester, 38
  manhattan.circos, 38
  manhattan.plot, 40
  map.fill.ggplot, 42
  map.plot, 43
  mapbar.plot, 44
  mapone, 45
  multienv, 47
  multloc, 48
  northcaro1, 50
  northcaro2, 51
  onemap, 4
  onemap2mapchart, 52
  onfarm, 53
  parentoffs, 54
  peanut, 55
  phenosim, 57
  plantbreeding (plantbreeding-package), 3
  plantbreeding-package, 3
  plotblock, 58
  plotgen, 59
  plotwith.map, 60
  polar.genome, 62
  popvisd, 63
  qtl, 4
  rcbsingle, 64
  respdataf, 64
  rowcoldata, 65
  rqtl2mapchart, 66
  rqtldata, 67
selement.index, 69
selindex, 70
shaded.normal, 70
stability, 72

table.creator, 73
variability, 74
wintwheat, 78