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` *Analysis and visualization of data from plant breeding and genetics experiments*

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

Allard R.W.(1999) Principles of Plant Breeding, John Wiley and Sons, May 10, 1999 - 254 pages

Sleper D.A.(2006) Poehlman J.M. Breeding Field Crops, Blackwell Pub., Jul 25, 2006 - 424 pages

Hill J., Becker H.C., Tigerstedt P.M. A. (1998) Quantitative and Ecological Aspects of Plant Breeding, Springer, 1998 - 275 pages

Lynch M., Walsh B. (1998). Genetics and Analysis of Quantitative Traits. Sinauer, Sunderland, MA

Falconer D. S., Mackay T.F.C. (1996). Introduction to Quantitative Genetics. Fourth edition. Addison Wesley Longman, Harlow, Essex, UK.

Mather K., Jinks J.L. (1971). Biometrical Genetics. Chapman & Hall, London.

Sorensen D., Gianola D.(2002). Likelihood, Bayesian, and MCMC Methods in Quantitative Genetics. Springer 739 pp.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Saxton A. (2004) Genetic Analysis of Complex Traits Using SAS. SAS Institute, Inc.

Littell R.C.(2006) SAS for Mixed Models,SAS Institute, Inc.

Broman K.W., Sen S. (2009) A Guide to QTL Mapping with R/qtl. SBH/Statistics for Biology and Health. Springer

Wickham H. (2009) ggplot: Elegant Graphics for Data Analysis. Use R. Springer.

Sarkar D. (2008) Lattice: Multivariate Data Visualization with R. Springer, New York

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 <-diallele1(dataframe = fulldial, male = "MALE", female = "FEMALE",
 progeny = "TRT", replication = "REP", yvar = "YIELD" )
print(out)
out$anvout # analysis of variance
out$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 Augumented 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 = 1: 2000*20 , 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 = "regular", colour = "multicolor" , geom = c("line","point"))

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) )



### Example 5: plot maps with additional informations
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
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 = 10,  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")
```

`adesign` *Design Augmented Block Design experiment*

Description

The function generates randomized plans for augmented block design.

Usage

```
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 the each block consequetively)
`r`	total number of replications (single number)
`seed`	Random seed

Details

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

Value

Returns dataframe with randomization with plot number, blocks and treatments

Author(s)

Umesh R. Rosyara

References

Mead, R. 1997. Design of plant breeding trials. In Statistical Methods for Plant Variety Evaluation. eds. Kempton and Fox. Chapman and Hall. London.

See Also

augblock

Examples

```
## 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")
newtrt1 = c("SD101", "SD102", "SD302", "MN102", "MI6789", "KS2034","SD134", "SD402", "SD4342",
 "MN232", "MI69", "KS234",  "SD451", "SD892", "SD212", "MN344", "MI649", "KS336",
 "SD345", "SD425", "SD5662", "MN892", "MI902", "KS4",  "SD333", "SD1212", "SD021",
 "MN223L", "MI69n", "KS2123","SD145", "SDJ1", "SD4234", "MN90", "MI4567", "KS956",
"SD9901", "SD6602", "SD2202", "MN4402", "MI892", "KS2421","SD400", "SD4029", "SD987",
 "MN2333", "MI690", "KS214")
r1 = 4
block.size1 = c(16, 16, 16, 16)
print(EX2 <- auugmentdesign (checks = checks1, newtrt = newtrt1,
block.size = block.size1, r = r1,  seed = 124))
```

---

alphasim *Alpha Design Dataset*

Description

The dataset is simulated alpha design dataset with 30 treatments, 2 replications, k = 3 and s = 10.

Usage

```
data(alphasim)
```

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

`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:

Mendiburu F (2010) agricolae: Statistical procedures for agricultural research, The Comprehensive R Archive Network

Source

simulated dataset

References

Mendiburu F (2010) agricolae: Statistical procedures for agricultural research, The Comprehensive R Archive Network

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 evironmentenvironment 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

Gauch H.G.(1992). Statistical analysis of regional yield trials:AMMI analysis of factorial designs. Elsevier, Amsterdam.

Gauch H.G. (2006). Statistical analysis of yield trials by AMMI and GGE. Crop Sci. 46:1488-1500.

Gauch H.G., Zobel.R.W. (1996). AMMI analysis of yield trials. p.85-122. In M.S. Kang and H.G. Gauch, Jr. (ed.) Genotype-byenvironment interaction. CRC Press, Boca Raton, FL.

Gower J.C., Hand D.J. (1996). Biplots. Monographs on Statistics and Applied Probability. London, UK: Chapman & Hall

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) +
```

```
  ylab ("PC2") + xlab ("PC1") + theme_bw()
  print(d2)

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

`assambly.plot` *Assembly plot*

Description

The function develops assambly plot of Displaying multiple assambled line segments such as plotting of chrosmomechromosome 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 segrementssegments
`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, defualtdefault 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, 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))
```

```
assambly.plot (data = dat, yvar = "yvar", xstart = "start", xend = "end",
id = "id", xlab = "position",  ylab = "", linecol = "cyan4", lwd = 6, lend = 2)
abline ( v = 15,  col = "pink4")

 assambly.plot (data = dat, yvar = "yvar", xstart = "start", xend = "end",
 id = "id", xlab = "position",  ylab = "Fragments",
 linecol = "red", lwd = 4, lend = 2)

 abline ( h = 5, col = "pink4")
```

`assoc.unr` *Association mapping in unrelated or unstructured mapping population using linear model*

Description

The function pefroms mapping in unrelated or without clear population structure population using linear model. Both binomial (eg. susceptible or resistance, diseased or disease free, present or abscent) 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

```
# simulated example
set.seed(3456)
id <- c(1:115)
snpmat<- data.frame(matrix(sample(c("AA","AB","BB"), 115000, replace = TRUE), ncol = 1000))
names(snpmat) <- c(paste ("SNP",1:1000, 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 = 6:10,
 covariate = FALSE, cvar = NA, binomial = FALSE, model = "ADD")
out1.add

out1.dom <- assoc.unr(dataframe = snpdata,  yvar = "trait1", xvars = 6:10,
covariate = FALSE, cvar = NA, binomial = FALSE, model = "DOM")
out1.dom

out1.dom <- assoc.unr(dataframe = snpdata,  yvar = "trait1", xvars = 6:10,
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), 100),]
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 probablities
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

```
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

Jeger M.J., Viljanen-Rollinson S.L.H. (2001) The use of the area under the disease-progress curve (AUDPC) to assess quantitative disease resistance in crop cultivars, Theor Appl Genet 102:32-40

Madden L.V., Hughes, G., van den Bosch, F. (2007) The study of plant disease epidemics. The American Phytopathological Society, APS Press St. Paul, Minnesota.

Examples

```
# 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)
```

---

aug.rcb *Analysis of augmented random block design*

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 varitiesvarieties / 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")
```

aug.rowcol *Analysis of Augmented row and column design*

Description

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

Usage

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

Arguments

dataframe	dataframe object with at least columns with information of rows, columns, genotypes / entries/ varieties / or treatments and yield (yvariable)
rows	name of numbericnumeric variable with rows number
columns	name of numeric variable with column number
genotypes	name of column with with treatments / genotypes (factor)
yield	name of column with yield or any y variable

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 / varitiesvarieties / 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

```
# 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

```
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

```
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 augumented block design

Usage

```
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 can not 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

Mead, R. 1997. Design of plant breeding trials. In Statistical Methods for Plant Variety Evaluation. eds. Kempton and Fox. Chapman and Hall. London.

Examples

```
## 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")

newtrt1 = c("SD101", "SD102", "SD302", "MN102", "MI6789", "KS2034","SD134",
 "SD402", "SD4342", "MN232", "MI69", "KS234",
 "SD451", "SD892", "SD212", "MN344", "MI649", "KS336","SD345",
  "SD425", "SD5662", "MN892", "MI902", "KS4",
"SD333", "SD1212", "SD021", "MN223L", "MI69n", "KS2123","SD145", "SDJ1",
 "SD4234", "MN90", "MI4567", "KS956","SD9901", "SD6602", "SD2202", "MN4402",
 "MI892", "KS2421","SD400", "SD4029", "SD987", "MN2333", "MI690", "KS214")
r1 = 4
block.size1 = c(16, 16, 16, 16)
print(EX2 <- auugmentdesign (checks = checks1, newtrt = newtrt1,
block.size = block.size1, r = r1,  seed = 124))
```

`balincom` *Data from a balanced incomplete block design*

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

Littel, R. C., Milliken, G. A., Stroup, W. W., and Wolfinger, R. D. (1996), SAS System for Mixed Models, SAS Institute (Data Set 5.4)

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 perfromsperforms 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 sould 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

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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Mather K., Jinks J.L. (1971). Biometrical Genetics. Chapman & Hall, London.

Saxton A. (2004) Genetic Analysis of Complex Traits Using SAS. SAS Institute, Inc.

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 # femal 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 )

p2
p2$model
p2$`BULP estimates`
```

`carolina2` *Analysis of North Carolina design II*

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 variablesevariables 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

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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers.

Mather K., Jinks J.L. (1971). Biometrical Genetics. Chapman & Hall, London.

Saxton A. (2004) Genetic Analysis of Complex Traits Using SAS. SAS Institute, Inc.

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

Littel, R. C., Milliken, G. A., Stroup, W. W., and Wolfinger, R. D. (1996), SAS System for Mixed Models, SAS Institute (Data Set 1.5.1).

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 *Analysis of Diallel data*

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

Griffing, B. 1956. Concept of general and specific combining ability in relation to diallel crossing systems. Austr. J. Biol. Sci. 9, 463-493.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Mather K., Jinks J.L. (1971). Biometrical Genetics. Chapman & Hall, London.

Examples

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

print(out)
out$anvout # analysis of variance
out$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 pricipal component analysis, cluster analysis and development of heatmap plot.

Usage

```
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 pricipal 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 heapmap plot
rc	vector with row colour for heatmap
cc	vector with column colour for heatmap

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 Pricipal 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 samething 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()[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", "#8B0A50", "#696969",
 "#ADFF2F", "#BEBEBE", "#8B0000", "#FFB90F")

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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

References

Griffing, B. 1956. Concept of general and specific combining ability in relation to diallel crossing systems. Austr. J. Biol. Sci. 9, 463-493.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Mather K., Jinks J.L. (1971). Biometrical Genetics. Chapman & Hall, London.

Examples

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

print(out)
out$anvout # analysis of variance
out$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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Falconer D. S., Mackay T.F.C. (1996). Introduction to Quantitative Genetics. Fourth edition. Addison Wesley Longman, Harlow, Essex, UK.

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, 42,45,43,45,43,43,42,43,42,40,43,41), 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
```

---

geno.convert *Recoding genotypes*

Description

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

Usage

```
geno.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 fileoutput files to be produced.
outsep	The output file seperator. Use "," for comma delinimated file, " " space delimi-nated and other delimiter is also supported
na.strings	what spring 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)
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)

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
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
# 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 alelele format*

Description

Converting SNP or other two letter genotype format to alelele format

Usage

```
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 seperator - eg "," for csv file input or " " space deliminated (this is seperator 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 seperator 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

```
# Example 1
A1 <- c("A/B", "A/A", "B/B", "A/A")
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)

# Example 2
ID <- 1:4
pos <- c(0, 245, 567, 871)
A1 <- c("A/B", "A/A", "B/B", "A/A")
B1 <- c("B/B", "C/C", "C/B", "D/A")
C1 <- c("B/B", "C/C", "-/-", "D/A")
```

```
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

```
graphicalgeno(dataframe = dataframe, group = "group", position = "position",
 yvar = "yvar", ycat = "ycat", namevar = "namevar", subset = TRUE,
 subsetdata,  panel.margin = 0.1, 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 indivudalindividual id
`subset`	Logical variable whether to subset data or not
`subsetdata`	Name of subset dataframe. WheverWhenever subsettted 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 interepretedinterpreted as number of lines
`strip.background`	Color of strip background
`filllow`	Lower color for the heatmap
`fillhigh`	Higher color for the heatmap

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
 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 >= 30 & position <= 50) |
 (group == 3 & position >= 20 & position <= 60))
datas1 = subset(dataframe,(group == 1 & position >= 30 & position <= 50) |
(group == 3 & position >= 20 & position <= 60) |
(group == 3 & position >= 80 & 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")

datas1 = subset(dataframe,(group == 1 & position >= 30 & position <= 50) |
  (group == 3 & position >= 20 & position <= 60)
   |(group == 5 & position >= 80 & position <= 85) )

graphicalgeno (dataframe = dataframe, group = "group",position = "position", yvar = "yvar",
  ycat = "ycat",namevar = "namevar",subset = TRUE, subsetdata = datas1, 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 = datas, 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",
```

```
   ycat = "ycat",namevar = "namevar",subset = FALSE, subsetdata = datas, panel.margin = 0 ,
    strip.background = "gray80", filllow = "yellow", fillhigh = "pink4", textcolor = "darkgreen",
     chr.arrange = "TB")

graphicalgeno (dataframe = dataframe, group = "group",position = "position", yvar = "yvar",
    ycat = "ycat",namevar = "namevar",subset = FALSE, subsetdata = datas, panel.margin = 0 ,
     strip.background = "gray80", filllow = "yellow", fillhigh = "midnightblue",
      textcolor = "red", chr.arrange = "LR")

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 = FALSE,
      textcolor = NA, chr.arrange = "LR")

# Example 2
# data
ID <- c("A", "B", "C", "D")
chromosomes <- rep (c(1,   1, 1,   1,   1,  1,    2,  2,    2,   2,  2,   2,
3,  3,   3,  3, 3), each = length(ID))
markposition <- rep(c(1,  5, 10,   20, 35,  40,   1,  15,  18,  20, 30,  60,
   1, 15,  20,  25, 26), each = length(ID))
genotypes <- c(sample (c("AC", "CC", "AA"), length(markposition), replace = TRUE))

genotypicprob <- rnorm (length(markposition), 0.5, 0.1)
idvar <- rep(ID, length(markposition)/4)
genoprob.data <- data.frame (chromosomes, markposition, genotypes, genotypicprob, idvar)

# plotting
graphicalgeno (dataframe = genoprob.data, group = "chromosomes",
position = "markposition",   yvar = "genotypicprob", ycat = "genotypes",
namevar = "idvar",subset = FALSE,subsetdata = datas, panel.margin = 0.1 ,
strip.background = "lightpink", filllow = "yellow",
 fillhigh = "blue", textcolor = "blue", chr.arrange = "DF")
```

`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.kyazma.nl/index.php/mc.JoinMap/sc.Evaluate

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", "AA", "BB", "BB", "AA", "--", "BB")
loctype <-c(4, 3, 5, 5, 3,6,6, 0)

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

# F2 population
mark1 <- c("AA", "BB", "AB", "BB", "BB", "AB", "--", "BB")
mark2 <- c("BB", "AA", "AA", "BB", "BB", "AA", "--", "BB")
mark3 <- c("BB", "AA", "AA", "BB", "BB", "AA", "--", "BB")
mark4 <- c("AA", "BB", "AA", "BB", "BB", "AA", "--", "BB")
mark5 <- c("AA", "BB", "AA", "BB", "BB", "AA", "--", "BB")
mark6 <- c("--", "BB", "AA", "BB", "BB", "AA", "--", "BB")
mark7 <- c("AA", "--", "AA", "BB", "BB", "AA", "--", "BB")
mark8 <- c("BB", "AA", "AA", "BB", "BB", "AA", "--", "BB")
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(2,3,3,4,4)
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)
```

```
breedpop <- data.frame (population = rep(paste( "Pop", 1:3,sep = "-"), each = 100),
yvar = rnorm(300, 50, 30))

# arrow label
population = c("Pop-1", "Pop-1", "Pop-2", "Pop-2","Pop-3", "Pop-3" )
parents = c  ("Res101", "Suc102", "P1", "P2", "L101-3-2","L101-3-3")
yvar1 = c(0, 80, 14, 75, 14, 75)

histlab(dataframe = breedpop, classvar = "population", yvar = "yvar",
arrow_yvar = yvar1, arrow_label = parents, arrow_class = population,
  bwidth = 20, colour = "green", fill = "lightseagreen")

# example 3, single population
set.seed(123)
pop2 <- data.frame (population = rep ("RxL", 500), yvar2 = rnorm(500, 0, 1))

# arrow label
population = c("RxL", "RxL" )
parents = c  ("line1", "line2")
yvar2 = c(0.5, 1.5)

histlab(dataframe = pop2, classvar = "population", yvar = "yvar2", arrow_yvar = yvar2,
 arrow_label = parents, arrow_class = population,  bwidth = 0.1, colour = "salmon",
 fill = "lightsalmon")
```

`hsq.single` *Broad 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`	name of Y variable used in the model
`genovar`	name of genotype variable used in the analysis
`replication`	name 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
`hertiability`	Heritability

When REML = TRUE

`hertiability`	Heritability
`genovar`	Genotypic variance
`totalvar`	Total variance
`randomeffects`	Random effects

When exout is FALSE only heritability is returned.

Author(s)

Umesh Rosyara

References

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Saxton A. (2004) Genetic Analysis of Complex Traits Using SAS. SAS Institute, Inc.

Littell R.C.(2006) SAS for Mixed Models,SAS Institute, Inc.

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= TRUE, REML = TRUE)
```

`line.tester` *Line x Tester Analysis*

Description

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

Usage

```
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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Examples

```
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

```
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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Examples

```
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

```
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 consequtive groups, 0 if no gap is required
type	Value "polar" is polar plot need to be produced, "regular" if regular catisian 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*20 , 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",ymax = "maximum", ymin = 0, gapbp = 1000,
type = "regular", colour = "multicolor" , geom = c("line","point"))
```

`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

```
manhatton.plot(dataframe, SNPname, chromosome, position, pvcol, ymax = "maximum",
 ymin = "minimum", gapbp = 500, pch = c(18, 19, 20), 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 adjecent chromsomome 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 chrosomomes 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 chrosomomes 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 prameters (not conflicting with specified here) can be applied to plot.

Value

Produce Manhattan plot

Author(s)

Umesh Rosyara

Examples

```
# 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", "#009E73",
"#F0E442", "#0072B2", "#D55E00", "#CC79A7","#CD661D", "#FF00FF","#8B6508",
 "#D2691E","#008B00", "#8B1A1A",  "#8B3A62", "#8B864E", "#3CB371", "#8B5742",
  "#8B5A00", "#36648B")

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
dat2 <- data22[!is.na(data22$pval1),]
```

```
op <- par(mfrow=c(2,1), cex.axis = 0.75, font = 1, family = "serif")
par(op)
manhatton.plot(dataframe = dat2, SNPname = "snp", chromosome = "chr", position = "pos",
 pvcol = "pval1", line1 = 4, line2 = 8, ymax = "maximum", ymin = 0, gapbp = 2000)
title(main = "Mahattan plot of results for trait1", sub = "Method: Linear mixed model")

manhatton.plot(dataframe = dat2, SNPname = "snp", chromosome = "chr", position = "pos",
 pvcol = "pval2", line1 = 4, line2 = 8, ymax = "maximum", ymin = 0)
title(main = "Mahattan plot of results for trait2", sub = "Method: Linear mixed model")
```

`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 continous variable such as linkage disequilibrium (LD block plot). The output is simar 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 dataframe
`chr`	Name of chromosome variable within the map dataframe
`label`	Name of label variable in the map dataframe
`position`	Name of map position variable within the map dataframe
`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

```
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

```
### 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 disequilibiriumdisequilibrium. For example we can plot linkage disequilibrium information between adjecent 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 chromsome 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

chr1	Name for chromsome in the dataframe filld
fillcol	Name of variable in dataframe for filling colors (for example linkage disequibrium) - for n markers n-1 colors correspondng to the interval plotting should provided.

Details

The bar size can be changed by changing dimension of plot are (reduced heigh smaller bar, increased heigh 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 = 10, 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

Margarido GRA, Souza AP, Garcia AAF (2007) OneMap: software for genetic mapping in outcrossing species. Hereditas 144: 78-79

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 indetail 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 oder.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)
```

```
# 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

# pefroming 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- evironment data*

Description

Simulated Multi- evironment 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

Gauch H.G.(1992). Statistical analysis of regional yield trials:AMMI analysis of factorial designs. Elsevier, Amsterdam.

Gauch H.G. (2006). Statistical analysis of yield trials by AMMI and GGE. Crop Sci. 46:1488-1500.

Gauch, H.G., Zobel.R.W. (1996). AMMI analysis of yield trials. p.85-122. In M.S. Kang and H.G. Gauch, Jr. (ed.) Genotype x byenvironment interaction. CRC Press, Boca Raton, FL.

Eberhart S.A., Russell W.A. (1966) Stability parameters for comparing varieties. Crop Sci. 6: 36-40.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Kang M.S., Aggarwal V.D., Chirwa R.M.(2006) Adaptability and stability of bean cultivars as determined via yield-stability statistic and GGE biplot analysis. J. Crop Improv. 15:97-120

Examples

```
# 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()
 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) +
   ylab ("PC2") + xlab ("PC1") + theme_bw()
  print(d2)
```

`multloc` *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

Littel, R. C., Milliken, G. A., Stroup, W. W., and Wolfinger, R. D. (1996), SAS System for Mixed Models, SAS Institute (Data Set 2.8.1).

References

Gauch H.G.(1992). Statistical analysis of regional yield trials - AMMI analysis of factorial designs. Elsevier, Amsterdam.

Gauch H.G. (2006). Statistical analysis of yield trials by AMMI and GGE. Crop Sci. 46:1488-1500.

Gauch, H.G., Zobel.R.W. (1996). AMMI analysis of yield trials. p.85-122. In M.S. Kang and H.G. Gauch, Jr. (ed.) Genotype-byenvironment interaction. CRC Press, Boca Raton, FL.

Eberhart S.A., Russell W.A. (1966) Stability parameters for comparing varieties. Crop Sci. 6: 36-40.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Kang M.S., Aggarwal V.D., Chirwa R.M.(2006) Adaptability and stability of bean cultivars as determined via yield-stability statistic and GGE biplot analysis. J. Crop Improv. 15:97-120

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 (mydenv)), colour = "red",hjust=1.2, vjust=0) +
  ylab ("PC2") + xlab ("PC1") + theme_bw()
 print(d2)
```

`northcaro1` *North Carolia Design I*

Description

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

Usage

```
data(northcaro1)
```

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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani 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.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Mather K., Jinks J.L. (1971). Biometrical Genetics. Chapman & Hall, London.

Saxton A. (2004) Genetic Analysis of Complex Traits Using SAS. SAS Institute, Inc.

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 # femal 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)
```

---

northcaro2 *North Carolina Design II*

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` femail

`male` male

`plrv` plrv

`yield` yield

`tuber` tuber

`weight` weight

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.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Mather K., Jinks J.L. (1971). Biometrical Genetics. Chapman & Hall, London.

Saxton A. (2004) Genetic Analysis of Complex Traits Using SAS. SAS Institute, Inc.

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

Margarido GRA, Souza AP, Garcia AAF (2007) OneMap: software for genetic mapping in outcrossing species. Hereditas 144: 78-79

Voorrips, R.E., 2002. MapChart: Software for the graphical presentation of linkage maps and QTLs. The Journal of Heredity 93 (1): 77-78.

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` vilage - 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` trait 1 - 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
```

`parentoffs` *Parent offspring regression example data*

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

Allard R.W.(1999) Principles of Plant Breeding, John Wiley and Sons, May 10, 1999 - 254 pages

Sleper D.A.(2006) Poehlman J.M. Breeding Field Crops, Blackwell Pub., Jul 25, 2006 - 424 pages

Falconer D. S., Mackay T.F.C. (1996). Introduction to Quantitative Genetics. Fourth edition. Addison Wesley Longman, Harlow, Essex, UK.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Examples

```
data(parentoffs)

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

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


par(fig=c(0,0.8,0.55,1), new=TRUE)
boxplot(parentoffs$midparent, horizontal=TRUE, col = "red", axes=FALSE)
par(fig=c(0.65,1,0,0.8),new=TRUE)

boxplot(parentoffs$offspring, col = "blue", axes=FALSE)
mtext(paste("Parent ofspring regression \n 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 *Peanut data from multilocation trials*

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` yield - Y variable

`env` environment

`gen` genotypes

Source

Littell R.C.(2006) SAS for Mixed Models,SAS Institute, Inc.

References

Gauch H.G.(1992). Statistical analysis of regional yield trials:AMMI analysis of factorial designs. Elsevier, Amsterdam.

Gauch H.G. (2006). Statistical analysis of yield trials by AMMI and GGE. Crop Sci. 46:1488-1500.

Gauch, H.G., Zobel.R.W. (1996). AMMI analysis of yield trials. p.85-122. In M.S. Kang and H.G. Gauch, Jr. (ed.) Genotype-byenvironment interaction. CRC Press, Boca Raton, FL.

Eberhart S.A., Russell W.A. (1966) Stability parameters for comparing varieties. Crop Sci. 6: 36-40.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Kang M.S., Aggarwal V.D., Chirwa R.M.(2006) Adaptability and stability of bean cultivars as determined via yield-stability statistic and GGE biplot analysis. J. Crop Improv. 15:97-120

Examples

```
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
results_p <- ammi.full(dataframe = peanut, environment = "env", genotype = "geno",
 replication = "rep", yvar = "yield")

# heatmap plot
heatmap (results_p$means)

# plot bar plot

myd <- melt(results_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 (results_p$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)
```

---

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

Lynch M., Walsh B. (1998). Genetics and Analysis of Quantitative Traits. Sinauer, Sunderland, MA

Falconer D. S., Mackay T.F.C. (1996). Introduction to Quantitative Genetics. Fourth edition. Addison Wesley Longman, Harlow, Essex, UK.

Examples

```
# 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

```
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 suggessted 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", 1:20, 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 = 0.45, fill = "gray80", l.pos = 0.2)

plotblock(label = treatment, plotn = plot.number,  nrow = 5 , ncol = length (genotypes),
 g.col = 0.49, g.row = 0.49, fill = "gray80", 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

Allard R.W.(1999) Principles of Plant Breeding, John Wiley and Sons, May 10, 1999 - 254 pages

Sleper D.A.(2006) Poehlman J.M. Breeding Field Crops, Blackwell Pub., Jul 25, 2006 - 424 pages

Hill J., Becker H.C., Tigerstedt P.M. A. (1998) Quantitative and Ecological Aspects of Plant Breeding, Springer, 1998 - 275 pages

Lynch M., Walsh B. (1998). Genetics and Analysis of Quantitative Traits. Sinauer, Sunderland, MA

Falconer D. S., Mackay T.F.C. (1996). Introduction to Quantitative Genetics. Fourth edition. Addison Wesley Longman, Harlow, Essex, UK.

Mather K., Jinks J.L. (1971). Biometrical Genetics. Chapman & Hall, London.

Examples

```
set.seed (1234)
require(ggplot2)
mydf1 <-  data.frame (class = c(rep ("F2", 1000), rep("F3", 100)),
yield = c(rnorm (1000, 50, 20),rnorm (50, 65, 5),rnorm (50, 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 chromsome map with specified maker positions and labels. In addoton to map, aligned scatter plot will be produced for addition variable (such as LOD score, minor allele frequency). The scatter plot can have points or lines or area as user specified.

Usage

```
plotwith.map(mapdata, ydata, yvar, position, marker, type = "l", 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
...	More graphical parameters can be passed to the scatter plot, help(par)

Examples

```
# 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 = "gray20", 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",
```

```
position = "position", marker = "snpname", type = "o", pch = 19, ycol = "darkgreen",
mbar.col = "gray20", ylab = "Minor Alele Frequency", cex.lab = 3, chr.lab = 3)
```

`polar.genome` *Circular (polar) genome plot*

Description

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

Usage

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

Arguments

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

Author(s)

Umesh Rosyara

Examples

```
gr1 <- c(1, 5, 15, 20, 30, 40)
gr2 <- c(1, 15, 25, 30, 40)
gr3 <- c(1, 5, 10, 25, 40, 60, 80)

mapdataframe <- data.frame (group = c(rep(1, length(gr1)),
 rep(2, length(gr2)), rep(3, length(gr3))), position = c(gr1, gr2, gr3))

mapsubset <- data.frame (group = c(1,1,2,2, 3,3,3, 3),
position = c(25, 35, 5, 35, 8, 50, 65, 75))
```

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

---

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

Wickham H. (2009) ggplot: Elegant Graphics for Data Analysis. Use R. Springer.

Sarkar D. (2008) Lattice: Multivariate Data Visualization with R. Springer, New York

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 histpgram 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 demostrate in calculation of broad-sense hertitability 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",
 replication = "replication", exout= TRUE, REML = FALSE)

# genetic correlation

gencor.lm(dataframe = rcbsingle, yvar1 = "tgw", yvar2 = "grw", genovar = "genovar",
replication = "replication", exout = FALSE)

out <- gencor.lm(dataframe = rcbsingle, yvar1 = "tgw", yvar2 = "grw",
genovar = "genovar", replication = "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)
```

Format

A data frame with 4000 observations on the following 3 variables.

`iid` a numeric vector

`traitF2` a numeric vector

`traitF3` a numeric vector

Examples

```
data(respdataf)
F3s <- subset(respdataf, traitF2 >=65)  # selected lines from F2 that meet criteria
SDiff = mean(F3s$traitF2 ) - mean(respdataf$traitF2 )
traitF3v <- respdataf$traitF3
F3sv <- F3s$traitF3
F3rv <- sample(traitF3v, length(F3sv)) # random selection of equal number of progeny

# hertiability and genetic gain
hsqr = (mean(F3sv) - mean(F3rv))/ SDiff # heritability
hsqr
geneticgain = mean(F3sv) - mean(respdataf$traitF2) # genetic gain
geneticgain

# plotting the generation density and selected fractions
plot(density(respdataf$traitF2),xlim=c(0,100), main = paste("F2 and F3 distributions"))

dens <- density(respdataf$traitF2)
x1 <- min(which(dens$x >= 65))
x2 <- max(which(dens$x <  100))
with(dens, polygon(x=c(x[c(x1,x1:x2,x2)]), y= c(0, y[x1:x2], 0), col="green"))
abline(v= mean(respdataf$traitF2),  col = "black", lty = 1)
lines(density(respdataf$traitF3),lty=2, col= "blue")
abline(v= mean(respdataf$traitF3),  col = "blue", lty = 2)
lines(density(F3sv),lty=5, col = "green4")
abline(v= mean(F3sv),  col = "green4", lty = 5)

legend(73, 0.03, c("F2", "F3 not selected", "F3 from selected F2"),
col = c("black","blue","green4"), text.col = c("black","blue","green4"),
 lty = c(1, 2, 5), bg = 'gray90')
```

---

rowcoldata *Analysis of Row Column Experimental Design Data*

Description

The data us 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

```
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")
```

---

rqtl2mapchart *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

```
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

Broman K.W., Sen S. (2009) A Guide to QTL Mapping with R/qtl. SBH/Statistics for Biology and Health. Springer

Voorrips, R.E., 2002. MapChart: Software for the graphical presentation of linkage maps and QTLs. The Journal of Heredity 93 (1): 77-78. http://www.biometris.wur.nl/uk/Software/MapChart/

Examples

```
#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

```
data(rqtldata)
```

Format

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

Source

Simulated data

References

Broman K.W., Sen S. (2009) A guide to QTL mapping with R/qtl. Springer

Broman K.W., Wu H., Sen S., Churchill G.A. (2003) R/qtl-QTL mapping in experimental crosses. Bioinformatics 19, 889-890.

Lander E.S., Botstein D. (1989) Mapping Mendelian factors underlying quantitative traits using RFLP linkage maps. Genetics 121, 185-199.

Haley C.S., Knott S.A. (1992) A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. Heredity 69, 315-324.

Sen S., and Churchill G.A. (2001) A statistical framework for quantitative trait mapping. Genetics 159, 371-387.

Lincoln S.E., Lander E.S. (1992) Systematic detection of errors in genetic linkage data. Genomics 14, 604-610.

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

Examples

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

 # marker regression
  out.mr <- scanone(rqtldata, method="mr")
  out.mr
  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))

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

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

# Multiple imputation
 mydatacross <- sim.geno(rqtldata, step=1, n.draws=64, error.prob=0.001)
 out.imp <- scanone(rqtldata, 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(rqtldata, chr=2, n.boot=1000)
out.boot
plot(out.boot)
summary(out.boot)

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

# Two-dimensional, two-QTL scans
mycrossdata <- calc.genoprob(rqtldata, step=2.5, err=0.001)
out2 <- scantwo(rqtldata, verbose=FALSE, method = "em")
```

```
# multi-dimensional, multiple-QTL scans
mq.rqtldata <- mqmaugment(rqtldata, minprob=0.001) # data augumentation
mq.rqtldata.out <- mqmscan(mq.rqtldata)
summary(mq.rqtldata.out, lod =3)
plot(mq.rqtldata.out,out.cim.20, 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

```
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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Hill J., Becker H.C., Tigerstedt P.M. A. (1998) Quantitative and Ecological Aspects of Plant Breeding, Springer, 1998 - 275 pages

Lynch M., Walsh B. (1998). Genetics and Analysis of Quantitative Traits. Sinauer, Sunderland, MA

Smith H.F. (1936) A discriminant function for plant selection. Ann. Eugenicd, 7: 240-250.

Examples

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

selindex *Data for selection index*

Description

The data illustrates development of selection index outlined by Smith(1936) as outlined in Singh and Chaudhary (1985) using function selection.index.

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

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

Hill J., Becker H.C., Tigerstedt P.M. A. (1998) Quantitative and Ecological Aspects of Plant Breeding, Springer, 1998 - 275 pages

Lynch M., Walsh B. (1998). Genetics and Analysis of Quantitative Traits. Sinauer, Sunderland, MA

Examples

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

shaded.normal *Shading regions in theoritical normal curves or sample density curves for quantative traits*

Description

The function is useful for teaching and publication purpose.

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 theoritical 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 veritical lines added to plot
lat	Point of additional veritical 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 theoritical 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))
```

```
shaded.normal(shade = "percent", lowrp = 0.05, uprp = 0.05, avg = 50, sdev= 40)
shaded.normal(shade = "percent",lowrp = 0.2, uprp = 0.2, avg = 70, sdev= 20)
shaded.normal(shade = "percent",lowrp = 0.25, uprp = 0.25, avg = 80, sdev= 10)

# tait
trait <- rnorm(800, 50, 10)
shaded.normal (type = "TR", trait = trait, shade = "percent",  lowrp = 0.010,
 uprp = 0.1, Fcolfill = 2, Lcolfill = 4, lincolor = c("blue","red"), lat = c(45, 80))
data(respdataf)
 shaded.normal (type = "TR", trait = respdataf$traitF2, shade = "trunp",  lowrp = 40,
  uprp = 60, Fcolfill = "#CAFF70", Lcolfill = "#FF7F50")
```

`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

Eberhart S.A., Russell W.A. (1966) Stability parameters for comparing varieties. Crop Sci. 6: 36-40.

Singh R.K., Chaudhary B.D.(1985) Biometrical Methods in Quantitative Genetics Analysis, Kalyani Publishers

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, 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, 32.4,
```

```
            26.4, 27.7, 36.8, 21.5, 29.6, 31.5, 25.8, 17.3, 24.3, 24.3,
            22.6, 17.7, 35.5, 32.8, 25.8, 28.8, 28.0, 24.8, 26.7, 29.8,
            31.2, 20.2, 28.0, 21.3, 36.9, 41.2, 27.9, 20.6, 20.9, 20.8,
            25.4, 29.7, 26.3, 33.7, 29.8, 27.3, 25.9, 25.3, 30.2, 17.8,
            23.7, 23.9, 32.2, 34.7, 30.6, 28.3, 27.2, 23.9, 23.8, 15.0,
            24.3, 28.2, 20.3, 32.3, 18.5, 28.1, 22.0, 30.7, 32.4, 26.1,
            34.3, 30.2, 25.6, 28.1, 29.2, 40.1, 28.2, 27.7, 37.0, 32.4,
            36.5, 30.1, 35.1, 28.2, 34.5, 42.1, 38.7, 15.1, 25.4, 38.7 )


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(c("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

```
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

```
# Example 1:
 mydata1 <- data.frame (yvar1 = rnorm(2000, 15, 5), xv1 = rep(1:5, each = 400),
   xv2 = rep(1:10, 200), xv3 = rep(1:2, 1000), xv4 = rep(1:2, 1000))

table.creator (mydata = mydata1, yvar = NA, classvars = c("xv1", "xv2", "xv3"),
 ycut = FALSE)

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 = 1:4000, chromsome = as.factor (rep(1:10, each = 400)),
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)

 af1

 snpout <- table.creator (mydata = snpprop, yvar = "minorAF", classvars = c("chromsome","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 = "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

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

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,])
```

```
names(tvariability) <- c(paste ("GEN",1:10, sep='' ) )

# the following commands need pvclust package need to be installed first
require(pvclust)
plot(result <- pvclust(tvariability, method.dist="cor", method.hclust ="average", nboot=100))
pvrect(result, alpha=0.90)

# heat map plot with dendogram at margin, the following scripts need garphics and grDevices installed
tvar <- as.matrix(tvariability) ;
require(graphics);
require(grDevices)
rc <- rainbow(nrow(tvar), start=0, end=.3)
cc <- rainbow(ncol(tvar), start=0, end=.3)
hv <- heatmap(tvar, col = heat.colors(20), scale="column", RowSideColors = rc,
ColSideColors = cc, margins=c(5,10), xlab = "Genotypes", ylab= " Markers",
main = " Heat map plot of variability data")
```

`wintwheat` *Winter wheat data from mixed model data SAS*

Description

inter 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 10 2 3 4 5 6 7 8 9

`Yield` a numeric vector

`Moisture` a numeric vector

Source

Littell R.C.(2006) SAS for Mixed Models,SAS Institute, Inc.

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