Package 'lineup'

July 22, 2025

Version 0.44

Date 2024-07-15

Title Lining Up Two Sets of Measurements

Description Tools for detecting and correcting sample mix-ups between two sets of measurements, such as between gene expression data on two tissues. Broman et al. (2015) <doi:10.1534/g3.115.019778>.

Author Karl W Broman [aut, cre] (ORCID: <https://orcid.org/0000-0002-4914-6671>)

Maintainer Karl W Broman <broman@wisc.edu>

URL https://github.com/kbroman/lineup

BugReports https://github.com/kbroman/lineup/issues

Depends R (>= 2.10.1)

Imports qtl (>= 1.20-15), class, graphics, grDevices, utils, stats, parallel

Suggests roxygen2, knitr, rmarkdown, devtools, testthat

License GPL-3

VignetteBuilder knitr

LazyData true

ByteCompile true

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-07-15 19:50:02 UTC

Contents

calc.locallod . . . 2
combinedist . . . 4
corbetw2mat . . . 5
distee . . . 7
disteg . . . 8
expr-data . . . 11
f2cross . . . 12
find.gene.pseudomarker . . . 13
findCommonID . . . 14
fscale . . . 15
genepos . . . 16
lineupversion . . . 16
omitdiag . . . 17
plot.lineupdist . . . 18
plot2dist . . . 19
plotEGclass . . . 21
pmap . . . 22
pulldiag . . . 23
subset.lineupdist . . . 24
summary.lineupdist . . . 25

Index **27**

`calc.locallod` *Calculate LOD score at physical position of each gene*

Description

For gene expression data with physical positions of the genes, calculate the LOD score at those positions to assess evidence for local eQTL.

Usage

```
calc.locallod(
  cross,
  pheno,
  pmark,
  addcovar = NULL,
  intcovar = NULL,
  verbose = TRUE,
  n.cores = 1
)
```

Arguments

cross	An object of class "cross" containing data for a QTL experiment. See the help file for qtl::read.cross() in the R/qtl package (https://rqtl.org). There must be a phenotype named "id" or "ID" that contains the individual identifiers.
pheno	A data frame of phenotypes (generally gene expression data), stored as individuals x phenotypes. The row names must contain individual identifiers.
pmark	Pseudomarkers that are closest to the genes in pheno, as output by find.gene.pseudomarker().
addcovar	Additive covariates passed to qtl::scanone().
intcovar	Interactive covariates passed to qtl::scanone().
verbose	If TRUE, print tracing information.
n.cores	Number of CPU cores to use in the calculations. With n.cores=0, parallel::detectCores() is used to detect the number of available cores.

Details

cross and pheno must contain exactly the same individuals in the same order. (Use findCommonID() to line them up.)

We consider the expression phenotypes in batches: those whose closest pseudomarker is the same.

We use Haley-Knott regression to calculate the LOD scores.

Actually, we use a bit of a contortion of the data to force the qtl::scanone() function in R/qtl to calculate the LOD score at a single position.

We omit any transcripts that map to the X chromosome; we can only handle autosomal loci for now.

Value

A vector of LOD scores. The names indicate the gene names (columns in pheno).

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

find.gene.pseudomarker(), plotEGclass(), findCommonID(), disteg()

Examples

```
data(f2cross, expr1, genepos, pmap)
library(qtl)

# calc QTL genotype probabilities
f2cross <- calc.genoprob(f2cross, step=1)

# find nearest pseudomarkers
pmark <- find.gene.pseudomarker(f2cross, pmap, genepos, "prob")

# line up f2cross and expr1
```

```
id <- findCommonID(f2cross, expr1)

# calculate LOD score for local eQTL
locallod <- calc.locallod(f2cross[,id$first], expr1[id$second,], pmark)
```

`combinedist` *Combine distance matrices into a single such*

Description

Combine multiple distance matrices into a single distance matrix providing an overall summary

Usage

```
combinedist(..., method = c("median", "mean"))
```

Arguments

`...`	Set of distance matrices, as calculated by `distee()` or `disteg()`.
`method`	Indicates whether to summarize using the median or the mean.

Details

The row and column names of the input distance matrices define the individual IDs.

If the input distance matrices all have an attribute `"denom"` (for denominator) and `method="mean"`, we use a weighted mean, weighted by the denominators. This could be used to calculate an overall proportion.

Value

A distance matrix, with class `"lineupdist"`. The individual IDs are in the row and column names.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`distee()`, `disteg()`, `summary.lineupdist()`

Examples

```
library(qtl)

# load example data
data(f2cross, expr1, expr2, pmap, genepos)


# calculate QTL genotype probabilities
f2cross <- calc.genoprob(f2cross, step=1)

# find nearest pseudomarkers
pmark <- find.gene.pseudomarker(f2cross, pmap, genepos)

# line up individuals
id1 <- findCommonID(f2cross, expr1)
id2 <- findCommonID(f2cross, expr2)

# calculate LOD score for local eQTL
locallod1 <- calc.locallod(f2cross[,id1$first], expr1[id1$second,], pmark)
locallod2 <- calc.locallod(f2cross[,id2$first], expr2[id2$second,], pmark)

# take those with LOD > 25
expr1s <- expr1[,locallod1>25,drop=FALSE]
expr2s <- expr2[,locallod2>25,drop=FALSE]

# calculate distance between individuals
#     (prop'n mismatches between obs and inferred eQTL geno)
d1 <- disteg(f2cross, expr1s, pmark)
d2 <- disteg(f2cross, expr2s, pmark)

# combine distances
d <- combinedist(d1, d2)

# summary of problem samples
summary(d)
```

`corbetw2mat` *Calculate correlations between columns of two matrices*

Description

For matrices x and y, calculate the correlation between columns of x and columns of y.

Usage

```
corbetw2mat(
  x,
  y,
```

```
  what = c("paired", "bestright", "bestpairs", "all"),
  corthresh = 0.9
)
```

Arguments

x	A numeric matrix.
y	A numeric matrix with the same number of rows as x.
what	Indicates which correlations to calculate and return. See value, below.
corthresh	Threshold on correlations if what="bestpairs".

Details

Missing values (NA) are ignored, and we calculate the correlation using all complete pairs, as in stats::cor() with use="pairwise.complete.obs".

Value

If what="paired", the return value is a vector of correlations, between columns of x and the corresponding column of y. x and y must have the same number of columns.

If what="bestright", we return a data frame of size ncol(x) by 3, with the ith row being the maximum correlation between column i of x and a column of y, and then the y-column index and y-column name with that correlation. (In case of ties, we give the first one.)

If what="bestpairs", we return a data frame with five columns, containing all pairs of columns (with one in x and one in y) with correlation $\geq$ corthresh. Each row corresponds to a column pair, and contains the correlation and then the x- and y-column indices followed by the x- and y-column names.

If what="all", the output is a matrix of size ncol(x) by ncol(y), with all correlations between columns of x and columns of y.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

distee(), findCommonID()

Examples

```
data(expr1, expr2)


# correlations with paired columns
r <- corbetw2mat(expr1, expr2)
# top 10, by absolute value
r[order(abs(r), decreasing=TRUE)[1:10]]

# all pairs of columns with correlation >= 0.8
```

```
r_allpairs <- corbetw2mat(expr1, expr2, what="bestpairs", corthresh=0.6)

# for each column in left matrix, most-correlated column in right matrix
r_bestright <- corbetw2mat(expr1, expr2, what="bestright")
```

`distee` *Calculate distance between two gene expression data sets*

Description

Calculate a distance between all pairs of individuals for two gene expression data sets

Usage

```
distee(
  e1,
  e2 = NULL,
  d.method = c("rmsd", "cor"),
  labels = c("e1", "e2"),
  verbose = TRUE
)
```

Arguments

`e1`	Numeric matrix of gene expression data, as individuals x genes. The row and column names must contain individual and gene identifiers.
`e2`	(Optional) Like `e1`. An appreciable number of individuals and genes must be in common.
`d.method`	Calculate inter-individual distance as RMS difference or as correlation.
`labels`	Two character strings, to use as labels for the two data matrices in subsequent output.
`verbose`	if TRUE, give verbose output.

Details

We calculate the pairwise distance between all individuals (rows) in `e1` and all individuals in `e2`. This distance is either the RMS difference (`d.method="rmsd"`) or the correlation (`d.method="cor"`).

Value

A matrix with `nrow(e1)` rows and `nrow(e2)` columns, containing the distances. The individual IDs are in the row and column names. The matrix is assigned class `"lineupdist"`.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

pulldiag(), omitdiag(), summary.lineupdist(), plot2dist(), disteg(), corbetw2mat()

Examples

```
# load the example data
data(expr1, expr2)


# find samples in common
id <- findCommonID(expr1, expr2)

# calculate correlations between cols of x and cols of y
thecor <- corbetw2mat(expr1[id$first,], expr2[id$second,])

# subset at genes with corr > 0.8 and scale values
expr1s <- expr1[,thecor > 0.8]/1000
expr2s <- expr2[,thecor > 0.8]/1000

# calculate distance (using "RMS difference" as a measure)
d1 <- distee(expr1s, expr2s, d.method="rmsd", labels=c("1","2"))

# calculate distance (using "correlation" as a measure...really similarity)
d2 <- distee(expr1s, expr2s, d.method="cor", labels=c("1", "2"))

# pull out the smallest 8 self-self correlations
sort(pulldiag(d2))[1:8]

# summary of results
summary(d1)
summary(d2)

# plot histograms of RMS distances
plot(d1)

# plot histograms of correlations
plot(d2)

# plot distances against one another
plot2dist(d1, d2)
```

---

disteg *Calculate distance between two gene expression data sets*

Description

Calculate a distance between all pairs of individuals for two gene expression data sets

Usage

```
disteg(
  cross,
  pheno,
  pmark,
  min.genoprob = 0.99,
  k = 20,
  min.classprob = 0.8,
  classprob2drop = 1,
  repeatKNN = TRUE,
  max.selfd = 0.3,
  phenolabel = "phenotype",
  weightByLinkage = FALSE,
  map.function = c("haldane", "kosambi", "c-f", "morgan"),
  verbose = TRUE
)
```

Arguments

cross	An object of class "cross" containing data for a QTL experiment. See the help file for qtl::read.cross() in the R/qtl package (https://rqtl.org). There must be a phenotype named "id" or "ID" that contains the individual identifiers.
pheno	A data frame of phenotypes (generally gene expression data), stored as individuals x phenotypes. The row names must contain individual identifiers.
pmark	Pseudomarkers that are closest to the genes in pheno, as output by find.gene.pseudomarker().
min.genoprob	Threshold on genotype probabilities; if maximum probability is less than this, observed genotype taken as NA.
k	Number of nearest neighbors to consider in forming a k-nearest neighbor classifier.
min.classprob	Minimum proportion of neighbors with a common class to make a class prediction.
classprob2drop	If an individual is inferred to have a genotype mismatch with classprob > this value, treat as an outlier and drop from the analysis and then repeat the KNN construction without it.
repeatKNN	If TRUE, repeat k-nearest neighbor a second time, after omitting individuals who seem to not be self-self matches
max.selfd	Min distance from self (as proportion of mismatches between observed and predicted eQTL genotypes) to be excluded from the second round of k-nearest neighbor.
phenolabel	Label for expression phenotypes to place in the output distance matrix.
weightByLinkage	If TRUE, weight the eQTL to account for their relative positions (for example, two tightly linked eQTL would each count about 1/2 of an isolated eQTL)
map.function	Used if weightByLinkage is TRUE
verbose	if TRUE, give verbose output.

Details

We consider the expression phenotypes in batches, by which pseudomarker they are closest to. For each batch, we pull the genotype probabilities at the corresponding pseudomarker and use the individuals that are in common between `cross` and `pheno` and whose maximum genotype probability is above `min.genoprob`, to form a classifier of eQTL genotype from expression values, using k-nearest neighbor (the function `class::knn()`). The classifier is applied to all individuals with expression data, to give a predicted eQTL genotype. (If the proportion of the k nearest neighbors with a common class is less than `min.classprob`, the predicted eQTL genotype is left as `NA`.)

If `repeatKNN` is TRUE, we repeat the construction of the k-nearest neighbor classifier after first omitting individuals whose proportion of mismatches between observed and inferred eQTL genotypes is greater than `max.selfd`.

Finally, we calculate the distance between the observed eQTL genotypes for each individual in `cross` and the inferred eQTL genotypes for each individual in `pheno`, as the proportion of mismatches between the observed and inferred eQTL genotypes.

If `weightByLinkage` is `TRUE`, we use weights on the mismatch proportions for the various eQTL, taking into account their linkage. Two tightly linked eQTL will each be given half the weight of a single isolated eQTL.

Value

A matrix with `nind(cross)` rows and `nrow(pheno)` columns, containing the distances. The individual IDs are in the row and column names. The matrix is assigned class `"lineupdist"`.

The names of the genes that were used to construct the classifier are saved in an attribute `"retained"`.

The observed and inferred eQTL genotypes are saved as attributes `"obsg"` and `"infg"`.

The denominators of the proportions that form the inter-individual distances are in the attribute `"denom"`.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`distee()`, `summary.lineupdist()`, `pulldiag()`, `omitdiag()`, `findCommonID()`, `find.gene.pseudomarker()`, `calc.locallod()`, `plot.lineupdist()`, `class::knn()`, `plotEGclass()`

Examples

```
library(qtl)

# load example data
data(f2cross, expr1, pmap, genepos)


# calculate QTL genotype probabilities
f2cross <- calc.genoprob(f2cross, step=1)

# find nearest pseudomarkers
```

```
pmark <- find.gene.pseudomarker(f2cross, pmap, genepos)

# line up individuals
id <- findCommonID(f2cross, expr1)

# calculate LOD score for local eQTL
locallod <- calc.locallod(f2cross[,id$first], expr1[id$second,], pmark)

# take those with LOD > 25
expr1s <- expr1[,locallod>25,drop=FALSE]

# calculate distance between individuals
#     (prop'n mismatches between obs and inferred eQTL geno)
d <- disteg(f2cross, expr1s, pmark)

# plot distances
plot(d)

# summary of apparent mix-ups
summary(d)

# plot of classifier for and second eQTL
par(mfrow=c(2,1), las=1)
plotEGclass(d)
plotEGclass(d, 2)
```

`expr-data` *Example gene expression data*

Description

Matrices of simulated gene expression data, each for 98 individuals at 5,000 genes. Think of `expr1` and `expr2` as expression data on two different tissues.

Usage

```
data(expr1)
data(expr2)
```

Format

A matrix of integers, individuals as rows and genes as columns.

See Also

`genepos()`, `f2cross()`, `pmap()`

Examples

```
data(expr1)
data(expr2)

# identify the common individuals
id <- findCommonID(rownames(expr1), rownames(expr2))

# correlation between tissues for each gene
rho <- corbetw2mat(expr1[id$first,], expr2[id$second,])
hist(rho, breaks=100)
```

`f2cross` *Example experimental cross data*

Description

Simulated experimental cross data with some sample mix-ups. The only phenotype is an individual ID. There are 100 individuals genotyped at 1000 markers on 19 autosomes.

Usage

```
data(f2cross)
```

Format

An object of class `"cross"`. See `qtl::read.cross()` in the R/qtl package for details.

See Also

`expr1()`, `expr2()`, `genepos()`, `pmap()`

Examples

```
library(qtl)
data(f2cross)
summary(f2cross)
```

`find.gene.pseudomarker` *Find nearest peudomarker to each gene*

Description

Pull out the pseudomarker that is closest to the position of each of a series of genes.

Usage

```
find.gene.pseudomarker(cross, pmap, geneloc, where = c("prob", "draws"))
```

Arguments

`cross`	An object of class `"cross"` containing data for a QTL experiment. See the help file for `qtl::read.cross()` in the R/qtl package (https://rqtl.org).
`pmap`	A physical map of the markers in `cross`, with locations in Mbp. This is a list whose components are the marker locations on each chromosome.
`geneloc`	A data frame specifying the physical locations of the genes. There should be two columns, `chr` for chromosome and `pos` for position in Mbp. The rownames should indicate the gene names.
`where`	Indicates whether to pull pseudomarkers from the genotype probabilities (produced by `qtl::calc.genoprob()`) or from the imputed genotypes (produced by `qtl::sim.geno()`).

Details

We first convert positions (by interpolation) from those contained within `cross` to physical coordinates contained in `pmap`. We then use `qtl::find.pseudomarker()` to identify the closest pseudomarker to each gene location.

We also include the positions of the pseudomarkers, and we print a warning message if pseudomarkers are > 2 Mbp from the respective gene.

Value

A data frame with columns `chr` (the chromosome) and `pmark` (the name of the pseudomarker). The third column `pos` contains the Mbp position of the pseudomarker. The final column is the signed distance between the gene and the pseudomarker. The rownames indicate the gene names.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`qtl::find.pseudomarker()`, `qtl::find.pseudomarkerpos()`, `plotEGclass()`, `disteg()`, `calc.locallod()`

Examples

```
data(f2cross, expr1, genepos, pmap)
library(qtl)

# calc QTL genotype probabilities
f2cross <- calc.genoprob(f2cross, step=1)

# find nearest pseudomarkers
pmark <- find.gene.pseudomarker(f2cross, pmap, genepos, "prob")
```

`findCommonID` *Find individuals in common between a cross and a phenotype matrix*

Description

Identify which individuals are in common between a QTL mapping data set and a matrix of phenotypes, series of genes.

Usage

```
findCommonID(id1, id2)
```

Arguments

`id1`	A character vector of individual IDs. This can also be a QTL cross object (see `qtl::read.cross()`), in which case `qtl::getid()` is used to grab individual IDs, or a matrix or data frame, in which case the rownames are taken to be IDs.
`id2`	Like `id1`, can be a character vector, a cross or a matrix/data frame.

Value

A list with three components:

First, a data frame with rows corresponding to all individuals (across the two sets of individual IDs) and three columns: `indexInFirst` and `indexInSecond` contain numeric indices to the locations of the individuals within `cross` and `pheno`, and `inBoth` is a logical vector to indicate which individuals appear in both crosses. The row names are the individual identifiers.

The second and third components are vectors of indices in `id1` and `id2`, respectively, indicating the paired locations of the individuals that are in common.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`calc.locallod()`, `corbetw2mat()`

Examples

```
data(f2cross, expr1)

# align IDs
id <- findCommonID(f2cross, expr1)

# aligned data
f2cross_aligned <- f2cross[,id$first]
expr1_aligned <- expr1[id$second,]
```

`fscale` *Standardize the columns of a matrix*

Description

Standardize each column in a matrix, so that the columns have mean 0 and SD 1.

Usage

```
fscale(x)
```

Arguments

`x`	A numeric matrix.

Details

Missing values (`NA`) are ignored and left as is.

If there is just 1 non-missing value in a column, it is left as is.

This function uses a one-pass algorithm to calculate the mean and SD, which is fast but can show a bit of round-off error.

Value

A matrix of the same form as the input, but with columns transformed to have mean 0 and SD 1.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`base::scale()`

Examples

```
x <- matrix(1:10, ncol=2)
y <- fscale(x)
```

---

genepos *Genomic positions of genes in simulated expression data*

Description

A table with the genomic positions of genes in the simulated expression data, `expr1()` and `expr2()`.

Usage

```
data(genepos)
```

Format

A data frame with two columns, chromosome and physical position (in Mbp).

See Also

`expr1()`, `expr2()`, `f2cross()`, `pmap()`

Examples

```
data(genepos)

# interplot genetic positions
library(qtl)
data(pmap)
data(f2cross)
genepos_interp <- interpPositions(genepos, pmap, pull.map(f2cross))
genepos[1:5,] # 'newpos' column is the interpolated cM position
```

---

lineupversion *Installed version of R/lineup*

Description

Print the version number of the currently installed version of R/lineup.

Usage

```
lineupversion()
```

Value

A character string with the version number of the currently installed version of R/lineup.

Author(s)

Karl W Broman, <broman@wisc.edu>

Examples

```
lineupversion()
```

`omitdiag` *Replace the diagonal in a distance matrix with missing values*

Description

Replace the diagonal (that is, self-self distances) from a distance matrix calculated by `distee()` or `disteg()` with missing values (so that only self-nonself distances are left).

Usage

```
omitdiag(d)
```

Arguments

`d`	A distance matrix calculated by `distee()` or `disteg()`.

Details

We use the row and column names to identify which entries are self-self.

Value

A matrix of the same form as the input, but with self-self distances replaced with `NA`.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

`pulldiag()`, `distee()`, `disteg()`, `summary.lineupdist()`, `plot2dist()`, `plot.lineupdist()`

Examples

```
data(expr1, expr2)



# distance as RMS difference
d <- distee(expr1, expr2)

# focus on the self-nonself distances
# (replace self-self distances with NA)
d_selfnonself <- omitdiag(d)
```

`plot.lineupdist` *Plot summary of inter-individual distances*

Description

Plot histograms of self-self and self-nonself distances from a distance matrix calculated by `distee()` or `disteg()`.

Usage

```
## S3 method for class 'lineupdist'
plot(x, breaks = NULL, add.rug = TRUE, what = c("both", "ss", "sn"), ...)
```

Arguments

`x`	Output of `distee()` or `disteg()`.
`breaks`	Optional vector of breaks, passed to `graphics::hist()`, though if it is length 1, we interpret it as the number of breaks and ensure that both histograms use the same set of breaks.
`add.rug`	If true, also include `graphics::rug()` below histograms.
`what`	Indicates whether to plot both self-self and self-nonself distances (or correlations) or just one or the other. (`"ss"` indicates self-self and `"sn"` indicates self-nonself.)
`...`	Ignored at this point.

Details

We call `pulldiag()` and `omitdiag()` to get the self-self and self-nonself distances.

If all of the self-self distances are missing, we plot just the self-nonself distances.

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

pulldiag(), distee(), plot2dist()

Examples

```
data(expr1, expr2)



# distance as correlation
d <- distee(expr1, expr2, "cor")

# plot histograms of self-self and self-nonself correlations
plot(d)
```

---

plot2dist *Plot two sets of inter-individual distances against one another*

Description

Plot two sets of inter-individual distances against one another, colored by self and non-self distances.

Usage

```
plot2dist(
  d1,
  d2,
  hirow = NULL,
  hicol = NULL,
  xlab = NULL,
  ylab = NULL,
  smoothScatter = FALSE,
  colself = "black",
  colnonself = "gray",
  colhirow = "green",
  colhicol = "orange",
  ...
)
```

Arguments

d1	Output of distee().
d2	Output of distee().
hirow	Names of rows to highlight in green.
hicol	Names of columns to highlight in orange.
xlab	X-axis label (optional)
ylab	Y-axis label (optional)
smoothScatter	If TRUE, plot non-self distances with graphics::smoothScatter(); if FALSE, use base::plot().
colself	Color to use for the self-self points. If NULL, these aren't plotted.
colnonself	Color to use for the non-self points. If NULL, these aren't plotted.
colhirow	Color to use for the hirow points. If NULL, these aren't plotted.
colhicol	Color to use for the hicol points. If NULL, these aren't plotted.
...	Passed to base::plot() and graphics::points().

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

pulldiag(), distee(), summary.lineupdist()

Examples

```
data(expr1, expr2)



# distances as RMS difference and correlation
d_rmsd <- distee(expr1, expr2, "rmsd")
d_cor <- distee(expr1, expr2, "cor")

# plot distances against one another
plot2dist(d_rmsd, d_cor)
```

plotEGclass *Plot classifier of eQTL genotype from expression data*

Description

Diagnostic plot of one of the eQTL classifiers from the results of `disteg()`: generally expression phenotype against observed eQTL genotype, colored by inferred eQTL genotype.

Usage

```
plotEGclass(
  d,
  eqtl = 1,
  outercol = "inferred",
  innercol = "observed",
  thecolors = c("#7B68ED", "#1B9E78", "#CA3767", "#E59E00"),
  ...
)
```

Arguments

`d`	Output of `disteg()`.
`eqtl`	Numeric index or a character vector (of the form "1@102.35") indicating the eQTL to consider.
`outercol`	Indicates how to color the outer edge of the points: `"observed"` indicates to color based on observed genotypes; `"inferred"` indicates to color based on inferred genotypes; otherwise, give a color.
`innercol`	Like `outercol`, but indicating the interior of the points.
`thecolors`	The colors to use in the plot. The last element (after the number of genotypes) indicates the color to use for missing values.
`...`	Passed to `base::plot()` and `graphics::points()`.

Details

The function produces a diagnostic plot for studying one of the k-nearest neighbor classifiers underlying the output from `disteg()`.

In the case of one expression phenotype attached to the selected eQTL, the plot is a dot plot of gene expression against observed eQTL genotype.

In the case of two expression phenotypes, the plot is a scatterplot of the two expression phenotypes against each other.

In the case of more than two expression phenotypes, we use `graphics::pairs()` to produce a matrix of scatterplots.

Value

None.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

disteg(), plot.lineupdist(), plot2dist(), class::knn()

Examples

```
library(qtl)

# load example data
data(f2cross, expr1, pmap, genepos)


# calculate QTL genotype probabilities
f2cross <- calc.genoprob(f2cross, step=1)

# find nearest pseudomarkers
pmark <- find.gene.pseudomarker(f2cross, pmap, genepos)

# line up individuals
id <- findCommonID(f2cross, expr1)

# calculate LOD score for local eQTL
locallod <- calc.locallod(f2cross[,id$first], expr1[id$second,], pmark)

# take those with LOD > 25
expr1s <- expr1[,locallod>25,drop=FALSE]

# calculate distance between individuals
#     (prop'n mismatches between obs and inferred eQTL geno)
d <- disteg(f2cross, expr1s, pmark)

# plot of classifier for and second eQTL
par(mfrow=c(2,1), las=1)
plotEGclass(d)
plotEGclass(d, 2)
```

---

pmap *Physical map of markers*

Description

Physical map (Mbp positions) of the markers in f2cross()

Usage

```
data(pmap)
```

Format

A list of vectors, each containing the locations of markers in Mbp. (Technically, an object of class "map".)

See Also

expr1(), expr2(), f2cross(), genepos()

Examples

```
data(pmap)
summary(pmap)
plot(pmap)
```

---

pulldiag *Pull out the diagonal from a distance matrix*

Description

Pull out the diagonal from a distance matrix calculated by distee() (that is, self-self distances).

Usage

```
pulldiag(d)
```

Arguments

d	A distance matrix calculated by distee().

Details

We use the row and column names to identify which entries are self-self.

Value

A vector with the self-self distances.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

omitdiag(), distee(), disteg(), summary.lineupdist(), plot2dist(), plot.lineupdist()

Examples

```
data(expr1, expr2)



# distance as RMS difference
d <- distee(expr1, expr2)

# pull out the self-self distances
d_selfself <- pulldiag(d)

# samples with smallest self-self correlation
sort(d_selfself)[1:10]
```

`subset.lineupdist` *Subsetting distance matrix*

Description

Pull out a specified set of rows and columns from a distance matrix calculated by `distee()` or `disteg()`.

Usage

```
## S3 method for class 'lineupdist'
subset(x, rows = NULL, cols = NULL, ...)

## S3 method for class 'lineupdist'
x[rows = NULL, cols = NULL]
```

Arguments

`x`	A distance matrix object as obtained from `distee()` or `disteg()`.
`rows`	Optional vector of selected rows.
`cols`	Optional vector of selected columns.
`...`	Ignored at this point.

Value

The input distance matrix object, but with only the specified subset of the data.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

disteg(), distee(), pulldiag()

Examples

```
data(expr1, expr2)


# find samples in common
id <- findCommonID(expr1, expr2)

# calculate correlations between cols of x and cols of y
thecor <- corbetw2mat(expr1[id$first,], expr2[id$second,])

expr1s <- expr1[,thecor > 0.8]/1000
expr2s <- expr2[,thecor > 0.8]/1000

# calculate correlations among samples
d <- distee(expr1s, expr2s, d.method="cor")

# pull out distances for samples 24, 92, 44, 66
samp <- c("24", "92", "44", "66")
d[samp, samp]
```

---

summary.lineupdist *Summarize inter-individual distances*

Description

Summarize the results of distee() or disteg(), with inter-individual distances between two sets of gene expression data.

Usage

```
## S3 method for class 'lineupdist'
summary(
  object,
  cutoff = NULL,
  dropmatches = TRUE,
  reorder = c("alignmatches", "bydistance", "no"),
  ...
)
```

Arguments

object	Output of distee() or disteg().

cutoff	(Optional) Cutoff on correlation/distance, with rows in the results only being kept if the best distance/correlation is above this cutoff or the self-self result is not missing and is above this cutoff.
dropmatches	If TRUE, omit rows for which an individual's best match is itself.
reorder	If "bydistance", reorder rows by increasing distance (or decreasing correlation) to the best match and then by decreasing distance (or decreasing correlation) to self; if "alignmatches", group related errors together; if "no", leave as is.
...	Passed to base::print.data.frame().

Value

A list with two components: the distances summarized by row and the distances summarized by column.

For each individual, we calculate the minimum distance to others, next-smallest distance, the self-self distance, the mean and SD of the distances to others, and finally indicate the individual (or individuals) that is closest.

Author(s)

Karl W Broman, <broman@wisc.edu>

See Also

pulldiag(), omitdiag(), distee(), disteg(), plot2dist(), plot.lineupdist()

Examples

```
data(expr1, expr2)



# distance as correlation
d <- distee(expr1, expr2, "cor")

# summary of potential problems
summary(d)
```

Index

∗ **array**
corbetw2mat, 5
fscale, 15
omitdiag, 17
pulldiag, 23
∗ **datasets**
expr-data, 11
f2cross, 12
genepos, 16
pmap, 22
∗ **graphics**
plot2dist, 19
plotEGclass, 21
∗ **manip**
subset.lineupdist, 24
∗ **multivariate**
corbetw2mat, 5
∗ **print**
lineupversion, 16
∗ **univar**
corbetw2mat, 5
∗ **utilities**
calc.locallod, 2
combinedist, 4
distee, 7
disteg, 8
find.gene.pseudomarker, 13
findCommonID, 14
plot.lineupdist, 18
summary.lineupdist, 25
[.lineupdist *(*subset.lineupdist*)*, 24

base::print.data.frame(), *26*
base::scale(), *15*

calc.locallod, 2
calc.locallod(), *10*, *13*, *14*
class::knn(), *10*, *22*
combinedist, 4
corbetw2mat, 5
corbetw2mat(), *8*, *14*

distee, 7
distee(), *4*, *6*, *10*, *17–20*, *23–26*
disteg, 8
disteg(), *3*, *4*, *8*, *13*, *17*, *18*, *21–26*

expr-data, 11
expr1 *(*expr-data*)*, 11
expr1(), *12*, *16*, *23*
expr2 *(*expr-data*)*, 11
expr2(), *12*, *16*, *23*

f2cross, 12
f2cross(), *11*, *16*, *22*, *23*
find.gene.pseudomarker, 13
find.gene.pseudomarker(), *3*, *9*, *10*
findCommonID, 14
findCommonID(), *3*, *6*, *10*
fscale, 15

genepos, 16
genepos(), *11*, *12*, *23*
graphics::hist(), *18*
graphics::pairs(), *21*
graphics::points(), *20*, *21*
graphics::rug(), *18*
graphics::smoothScatter(), *20*

lineupversion, 16

omitdiag, 17
omitdiag(), *8*, *10*, *18*, *23*, *26*

parallel::detectCores(), *3*
plot.lineupdist, 18
plot.lineupdist(), *10*, *17*, *22*, *23*, *26*
plot2dist, 19
plot2dist(), *8*, *17*, *19*, *22*, *23*, *26*
plotEGclass, 21
plotEGclass(), *3*, *10*, *13*

pmap, 22
pmap(), *11*, *12*, *16*
pulldiag, 23
pulldiag(), *8*, *10*, *17–20*, *25*, *26*

qtl::calc.genoprob(), *13*
qtl::find.pseudomarker(), *13*
qtl::find.pseudomarkerpos(), *13*
qtl::getid(), *14*
qtl::read.cross(), *3*, *9*, *12–14*
qtl::scanone(), *3*
qtl::sim.geno(), *13*

stats::cor(), *6*
subset.lineupdist, 24
summary.lineupdist, 25
summary.lineupdist(), *4*, *8*, *10*, *17*, *20*, *23*