# Package 'doppelgangR'

April 16, 2025

**Title** Identify likely duplicate samples from genomic or meta-data

**Version** 1.36.0

**Description** The main function is doppelgangR(), which takes as minimal input a list of ExpressionSet object, and searches all list pairs for duplicated samples. The search is based on the genomic data (exprs(eset)), phenotype/clinical data (pData(eset)), and ``smoking guns'' - supposedly unique identifiers found in pData(eset).

**Depends** R (>= 3.5.0), Biobase, BiocParallel

**Imports** sva, impute, digest, mnormt, methods, grDevices, graphics, stats, SummarizedExperiment, utils

**Suggests** BiocStyle, knitr, rmarkdown, curatedOvarianData, testthat

**biocViews** ImmunoOncology, RNASeq, Microarray, GeneExpression, QualityControl

**License** GPL (>=2.0)

**Encoding** UTF-8

**URL** https://github.com/lwaldron/doppelgangR

**BugReports** https://github.com/lwaldron/doppelgangR/issues

**VignetteBuilder** knitr

**RoxygenNote** 7.1.0

**git_url** https://git.bioconductor.org/packages/doppelgangR

**git_branch** RELEASE_3_21

**git_last_commit** 092646d

**git_last_commit_date** 2025-04-15

**Repository** Bioconductor 3.21

**Date/Publication** 2025-04-16

**Author** Levi Waldron [aut, cre],
Markus Reister [aut, ctb],
Marcel Ramos [ctb]

**Maintainer** Levi Waldron <lwaldron.research@gmail.com>

# Contents



---

`doppelgangR-package` *Identify likely duplicate samples from genomic or meta-data*

---

## Description

The main function is doppelgangR(), which takes as minimal input a list of ExpressionSet object, and searches all list pairs for duplicated samples. The search is based on the genomic data (exprs(eset)), phenotype/clinical data (pData(eset)), and "smoking guns" - supposedly unique identifiers found in pData(eset).

## Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

## See Also

Useful links:

- https://github.com/lwaldron/doppelgangR
- Report bugs at https://github.com/lwaldron/doppelgangR/issues

`colFinder` *Calculate pairwise similarities of colData between samples for a list containing two DataFrame*

## Description

This function acts as a wrapper to colData to handle cases of one DataFrame, a list of two identical DataFrame, or a list of two different DataFrame

## Usage

```
colFinder(summex.list, ...)
```

## Arguments

| | |
|---|---|
| `summex.list` | input: a list of DataFrame with two elements, or a DataFrame. If the two elements are identical, return the correlation matrix for pairs of samples in the first element. If not identical, return pairs between the two elements. |
| `...` | Extra arguments passed on to colFinder |

## Value

A matrix of similarities between the colData of pairs of samples.

## Author(s)

Fabio Da Col, Marcel Ramos

`corFinder` *Calculate pair-wise correlations between samples using the expr() slots of a list of two ExpressionSets.*

## Description

This function acts as a wrapper around ComBat (sva package) and cor(), to calculate pairwise correlations within one or between two ExpressionSets.

## Usage

```
corFinder(eset.pair, separator = ":", use.ComBat = TRUE, ...)
```

### Arguments

| | |
|---|---|
| eset.pair | a list of ExpressionSets, with two elements. If the two elements are identical, return the correlation matrix for pairs of samples in the first element. If not identical, return pairs between the two elements. |
| separator | Separator between dataset name and sample name. Dataset names are added to sample names to keep track of dataset of origin. |
| use.ComBat | Use the sva::ComBat function for batch correction of the expr() data between the two datasets. |
| ... | Extra arguments passed to the cor() function. |

### Value

Returns a matrix of sample-wise Pearson Correlations.

### Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

### Examples

```
example("phenoFinder")

corFinder(esets2)
```

---

## DoppelGang-class *DoppelGang S4 class*

---

### Description

S4 class containing results of doppelgangR() function.

### Usage

```
## S4 method for signature 'DoppelGang'
summary(object)

## S4 method for signature 'DoppelGang'
show(object)

## S4 method for signature 'DoppelGang'
print(x)
```

### Arguments

| | |
|---|---|
| x, object | A DoppelGang class object |

### Objects from the Class

Objects can be created by calls of the form `new(DoppelGang ...)`

### Author(s)

Levi Waldron and Markus Riester

### See Also

`plot,DoppelGang-method`

---

`doppelgangR` *doppelgangR*

---

### Description

Identify samples with suspiciously high correlations and phenotype similarities

### Usage

```
doppelgangR(
  esets,
  separator = ":",
 corFinder.args = list(separator = separator, use.ComBat = TRUE, method = "pearson"),
 phenoFinder.args = list(separator = separator, vectorDistFun = vectorWeightedDist),
 outlierFinder.expr.args = list(bonf.prob = 0.5, transFun = atanh, tail = "upper"),
 outlierFinder.pheno.args = list(normal.upper.thresh = 0.99, bonf.prob = NULL, tail =
    "upper"),
  smokingGunFinder.args = list(transFun = I),
 impute.knn.args = list(k = 10, rowmax = 0.5, colmax = 0.8, maxp = 1500, rng.seed =
    362436069),
  manual.smokingguns = NULL,
  automatic.smokingguns = FALSE,
  within.datasets.only = FALSE,
  intermediate.pruning = FALSE,
  cache.dir = "cache",
  BPPARAM = bpparam(),
  verbose = TRUE
)
```

### Arguments

| | |
|---|---|
| `esets` | a list of ExpressionSets, containing the numeric and phenotypic data to be analyzed. |
| `separator` | a delimitor to use between dataset names and sample names |
| `corFinder.args` | a list of arguments to be passed to the corFinder function. |

`phenoFinder.args`
: a list of arguments to be passed to the phenoFinder function. If NULL, samples with similar phenotypes will not be searched for.

`outlierFinder.expr.args`
: a list of arguments to be passed to outlierFinder when called for expression data

`outlierFinder.pheno.args`
: a list of arguments to be passed to outlierFinder when called for phenotype data

`smokingGunFinder.args`
: a list of arguments to be passed to smokingGunFinder

`impute.knn.args`
: a list of arguments to be passed to impute::impute.knn. Set to NULL to do no knn imputation.

`manual.smokingguns`
: a character vector of phenoData columns that, if identical, will be considered evidence of duplication

`automatic.smokingguns`
: automatically look for "smoking guns." If TRUE, look for phenotype variables that are unique to each patient in dataset 1, also unique to each patient in dataset 2, but contain exact matches between datasets 1 and 2.

`within.datasets.only`
: If TRUE, only search within each dataset for doppelgangers.

`intermediate.pruning`
: The default setting FALSE will result in output with no missing values, but uses extra memory because all results from the expression, phenotype, and smoking gun doppelganger searches must be saved until the end. Setting this to TRUE will save memory for very large searches, but distance metrics will only be available if that value was identified as a doppelganger (for example, phenotype doppelgangers will have missing values for the expression and smoking gun similarity).

`cache.dir`
: The name of a directory in which to cache or look up results to save re-calculating correlations. Set to NULL for no caching.

`BPPARAM`
: Argument for BiocParallel::bplapply(), by default will use all cores of a multicore machine

`verbose`
: Print progress information

## Value

Returns an object of S4-class "DoppelGang"

## Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

## See Also

DoppelGang-class BiocParallelParam-class

## Examples

```
example("phenoFinder")

results2 <- doppelgangR(esets2, cache.dir = NULL)
results2
plot(results2)
summary(results2)

## Set phenoFinder.args=NULL to ignore similar phenotypes, and
## turn off ComBat batch correction:

## Not run:
results2 <- doppelgangR(testesets,
corFinder.args=list(use.ComBat=FALSE), phenoFinder.args=NULL,
    cache.dir=NULL)
summary(results2)

library(curatedOvarianData)
data(GSE32062.GPL6480_eset)
data(GSE32063_eset)
data(GSE12470_eset)
data(GSE17260_eset)

testesets <- list(JapaneseA = GSE32062.GPL6480_eset,
    JapaneseB = GSE32063_eset,
    Yoshihara2009 = GSE12470_eset,
    Yoshihara2010 = GSE17260_eset)

## standardize the sample ids to improve matching
## based on clinical annotation

testesets <- lapply(testesets, function(X) {
  X$alt_sample_name <-
    paste(X$sample_type, gsub("[^0-9]", "", X$alt_sample_name), sep = "_")
  pData(X) <-
    pData(X)[,!grepl("uncurated_author_metadata", colnames(pData(X)))]
  X[, 1:20]  ##speed computations
})

(results1 <- doppelgangR(testesets, cache.dir = NULL))
plot(results1)
summary(results1)


## End(Not run)
```

---

dst *Skew-t Distribution*

---

## Description

Density function, distribution function and random number generation for the skew-$t$ (ST) distribution. Functions copied from sn CRAN library v0.4.18 for argument name compatibility with st.mle function from the same version.

## Usage

```
dst(x, location = 0, scale = 1, shape = 0, df = Inf, dp = NULL, log = FALSE)

rst(n = 1, location = 0, scale = 1, shape = 0, df = Inf, dp = NULL)

pst(x, location = 0, scale = 1, shape = 0, df = Inf, dp = NULL, ...)

qst(
  p,
  location = 0,
  scale = 1,
  shape = 0,
  df = Inf,
  tol = 1e-06,
  dp = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| x | vector of quantiles. Missing values (NAs) are allowed. |
| location | vector of location parameters. |
| scale | vector of (positive) scale parameters. |
| shape | vector of shape parameters. With pst and qst, it must be of length 1. |
| df | degrees of freedom (scalar); default is df=Inf which corresponds to the skew-normal distribution. |
| dp | a vector of length 4, whose elements represent location, scale (positive), shape and df, respectively. If dp is specified, the individual parameters cannot be set. |
| log | logical; if TRUE, densities are given as log-densities. |
| n | sample size. |
| ... | additional parameters passed to integrate. |
| p | vector of probabililities |
| tol | a scalar value which regulates the accuracy of the result of qsn. |

## Value

Density (dst), probability (pst), quantiles (qst) and random sample (rst) from the skew-$t$ distribution with given location, scale, shape and df parameters.

## Details

Typical usages are

```
scale=1, shape=0, df=Inf, log=FALSE) dst(x, dp=, log=FALSE) pst(x,
location=0, scale=1, shape=0, df=Inf, ...) pst(x, dp=, log=FALSE) qst(p,
location=0, scale=1, shape=0, df=Inf, tol=1e-8, ...) qst(x, dp=, log=FALSE)
rst(n=1, location=0, scale=1, shape=0, df=Inf) rst(x, dp=, log=FALSE)
```

## References

Azzalini, A. and Capitanio, A. (2003). Distributions generated by perturbation of symmetry with emphasis on a multivariate skew-*t* distribution. *J.Roy. Statist. Soc. B* **65**, 367–389.

## See Also

st.mle

## Examples

```
pdf <- dst(seq(-4,4,by=0.1), shape=3, df=5)
rnd <- rst(100, 5, 2, -5, 8)
q <- qst(c(0.25,0.5,0.75), shape=3, df=5)
stopifnot(identical(all.equal(pst(q, shape=3, df=5), c(0.25,0.5,0.75)), TRUE))
```

---

# mst.mle *Maximum likelihood estimation for a (multivariate) skew-t distribution*

---

## Description

Fits a skew-t (ST) or multivariate skew-t (MST) distribution to data, or fits a linear regression model with (multivariate) skew-t errors, using maximum likelihood estimation. Functions copied from sn CRAN library v0.4.18 because they were later deprecated in that library.

## Usage

```
mst.mle(
  X,
  y,
  freq,
  start,
  fixed.df = NA,
  trace = FALSE,
  algorithm = c("nlminb", "Nelder-Mead", "BFGS", "CG", "SANN"),
  control = list()
)
```

```
st.mle(
  X,
  y,
  freq,
  start,
  fixed.df = NA,
  trace = FALSE,
  algorithm = c("nlminb", "Nelder-Mead", "BFGS", "CG", "SANN"),
  control = list()
)
```

## Arguments

| | |
|---|---|
| X | a matrix of covariate values. If missing, a one-column matrix of 1's is created; otherwise, it must have the same number of rows of y. If X is supplied, then it must include a column of 1's. |
| y | a matrix (for mst.mle) or a vector (for st.mle). If y is a matrix, rows refer to observations, and columns to components of the multivariate distribution. |
| freq | a vector of weights. If missing, a vector of 1's is created; otherwise it must have length equal to the number of rows of y. |
| start | for mst.mle, a list contaning the components beta,Omega, alpha, df of the type described below; for st.mle, a vector whose components contain analogous ingredients as before, with the exception that the scale parameter is the square root of Omega. In both cases, the dp component of the returned list from a previous call has the required format and it can be used as a new start. If the start parameter is missing, initial values are selected by the function. |
| fixed.df | a scalar value containing the degrees of freedom (df), if these must be taked as fixed, or NA (default value) if df is a parameter to be estimated. |
| trace | logical value which controls printing of the algorithm convergence. If trace=TRUE, details are printed. Default value is FALSE. |
| algorithm | a character string which selects the numerical optimization procedure used to maximize the loglikelihood function. If this string is set equal to "nlminb", then this function is called; in all other cases, optim is called, with method set equal to the given string. Default value is "nlminb". |
| control | this parameter is passed to the chose optimizer, either nlminb or optim; see the documentation of this function for its usage. |

## Details

If y is a vector and it is supplied to mst.mle, then it is converted to a one-column matrix, and a scalar skew-t distribution is fitted. This is also the mechanism used by st.mle which is simply an interface to mst.mle.

The parameter freq is intended for use with grouped data, setting the values of y equal to the central values of the cells; in this case the resulting estimate is an approximation to the exact maximum likelihood estimate. If freq is not set, exact maximum likelihood estimation is performed.

likelihood estimation, use st.mle.grouped.

Numerical search of the maximum likelihood estimates is performed in a suitable re-parameterization of the original parameters with aid of the selected optimizer (`nlminb` or `optim`) which is supplied with the derivatives of the log-likelihood function. Notice that, in case the optimizer is `optim`), the gradient may or may not be used, depending on which specific method has been selected. On exit from the optimizer, an inverse transformation of the parameters is performed. For a specific description on the re-parametrization adopted, see Section 5.1 and Appendix B of Azzalini \& Capitanio (2003).

## Value

A list containing the following components:

| | |
|---|---|
| `call` | a string containing the calling statement. |
| `dp` | for `mst.mle`, this is a list containing the direct parameters `beta`, `Omega`, `alpha`. Here, `beta` is a matrix of regression coefficients with `dim(beta)=c(ncol(X),ncol(y))`, `Omega` is a covariance matrix of order `ncol(y)`, `alpha` is a vector of shape parameters of length `ncol(y)`. For `st.mle`, `dp` is a vector of length `ncol(X)+3`, containing `c(beta, omega, alpha,df)`, where `omega` is the square root of `Omega`. |
| `se` | a list containing the components `beta`, `alpha`, `info`. Here, `beta` and `alpha` are the standard errors for the corresponding point estimates; `info` is the observed information matrix for the working parameter, as explained below. |
| `algorithm` | the list returned by the chose optimizer, either `nlminb` or `optim`, plus an item with the `name` of the selected algorithm; see the documentation of either `nlminb` or `optim` for explanation of the other components. |

## Background

The family of multivariate skew-t distributions is an extension of the multivariate Student's t family, via the introduction of a `shape` parameter which regulates skewness; when `shape=0`, the skew-t distribution reduces to the usual t distribution. When `df=Inf` the distribution reduces to the multivariate skew-normal one; see `dmsn`. See the reference below for additional information.

## References

Azzalini, A. and Capitanio, A. (2003). Distributions generated by perturbation of symmetry with emphasis on a multivariate skew *t* distribution. The full version of the paper published in abriged form in *J.Roy. Statist. Soc. B* **65**, 367–389, is available at http://azzalini.stat.unipd.it/SN/se-ext.ps

## See Also

`dst`

## Examples

```
dat <- rt(100, df=5, ncp=100)
fit <- st.mle(y=dat)
fit
```

`outlierFinder` *Identifies outliers in a similarity matrix.*

## Description

By default uses the Fisher z-transform for Pearson correlation (atanh), and identifies outliers as those above the quantile of a skew-t distribution with mean and standard deviation estimated from the z-transformed matrix. The quantile is calculated from the Bonferroni-corrected cumulative probability of the upper tail.

## Usage

```
outlierFinder(
  similarity.mat,
  bonf.prob = 0.05,
  transFun = atanh,
  normal.upper.thresh = NULL,
  tail = "upper"
)
```

## Arguments

| | |
|---|---|
| `similarity.mat` | A matrix of similarities - larger values mean more similar. |
| `bonf.prob` | Bonferroni-corrected probability. A raw.prob is calculated by dividing this by the number of non-missing values in similarity.mat, and the rejection threshold is qnorm(1-raw.prob, mean, sd) where mean and sd are estimated from the transFun-transformed similarity.mat. |
| `transFun` | A function applied to the numeric values of similarity.mat, that should result in normally-distributed values. |
| `normal.upper.thresh` | Instead of specifying bonf.prob and transFun, an upper similarity threshold can be set, and values above this will be considered likely duplicates. If specified, this over-rides bonf.prob. |
| `tail` | "upper" to look for samples with very high similarity values, "lower" to look for very low values, or "both" to look for both. |

## Value

Returns either NULL or a dataframe with three columns: sample1, sample2, and similarity.

## Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

## Examples

```
library(curatedOvarianData)
data(GSE32063_eset)
cormat <- cor(exprs(GSE32063_eset))
outlierFinder(cormat, bonf.prob = 0.05)
```

---

`phenoDist` *Calculate distance between two vectors, rows of one matrix/dataframe, or rows of two matrices/dataframes.*

---

## Description

This function does some simple looping to allow x and y to be various combinations of vectors and matrices/dataframes.

## Usage

```
phenoDist(x, y = NULL, bins = 10, vectorDistFun = vectorWeightedDist, ...)
```

## Arguments

| | |
|---|---|
| `x` | A vector, matrix or dataframe |
| `y` | NULL, a vector, matrix, or dataframe. If x is a vector, y must also be specified. |
| `bins` | discretize continuous fields in the specified number of bins |
| `vectorDistFun` | A function of two vectors that returns the distance between those vectors. |
| `...` | Extra arguments passed on to vectorDistFun |

## Value

a matrix of distances between pairs of rows of x (if y is unspecified), or between all pairs of rows between x and y (if both are provided).

## Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

## Examples

```
example("phenoFinder")

pdat1 <- pData(esets2[[1]])
pdat2 <- pData(esets2[[2]])

## Use phenoDist() to calculate a weighted distance matrix
distmat <- phenoDist(as.matrix(pdat1), as.matrix(pdat2))
## Note outliers with identical clinical data, these are probably the same patients:
```

```
graphics::boxplot(distmat)

## Not run:
   library(curatedOvarianData)
   data(GSE32063_eset)
   data(GSE17260_eset)
   pdat1 <- pData(GSE32063_eset)
   pdat2 <- pData(GSE17260_eset)
   ## Curation of the alternative sample identifiers makes duplicates stand out more:
   pdat1$alt_sample_name <-
     paste(pdat1$sample_type,
           gsub("[^0-9]", "", pdat1$alt_sample_name),
           sep = "_")
   pdat2$alt_sample_name <-
     paste(pdat2$sample_type,
           gsub("[^0-9]", "", pdat2$alt_sample_name),
           sep = "_")
  ## Removal of columns that cannot possibly match also helps duplicated patients to stand out
   pdat1 <-
     pdat1[,!grepl("uncurated_author_metadata", colnames(pdat1))]
   pdat2 <-
     pdat2[,!grepl("uncurated_author_metadata", colnames(pdat2))]
   ## Use phenoDist() to calculate a weighted distance matrix
   distmat <- phenoDist(as.matrix(pdat1), as.matrix(pdat2))
   ## Note outliers with identical clinical data, these are probably the same patients:
   graphics::boxplot(distmat)

## End(Not run)
```

---

phenoFinder *Calculate pairwise similarities of phenoData between samples for a list containing two ExpressionSets*

---

## Description

This function acts as a wrapper to phenoDist to handle cases of one ExpressionSet, a list of two identical ExpressionSets, or a list of two different ExpressionSets.

## Usage

```
phenoFinder(eset.pair, separator = ":", ...)
```

## Arguments

| | |
|---|---|
| eset.pair | input: a list of ExpressionSets with two elements, or an ExpressionSet. If the two elements are identical, return the correlation matrix for pairs of samples in the first element. If not identical, return pairs between the two elements. |

| | |
|---|---|
| separator | a separator between dataset name (taken from the list names) and sample name (taken from sampleNames(eset), to keep track of which samples come from which dataset. |
| ... | Extra arguments passed on to phenoDist |

## Value

A matrix of similarities between the phenotypes of pairs of samples.

## Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

## Examples

```
library(curatedOvarianData)
data(GSE32063_eset)
data(GSE17260_eset)
esets2 <- list(JapaneseB=GSE32063_eset,
               Yoshihara2010=GSE17260_eset)

## standardize the sample ids to improve matching based on clinical annotation
esets2 <- lapply(esets2, function(X){
   X$alt_sample_name <- paste(X$sample_type, gsub("[^0-9]", "", X$alt_sample_name), sep="_")

## Removal of columns that cannot possibly match also helps duplicated patients to stand out
    pData(X) <- pData(X)[, !grepl("uncurated_author_metadata", colnames(pData(X)))]
    X <- X[, 1:20]  ##speed computations
    return(X) })

## See first six samples in both rows and columns
phenoFinder(esets2)[1:6, 1:6]
```

---

# plot-methods *Histograms of all pairwise sample correlations, showing identified doppelgangers.*

---

## Description

Identified doppelgangers are shown with a red vertical line overlaid on a histogram of pairwise sample correlations. One plot is made per pair of datasets.

## Usage

```
## S4 method for signature 'DoppelGang,ANY'
plot(x, skip.no.doppels = FALSE, plot.pair = NULL, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class DoppelGang |
| skip.no.doppels | (default FALSE) If TRUE, do not plot histograms where no doppelgangers were identified. |
| plot.pair | An optional character vector of length two, providing the names of two datasets. If provided, only the comparison of these two datasets will be plotted. |
| ... | Additional arguments passed on to hist. |

## Value

None

## Methods

**list("signature(x = \"DoppelGang\")")** Histograms of all pairwise sample correlations, showing identified doppelgangers.

## Author(s)

Levi Waldron

## Examples

```
library(curatedOvarianData)
data(TCGA_eset)
data(GSE26712_eset)
## Remove some TCGA samples to speed computation:
keep.tcga <-
c("TCGA.13.2060", "TCGA.24.2290", "TCGA.25.2392", "TCGA.25.2404",
"TCGA.59.2349", "TCGA.09.2044", "TCGA.24.2262", "TCGA.24.2293",
"TCGA.25.2393", "TCGA.25.2408", "TCGA.59.2350", "TCGA.09.2045",
"TCGA.24.2267", "TCGA.59.2351", "TCGA.09.2048", "TCGA.24.2271",
"TCGA.24.2298", "TCGA.25.2398", "TCGA.59.2354", "TCGA.09.2050",
"TCGA.24.2281", "TCGA.09.2051", "TCGA.29.2428", "TCGA.09.2055",
"TCGA.24.2289", "TCGA.29.2414", "TCGA.59.2352", "TCGA.36.2532",
"TCGA.36.2529", "TCGA.36.2551", "TCGA.42.2590", "TCGA.13.2071",
"TCGA.29.2432", "TCGA.36.2537", "TCGA.36.2547", "TCGA.04.1369",
"TCGA.42.2591", "TCGA.23.2641", "TCGA.29.2434", "TCGA.36.2538",
"TCGA.36.2548", "TCGA.04.1516", "TCGA.42.2593", "TCGA.36.2549",
"TCGA.04.1644", "TCGA.13.2057", "TCGA.23.2647", "TCGA.36.2530",
"TCGA.36.2552", "TCGA.42.2587", "TCGA.13.2061", "TCGA.42.2588",
"TCGA.36.2544", "TCGA.42.2589", "TCGA.13.2066", "TCGA.61.2613",
"TCGA.61.2614", "TCGA.24.1852", "TCGA.29.1704", "TCGA.13.1819"
)
keep.tcga <- unique(c(keep.tcga, sampleNames(TCGA_eset)[1:200]))
testesets <- list(Bonome08=GSE26712_eset, TCGA=TCGA_eset[, keep.tcga])
results1 <- doppelgangR(testesets,
    corFinder.args=list(use.ComBat=FALSE), phenoFinder.args=NULL,
    cache.dir=NULL)
plot(results1)
```

smokingGunFinder *Find doppelgangers based on "smoking gun" phenotypes - those that should be unique to each patient.*

## Description

Checks all pairwise combinations of samples for values of the "smoking" gun phenotypes that are identical.

## Usage

```
smokingGunFinder(eset.pair, smokingguns, transFun = I, separator = ":")
```

## Arguments

| | |
|---|---|
| eset.pair | a list of ExpressionSets, with two elements. If the two elements are identical, the function will check for duplicate IDs within one element. If not identical, it will check for duplicate IDs between elements. |
| smokingguns | phenoData column names found in multiple elements of eset.pair that may contain "smoking guns" such as identifiers that should be unique to each sample. |
| transFun | a function to apply to IDs before comparing. By default apply no transformation. |
| separator | Separator between dataset name and sample name. Dataset names are added to sample names to keep track of dataset of origin. |

## Value

Returns an adjacency matrix for samples where matches have value 1, non-matches have value zero. Value for a sample against itself is NA.

## Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

## Examples

```
example("phenoFinder")

smokingGunFinder(esets2, "days_to_death")
```

`vectorHammingDist` *Calculate Hamming Distance between two vectors, using pairwise complete observations.*

## Description

Simple function to count the fraction of different elements (in the same position) between two vectors of the same length, after removing elements from both vectors corresponding to positions that are NA in either vector.

## Usage

```
vectorHammingDist(x, y, k, l)
```

## Arguments

| | |
|---|---|
| x | a matrix |
| y | a matrix with the same number of columns as x |
| k | row in x to test for differences |
| l | row in y to test for differences |

## Value

Returns a numeric value, the Hamming Distance (the number of non-equal values between x and y).

## Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

## Examples

```
(mat <- matrix(c(paste0("A", 1:5), paste0("A", 5:1)),
    nrow = 2, byrow = TRUE))

stopifnot(vectorHammingDist(mat, mat, 1, 2) == 0.8)
stopifnot(vectorHammingDist(mat, mat, 1, 1) == 0)

mat[1, 1] <- NA

stopifnot(vectorHammingDist(mat, mat, 1, 2) == 0.75)
stopifnot(vectorHammingDist(mat, mat, 1, 1) == 0)

mat[1, 3] <- NA

stopifnot(vectorHammingDist(mat, mat, 1, 2) == 1)
```

`vectorWeightedDist` *Calculate a weighted distance between two vectors, using pairwise complete observations.*

## Description

Simple function to count the fraction of different elements (in the same position) between two vectors of the same length, after removing elements from both vectors corresponding to positions that are NA in either vector. Distance is the probability for observing the matches and mismatches in two random patients.

## Usage

```
vectorWeightedDist(x, y, k, l)
```

## Arguments

| | |
|---|---|
| x | a matrix |
| y | a matrix with the same number of columns as x |
| k | row in x to test for differences |
| l | row in y to test for differences |

## Value

Returns a numeric value, the log of the probability of observing the matches in x and y

## Author(s)

Levi Waldron, Markus Riester, Marcel Ramos

## Examples

```
mymat1 <- matrix(rnorm(20), ncol = 5)
mymat1[1, 4] <- NA
mymat2 <- matrix(rnorm(20), ncol = 5)
vectorWeightedDist(mymat1, mymat2, 1, 2)
```

# Index